

# YOU ONLY LOGIN ONCE

Presented by: chsh

The bottom half of the slide features a decorative graphic consisting of numerous thin, parallel orange lines that flow and curve across the width of the image, creating a sense of motion and depth.

# SINGLE SIGN ON



# **CONTENT**

- **OpenID Connect protocol (OIDC)**
- **OIDC bug class that I think is cool**
- **OIDC bug that I saw in the wild (less cool)**

# LOGGING IN TO SLACK



https://slack.com


slack


# First of all, enter your email address

We suggest using the email address that you use at work.

Continue

OR


 Continue with Google

 Continue with Apple


Already using Slack?  
[Sign in to an existing workspace](#)

https://accounts.google.com/...

← → ↻ 🔒 🔓 🌐 https://accounts.google.com/o/oauth2/v2/auth/oauthchooseaccount?client\_id=6060... 🔍 ⬆️ ⬆️ ⬆️


 Sign in with Google

---




# Choose an account

to continue to [Slack](#)

 **chshtesting!**  
chshtesting@gmail.com

---

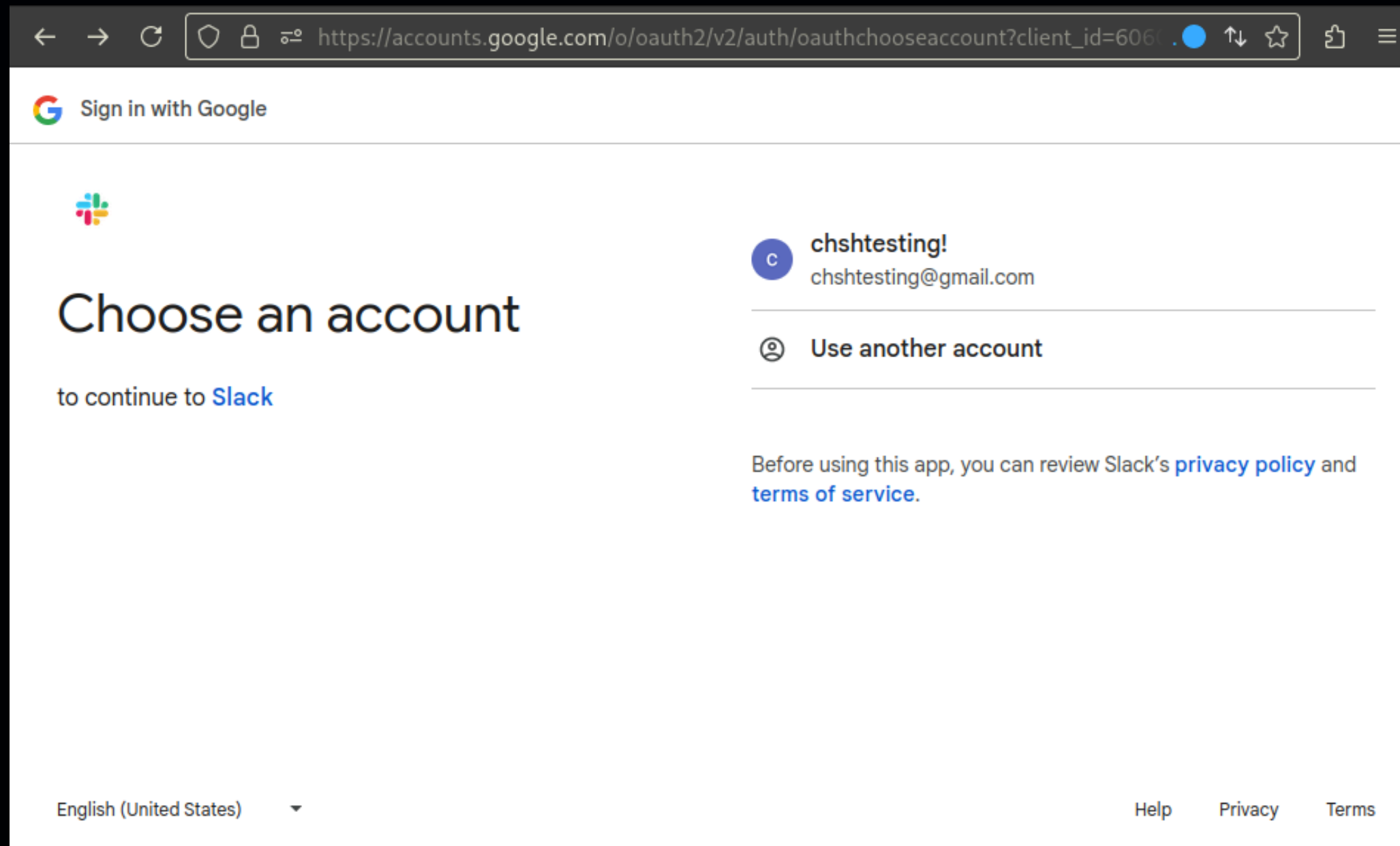
 Use another account

---

Before using this app, you can review Slack's [privacy policy](#) and [terms of service](#).

