Welcome

OpenID: Why SSO Serious?

chsh

Topics we will cover

1. Introduction to OAuth and OpenID
2. Deep dive into the OAuth protocol
3. History of vulnerabilities in OAuth and OpenID
4. Modern vulnerabilities in OpenID

Learning outcomes

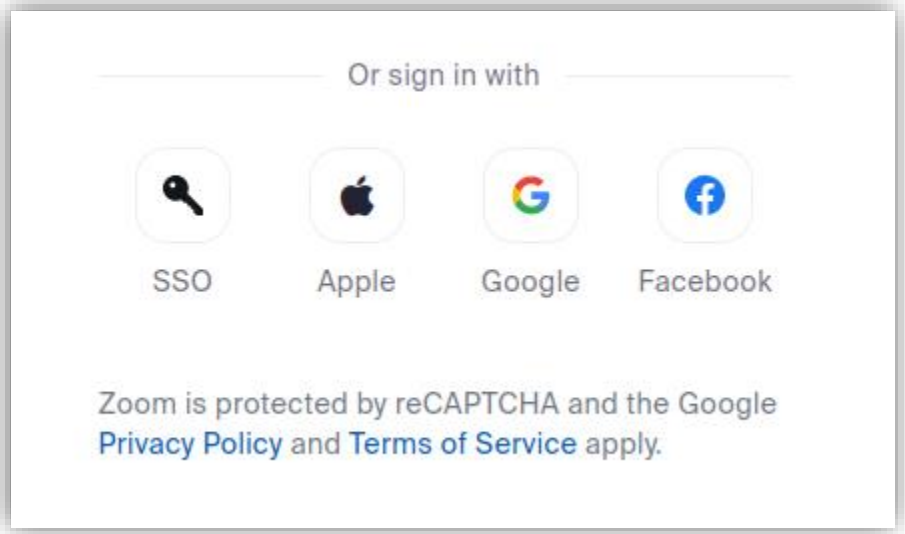
To gain a better understanding of the topics discussed, and hopefully answer the following questions:

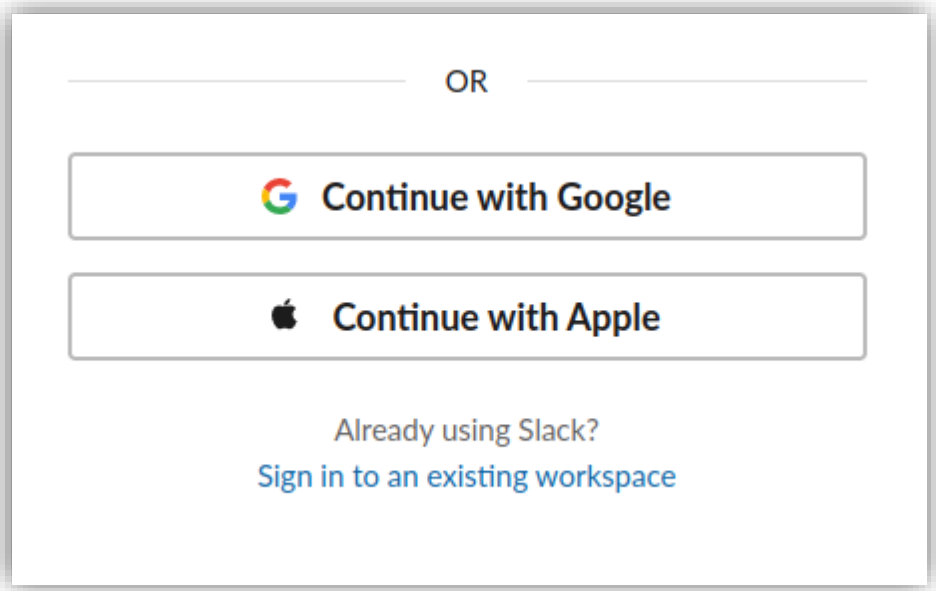
1. What is the OAuth protocol? What is it used for?
2. What is the OpenID Connect protocol? How does it relate to OAuth?
3. Where have vulnerabilities occurred in OAuth and OpenID Connect?

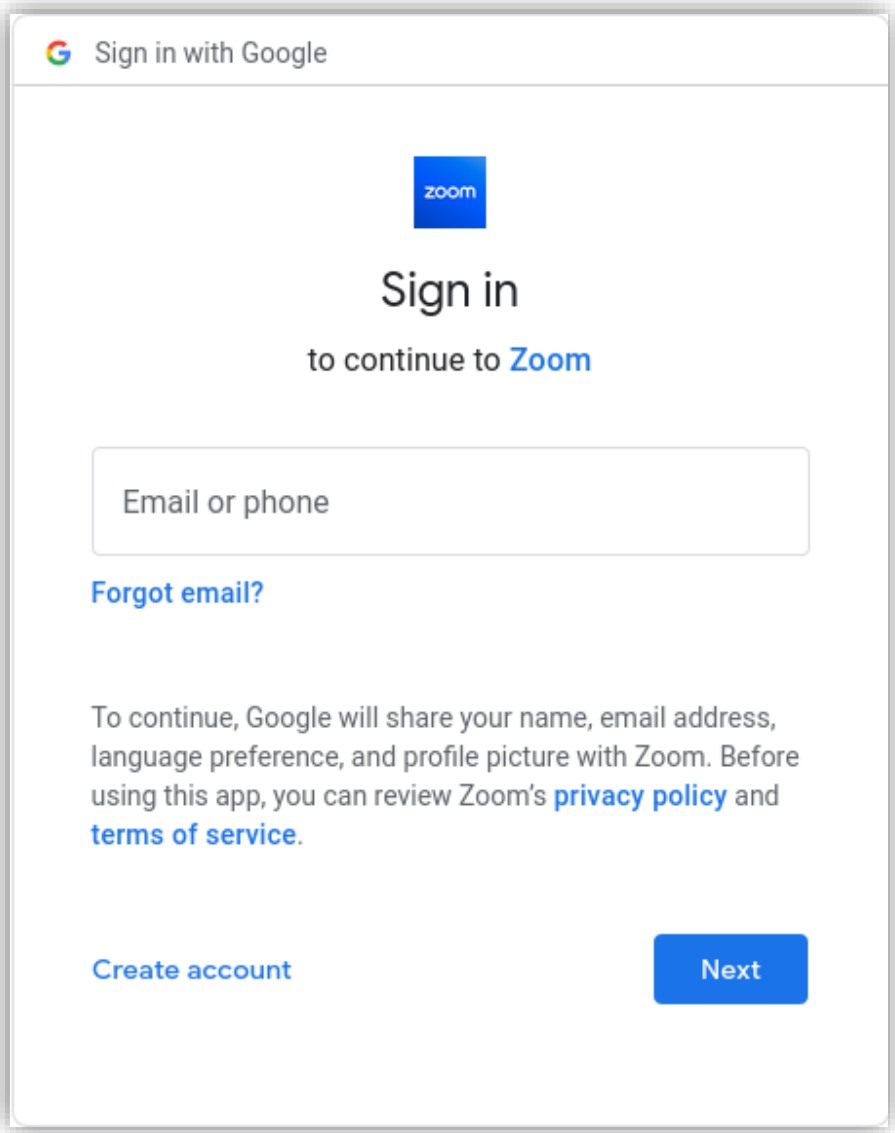
Learning outcomes

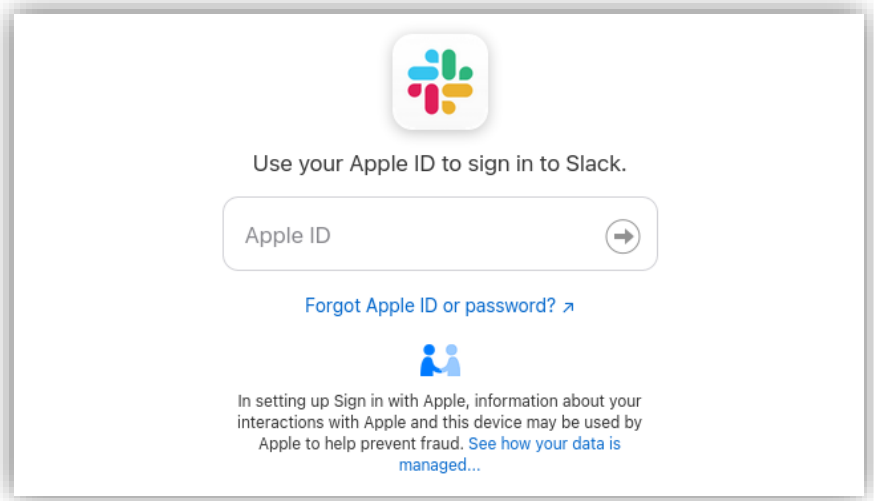
Let’s Begin!

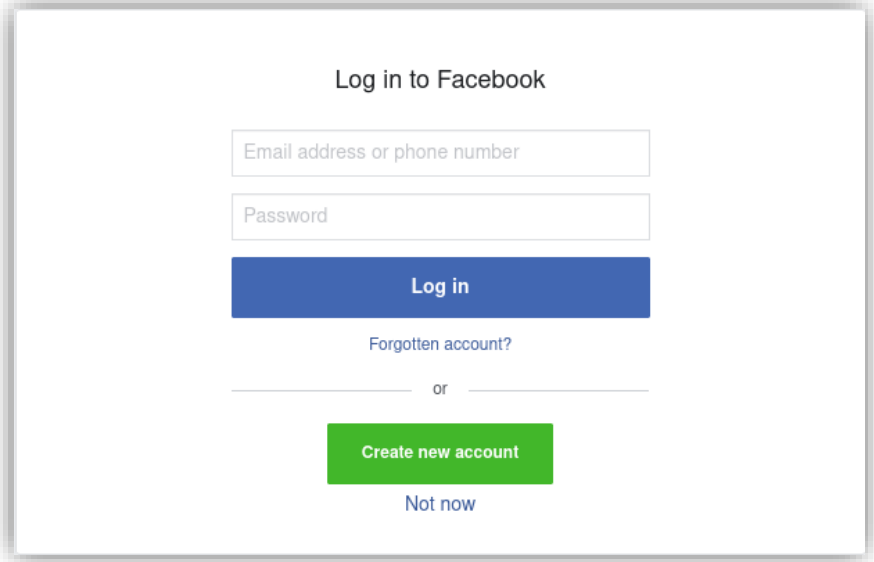
Single sign-on





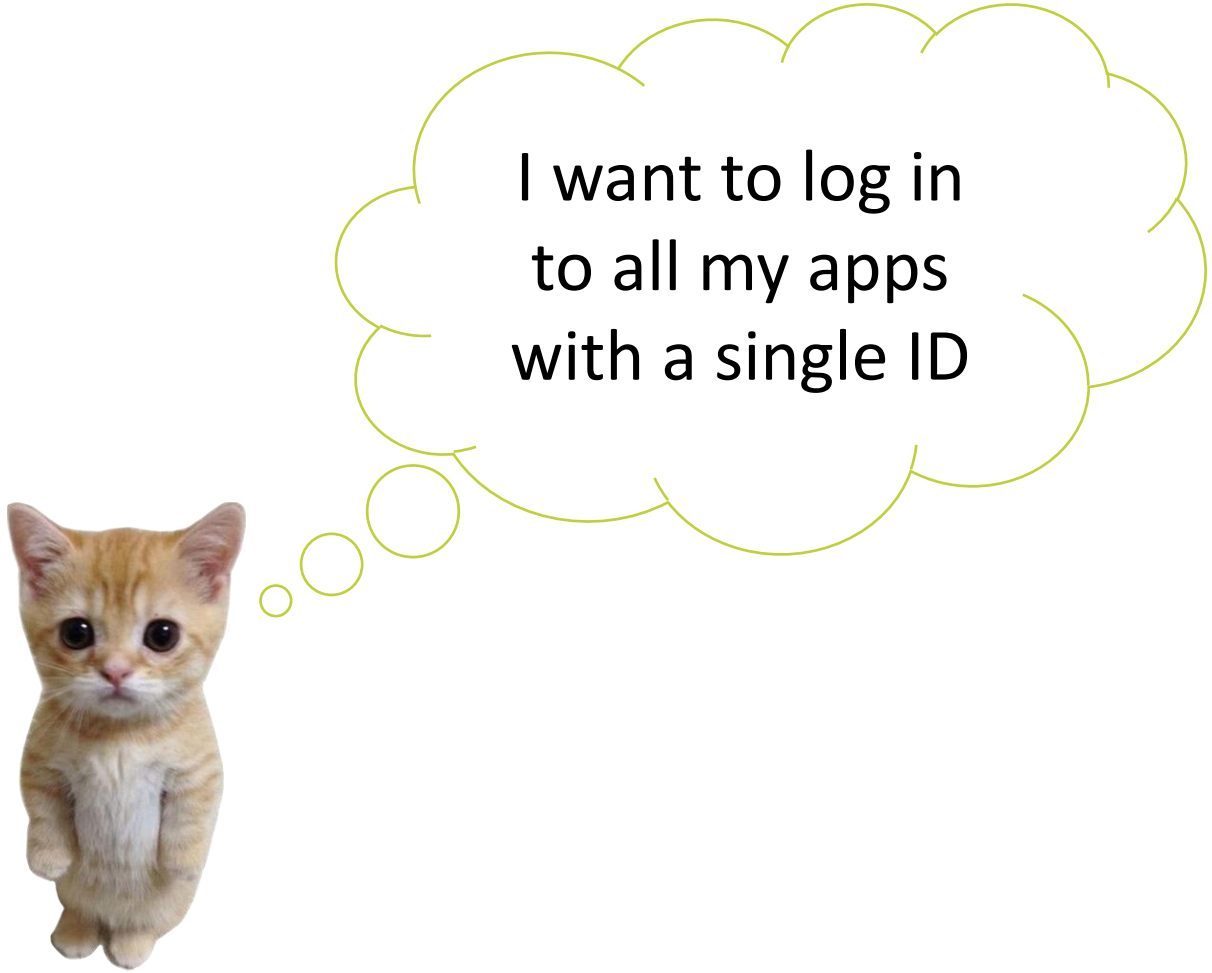

Sign in with Google
zoom
Sign in
to continue to Zoom
Email or phone
Forgot email?
To continue, Google will share your name, email address, language preference, and profile picture with Zoom. Before using this app, you can review Zoom's privacy policy and terms of service.
Create account
Next


Use your Apple ID to sign in to Slack.
Apple ID
Forgot Apple ID or password?
In setting up Sign in with Apple, information about your interactions with Apple and this device may be used by Apple to help prevent fraud. See how your data is managed...


Log in to Facebook
Email address or phone number
Password
Log in
Forgotten account?
or
Create new account
Not now

Single sign-on

- Problem:



Why use SSO?

Pros

- **Password management is hard**
- Let big corporations manage your biggest risk
- Faster and easier authentication

Cons

- Increases the impact of account compromise
- Increased risk of phishing threats
- And more…

SSO Implementations

- Security Assertion Markup Language (SAML)
- Kerberos
  - for private networks
- **OpenID Connect**

AuthN vs AuthZ

- **Authentication**
  - Identifies user
  - e.g. passport

- **Authorization**
  - Defines user permissions (rights to do an action)
  - e.g. visa, license

OIDC and OAuth

- OIDC is based on OAuth 2.0
- OAuth 2.0 provides **authorization** to access resources
- OIDC provides user **authentication**
- Often combined into single service (IAM)

Introduction: OAuth

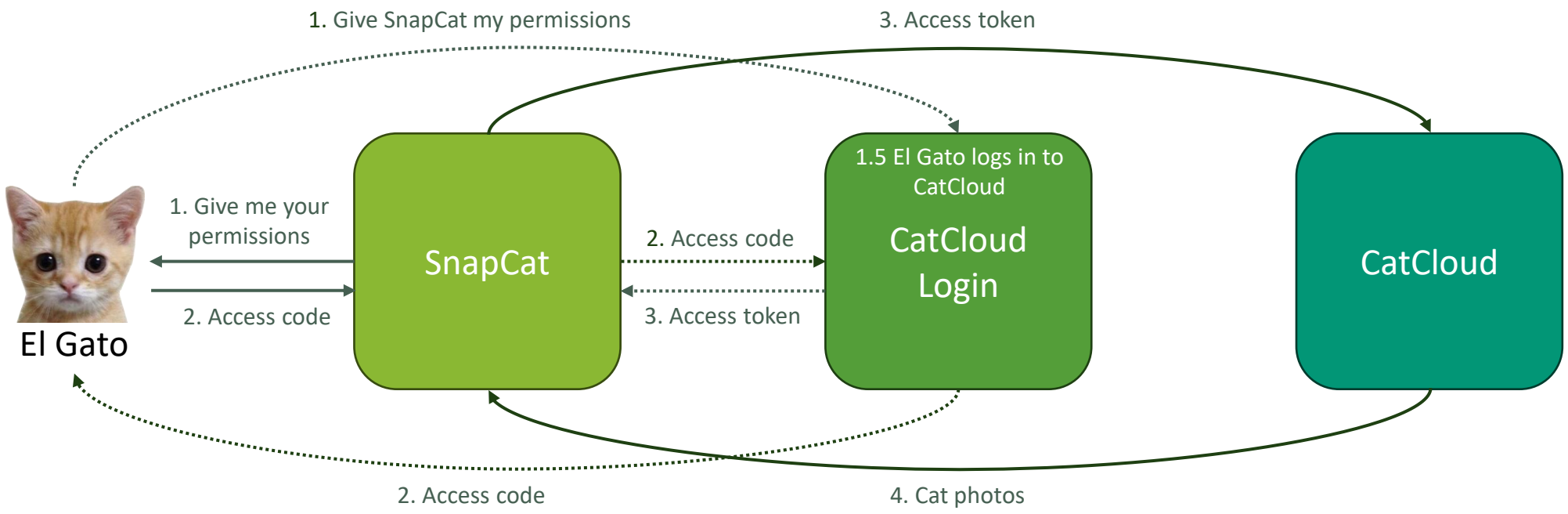
OAuth

- Designed to share resource access (authorization) to third parties
  - e.g., a third-party mail application sending mail on my behalf
- Third-party applications shouldn’t need to own my account to be authorised to perform certain actions
  - e.g., the third-party mail application shouldn’t need my email password or be able to read my emails

OAuth

Problem:

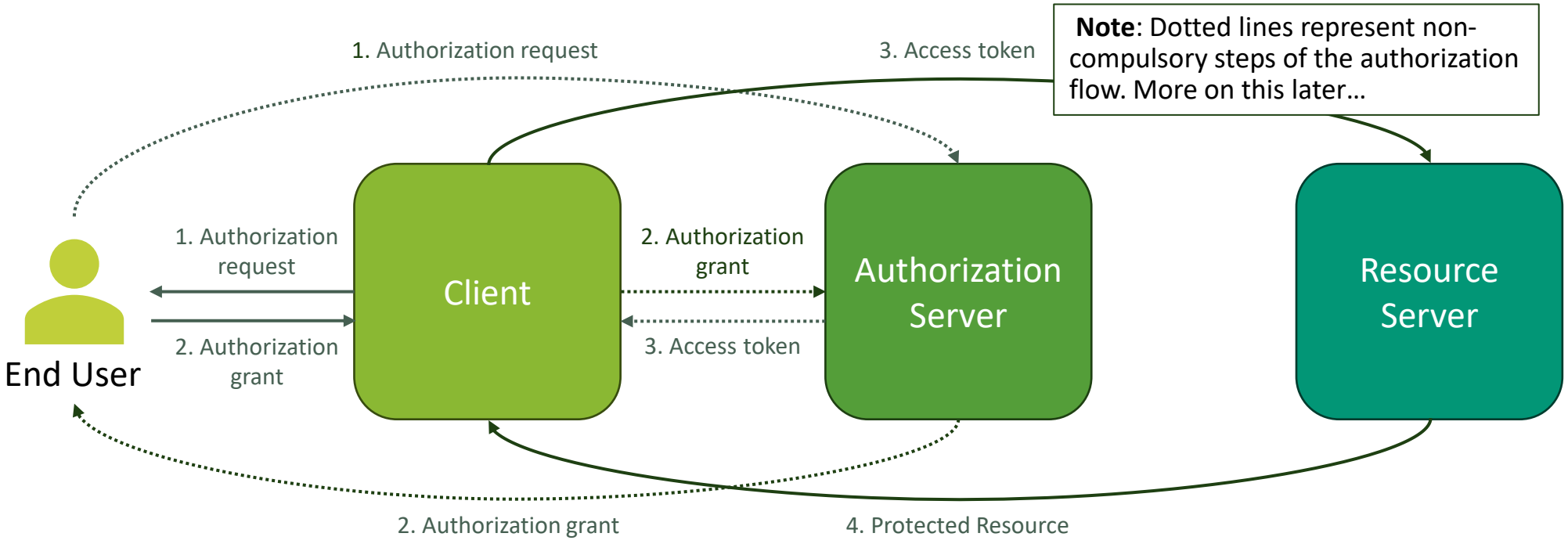
- **SnapCat** wants to access **El Gato's** photos from the **CatCloud**, which manages access using **CatCloud login**



OAuth

Problem:

- **Client** wants to access resources from the **Resource Server** as **End User**, whose access is managed by **Authorization Server**



Introduction: OpenID Connect

OIDC

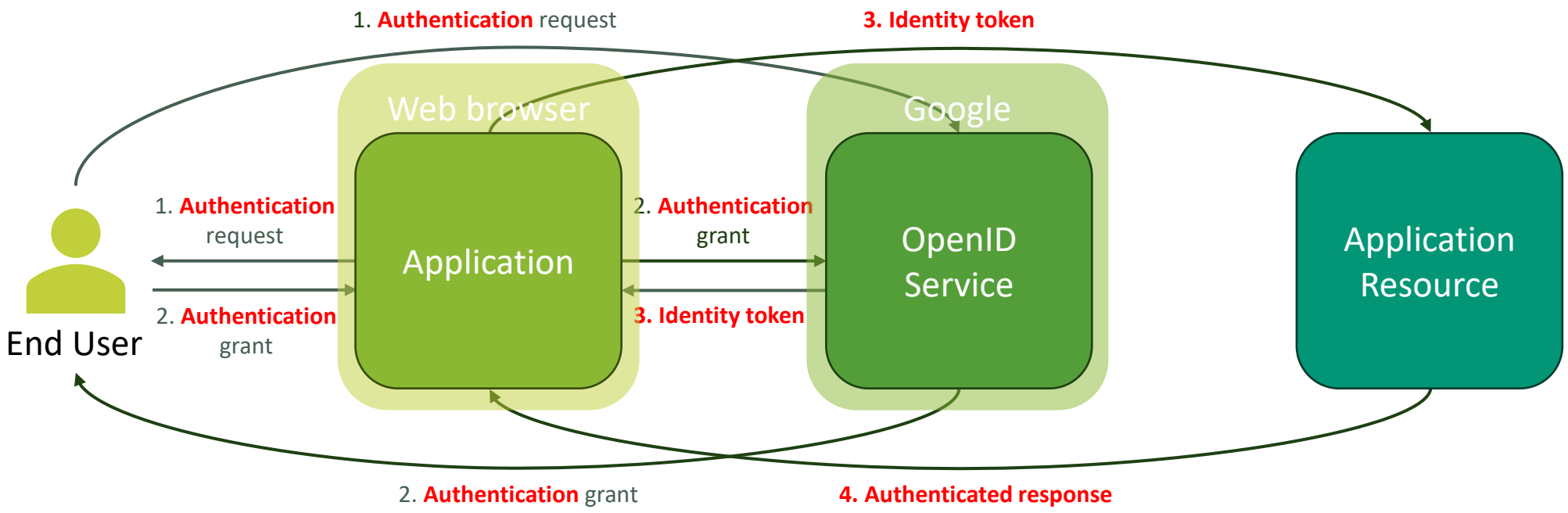
Problem:

- **Application** needs to know the identity of an **End User**

- Third party application **Identity Service** can identify the End User
  - By email: Google, Microsoft
  - By pre-existing account: Facebook
  - By integrating with identity management service: Okta

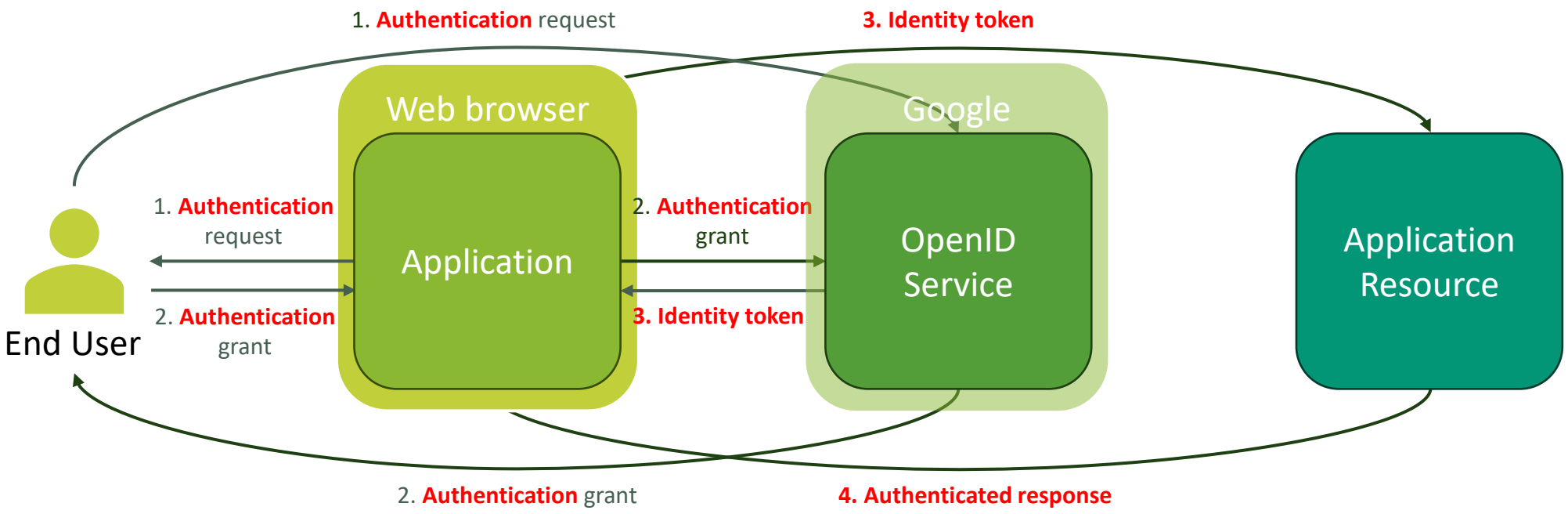
OIDC

- OAuth was designed to share resource access to a third party
- OIDC uses OAuth to “share” information about a user’s identity
  - Issued by an authority, e.g., Google



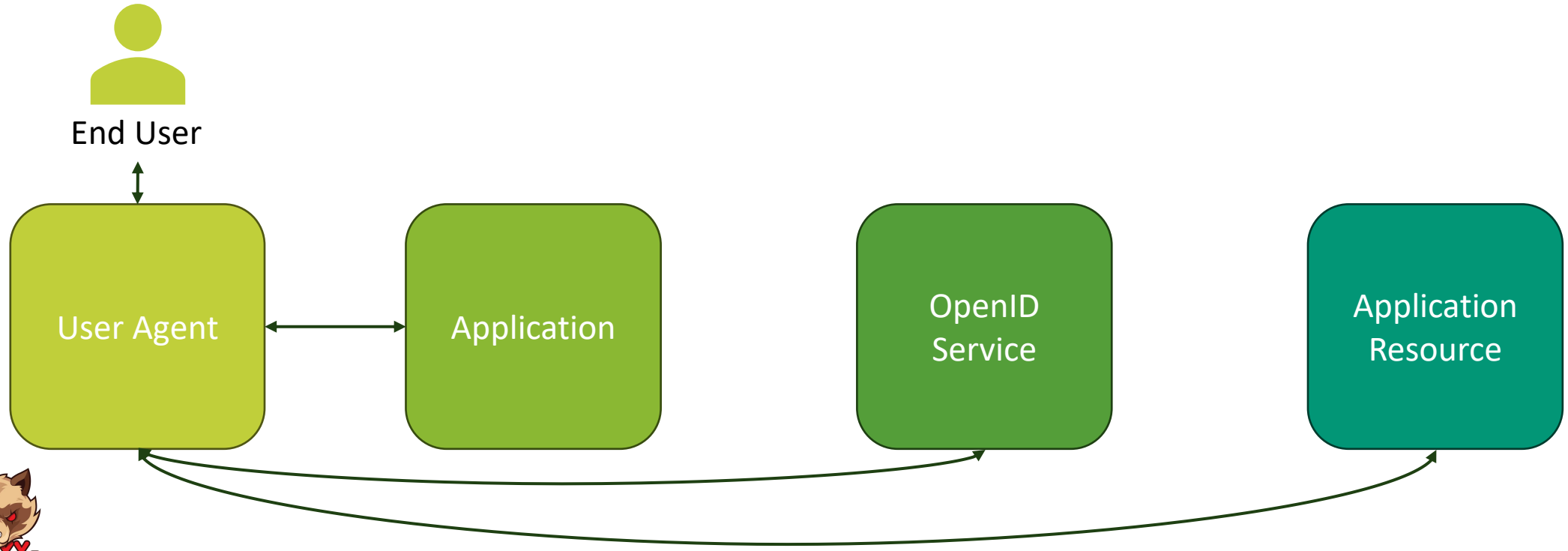
OIDC

- The web browser needs to have certain functionality, e.g.
  - Make “redirects”
  - Hold on to information (cookies)



OIDC

- Abstractly, the browser is sometimes referred to as the User Agent
  - Not in scope for this class



OpenID Configuration

- Provider configurations can be found at:

  *example.com**/.well-known/openid-configuration***

https://accounts.google.com/.well-known/openid-configuration


```
{
 "issuer": "https://accounts.google.com",
 "authorization_endpoint": "https://accounts.google.com/o/oauth2/v2/auth",
 "device_authorization_endpoint": "https://oauth2.googleapis.com/device/code",
 "token_endpoint": "https://oauth2.googleapis.com/token",
 "userinfo_endpoint": "https://openidconnect.googleapis.com/v1/userinfo",
 "revocation_endpoint": "https://oauth2.googleapis.com/revoke",
 "jwks_uri": "https://www.googleapis.com/oauth2/v3/certs",
 "response_types_supported": [
  "code",
  "token",
  "id_token",
  "code token",
  "code id_token",
  "token id_token",
  "code token id_token",
  "none"
 ],
 "subject_types_supported": [
  "public"
 ],
 "id_token_signing_alg_values_supported": [
  "RS256"
 ],
 "scopes_supported": [
  "openid",
  "email",
  "profile"
 ],
 "token_endpoint_auth_methods_supported": [
  "client_secret_post",
  "client_secret_basic"
 ],
 "claims_supported": [
  "aud",
  "email",
  "email_verified",
  "exp",
  "family_name",
  "given_name",
  "iat",
  "iss",
  "locale",
  "name",
  "picture",
  "sub"
 ],
 "code_challenge_methods_supported": [
  "plain",
  "S256"
 ],
 "grant_types_supported": [
  "authorization_code",
  "refresh_token",
  "urn:ietf:params:oauth:grant-type:device_code",
  "urn:ietf:params:oauth:grant-type:jwt-bearer"
 ]
}
```

Network Traffic

Burp Proxy

Burp Project Intruder Repeater Window Help

Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Settings

Organizer Extensions Learn IIS Tilde Enum Scanner Autorize

Intercept HTTP history WebSockets history Proxy settings

Filter: Hiding CSS, image and general binary content

#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Ex
222	https://www.gstatic.com	GET	/_/mss/boq-identity/_/js/k=boq-identi...			200	7151	script	
221	https://accounts.google.com	GET	/_/bscframe			200	1287	HTML	
220	https://accounts.youtube.com	GET	/accounts/CheckConnection?pmpo=...	✓		200	36110	HTML	
219	https://www.gstatic.com	GET	/_/mss/boq-identity/_/js/k=boq-identi...			200	4216	script	
218	https://www.gstatic.com	GET	/_/mss/boq-identity/_/js/k=boq-identi...			200	2505	script	
217	https://www.gstatic.com	GET	/_/mss/boq-identity/_/js/k=boq-identi...			200	2219	script	
216	https://www.gstatic.com	GET	/_/mss/boq-identity/_/js/k=boq-identi...			200	4147	script	
215	https://www.gstatic.com	GET	/_/mss/boq-identity/_/js/k=boq-identi...			200	15100	script	
214	https://www.gstatic.com	GET	/_/mss/boq-identity/_/js/k=boq-identi...			200	38488	script	
213	https://www.gstatic.com	GET	/_/mss/boq-identity/_/js/k=boq-identi...			200	124022	script	
211	https://accounts.google.com	GET	/v3/signin/identifier?dsh=S154396740...	✓		200	613339	HTML	
210	https://accounts.google.com	GET	/o/oauth2/v2/auth?client_id=6060929...	✓		302	4474	HTML	
209	https://slack.com	GET	/signin/oauth/google/start?email_firs...	✓		302	1375	HTML	
208	https://slack.com	POST	/api/signin.findWorkspaces?_x_id=no...	✓		200	1304	JSON	
207	https://a.slack-edge.com	GET	/bv1-10/slack_logo-ebd02d1.svg			200	5808	XML	svg
205	https://a.slack-edge.com	GET	/80588/fonts/lato-2-compressed/lato-...			200	220334		wof
204	https://a.slack-edge.com	GET	/bv1-10/slack-icons-v2-fe043a5.woff2			200	69301		wof
203	https://a.slack-edge.com	GET	/80588/fonts/lato-2-compressed/lato-...			200	203150		wof
202	https://cdn.cookielaw.org	GET	/scripttemplates/202211.1.0/assets/v...			200	63081	JSON	jsor
201	https://cdn.cookielaw.org	GET	/scripttemplates/202211.1.0/assets/ot...			200	14031	JSON	jsor
200	https://a.slack-edge.com	GET	/bv1-10/get-started.f8133a0.primer.m...			200	8294753	script	js
199	https://a.slack-edge.com	GET	/bv1-10/primer-vendor.6218cc2.prime...			200	415159	script	js
198	https://cdn.cookielaw.org	GET	/consent/3bcd90cf-1e32-46d7-adbd-6...			200	126790	JSON	jsor
197	https://geolocation.onetrust.c...	GET	/cookieconsentpub/v1/geo/location			200	422	JSON	
196	https://a.slack-edge.com	GET	/bv1-10/manifest.b039310.primer.min...			200	3362	script	js
195	https://a.slack-edge.com	GET	/80588/fonts/lato-2-compressed/lato-...			200	204242		wof

Firefox

Inspector Console Debugger Network Style Editor Performance Memory Storage Accessibility Application

Filter URLs

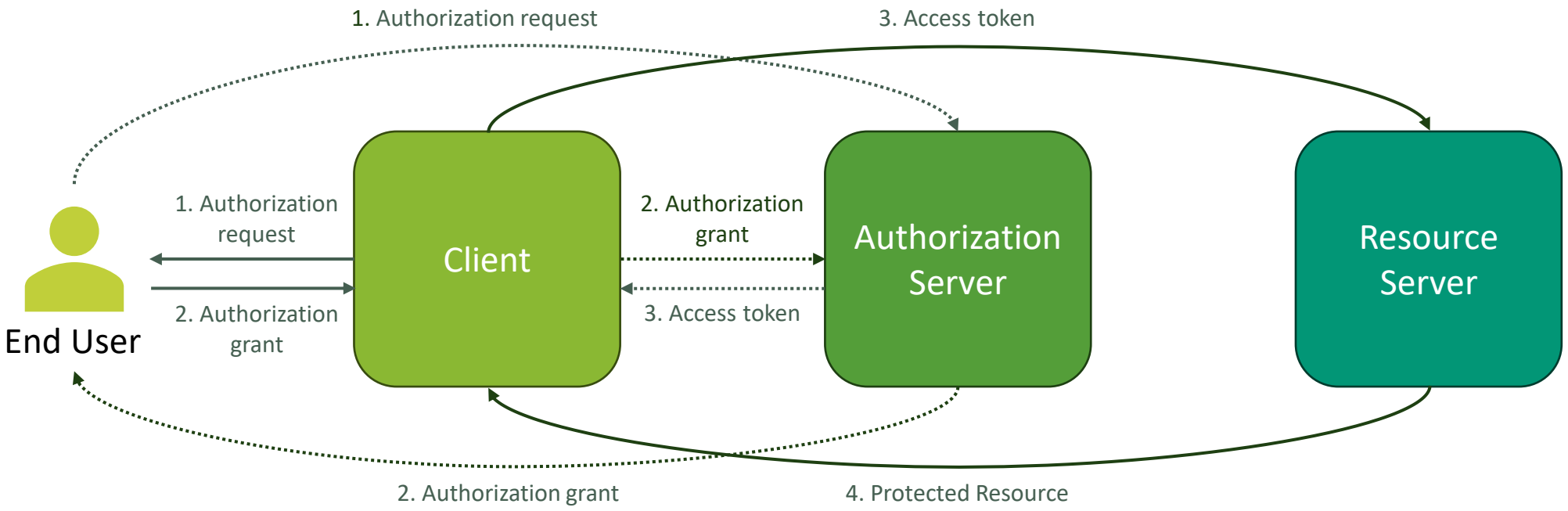
Status	Method	Domain	File
200	GET	duckduckgo.com	/?t=ffab&q=test
200	GET	links.duckduckgo.com	d.js?q=test&t=D&l=au-en&s=0&a=ffab&ct=AU&vqd=4-267680164436175476272800067280757
200	GET	duckduckgo.com	s2167.css
200	GET	duckduckgo.com	r2167.css
200	GET	duckduckgo.com	wbml2167.css
200	GET	duckduckgo.com	b218.js
200	GET	duckduckgo.com	ProximaNova-Reg-webfont.woff2
200	GET	duckduckgo.com	ProximaNova-Sbold-webfont.woff2
200	GET	duckduckgo.com	l132.js
200	GET	duckduckgo.com	duckduckgo14.js
200	GET	duckduckgo.com	u720.js
200	GET	duckduckgo.com	wbvm19.js
200	GET	duckduckgo.com	wbmm134.js
200	GET	duckduckgo.com	d3222.js
	GET	duckduckgo.com	g3000.js
	GET	duckduckgo.com	logo_header.v109.svg
200	GET	duckduckgo.com	t.js?q=test&t=D&l=au-en&s=0&ct=AU&p_ent=&ex=-1&dfrsp=1&biaexp=b&eclsexp=b&litexp=b&
200	GET	duckduckgo.com	post3.html
200	GET	duckduckgo.com	country.json
200	GET	duckduckgo.com	p104.js
200	GET	duckduckgo.com	DDG-iOS-icon_152x152.png?v=2
200	GET	duckduckgo.com	favicon.ico
200	GET	duckduckgo.com	s2490.js
200	GET	external-content.duckduckgo.com	www.speedtest.net.ico

Introduction: Recap

- OpenID Connect (OIDC) is one of many SSO implementations
- OIDC uses the OAuth 2.0 protocol, which was designed for authorization not authentication
- OAuth 2.0 actors and flow
  - Authorization request
  - Authorization grant
  - Access token

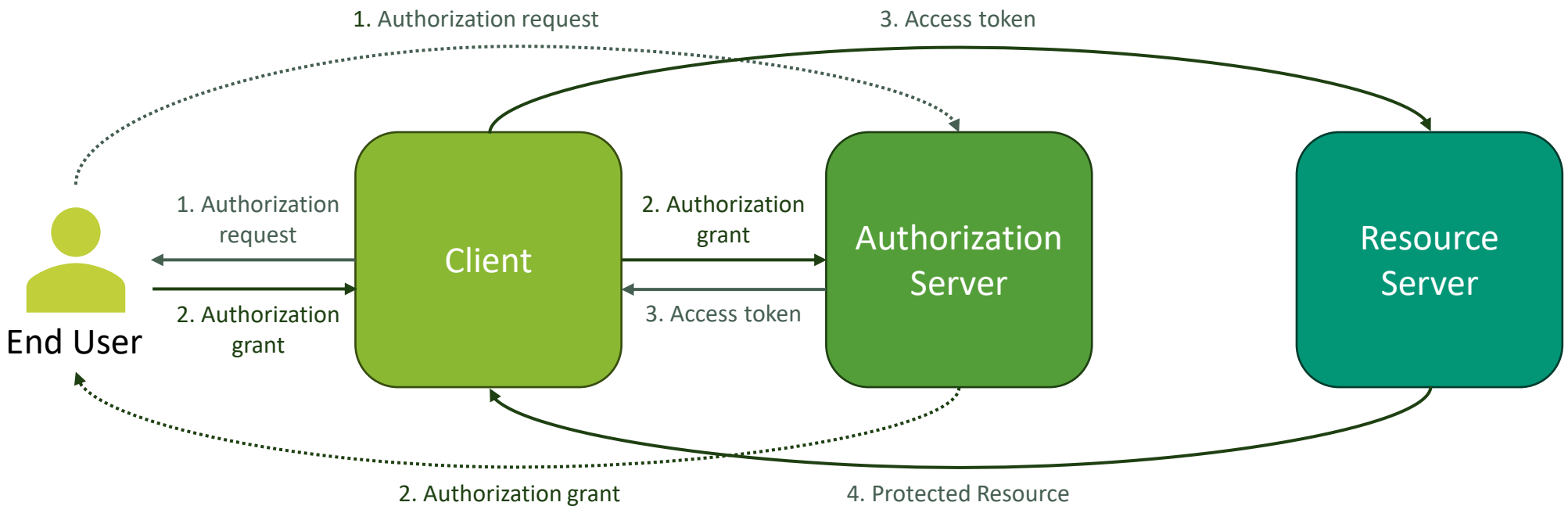
OAuth 2.0
(the OAuth Dance)