English (United States) ▼ Help Privacy Terms

https://accounts.google.com/o/oauth2/v2/auth/oauthchooseaccount?client\_id=606092904014-s1u3idjanlbhr4ns5b1hcjgfn63cr9nh.apps.googleusercontent.com&redirect\_uri=https://oauth2.slack.com/signin/oauth/google/end&scope=openid email profile&response\_type=code&access\_type=offline&state={"provider":"google","origin":"signin","timestamp":1724683119,"visitor":"796e18fba4f59969ef46afe47b252c14"}|0e201ea7bbed1ab29fba6000258397c02b391792d110740d86561bfadbf3c3a1&prompt=consent&service=iso&o2v=2&ddm=0&flowName=GeneralOAuthFlow



The screenshot shows a web browser window displaying the Google account selection interface. The address bar shows the URL: https://accounts.google.com/o/oauth2/v2/auth/oauthchooseaccount?client\_id=606092904014-s1u3idjanlbhr4ns5b1hcjgfn63cr9nh.apps.googleusercontent.com&redirect\_uri=https://oauth2.slack.com/signin/oauth/google/end&scope=openid email profile&response\_type=code&access\_type=offline&state={"provider":"google","origin":"signin","timestamp":1724683119,"visitor":"796e18fba4f59969ef46afe47b252c14"}|0e201ea7bbed1ab29fba6000258397c02b391792d110740d86561bfadbf3c3a1&prompt=consent&service=iso&o2v=2&ddm=0&flowName=GeneralOAuthFlow. The page header includes the Google logo and "Sign in with Google". The main content area features the Slack logo and the heading "Choose an account" with the subtext "to continue to Slack". On the right, a user profile is shown for "chshtesting!" with the email "chshtesting@gmail.com". Below this, there is a link "Use another account". At the bottom, there is a note: "Before using this app, you can review Slack's [privacy policy](#) and [terms of service](#)." The footer contains the language selection "English (United States)" and links for "Help", "Privacy", and "Terms".

https://accounts.google.com/o/oauth2/v2/auth/oauthchooseaccount?client\_id=606092904014-s1u3idjanlbhr4ns5b1hcjgfn63cr9nh.apps.googleusercontent.com&redirect\_uri=https://oauth2.slack.com/signin/oauth/google/end&scope=openid email profile&response\_type=code&access\_type=offline&state={"provider":"google","origin":"signin","timestamp":1724683119,"visitor":"796e18fba4f59969ef46afe47b252c14"}|0e201ea7bbed1ab29fba6000258397c02b391792d110740d86561bfadbf3c3a1&prompt=consent&service=iso&o2v=2&ddm=0&flowName=GeneralOAuthFlow



Relaying  
Party

A screenshot of a web browser showing a Google account selection screen. The browser's address bar contains the URL: https://accounts.google.com/o/oauth2/v2/auth/oauthchooseaccount?client\_id=606092904014-s1u3idjanlbhr4ns5b1hcjgfn63cr9nh.apps.googleusercontent.com&redirect\_uri=https://oauth2.slack.com/signin/oauth/google/end&scope=openid email profile&response\_type=code&access\_type=offline&state={"provider":"google","origin":"signin","timestamp":1724683119,"visitor":"796e18fba4f59969ef46afe47b252c14"}|0e201ea7bbed1ab29fba6000258397c02b391792d110740d86561bfadbf3c3a1&prompt=consent&service=iso&o2v=2&ddm=0&flowName=GeneralOAuthFlow. The page header says "Sign in with Google". Below that is the Slack logo. The main heading is "Choose an account" with the subtext "to continue to Slack". On the right, there is a profile card for "chshtesting!" with the email "chshtesting@gmail.com" and a "Use another account" option. At the bottom, there is a language selector set to "English (United States)" and links for "Help", "Privacy", and "Terms".

← → ↻ 🔒 🗄️ https://accounts.google.com/o/oauth2/v2/auth/oauthchooseaccount?client\_id=606092904014-s1u3idjanlbhr4ns5b1hcjgfn63cr9nh.apps.googleusercontent.com&redirect\_uri=https://oauth2.slack.com/signin/oauth/google/end&scope=openid email profile&response\_type=code&access\_type=offline&state={"provider":"google","origin":"signin","timestamp":1724683119,"visitor":"796e18fba4f59969ef46afe47b252c14"}|0e201ea7bbed1ab29fba6000258397c02b391792d110740d86561bfadbf3c3a1&prompt=consent&service=iso&o2v=2&ddm=0&flowName=GeneralOAuthFlow

Sign in with Google

Choose an account

to continue to [Slack](#)

**chshtesting!**  
chshtesting@gmail.com

Use another account

Before using this app, you can review Slack's [privacy policy](#) and [terms of service](#).