A deeper dive into the protocol

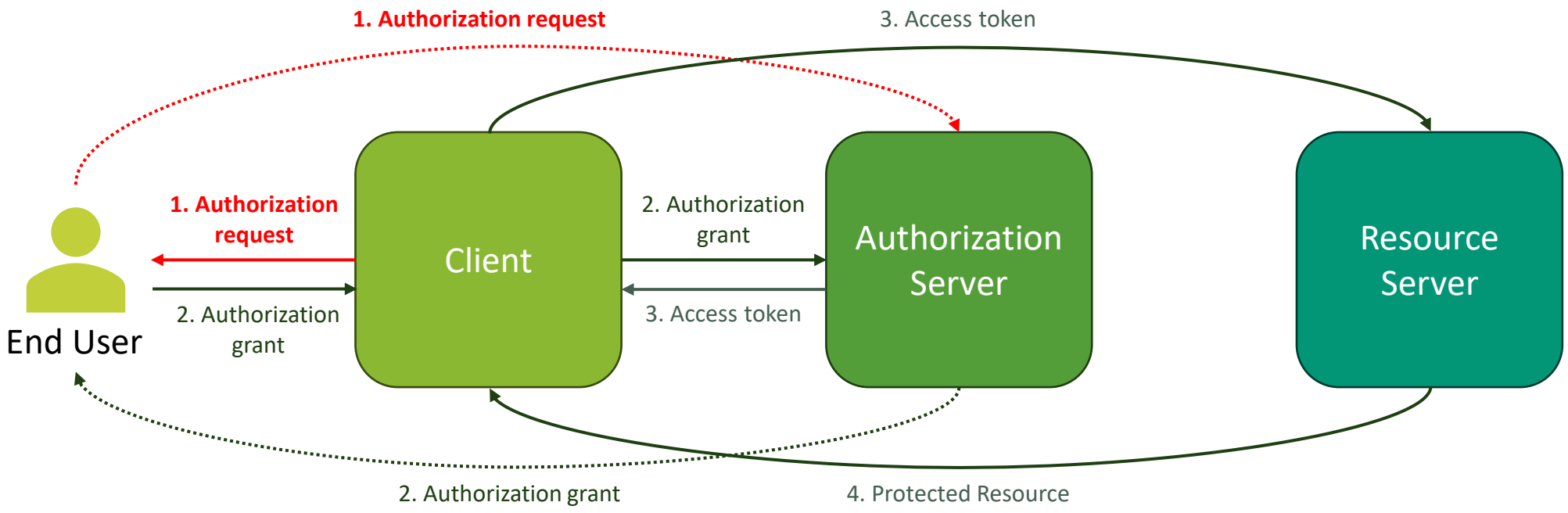


OAuth 2.0: Authorization request

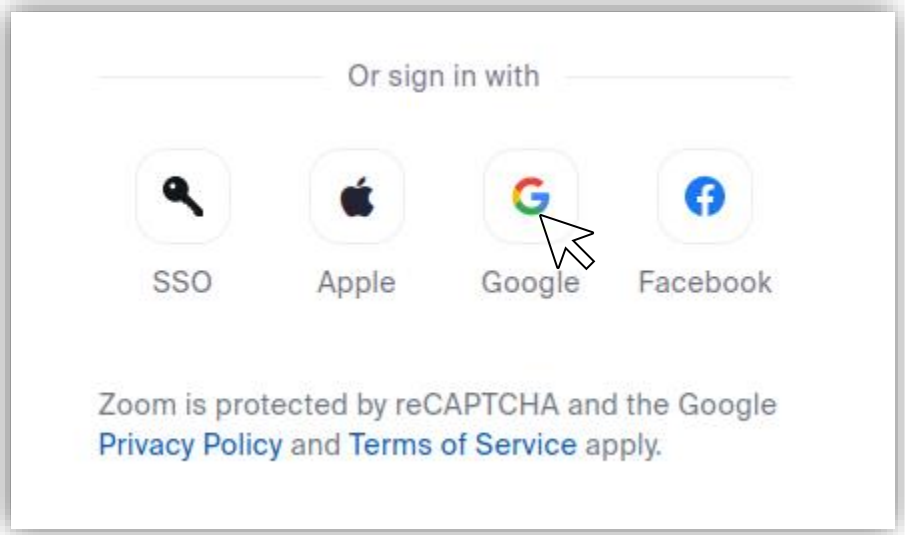
Authorization Request

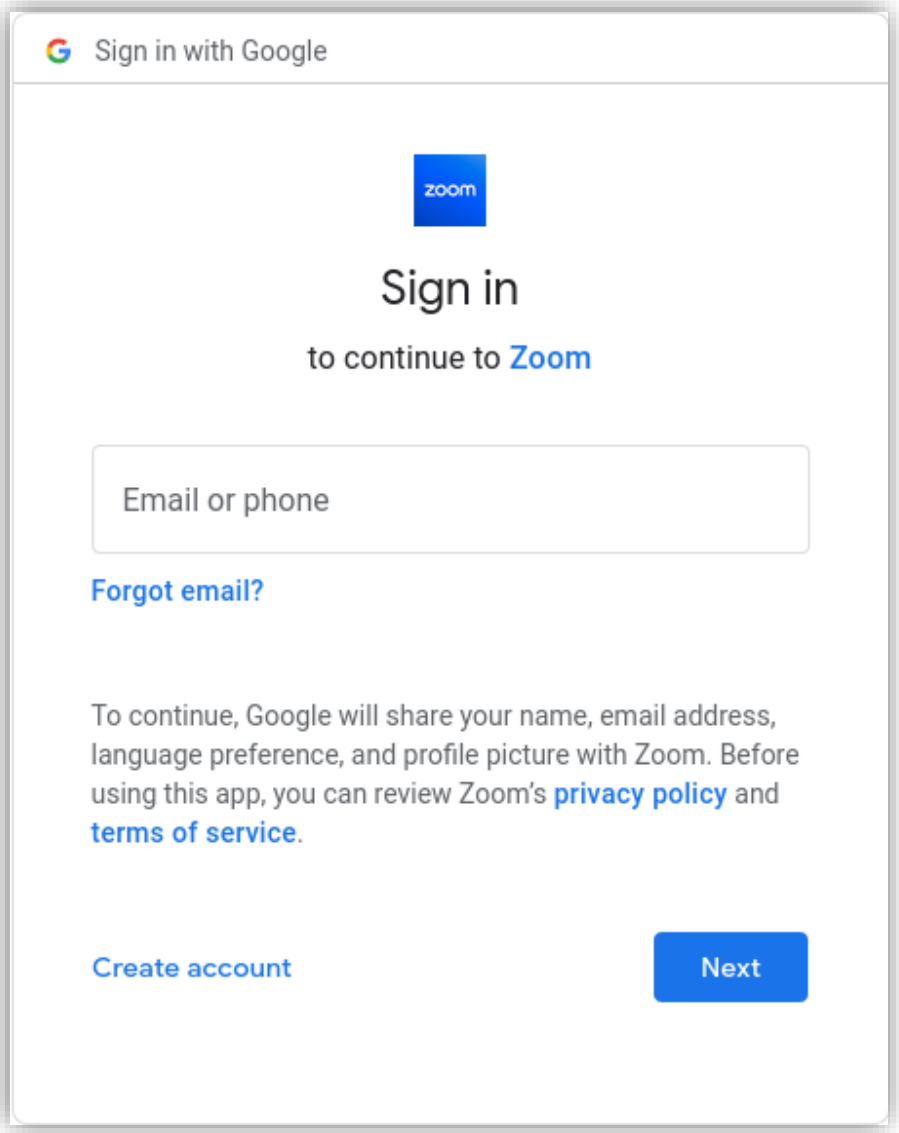


Authorization Request



Authorization Request





Authorization Request

- Initiated from the client (e.g., Zoom web or the Zoom app)
  - After clicking “Sign in with Google”
- Redirects end user to authorization server
  - Usually by popping up a new window/tab with the login portal
- Prompts user to log in using username and password
  - Browser could also have stored identity e.g., already logged in to Google

Example1: Zoom

14	https://zoom.us	GET	/google_oauth_signin	302	3049

Request

Pretty Raw Hex

```
1 GET /google_oauth_signin HTTP/1.1
2 Host: zoom.us
3 Cookie: _zm_ssid=aw1_c_Q2dTQSJbTQa04wCyFZP2Iw; cred=
C86E290474DB938A99547993070F0BB4; _zm_ctaid=
Gt3XDdPrThyieNAZNAwBUg.1687222792540.23bf44959f2bb129d75fb69
7b1e8e829; _zm_chtaid=326; _zm_mtk_guid=
2ac96641bf3b4563931b6ce5d2896ea5; _zm_csp_script_nonce=
51baTnNUTma8FmSClrgbMw; _zm_currency=AUD; _zm_visitor_guid=
2ac96641bf3b4563931b6ce5d2896ea5; __cf_bm=
wGWfoSMVrZWthK0hdgvhgjmCEpz9GIHdw1rbMKQxDGs-1687222793-0-Abg
Ks2oET0sd/S1+JAw2+9uIcEOMgFChfY6dLYfzDCzQXGHQ63gtrO1vaGki2s7
ox3dU0tk+jr9LHUhx267UPeE=; wULrMv6t=
A8O2UNaIAQAA2bR-ubEuNGvVP5Qkwk9h6JD9u6H47M7yVj35PTby1Rrz9UQA
AS2E4YCcuBZ2wH8AADQwAAAAAA|1|0|1a87c49a82342febb0385064b176f
755c84261db; _zm_page_auth=aw1_c_j3Uv8jh2R2-w54qHH6cmzA;
_zm_lang=en-US; _zm_cdn_blocked=unlog_unblk;
cdn_detect_result=enable; _zm_fingerprint=
ddf0b9fe8325be79cc0709db740625e0; OptanonConsent=
isGpcEnabled=0&datestamp=Tue+Jun+20+2023+11%3A00%3A50+GMT%2B
1000+(Australian+Eastern+Standard+Time)&version=6.21.0&isIAB
Global=false&hosts=&consentId=5e03764f-e068-469f-bcfd-5e2baa
4566ed&interactionCount=2&landingPath=NotLandingPage&groups=
C0004%3A0%2CC0003%3A0%2CC0002%3A0%2CC0001%3A1&AwaitingRecons
ent=false&geolocation=AU%3BVIC; OnetrustActiveGroups=C0001;
OptanonAlertBoxClosed=2023-06-20T00:59:48.746Z; _zm_o2state=
0MdPa2x4SXSiUd4AZ1MUvA; zm_oauth_back_params=
eyJiYXN1X3VybCI6Imh0dHBzOi8vem9vbS51cyJ9
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0)
```

Response

Pretty Raw Hex Render

```
1 HTTP/2 302 Found
2 Date: Tue, 20 Jun 2023 01:01:12 GMT
3 Content-Length: 0
4 Location:
https://accounts.google.com/o/oauth2/v2/auth?response_type=c
ode&access_type=offline&client_id=849883241272-ed6lnodi1grno
omiuknqkq2rbvd2udku.apps.googleusercontent.com&prompt=select
_account&scope=profile%20email&redirect_uri=https%3A%2F%2Fzo
om.us%2Fgoogle%2Foauth&state=ME1kUGEyeDRTWFNpVWQ0QVoxTVV2QSx
nb29nbGVfc2lnbmlu&_x_zm_rtaid=Gt3XDdPrThyieNAZNAwBUg.1687222
792540.23bf44959f2bb129d75fb697b1e8e829&_x_zm_rhtaid=326
5 X-Zm-Trackingid:
v=2.0;clid=aw1;rid=WEB_74258959a2b077a9355b458697834811
6 X-Content-Type-Options: nosniff
7 Content-Security-Policy: upgrade-insecure-requests;
default-src https://*.zoom.us https://zoom.us blob: 'self';
img-src https: about: blob: data: 'self'; style-src https:
safari-extension: chrome-extension: 'unsafe-inline' data:
'self'; font-src https: safari-extension: chrome-extension:
blob: data: 'self'; connect-src * about: blob: data: 'self';
 media-src * rtmp: blob: data: 'self'; frame-src https:
ms-appx-web: zoommtg: zoomus: wvjbscheme: data: blob:
'self'; object-src 'none'; base-uri 'none';
8 X-Frame-Options: SAMEORIGIN
9 Set-Cookie: zm_aid=""; Domain=.zoom.us; Expires=Thu,
01-Jan-1970 00:00:10 GMT; Path=/; Secure; HttpOnly
10 Set-Cookie: zm_haid=""; Domain=.zoom.us; Expires=Thu,
01-Jan-1970 00:00:10 GMT; Path=/; Secure; HttpOnly
```

Example2: Slack

Request

Pretty Raw Hex

```
1  GET /get-started/oauth/google/start?geocode=en-au HTTP/1.1
2  Host: slack.com
3  Cookie: b=bf5604dfbb577c02a6d74a04778ea1f1; OptanonConsent=
   isGpcEnabled=0&datestamp=Tue+Jun+20+2023+14%3A40%3A48+GMT%2B
   1000+(Australian+Eastern+Standard+Time)&version=202211.1.0&i
   sIABGlobal=false&hosts=&consentId=39466cbb-d27d-4d39-9a28-6a
   6c4a34cc9e&interactionCount=1&landingPath=https%3A%2F%2Fslac
   k.com%2Fintl%2Fen-au%2Fget-started%23%2Fcreatenew&groups=1%3
   A1%2C3%3A1%2C2%3A1%2C4%3A0; x=
   bf5604dfbb577c02a6d74a04778ea1f1.1687236063
4  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0)
   Gecko/20100101 Firefox/112.0
5  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/
   avif,image/webp,*/*;q=0.8
6  Accept-Language: en-US,en;q=0.5
7  Accept-Encoding: gzip, deflate
8  Referer: https://slack.com/intl/en-au/get-started
9  Dnt: 1
10 Upgrade-Insecure-Requests: 1
11 Sec-Fetch-Dest: document
12 Sec-Fetch-Mode: navigate
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-User: ?1
15 Te: trailers
16 Connection: close
17
18
```

Response

Pretty Raw Hex Render

```
1  HTTP/2 302 Found
2  Date: Tue, 20 Jun 2023 04:41:15 GMT
3  Server: Apache
4  Vary: Accept-Encoding
5  Location:
   https://accounts.google.com/o/oauth2/v2/auth?client_id=60609
   2904014-s1u3idjanlbhr4ns5b1hcjgfn63cr9nh.apps.googleusercont
   ent.com&redirect_uri=https%3A%2F%2Foauth2.slack.com%2Fget-st
   arted%2Foauth%2Fgoogle%2Fend&scope=openid+email+profile&resp
   onse_type=code&access_type=offline&state=%7B%22provider%22%3
   A%22google%22%2C%22origin%22%3A%22get-started%22%2C%22timest
   amp%22%3A1687236075%2C%22visitor%22%3A%22bf5604dfbb577c02a6d
   74a04778ea1f1%22%2C%22extra%22%3A%7B%22params%22%3A%7B%22geo
   code%22%3A%22en-au%22%7D%7D%7D%7Cbd74d58a478520f78bd06d5a6b7
   6191f382856e5bc999d87c44a7732e4ca3b6a&prompt=consent
6  Strict-Transport-Security: max-age=31536000;
   includeSubDomains; preload
7  X-Slack-Unique-Id: ZJEt64QlXBJq7Eivx6K_EgAAAA8
8  X-Slack-Backend: r
9  Referrer-Policy: no-referrer
10 X-Frame-Options: SAMEORIGIN
11 Content-Type: text/html
12 Content-Length: 0
13 Set-Cookie: x=bf5604dfbb577c02a6d74a04778ea1f1.1687236063;
   expires=Tue, 20-Jun-2023 04:56:15 GMT; Max-Age=900; path=/;
   domain=.slack.com; secure; SameSite=None
14 Via: 1.1 slack-prod.tinyspeck.com, envoy-www-iad-lmbrqjgu,
   envoy-edge-syd-efjauccr
```

Google’s perspective

Parameters:

- response_type
- access_type
- client_id
- prompt
- scope
- redirect_uri
- state

Request

Pretty | Raw | Hex

```
1  GET /o/oauth2/v2/auth?response_type=code&access_type=offline&client_id=
   849883241272-ed6lnodi1grnoomiuknqkq2rbvd2udku.apps.googleusercontent.com
   &prompt=select_account&scope=profile%20email&redirect_uri=
   https%3A%2F%2Fzoom.us%2Fgoogle%2Foauth&state=
   ME1kUGEyeDRTWFNpVWQ0QVoxTVV2QSxnb29nbGVfc2lnbmlu&_x_zm_rtaid=
   Gt3XDdPrThyieNAZNAwBUg.1687222792540.23bf44959f2bb129d75fb697b1e8e829&
   _x_zm_rhtaid=326 HTTP/1.1
2  Host: accounts.google.com
3  Cookie: 1P_JAR=2023-06-20-00; AEC=
   AUEFqZdQ25HcULnd3VgPck7tD_uDbKIWuwzE9BjnWFcPUwog6u_n8q95oA; NID=
   511=tbaGrcWVqpS7GL7DNVINtcDYSnOET1_-NG_Ca4PxeQcrwjpPjeeFXDhOp-nPO4fqZ8Fa
   IORxngusJedMZ0GCGuBoHaRwxqx_O2Ee07LpN01PoxaVzYdWN_M8uOUaolhi7VaCB14MzRdf
   aDxFzUmHDHaGWskcvbeXZ71hRZvwLU0; __Host-GAPS=
   1:VXmg8iKhhzCewpSuc5vduFvXgWqjuA:ZBJ64ZBnXvOQVWDS; OTZ=
   7081981_16_12_133440_16_395520
4  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101
   Firefox/112.0
5  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/w
   ebp,*/*;q=0.8
6  Accept-Language: en-US,en;q=0.5
7  Accept-Encoding: gzip, deflate
8  Referer: https://zoom.us/
9  Dnt: 1
10 Upgrade-Insecure-Requests: 1
11 Sec-Fetch-Dest: document
12 Sec-Fetch-Mode: navigate
13 Sec-Fetch-Site: cross-site
14 Sec-Fetch-User: ?1
15 Te: trailers
16 Connection: close
17
18
```

Google’s perspective

- Does Google know if the request comes from Zoom or Slack? If yes, how?
- Does Google **need** to know? Why?

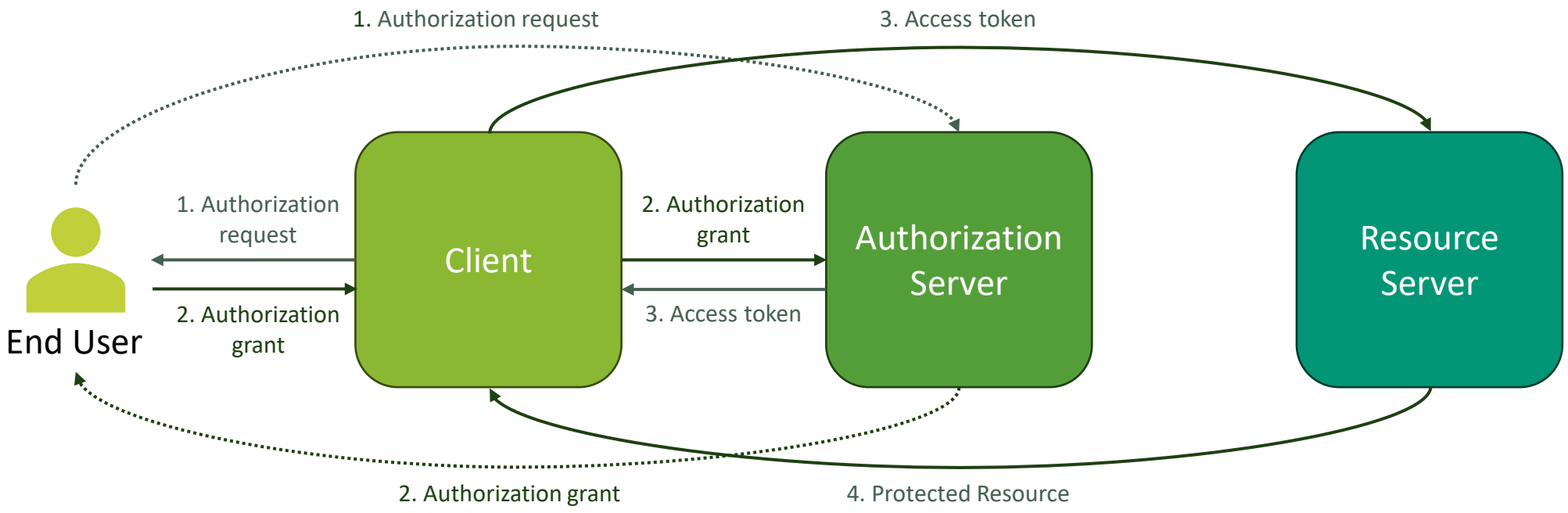
Request

Pretty | Raw | Hex

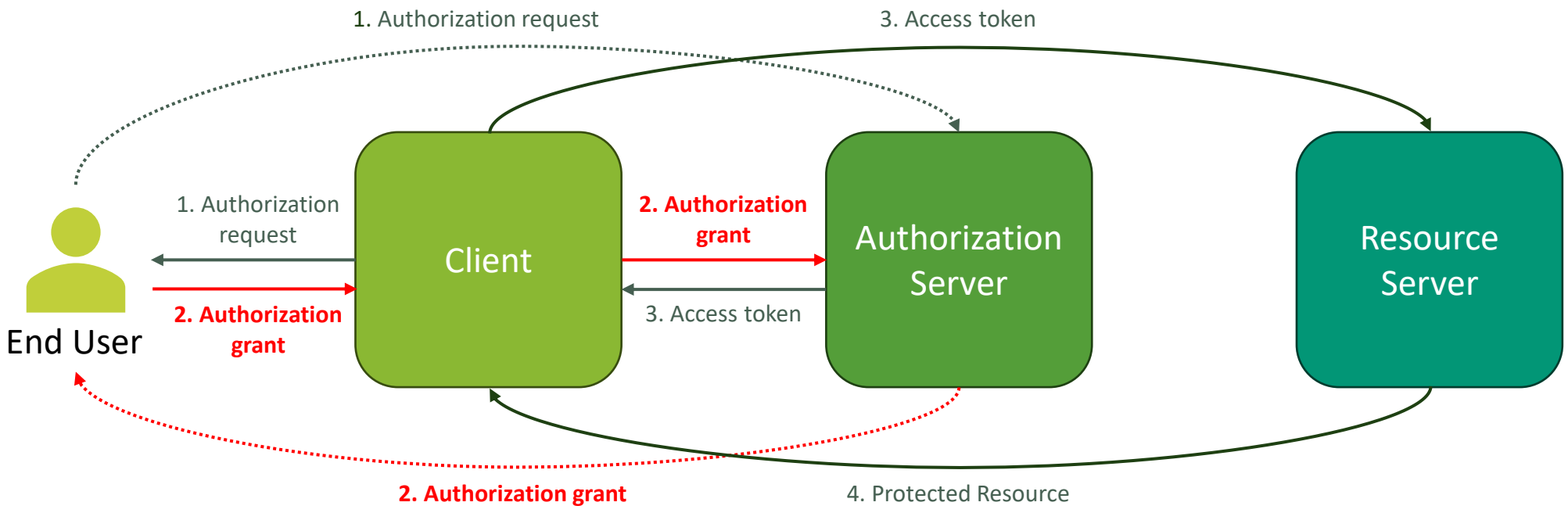
```
1  GET /o/oauth2/v2/auth?response_type=code&access_type=offline&client_id=
   849883241272-ed6lnodi1grnoomiuknqkq2rbvd2udku.apps.googleusercontent.com
   &prompt=select_account&scope=profile%20email&redirect_uri=
   https%3A%2F%2Fzoom.us%2Fgoogle%2Foauth&state=
   ME1kUGEyeDRTWFNpVWQ0QVoxTVV2QSxnb29nbGVfc2lnbmlu&_x_zm_rtaid=
   Gt3XDdPrThyieNAZNAwBUg.1687222792540.23bf44959f2bb129d75fb697b1e8e829&
   _x_zm_rhtaid=326 HTTP/1.1
2  Host: accounts.google.com
3  Cookie: 1P_JAR=2023-06-20-00; AEC=
   AUEFqZdQ25HcULnd3VgPck7tD_uDbKIWuwzE9BjnWFcPUwog6u_n8q95oA; NID=
   511=tbaGrcWVqpS7GL7DNVINtcDYSnOET1_-NG_Ca4PxeQcrwjpPjeeFXDhOp-nPO4fqZ8Fa
   IORxngusJedMZ0GCGuBoHaRwxqx_O2Ee07LpN01PoxaVzYdWN_M8uOUaolhi7VaCB14MzRdf
   aDxFzUmHDHaGWskcvbeXZ71hRZvwLU0; __Host-GAPS=
   1:VXmg8iKhhzCewpSuc5vduFvXgWqjuA:ZBJ64ZBnXvOQVWDS; OTZ=
   7081981_16_12_133440_16_395520
4  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101
   Firefox/112.0
5  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/w
   ebp,*/*;q=0.8
6  Accept-Language: en-US,en;q=0.5
7  Accept-Encoding: gzip, deflate
8  Referer: https://zoom.us/
9  Dnt: 1
10 Upgrade-Insecure-Requests: 1
11 Sec-Fetch-Dest: document
12 Sec-Fetch-Mode: navigate
13 Sec-Fetch-Site: cross-site
14 Sec-Fetch-User: ?1
15 Te: trailers
16 Connection: close
17
18
```

OAuth 2.0: Authorization grant

Authorization Grant



Authorization Grant



Authorization Grant

- Credential representing the end user
- Obtained by from authorization server (recommended)
- Lots of different types

Grant types

- 4 grant types defined with extensibility
  - Authorization code
  - Client credentials
  - Implicit
  - Password

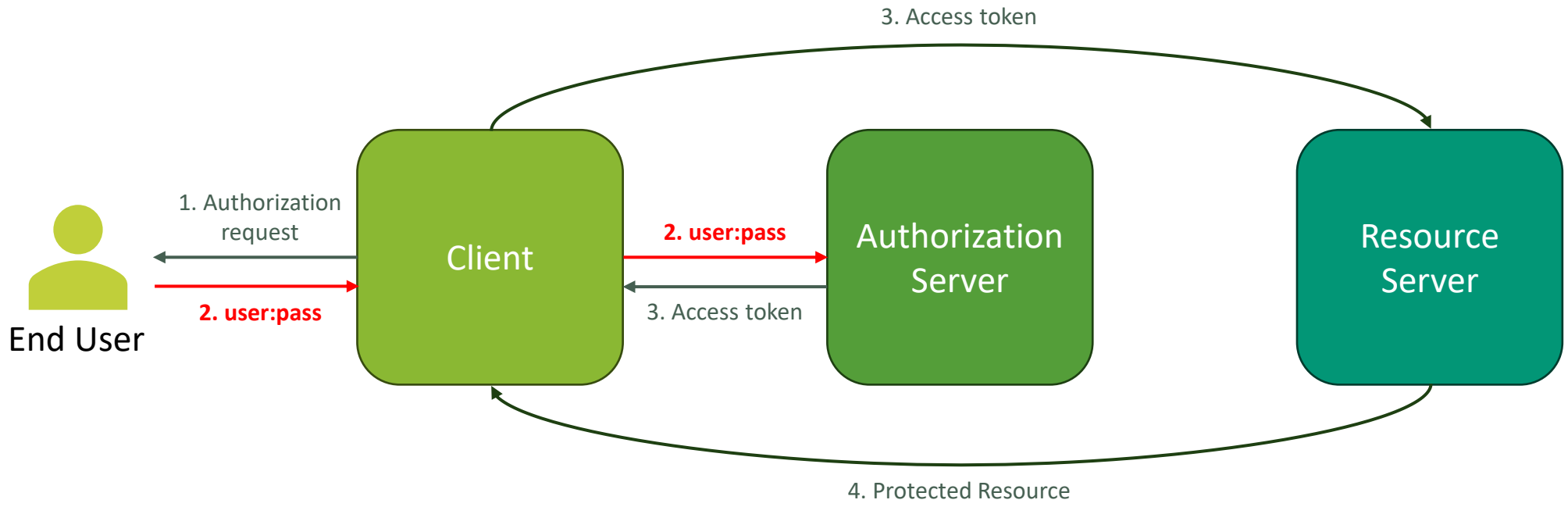
- Not all grant types are equal

https://accounts.google.com/.well-known/openid-configuration


```
{
 "issuer": "https://accounts.google.com",
 "authorization_endpoint": "https://accounts.google.com/o/oauth2/v2/auth",
 "device_authorization_endpoint": "https://oauth2.googleapis.com/device/code",
 "token_endpoint": "https://oauth2.googleapis.com/token",
 "userinfo_endpoint": "https://openidconnect.googleapis.com/v1/userinfo",
 "revocation_endpoint": "https://oauth2.googleapis.com/revoke",
 "jwks_uri": "https://www.googleapis.com/oauth2/v3/certs",
 "response_types_supported": [
  "code",
  "token",
  "id_token",
  "code token",
  "code id_token",
  "token id_token",
  "code token id_token",
  "none"
 ],
 "subject_types_supported": [
  "public"
 ],
 "id_token_signing_alg_values_supported": [
  "RS256"
 ],
 "scopes_supported": [
  "openid",
  "email",
  "profile"
 ],
 "token_endpoint_auth_methods_supported": [
  "client_secret_post",
  "client_secret_basic"
 ],
 "claims_supported": [
  "aud",
  "email",
  "email_verified",
  "exp",
  "family_name",
  "given_name",
  "iat",
  "iss",
  "locale",
  "name",
  "picture",
  "sub"
 ],
 "code_challenge_methods_supported": [
  "plain",
  "S256"
 ],
 "grant_types_supported": [
  "authorization_code",
  "refresh_token",
  "urn:ietf:params:oauth:grant-type:device_code",
  "urn:ietf:params:oauth:grant-type:jwt-bearer"
 ]
}
```

Password grant type

- Directly uses the end user's username and password
- Requires high level of trust between end user and client



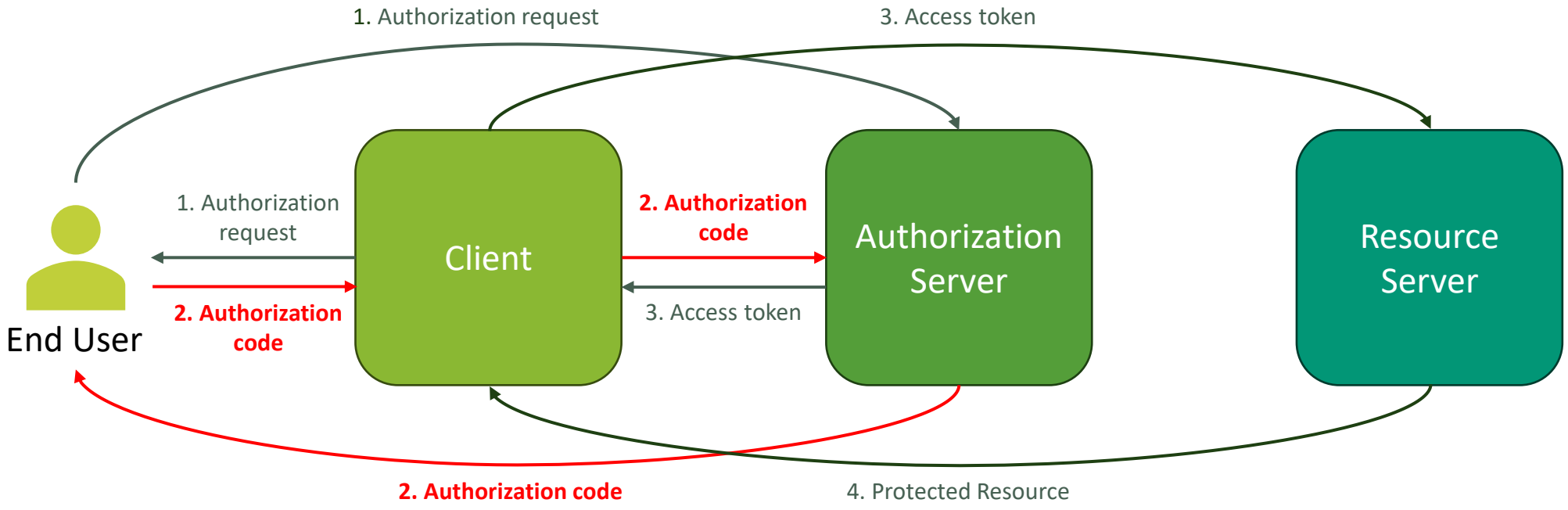
Password grant type

- Credentials only need to be supplied once
  - Access token is used for most transactions
- Credentials can be logged on the single transaction
- Avoid

Authorization code grant type

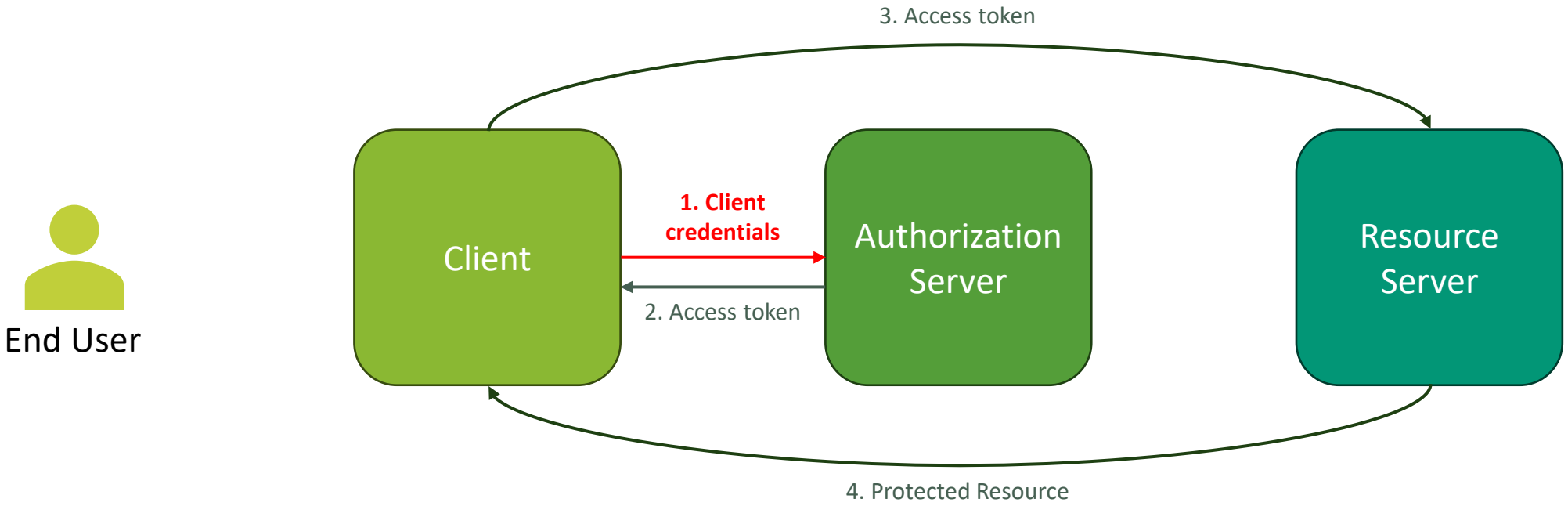
- End user obtains one-time code from authorization server

- Client uses code to exchange for access token



Client credentials grant type

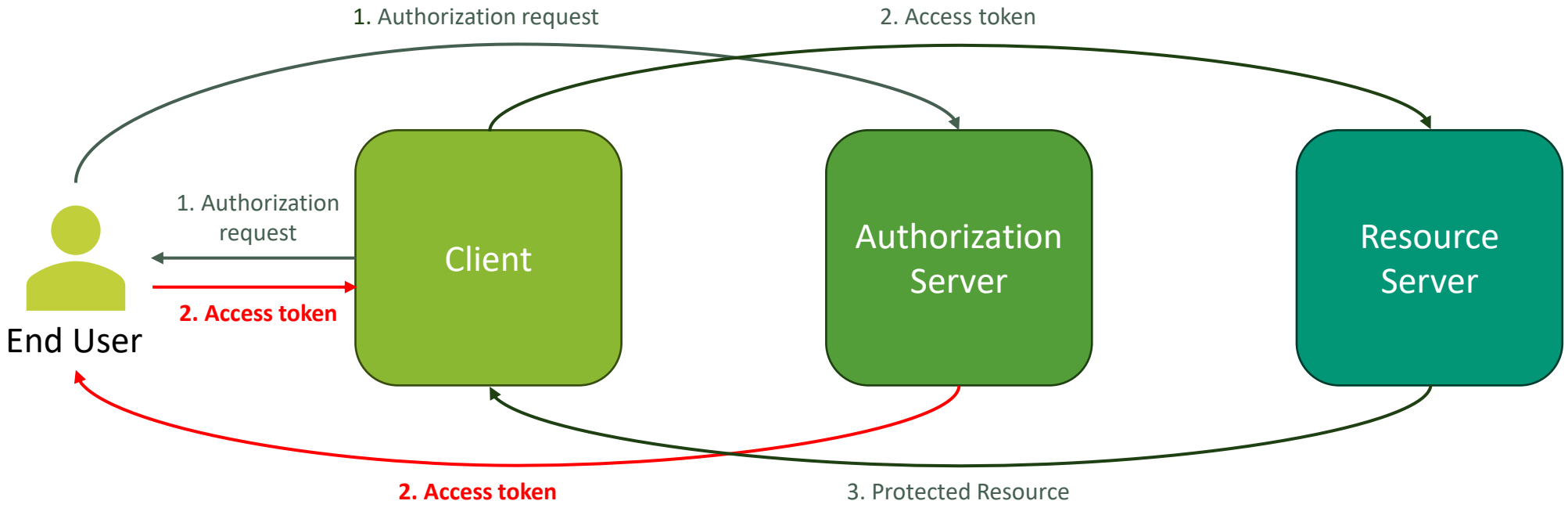
- Client is authorised to access resources
- No extra access needed from end user



Implicit grant type

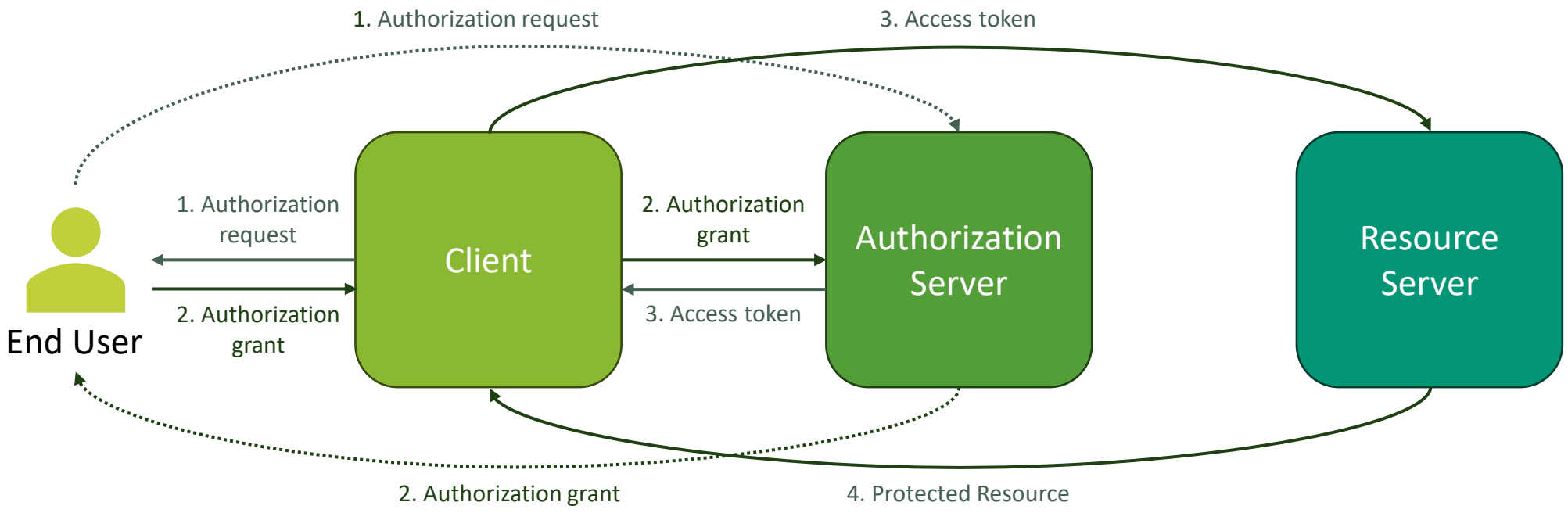
- Directly returns access token as authorization grant

- Insecure - will be deprecated in next OAuth version

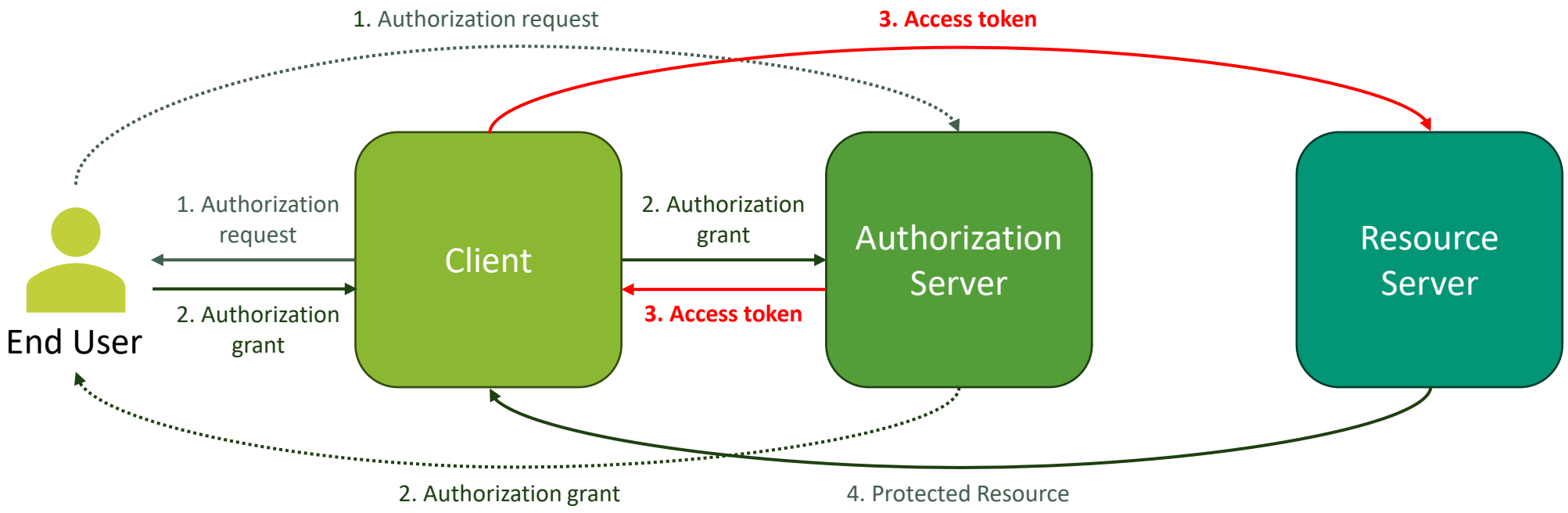


OAuth 2.0: Access token

Access token



Access token



Access token

- 2 token types defined by the OAuth protocol (with extensibility)
  - Access token
  - Refresh token

- Extra ID token type defined by OpenID

Request

Pretty | Raw | Hex

```
1  GET /o/oauth2/v2/auth?response_type=code&access_type=offline&client_id=
   849883241272-ed61nodi1grnoomiuknqkq2rbvd2udku.apps.googleusercontent.com
   &prompt=select_account&scope=profile%20email&redirect_uri=
   https%3A%2F%2Fzoom.us%2Fgoogle%2Foauth&state=
   ME1kUGEyeDRTWFNpVWQ0QVoxTVV2QSxnb29nbGVfc2lnbmlu&_x_zm_rtaid=
   Gt3XDdPrThyieNAZNAwBUg.1687222792540.23bf44959f2bb129d75fb697b1e8e829&
   _x_zm_rhtaid=326 HTTP/1.1
2  Host: accounts.google.com
3  Cookie: 1P_JAR=2023-06-20-00; AEC=
   AUEFqZdQ25HcULnd3VgPck7tD_uDbKIWuwzE9BjnWFcPUwog6u_n8q95oA; NID=
   511=tbaGrcWVqpS7GL7DNVINtcDYSnOET1_-NG_Ca4PxeQcrwjpPjeeFXDhOp-nPO4fqZ8Fa
   IORxngusJedMZ0GCGuBoHaRwxqx_O2Ee07LpN01PoxaVzYdWN_M8uOUaolhi7VaCB14MzRdf
   aDxFzUmHDHaGWskcvbeXZ7lhRZvwLU0; __Host-GAPS=
   1:VXmg8iKhhzCewpSuc5vduFvXgWqjuA:ZBJ64ZBnXvOQVWDS; OTZ=
   7081981_16_12_133440_16_395520
4  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101
   Firefox/112.0
5  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/w
   ebp,*/*;q=0.8
6  Accept-Language: en-US,en;q=0.5
7  Accept-Encoding: gzip, deflate
8  Referer: https://zoom.us/
9  Dnt: 1
10 Upgrade-Insecure-Requests: 1
11 Sec-Fetch-Dest: document
12 Sec-Fetch-Mode: navigate
13 Sec-Fetch-Site: cross-site
14 Sec-Fetch-User: ?1
15 Te: trailers
16 Connection: close
17
18
```

https://accounts.google.com/.well-known/openid-configuration


```
{
 "issuer": "https://accounts.google.com",
 "authorization_endpoint": "https://accounts.google.com/o/oauth2/v2/auth",
 "device_authorization_endpoint": "https://oauth2.googleapis.com/device/code",
 "token_endpoint": "https://oauth2.googleapis.com/token",
 "userinfo_endpoint": "https://openidconnect.googleapis.com/v1/userinfo",
 "revocation_endpoint": "https://oauth2.googleapis.com/revoke",
 "jwks_uri": "https://www.googleapis.com/oauth2/v3/certs",
 "response_types_supported": [
  "code",
  "token",
  "id_token",
  "code token",
  "code id_token",
  "token id_token",
  "code token id_token",
  "none"
 ],
 "subject_types_supported": [
  "public"
 ],
 "id_token_signing_alg_values_supported": [
  "RS256"
 ],
 "scopes_supported": [
  "openid",
  "email",
  "profile"
 ],
 "token_endpoint_auth_methods_supported": [
  "client_secret_post",
  "client_secret_basic"
 ],
 "claims_supported": [
  "aud",
  "email",
  "email_verified",
  "exp",
  "family_name",
  "given_name",
  "iat",
  "iss",
  "locale",
  "name",
  "picture",
  "sub"
 ],
 "code_challenge_methods_supported": [
  "plain",
  "S256"
 ],
 "grant_types_supported": [
  "authorization_code",
  "refresh_token",
  "urn:ietf:params:oauth:grant-type:device_code",
  "urn:ietf:params:oauth:grant-type:jwt-bearer"
 ]
}
```

Access token

- Provided by the authorization server and verified by resource server
- Commonly implemented using JSON Web Tokens (JWT)
- Represent **scope** and **duration** of resource access
  - **Scope**: What permissions are granted?
  - **Duration**: How long do these permissions last?