English (United States) ▼ Help Privacy Terms



https://accounts.google.com/o/oauth2/v2/auth/oauthchooseaccount?client\_id=606092904014-s1u3idjanlbhr4ns5b1hcjgfn63cr9nh.apps.googleusercontent.com&redirect\_uri=https://oauth2.slack.com/signin/oauth/google/end&scope=openid email profile&response\_type=code&access\_type=offline&state={"provider":"google","origin":"signin","timestamp":1724683119,"visitor":"796e18fba4f59969ef46afe47b252c14"}|0e201ea7bbed1ab29fba6000258397c02b391792d110740d86561bfadbf3c3a1&prompt=consent&service=iso&o2v=2&ddm=0&flowName=GeneralOAuthFlow

The screenshot shows a web browser window displaying the Google sign-in page for Slack. The browser's address bar shows the URL: https://accounts.google.com/signin/oauth/id?authuser=0&part=AJi8hAPmiQpkFeT... . The page header includes the Google logo and the text "Sign in with Google". The main content area features the Slack logo, the heading "Sign in to Slack", and a dropdown menu showing the email address "chshtesting@gmail.com". To the right of the email field, there is a text block: "By continuing, Google will share your name, email address, language preference and profile picture with Slack. See Slack's [privacy policy](#) and [Terms of Service](#). You can manage Sign in with Google in your [Google Account](#)." Below this text are two buttons: "Cancel" and "Continue". At the bottom of the page, there is a language selector set to "English (United Kingdom)" and links for "Help", "Privacy", and "Terms".

https://accounts.google.com/o/oauth2/v2/auth/oauthchooseaccount?client\_id=606092904014-s1u3idjanlbhr4ns5b1hcjgfn63cr9nh.apps.googleusercontent.com&redirect\_uri=https://oauth2.slack.com/signin/oauth/google/end&scope=openid email profile&response\_type=code&access\_type=offline&state={"provider":"google","origin":"signin","timestamp":1724683119,"visitor":"796e18fba4f59969ef46afe47b252c14"}|0e201ea7bbed1ab29fba6000258397c02b391792d110740d86561bfadbf3c3a1&prompt=consent&service=iso&o2v=2&ddm=0&flowName=GeneralOAuthFlow

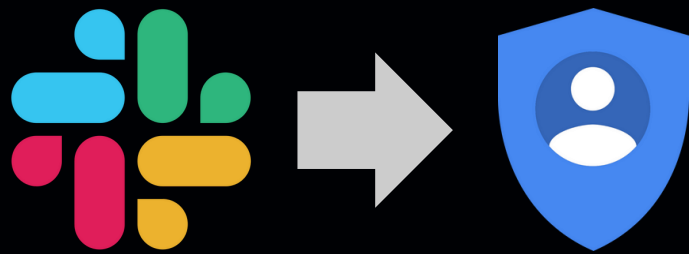
The screenshot shows a web browser at `https://slack.com/signin#workspaces`. The Slack logo is visible at the top. The developer tools network tab is open, showing a list of requests. A tooltip is displayed over a GET request to `slack.com/signin`, showing the original and decoded URLs.

Status	Method	Domain	File	Initiator	Type	Transferred	Size	Time	Other
302	POST	accounts.google.com	id?as=51679627720:1724683146029109&authuser=0&...	fetch	html	38.67 kB	3...	212 ms	
302	GET	accounts.google.com	consent?as=51679627720:1724683146029109&authu...	document	html	37.03 kB	3...	246 ms	
Blocked	POST	accounts.google.com	browserinfo?f.sid=8456323435363838408&bl=boq_id...	m= b, tp:328 (xhr)		Blocked By uBloc...			
302	GET	oauth2.slack.com	end?state={"provider":"google","origin":"signi...	document	html	37.52 kB	3...	546 ms	
200	GET	slack.com	signin						
	GET	a.slack-edge.com	rollup-sla...						
	GET	a.slack-edge.com	onetrust...						
	GET	a.slack-edge.com	lato-2-co...						
	GET	a.slack-edge.com	_generic...						
	GET	a.slack-edge.com	rollup-sla...						
	GET	a.slack-edge.com	rollup-sla...						
200	GET	a.slack-edge.com	manifest...						