Access token

- Common JWT claims:

Abbreviation	Purpose	Requirement
iss	identifies JWT **Issuer** (Authorization server)	OPTIONAL
sub	identifies **subject** of JWT (End User)	OPTIONAL
aud	**audience** of JWT (Resource server)	OPTIONAL
exp	**expiration** time of JWT (Unix timestamp)	OPTIONAL
nbf	**not before** time of JWT (Unix timestamp)	OPTIONAL
iat	**issued at** time of JWT (Unix timestamp)	OPTIONAL
scope	list of actions the application is allowed to perform	OPTIONAL

Refresh token

- Can be implemented using JWT
- Used to obtain new or additional access tokens
  - Current access token is or expired
  - Obtain access token with narrower scope

https://accounts.google.com/.well-known/openid-configuration

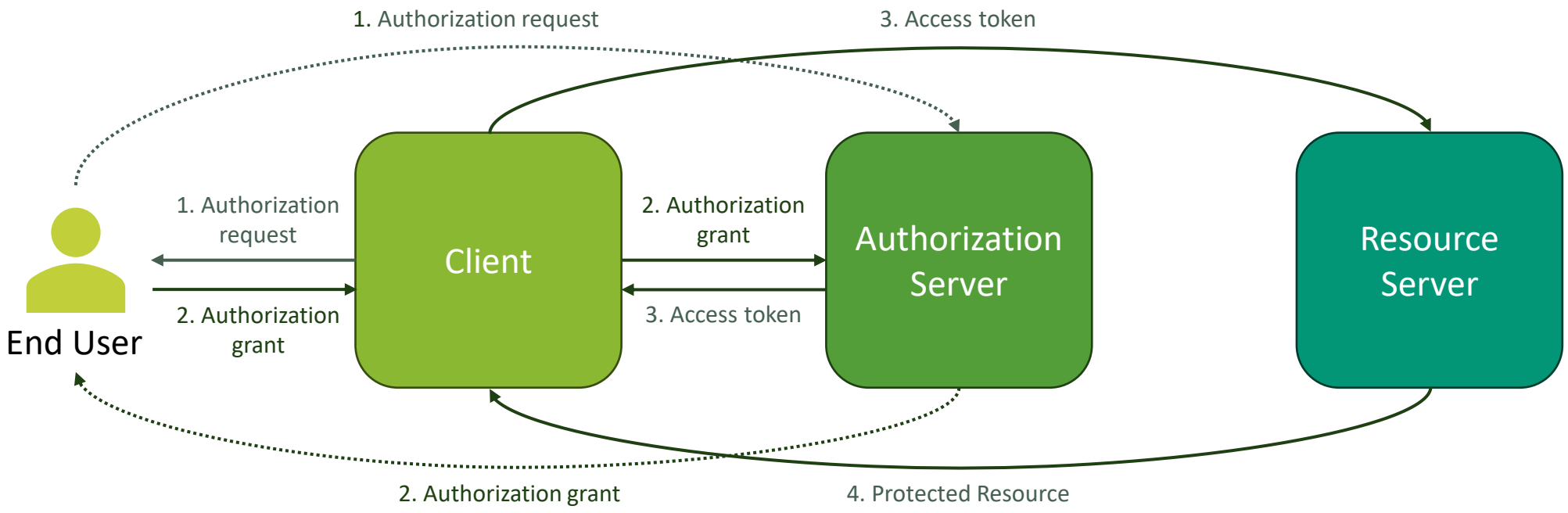

```
{
 "issuer": "https://accounts.google.com",
 "authorization_endpoint": "https://accounts.google.com/o/oauth2/v2/auth",
 "device_authorization_endpoint": "https://oauth2.googleapis.com/device/code",
 "token_endpoint": "https://oauth2.googleapis.com/token",
 "userinfo_endpoint": "https://openidconnect.googleapis.com/v1/userinfo",
 "revocation_endpoint": "https://oauth2.googleapis.com/revoke",
 "jwks_uri": "https://www.googleapis.com/oauth2/v3/certs",
 "response_types_supported": [
  "code",
  "token",
  "id_token",
  "code token",
  "code id_token",
  "token id_token",
  "code token id_token",
  "none"
 ],
 "subject_types_supported": [
  "public"
 ],
 "id_token_signing_alg_values_supported": [
  "RS256"
 ],
 "scopes_supported": [
  "openid",
  "email",
  "profile"
 ],
 "token_endpoint_auth_methods_supported": [
  "client_secret_post",
  "client_secret_basic"
 ],
 "claims_supported": [
  "aud",
  "email",
  "email_verified",
  "exp",
  "family_name",
  "given_name",
  "iat",
  "iss",
  "locale",
  "name",
  "picture",
  "sub"
 ],
 "code_challenge_methods_supported": [
  "plain",
  "S256"
 ],
 "grant_types_supported": [
  "authorization_code",
  "refresh_token",
  "urn:ietf:params:oauth:grant-type:device_code",
  "urn:ietf:params:oauth:grant-type:jwt-bearer"
 ]
}
```

Refresh token

- Optional – clients do not need to use both tokens
- Used to provide long term access
  - Mobile apps can store refresh tokens for persistent access, so that the user isn’t logged out after closing the app
- Should **never** be sent to any resource server

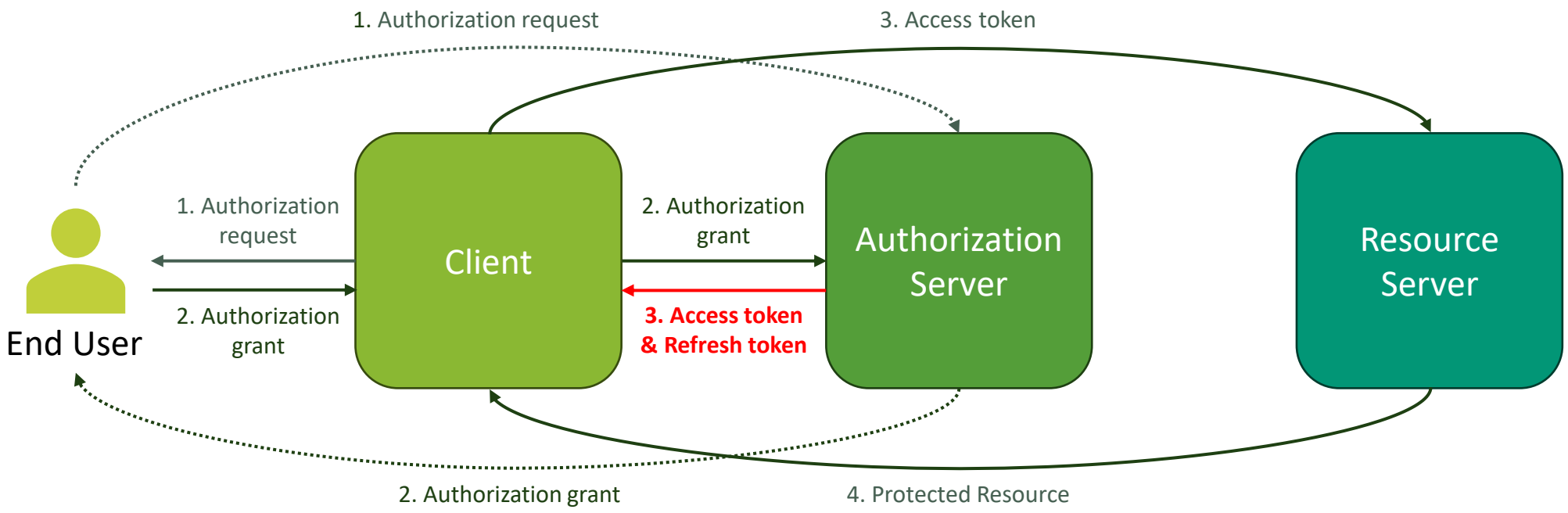
Refresh token

- Returned to client alongside access token



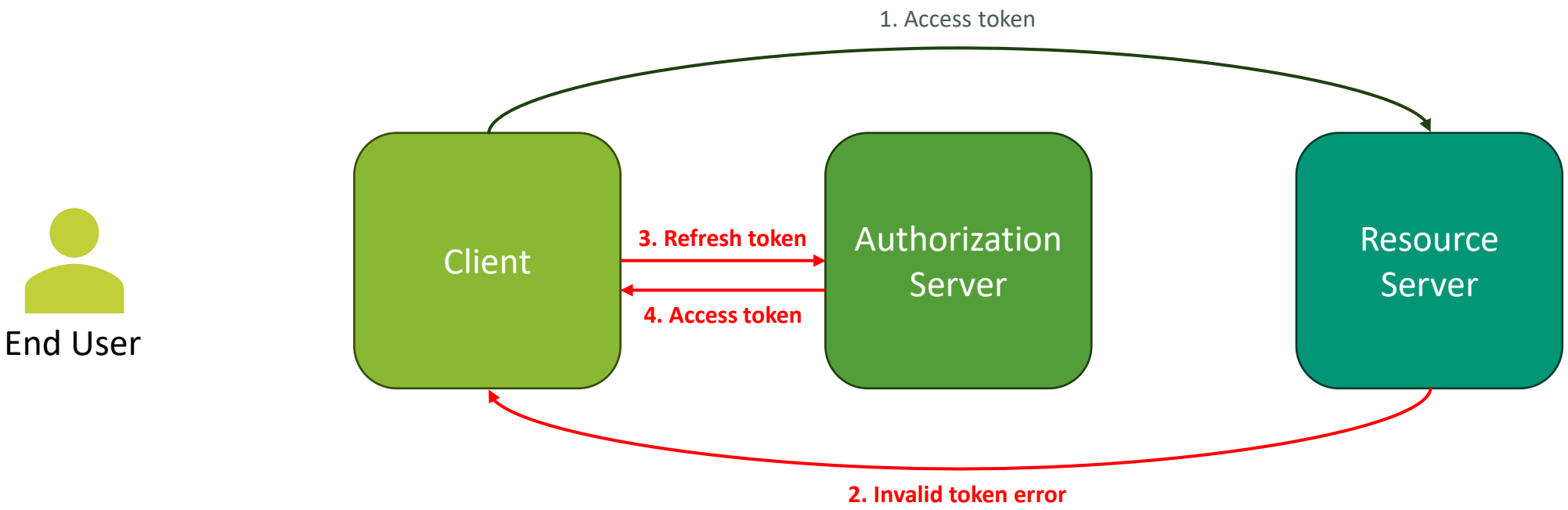
Refresh token

- Returned to client alongside access token



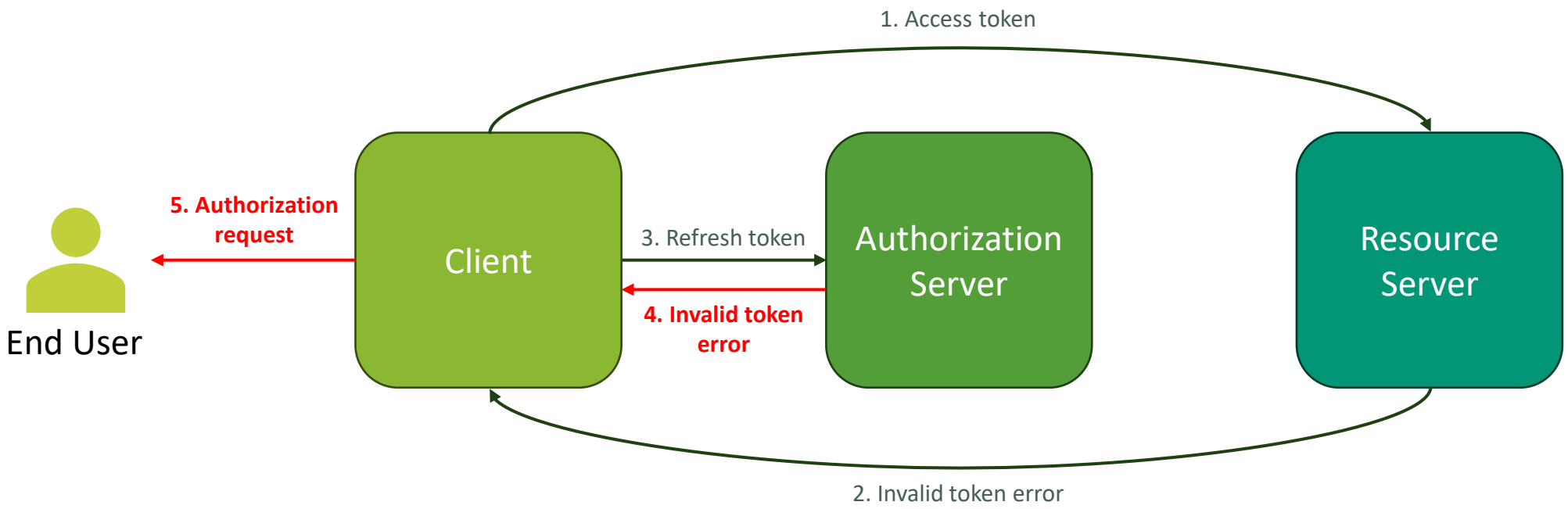
Refresh token

- Exchange for access tokens occurs between client and authorization server



Refresh token

- When a refresh token expires, the client must ask the end user to re-authenticate



Access & Refresh tokens

Access Token

- Short expiry
- Audience is a single resource server
- Used to obtain resources

Refresh Token

- Long expiry
- “Audience” is the authorization server
- Used to obtain access tokens

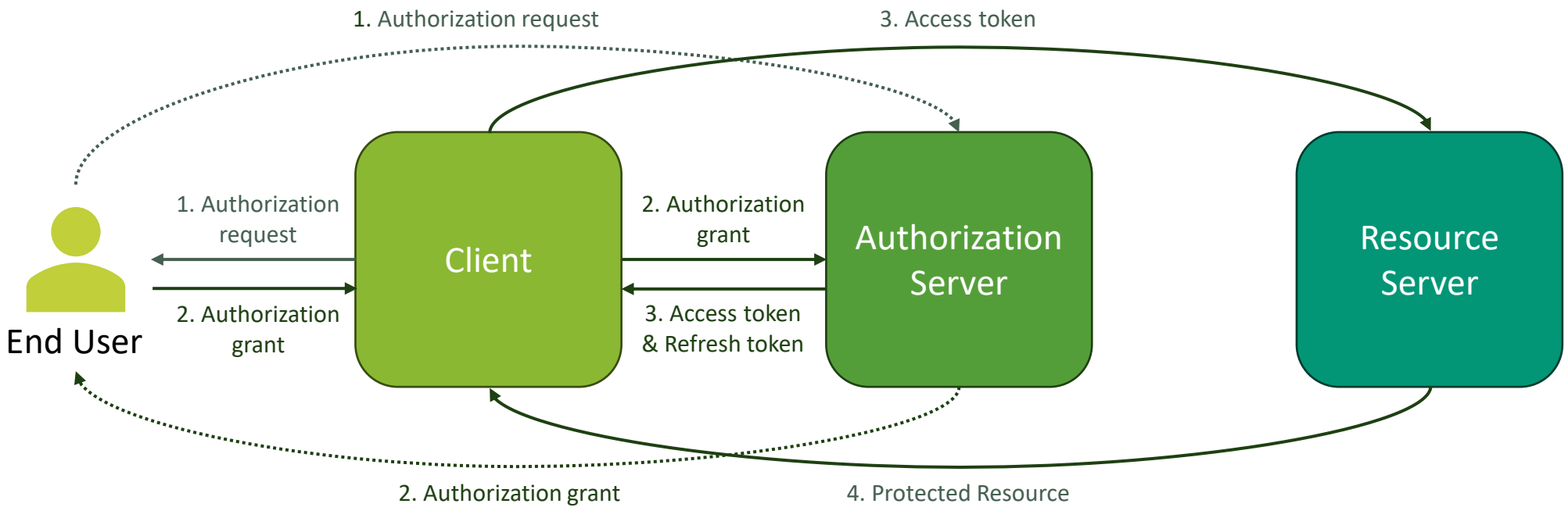
ID token

- Not defined in OAuth RFC

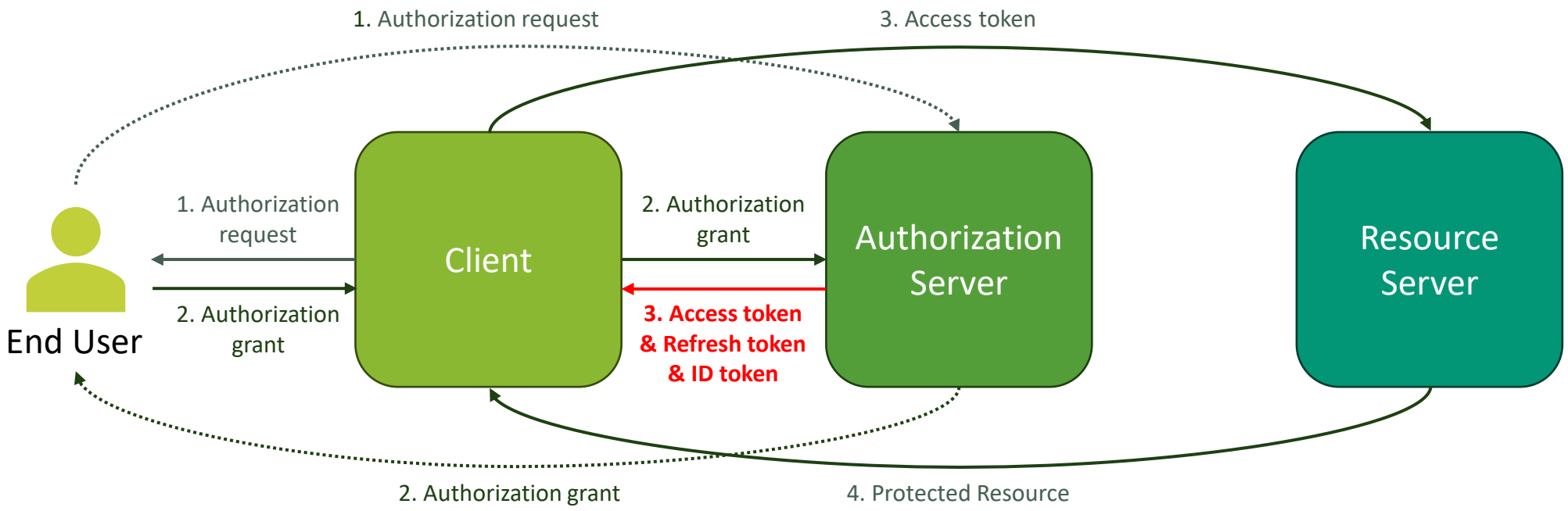
- Contains signed claims about their identity
  - e.g., a Google issued OpenID token can contain the “email address” claim
  - Users who are identified by their email can use this as **authentication**

- Usually have a very short lifetime
  - No refresh token equivalent

ID token


1. Authorization request
3. Access token
1. Authorization request
2. Authorization grant
Client
2. Authorization grant
3. Access token & Refresh token
Authorization Server
Resource Server
End User
2. Authorization grant
4. Protected Resource

ID token


1. Authorization request
3. Access token
1. Authorization request
2. Authorization grant
End User
Client
2. Authorization grant
3. Access token & Refresh token & ID token
Authorization Server
Resource Server
2. Authorization grant
4. Protected Resource

Example: Google ID token


```
{
  "iss": "https://accounts.google.com",
  "azp": "1234987819200.apps.googleusercontent.com",
  "aud": "1234987819200.apps.googleusercontent.com",
  "sub": "107691503500061507151130823 67",
  "at_hash": "HK6E_P6Dh8Y93mRNtsDB1Q",
  "hd": "example.com",
  "email": "jsmith@example.com",
  "email_verified": "true",
  "iat": 1353601026,
  "exp": 1353604926,
  "nonce": "0394852-3190485-2490358"
}
```

ID token scopes

- OIDC defines a set of scopes
  - profile
  - email
  - phone
  - address
  - **openid**

https://accounts.google.com/.well-known/openid-configuration


```
{
 "issuer": "https://accounts.google.com",
 "authorization_endpoint": "https://accounts.google.com/o/oauth2/v2/auth",
 "device_authorization_endpoint": "https://oauth2.googleapis.com/device/code",
 "token_endpoint": "https://oauth2.googleapis.com/token",
 "userinfo_endpoint": "https://openidconnect.googleapis.com/v1/userinfo",
 "revocation_endpoint": "https://oauth2.googleapis.com/revoke",
 "jwks_uri": "https://www.googleapis.com/oauth2/v3/certs",
 "response_types_supported": [
  "code",
  "token",
  "id_token",
  "code token",
  "code id_token",
  "token id_token",
  "code token id_token",
  "none"
 ],
 "subject_types_supported": [
  "public"
 ],
 "id_token_signing_alg_values_supported": [
  "RS256"
 ],
 "scopes_supported": [
  "openid",
  "email",
  "profile"
 ],
 "token_endpoint_auth_methods_supported": [
  "client_secret_post",
  "client_secret_basic"
 ],
 "claims_supported": [
  "aud",
  "email",
  "email_verified",
  "exp",
  "family_name",
  "given_name",
  "iat",
  "iss",
  "locale",
  "name",
  "picture",
  "sub"
 ],
 "code_challenge_methods_supported": [
  "plain",
  "S256"
 ],
 "grant_types_supported": [
  "authorization_code",
  "refresh_token",
  "urn:ietf:params:oauth:grant-type:device_code",
  "urn:ietf:params:oauth:grant-type:jwt-bearer"
 ]
}
```

ID token scopes

- OIDC defines a set of scopes
  - profile
  - email
  - phone
  - address
  - **openid**

- OpenID scope can be used to access UserInfo endpoint

https://accounts.google.com/.well-known/openid-configuration


```
{
 "issuer": "https://accounts.google.com",
 "authorization_endpoint": "https://accounts.google.com/o/oauth2/v2/auth",
 "device_authorization_endpoint": "https://oauth2.googleapis.com/device/code",
 "token_endpoint": "https://oauth2.googleapis.com/token",
 "userinfo_endpoint": "https://openidconnect.googleapis.com/v1/userinfo",
 "revocation_endpoint": "https://oauth2.googleapis.com/revoke",
 "jwks_uri": "https://www.googleapis.com/oauth2/v3/certs",
 "response_types_supported": [
  "code",
  "token",
  "id_token",
  "code token",
  "code id_token",
  "token id_token",
  "code token id_token",
  "none"
 ],
 "subject_types_supported": [
  "public"
 ],
 "id_token_signing_alg_values_supported": [
  "RS256"
 ],
 "scopes_supported": [
  "openid",
  "email",
  "profile"
 ],
 "token_endpoint_auth_methods_supported": [
  "client_secret_post",
  "client_secret_basic"
 ],
 "claims_supported": [
  "aud",
  "email",
  "email_verified",
  "exp",
  "family_name",
  "given_name",
  "iat",
  "iss",
  "locale",
  "name",
  "picture",
  "sub"
 ],
 "code_challenge_methods_supported": [
  "plain",
  "S256"
 ],
 "grant_types_supported": [
  "authorization_code",
  "refresh_token",
  "urn:ietf:params:oauth:grant-type:device_code",
  "urn:ietf:params:oauth:grant-type:jwt-bearer"
 ]
}
```

OAuth 2.0: Client types

OAuth client types

Confidential client

- Implemented on a secure server
- Can maintain confidentiality of their own credentials
  - client_id
  - **client_secret**

Public client

- Incapable of secure client authentication
- No confidential method of storing credentials

Example client types

Example application	Client type
Client-side web application • Zoom web	Public
Native application • Zoom application (desktop, mobile)	Public
Server-side web application	Confidential

Answer to previous questions

- Google recognises the client from the “client_id” parameter

- In most cases, this isn’t necessary
  - The same “passport” can be shown to different apps for identity
  - Becomes necessary when some apps require more identity documents than others (confidential clients)
  - Helps with implementing security controls

Request

Pretty Raw Hex

```
1  GET /o/oauth2/v2/auth?response_type=code&access_type=offline&client_id=
   849883241272-ed61nodi1grnoomiuknqkq2rbvd2udku.apps.googleusercontent.com
   &prompt=select_account&scope=profile%20email&redirect_uri=
   https%3A%2F%2Fzoom.us%2Fgoogle%2Foauth&state=
   ME1kUGEyeDRTWFNpVWQ0QVoxTVV2QSxnb29nbGVfc2lnbm1u&_x_zm_rtaid=
   Gt3XDdPrThyieNAZNAwBUg.1687222792540.23bf44959f2bb129d75fb697b1e8e829&
   _x_zm_rhtaid=326 HTTP/1.1
2  Host: accounts.google.com
3  Cookie: 1P_JAR=2023-06-20-00; AEC=
   AUEFqZdQ25HcULnd3VgPck7tD_uDbKIWuwzE9BjnWFcPUwog6u_n8q95oA; NID=
   511=tbaGrcWVqpS7GL7DNVINtcDYSnOET1_-NG_Ca4PxeQcrwjpPjeeFXDhOp-nPO4fqZ8Fa
   IORxngusJedMZ0GCGuBoHaRwxqx_O2Ee07LpN01PoxaVzYdWN_M8uOUaolhi7VaCB14MzRdf
   aDxFzUmHDHaGWskcvbeXZ7lhRZvwLU0; __Host-GAPS=
   1:VXmg8iKhhzCewpSuc5vduFvXgWqjuA:ZBJ64ZBnXvOQVWDS; OTZ=
   7081981_16_12_133440_16_395520
4  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101
   Firefox/112.0
5  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/w
   ebp,*/*;q=0.8
6  Accept-Language: en-US,en;q=0.5
7  Accept-Encoding: gzip, deflate
8  Referer: https://zoom.us/
9  Dnt: 1
10 Upgrade-Insecure-Requests: 1
11 Sec-Fetch-Dest: document
12 Sec-Fetch-Mode: navigate
13 Sec-Fetch-Site: cross-site
14 Sec-Fetch-User: ?1
15 Te: trailers
16 Connection: close
17
18
```

OAuth 2.0 Recap

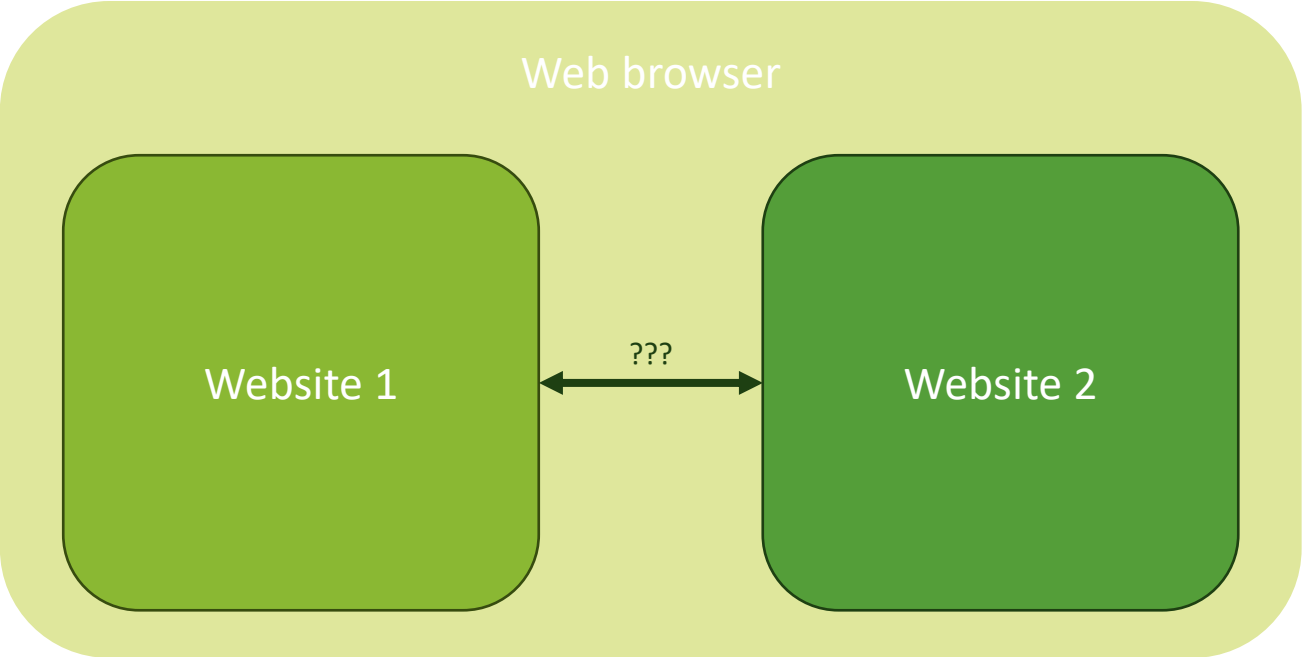
- Authorization request takes the user to the authorization server
- Different types of authorization grants
  - Authorization code
  - Client credentials
  - Implicit
  - Password
- Three types of tokens
  - Access tokens
  - Refresh tokens
  - OpenID tokens
- Two types of clients
  - Public
  - Confidential

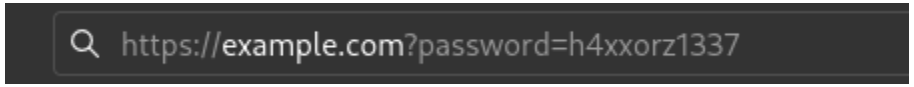
A History of Security Issues

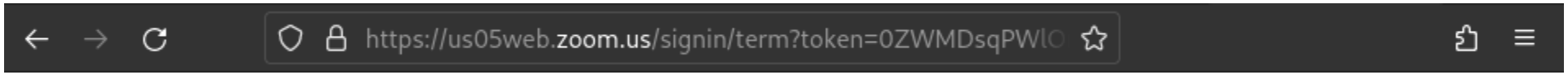
Food for thought

- The web was not designed for inter-site communication

- Redirect URLs are just links
  - Visible on screen (and to many other parties)
  - Phishing







A History of Security Issues: Cross-Site Request Forgery

Cross-Site Request Forgery (CSRF)

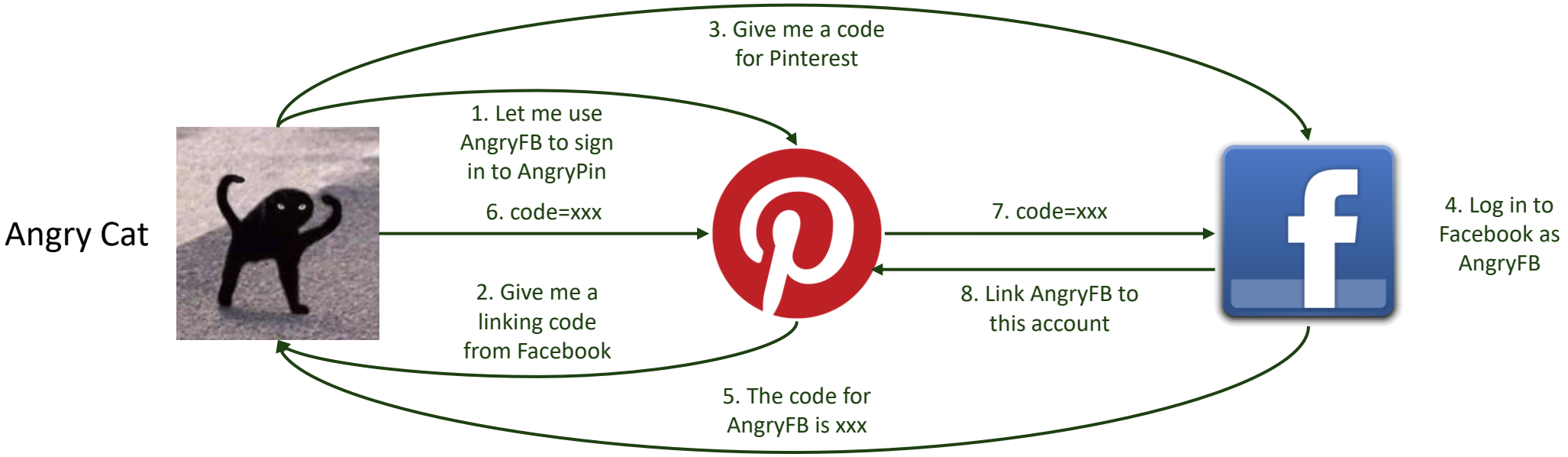
- Described in blog post by Egor Homakov (@homakov) *(July 3, 2012)*
- Enable SSO on third-party site, e.g., log in to Pinterest using Facebook
- Malicious user can create a link to enable SSO on the target's account
  - "1-click" phishing attack

Reference: https://homakov.blogspot.com/2012/07/saferweb-most-common-oauth2.html

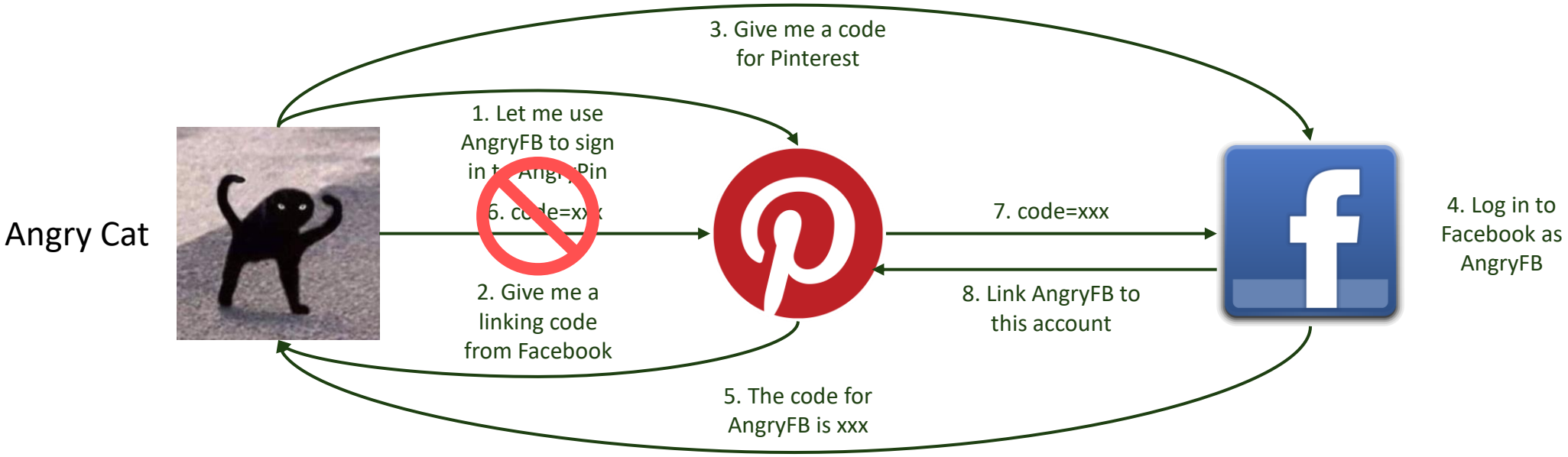
Cross-Site Request Forgery (CSRF)

- Enable Facebook SSO login in Pinterest
  - Pinterest confirms ownership of Facebook account using OAuth code flow
- Attacker wants to link their Facebook account, AttackerFB, to Target's Pinterest account, TargetPin
- Requires that Target is already logged into Client (Pinterest)

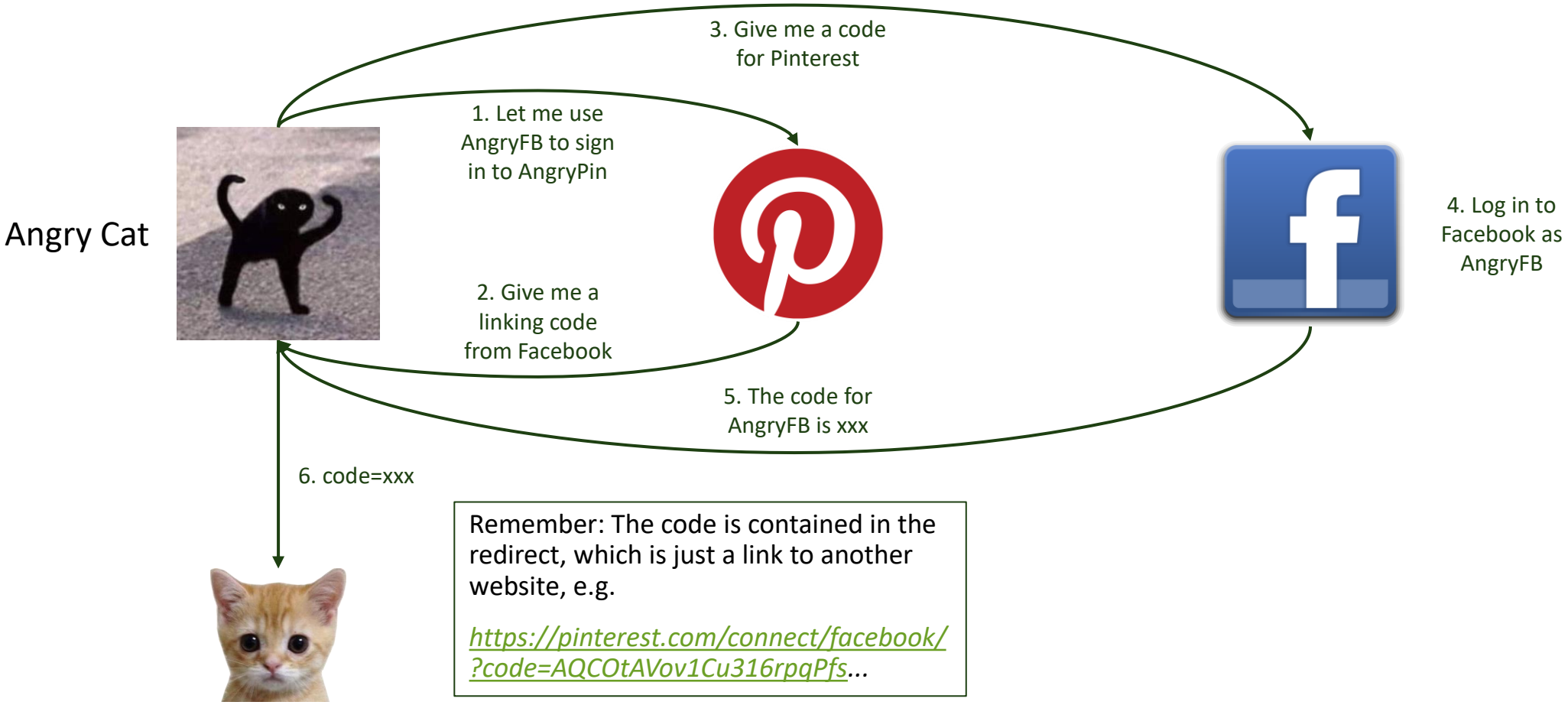
Cross-Site Request Forgery (CSRF)



Cross-Site Request Forgery (CSRF)



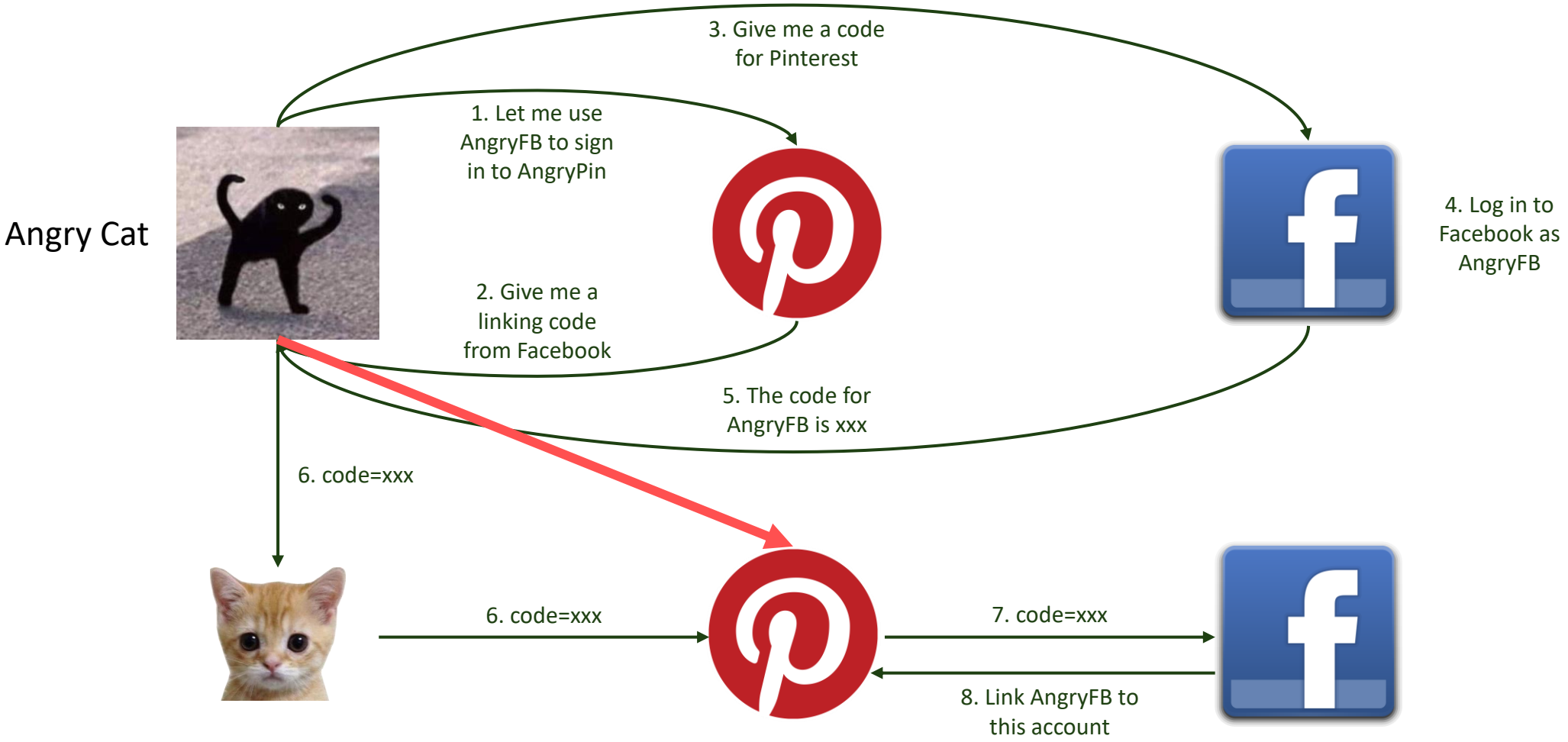
Cross-Site Request Forgery (CSRF)



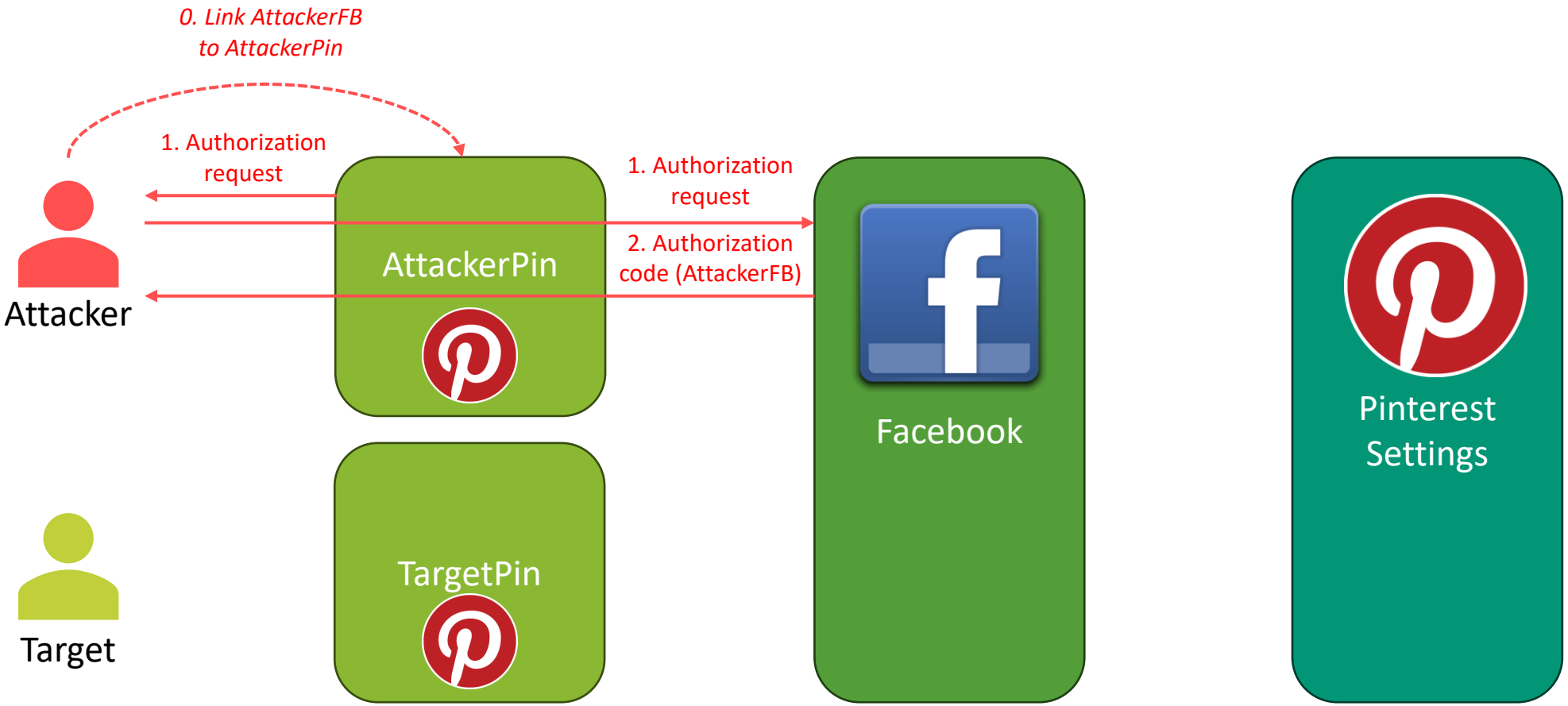
Remember: The code is contained in the redirect, which is just a link to another website, e.g.

https://pinterest.com/connect/facebook/?code=AQCOtAVov1Cu316rpqPfs...

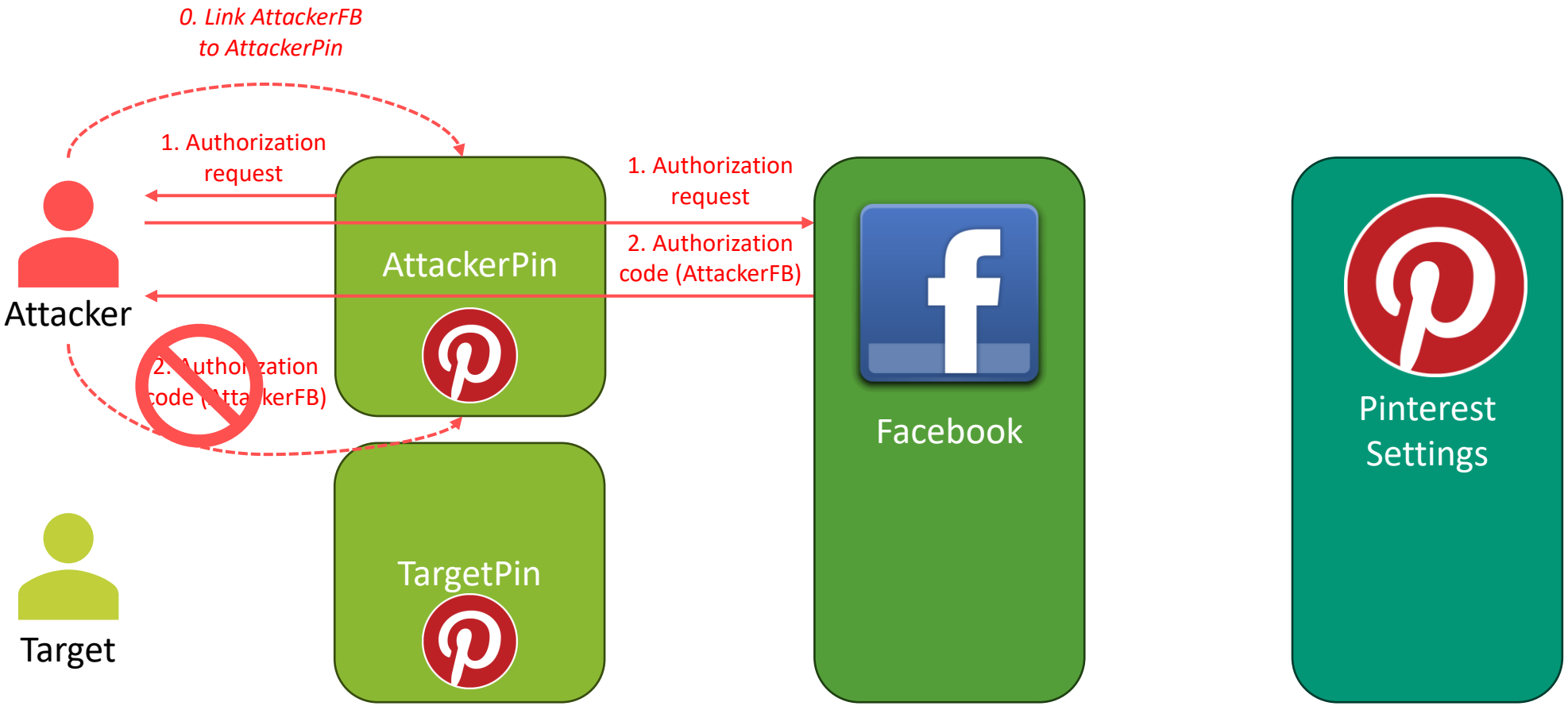
Cross-Site Request Forgery (CSRF)



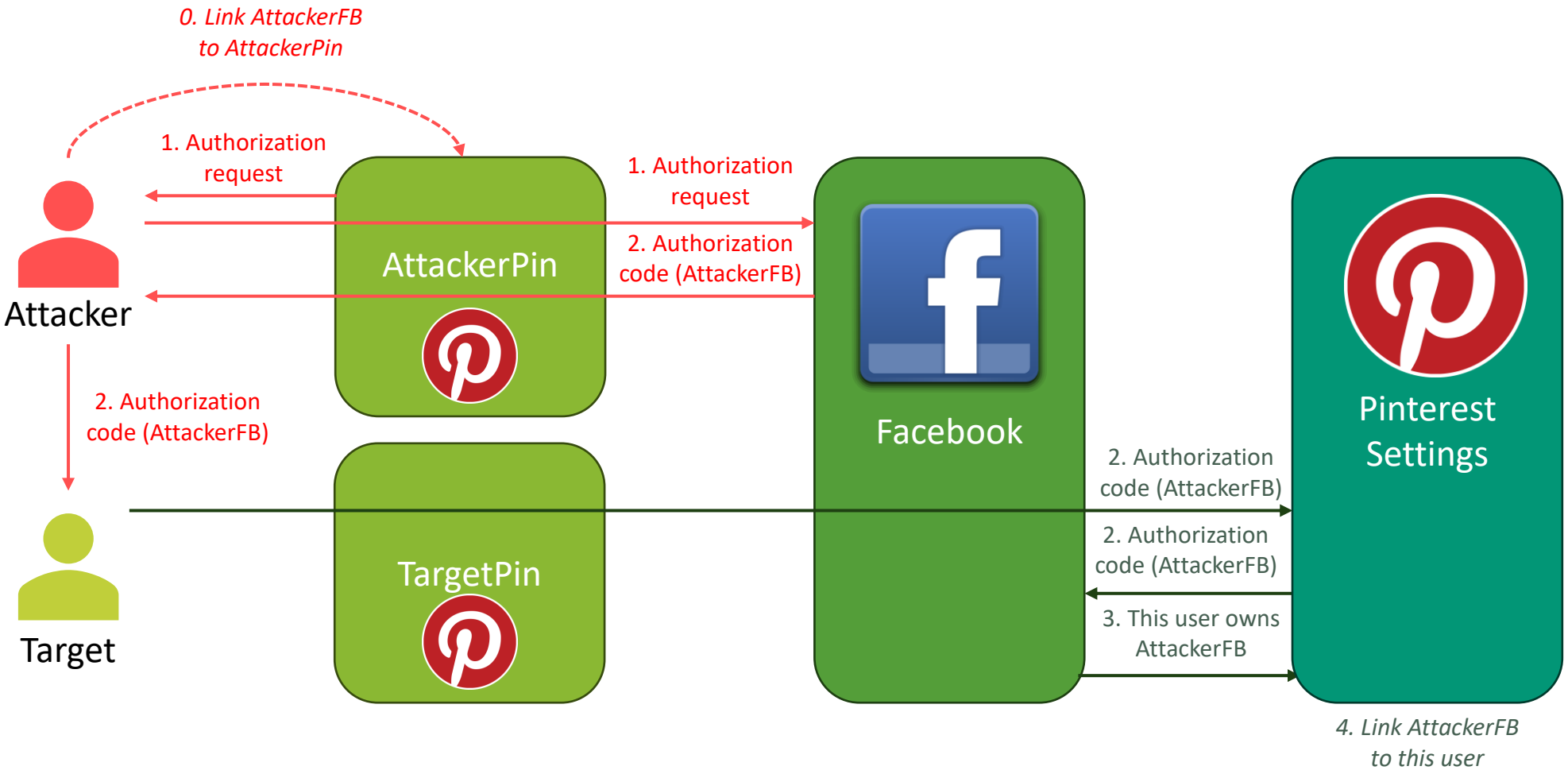
Cross-Site Request Forgery (CSRF)



Cross-Site Request Forgery (CSRF)



Cross-Site Request Forgery (CSRF)



Cross-Site Request Forgery (CSRF)

- Attacker can now log in to TargetPin using AttackerFB SSO

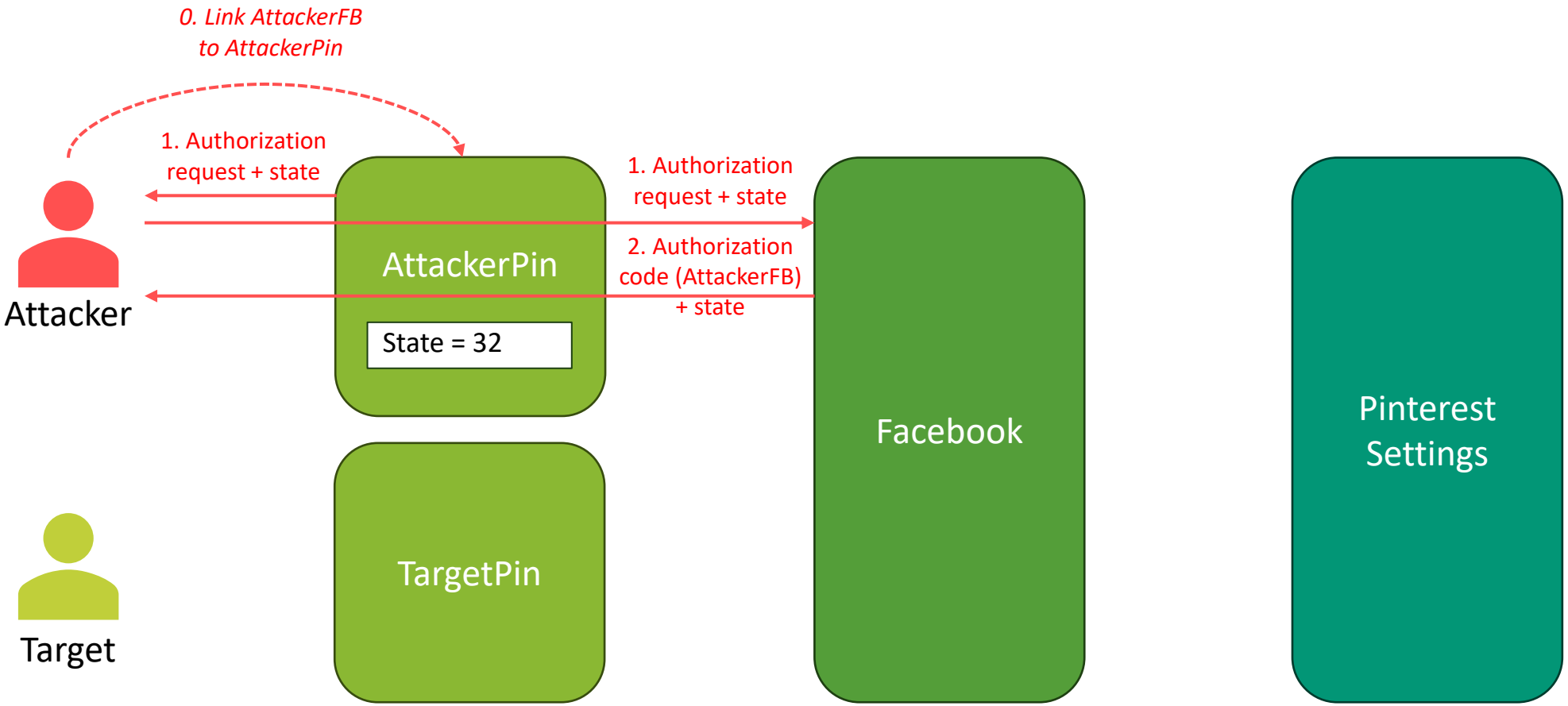
- Target does not need to know the link has been clicked, as long as browser visits the link
  - e.g., link can be embedded into the source of an <img>

- Is there a way to ensure that the user who requested the SSO update is the same user who completes the flow?
  - Discuss

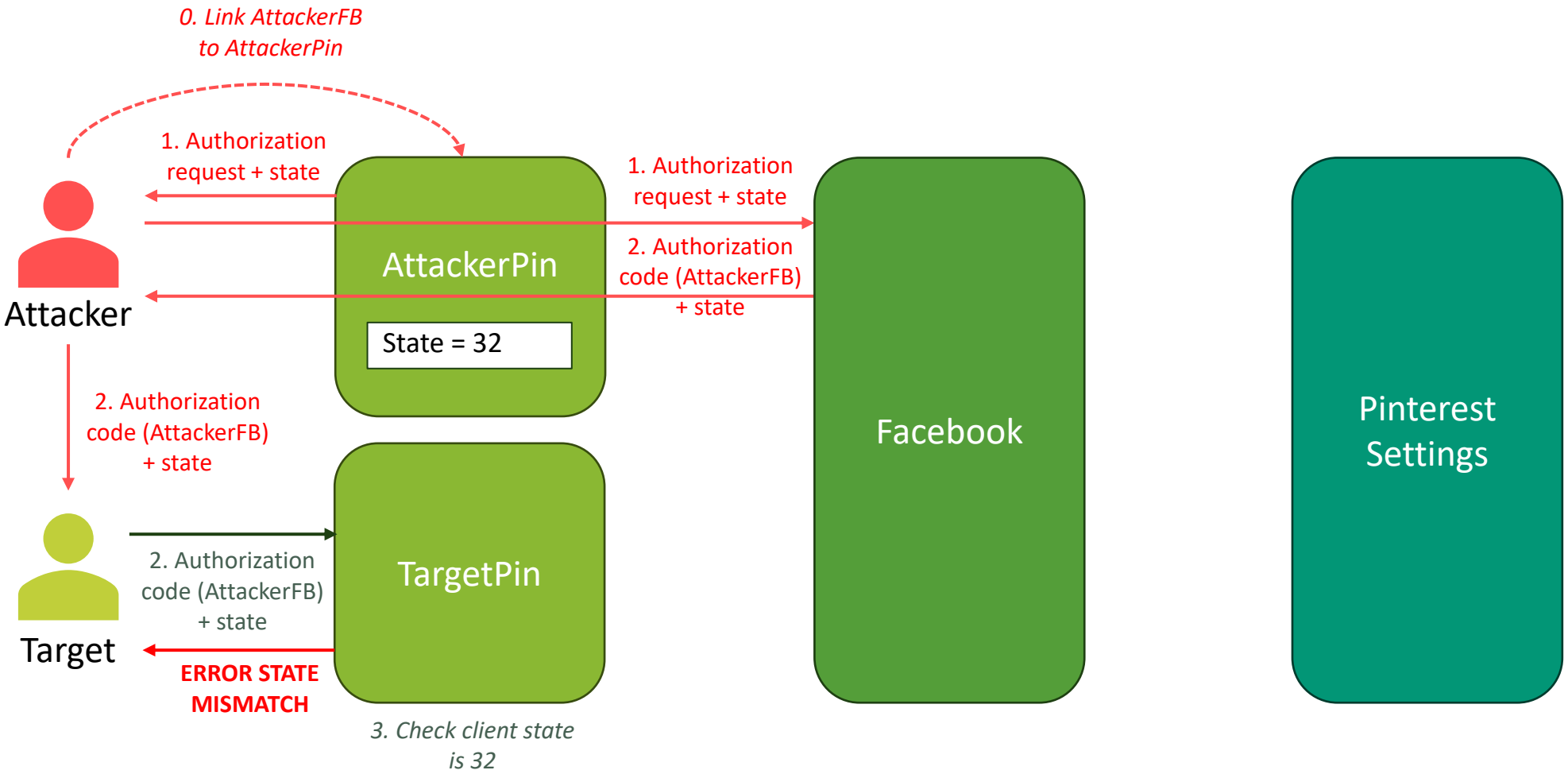
State

- Ensures that the user who initiated the request is the same user that completes the request
- Usually a random value generated by and stored within the client
- At the end of the Auth flow, the client checks that the state value contained in the redirect is the same as the local state value

State


0. Link AttackerFB to AttackerPin
1. Authorization request + state
1. Authorization request + state
2. Authorization code (AttackerFB) + state
Attacker
AttackerPin
State = 32
Facebook
Pinterest Settings
TargetPin
Target

State


0. Link AttackerFB to AttackerPin
1. Authorization request + state
1. Authorization request + state
2. Authorization code (AttackerFB) + state
Attacker
AttackerPin
State = 32
Facebook
Pinterest Settings
2. Authorization code (AttackerFB) + state
2. Authorization code (AttackerFB) + state
Target
TargetPin
ERROR STATE MISMATCH
3. Check client state is 32

A History of Security Issues: Unverified redirect_uri

Unverified redirect_uri

- Described in blog post by Egor Homakov *(August 1, 2012)*

- “redirect_uri” is a parameter sent in the authorization request

Remember: The authorization provider only knows what you tell it in the initial redirect, e.g.,

https://accounts.google.com/o/oauth2/v2/auth?redirect_uri=https://zoom.us/google/oauth

Request

Pretty Raw Hex

```
1 GET /o/oauth2/v2/auth?response_type=code&access_type=offline&client_id=
  849883241272-ed6lnodi1grnoomiuknqkq2rbvd2udku.apps.googleusercontent.com
  &prompt=select_account&scope=profile%20email&redirect_uri=
  https%3A%2F%2Fzoom.us%2Fgoogle%2Foauth&state=
  ME1kUGEyeDRTWFNpVWQ0QVoxTVV2QSxnb29nbGVfc2lnbmlu&_x_zm_rtaid=
  Gt3XDdPrThyieNAZNAwBUg.1687222792540.23bf44959f2bb129d75fb697b1e8e829&
  _x_zm_rhtaid=326 HTTP/1.1
2 Host: accounts.google.com
```

Reference: https://homakov.blogspot.com/2012/08/saferweb-oauth2a-or-lets-just-fix-it.html

Unverified redirect_uri

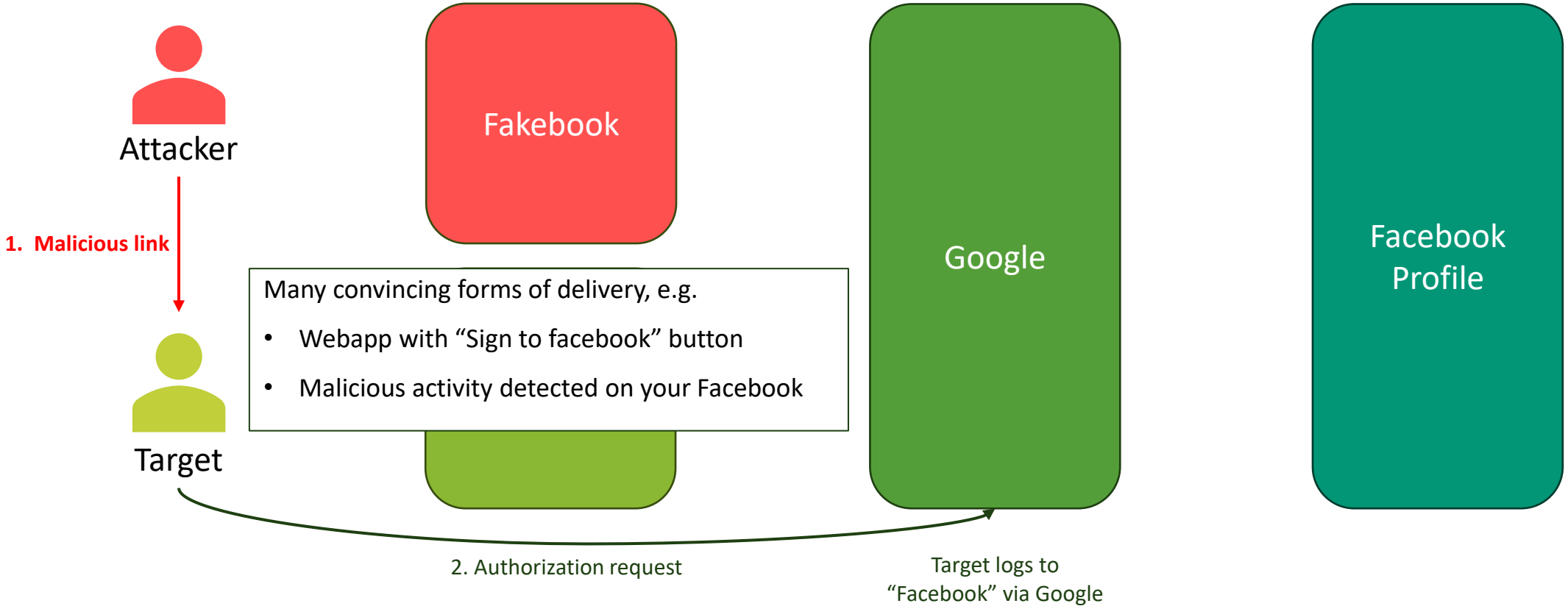
Legitimate login link

- https://accounts.google.com/o/oauth2/v2/auth?client_id=<facebook_client_id>**redirect_uri=https://facebook.com/oauth/google**

Malicious login link

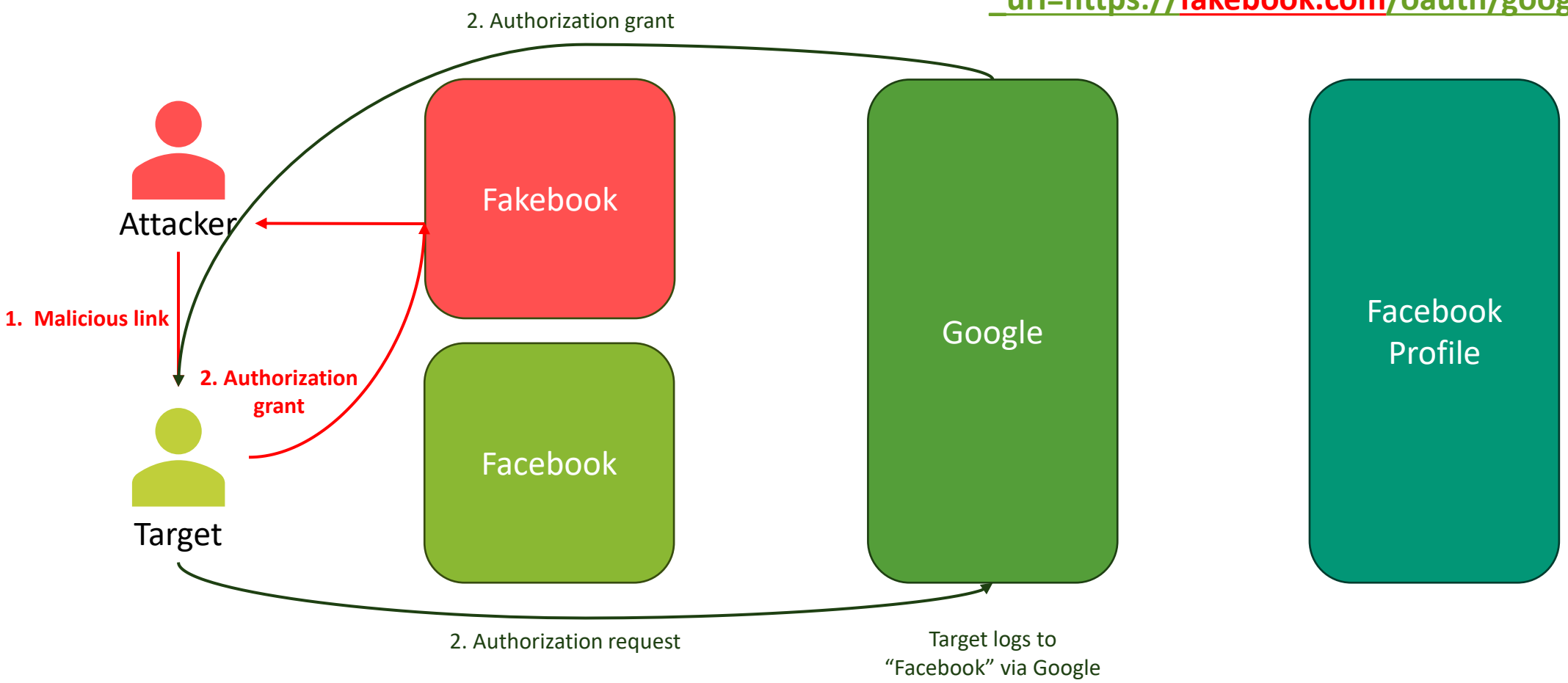
- https://accounts.google.com/o/oauth2/v2/auth?client_id=<facebook_client_id>**redirect_uri=https://fakebook.com/oauth/google**

Unverified redirect_uri



Unverified redirect_uri

https://accounts.google.com/o/oauth2/v2/auth?client_id=<facebook_client_id>**redirect_uri=https://fakebook.com/oauth/google**



Unverified redirect_uri

- Malicious application (Fakebook) receives authorization grant, usually a code that can be exchanged for an access token

- OAuth **provider** (Google) needs validate the “redirect_uri” parameter
  - Bugs can occur in validation logic

- Can’t be prevented by state parameter
  - Authorization grant sent directly to malicious application

Unverified redirect_uri

March 29 2013

How I Hacked Facebook OAuth To Get Full Permission On Any Facebook Account (Without App "Allow" Interaction)

Hello there, I've decided to share one of my favorite flaws in facebook.com. This flaw essentially let me take over any Facebook account. [...]

March 29 2013

How I Hacked Any Facebook Account...Again!

This post is the second one I've done in regard to Facebook OAuth Vulnerabilities. But, just so everything's clear from the start, no [...]

Share Share

April 3 2013

The Unfix Bug in Facebook OAuth

For starters, I want to reiterate that I have finished my tenure with Bug Bounty Programs, but, as promised, I will continue to publish [...]

Share Share

A History of Security Issues: Open Redirect

Open Redirect

- Common vulnerability that is very easily chained with CSRF attacks
- Allows anyone to create a link on a website that automatically redirects to a second website
  - The second website can be anywhere on the internet
- Example: returnUrl in ASP.NET used to allow free open redirects
  - http://example.com?returnUrl=http://evildomain.com
  - Browser redirects to http://evildomain.com

More issues

- Confused Deputy attacks
  - Authorization server doesn’t differentiate between clients

- Misuse of Implicit flow
  - Multiple issues with using implicit flow for authentication, detailed in https://homakov.blogspot.com/2012/08/oauth2-one-accesstoken-to-rule-them-all.html
  - Implicit flow will be deprecated but still supported by large vendors such as Google

A History of Security Issues Recap

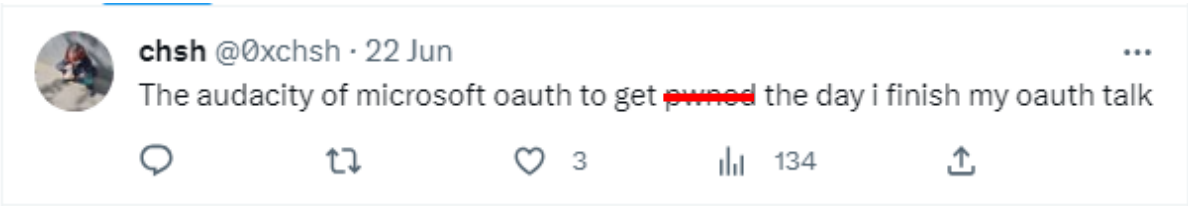
- The following issues were covered:
  - Cross-Site Request Forgery (CSRF)
  - Unverified redirect_uri
  - Open Redirect

- The following issues are interesting for those who want to dig deeper
  - Confused Deputy Attack
  - Misuse of Implicit flow

Modern Security Issues

Modern Security Issues: NoAuth

NoAuth (June 21, 2023)



NoAuth (June 21, 2023)

- Found in Microsoft Azure AD
- Information returned in the “email” claim could be changed
- Applications who were identifying users based on the email claim were vulnerable to impersonation

Reference: https://www.descope.com/blog/post/noauth

NoAuth (June 21, 2023)

- The “email” field was intended to be a contact email

MJ Marcus Johnson
1

Apr 28, 2022, 5:25 AM

After many back and forth emails and sessions with Microsoft support, this is resolved. The resolution is bizarre.
Yes, you can change the email address in the Azure Portal contact information area. NO YOU CANNOT use the user's email address.
However, if you use any other address, in this case I used my gmail email address. Azure happily reports that it the change is saved, but does NOT use the gmail address. It replaces it with the correct user address. Riddle me that Bat Man.

Please sign in to rate this answer.

0 comments Report a concern

Sign in to comment

Image Source: https://learn.microsoft.com/en-us/answers/questions/802766/how-to-change-azuread-user-contact-email-address

NoAuth (June 21, 2023)


```
1 {
2   "aud": "5b445490-7b86-49e7-a32f-a4dbffead36b",
3   "iss": "https://login.microsoftonline.com/cd9fcd8c-84a
  e-4f9a-b5c4-bf54926bb7d3/v2.0",
4   "email": "badguy@l33th4x0r.onmicrosoft.com",
5   "name": "Bad Guy",
6   "oid": "4ddd99a0-47d5-4680-85d4-e9fb8da0a032",
7   "preferred_username": "badguy@l33th4x0r.onmicrosoft.co
  m",
8   "rh": "0.AVIAjM2fza6Emk-1xL9Ukmu305BURFuGe-dJoy-k2__q0
  2u6AK4.",
9   "sub": "Ok7XfFn75AC33fXNDhay20rsdWarDOAgoNM40Nr3Py4",
10  "tid": "cd9fcd8c-84ae-4f9a-b5c4-bf54926bb7d3",
11  "ver": "2.0"
12 }
```

```
1 {
2   "aud": "5b445490-7b86-49e7-a32f-a4dbffead36b",
3   "iss": "https://login.microsoftonline.com/cd9fcd8c-84a
  e-4f9a-b5c4-bf54926bb7d3/v2.0",
4   "email": "omer@descope.com",
5   "name": "Bad Guy",
6   "oid": "4ddd99a0-47d5-4680-85d4-e9fb8da0a032",
7   "preferred_username": "badguy@l33th4x0r.onmicrosoft.co
  m",
8   "rh": "0.AVIAjM2fza6Emk-1xL9Ukmu305BURFuGe-dJoy-k2__q0
  2u6AK4.",
9   "sub": "Ok7XfFn75AC33fXNDhay20rsdWarDOAgoNM40Nr3Py4",
10  "tid": "cd9fcd8c-84ae-4f9a-b5c4-bf54926bb7d3",
11  "ver": "2.0"
12 }
```


Image Source: https://www.descope.com/blog/post/noauth

NoAuth (June 21, 2023)

Subject

Next, to determine if the token subject, such as the user (or app itself for an app-only token), is authorized, either check for specific `sub` or `oid` claims, or check that the subject belongs to an appropriate role or group with the `roles`, `groups`, `wids` claims.

For example, use the immutable claim values `tid` and `oid` as a combined key for application data and determining whether a user should be granted access.

The `roles`, `groups` or `wids` claims can also be used to determine if the subject has authorization to perform an operation. For example, an administrator may have permission to write to an API, but not a normal user, or the user may be in a group allowed to do some action.

⚠ **Warning**

Never use `email` or `upn` claim values to store or determine whether the user in an access token should have access to data. Mutable claim values like these can change over time, making them insecure and unreliable for authorization.

Image Source: https://www.descope.com/blog/post/noauth

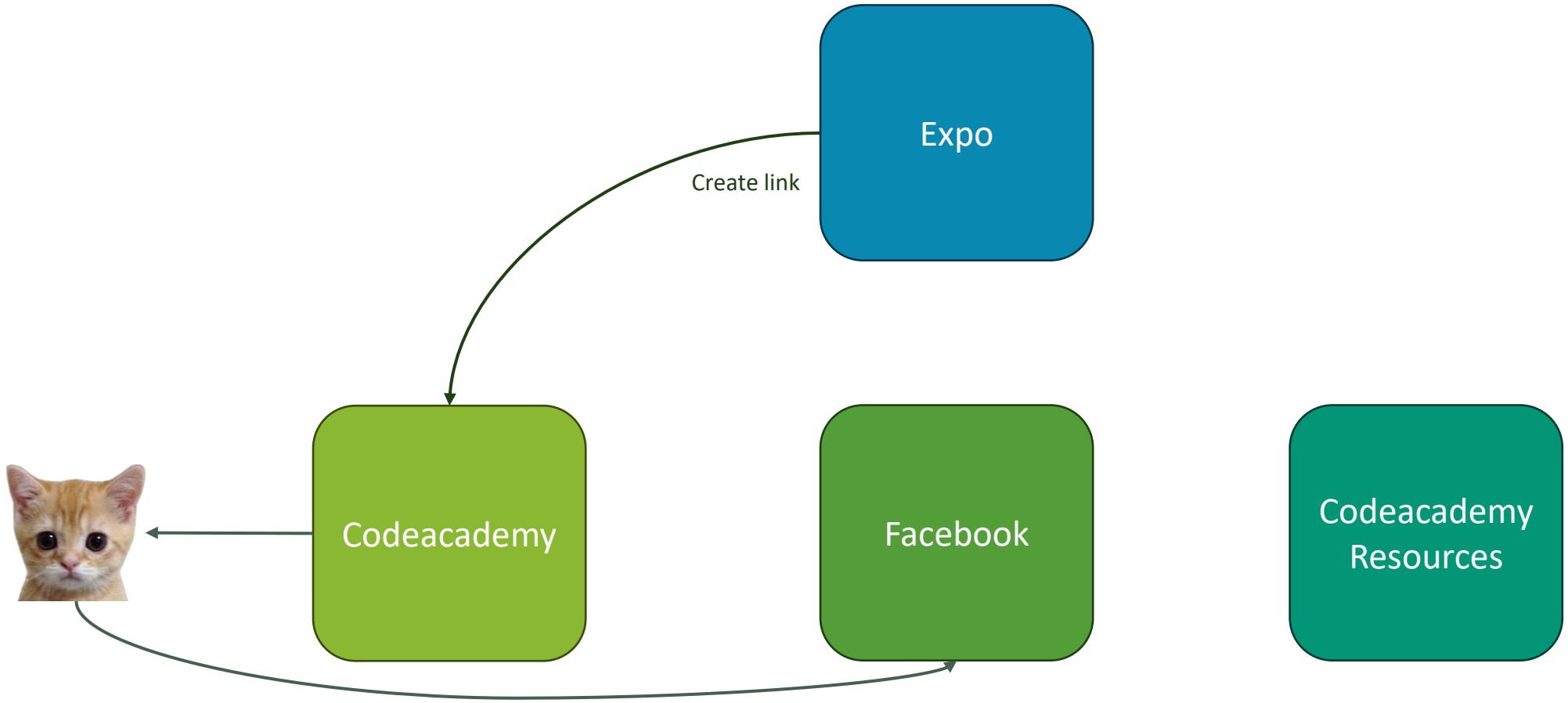
Modern Security Issues: CVE-2023-28131

CVE-2023-28131 (March 10, 2023)

- Initially discovered in Codeacademy.com
- Vulnerability occurred in Expo framework, which is used by hundreds of companies
- Leverages the following issues:
  - Unverified redirectUri
  - Misconfigured implicit grant

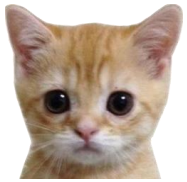
Reference: https://salt.security/blog/a-new-oauth-vulnerability-that-may-impact-hundreds-of-online-services

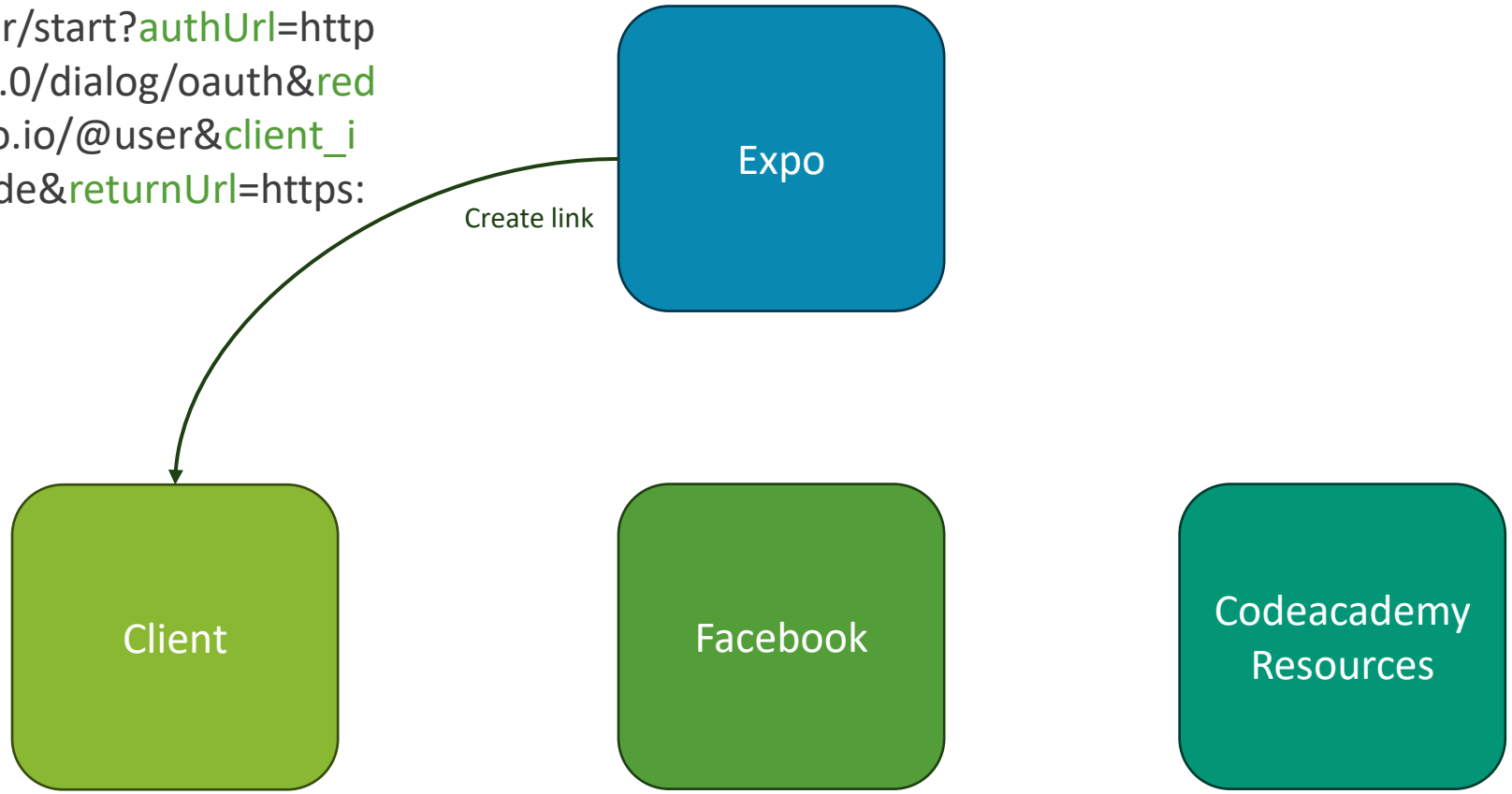
CVE-2023-28131 (March 10, 2023)



CVE-2023-28131 (March 10, 2023)

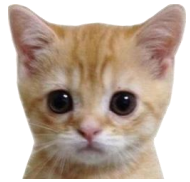
https://auth.expo.io/@user/start?authUrl=https://www.facebook.com/v6.0/dialog/oauth&redirect_uri=https://auth.expo.io/@user&client_id=328&response_type=code&returnUrl=https://codeacademy.com





CVE-2023-28131 (March 10, 2023)

https://auth.expo.io/@user/start?authUrl=https://www.facebook.com/v6.0/dialog/oauth&redirect_uri=https://auth.expo.io/@user&client_id=328&response_type=code,**token**&**returnUrl=hTTps://attacker.com**



Expo

Attacker.com

Create link

https://www.facebook.com/v6.0/dialog/oauth?redirect_uri=https://auth.expo.io/@user&client_id=328…

Client

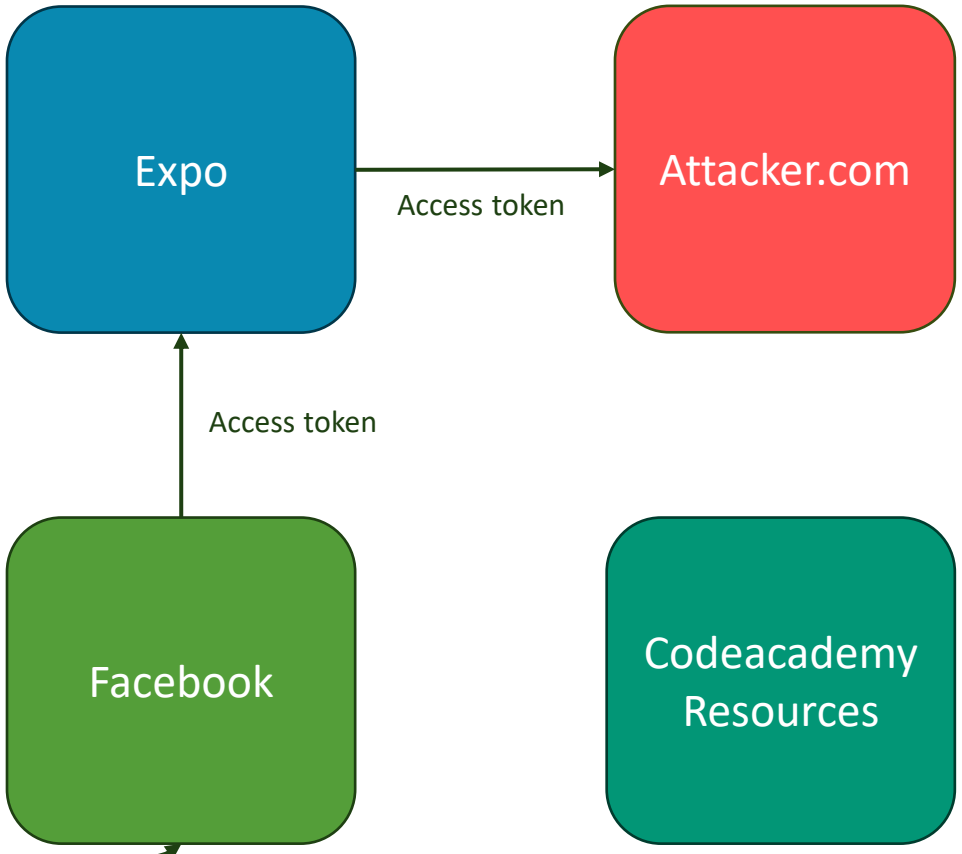
Facebook

Codeacademy Resources

CVE-2023-28131 (March 10, 2023)

https://auth.expo.io/@user/start?authUrl=https://www.facebook.com/v6.0/dialog/oauth&redirect_uri=https://auth.expo.io/@user&client_id=328&response_type=code,**token**&**returnUrl=hTTps://attacker.com**

https://www.facebook.com/v6.0/dialog/oauth?redirect_uri=https://auth.expo.io/@user&client_id=328…



CVE-2023-28131 (March 10, 2023)

- Open redirection occurred in a “Redirect Proxy”, which allows the user to first send the code to auth.expo.io instead of the client application

- Another parameter “returnUrl” was used by Expo to redirect the code to the user’s application

- The returnUrl parameter was not validated by Expo

Reference: https://salt.security/blog/a-new-oauth-vulnerability-that-may-impact-hundreds-of-online-services

Modern Security Issues: Booking.com

Account Takeover on Booking.com (March 2, 2023)

- A series of misconfigurations allowed an attacker to create a malicious link

- Link AttackerFB to Target's Booking.com account

- Leverages the following issues:
  - Unverified redirect_uri
  - Open redirect
  - Cross-site request forgery

Reference: https://salt.security/blog/traveling-with-oauth-account-takeover-on-booking-com

Account Takeover on Booking.com (March 2, 2023)

- Booking.com did not validate redirect_uri in the authorization request

- Valid:
  - *https://www.facebook.com/v3.0/dialog/oauth?redirect_uri=https://account.booking.com/social/result/facebook&scope=email&client_id=210068525731476&state=[large_object]&response_type=code*

- Also Valid:
  - *https://www.facebook.com/v3.0/dialog/oauth?redirect_uri=https://account.booking.com/any/path/an/attacker/wants&scope=email&client_id=210068525731476&state=[large_object]&response_type=code*

Reference: https://salt.security/blog/traveling-with-oauth-account-takeover-on-booking-com

Account Takeover on Booking.com (March 2, 2023)

- An Open Redirect was discovered on Booking.com
  - Reminder: an open redirect is a link on a website that can be configured to take a user to any other website on the internet

- State token contained the following string: *https://account.booking.com/oauth2/authorize?aid=123;client_id=d1cDdLj40ACItEtxJLTo;redirect_uri=https://account.booking.com/settings/oauth_callback;response_type=code;state=**eyJteXNldHRpbmdzX3BhdGgiOiIvbXlzZXR0aW5ncy9wZXJzb25hbCIsImFpZCI6IjEyMyJ9***

Reference: https://salt.security/blog/traveling-with-oauth-account-takeover-on-booking-com

Account Takeover on Booking.com (March 2, 2023)

- Decoded state token:

- Misuse of state token also failed to prevent CSRF

```
{
  "mysettings_path":"/mysettings/personal",
  "aid":"123"
}
```

Reference: https://salt.security/blog/traveling-with-oauth-account-takeover-on-booking-com

Modern Security Issues Recap

The following issues were covered:

- Account Takeover on Booking.com (discovered March 2, 2023)
- CVE-2023-28131 (discovered March 10, 2023)
- Microsoft NoAuth (discovered June 21, 2023)

Conclusion

Learning outcomes

1. What is the OAuth protocol? What is it used for?

2. What is the OpenID Connect protocol? How does it relate to OAuth?

3. Where have vulnerabilities occurred in OAuth and OpenID Connect?

Conclusion

- OAuth has a long and ugly history of vulnerabilities
- Bugs can occur in both providers and clients
- Historic bugs (2013 and before) are still found in modern applications

Resources

- The OAuth 2.0 Authorization Framework: https://www.rfc-editor.org/rfc/rfc6749
- OAuth 2.0 authentication vulnerabilities: https://portswigger.net/web-security/oauth
- PKCE vs. Nonce: Equivalent or Not?: https://danielfett.de/2020/05/16/pkce-vs-nonce-equivalent-or-not/

Blogs

- Account hijacking using "dirty dancing" in sign-in OAuth-flows: https://labs.detectify.com/2022/07/06/account-hijacking-using-dirty-dancing-in-sign-in-oauth-flows/
- Multiple bugs chained to takeover Facebook Accounts which uses Gmail: https://ysamm.com/?p=763
- Recommended to read the blogs referenced on the bottom of slides

Wrap-Up

- Questions?
- PollEv.com/haxxpanda098

Thanks for listening!

Add me on twitter @0xchsh ☺