Original: `https://oauth2.slack.com/signin/oauth/google/end?state=%7B%22provider%22%3A%22google%22%2C%22origin%22%3A%22signin%22%7D&prompt=consent&service=iso&o2v=2&ddm=0&flowName=GeneralOAuthFlow`

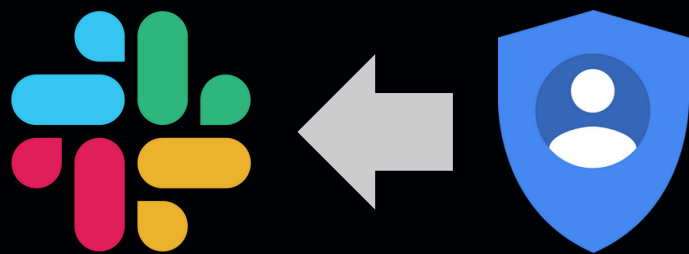
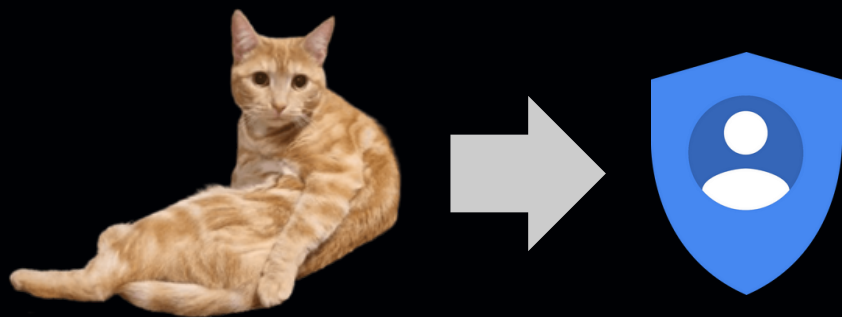
Decoded: `https://oauth2.slack.com/signin/oauth/google/end?state={"provider":"google","origin":"signin","timestamp":1724683119,"visitor":"796e18fba4f59969ef46afe47b252c14"}|0e201ea7bbed1ab29fba6000258397c02b391792d110740d86561bfadbf3c3a1&code=4www.googleapis.com/auth/userinfo.profile+https://www.googleapis.com/auth/userinfo.email&authuser=0&prompt=consent`

43 requests | 1.45 MB / 1.17 MB transferred | Finish: 36.59 s | DOMContentLoaded: 3.92 s | load: 4.08 s



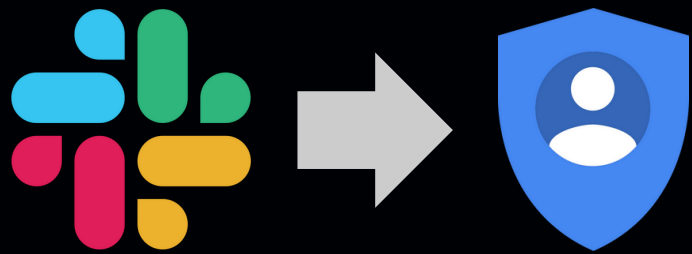
[https://accounts.google.com/o/oauth2/v2/auth/oauthchooseaccount?client\\_id=606092904014-s1u3idjanlbhr4ns5b1hcjgfn63cr9nh.apps.googleusercontent.com&redirect\\_uri=https://oauth2.slack.com/signin/oauth/google/end&scope=openid email profile&response\\_type=code&access\\_type=offline&state={"provider":"google","origin":"signin","timestamp":1724683119,"visitor":"796e18fba4f59969ef46afe47b252c14"}|0e201ea7bbbed1ab29fba6000258397c02b391792d110740d86561bfadbf3c3a1&prompt=consent&service=iso&o2v=2&ddm=0&flowName=GeneralOAuthFlow](https://accounts.google.com/o/oauth2/v2/auth/oauthchooseaccount?client_id=606092904014-s1u3idjanlbhr4ns5b1hcjgfn63cr9nh.apps.googleusercontent.com&redirect_uri=https://oauth2.slack.com/signin/oauth/google/end&scope=openid email profile&response_type=code&access_type=offline&state={)

*Authentication request*



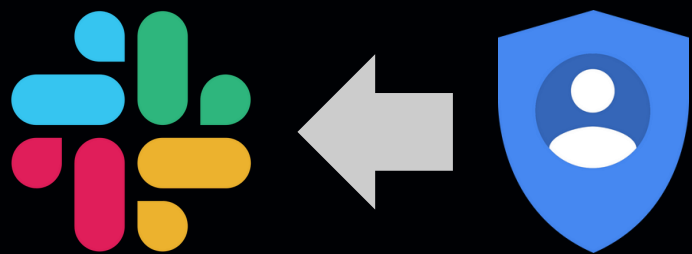
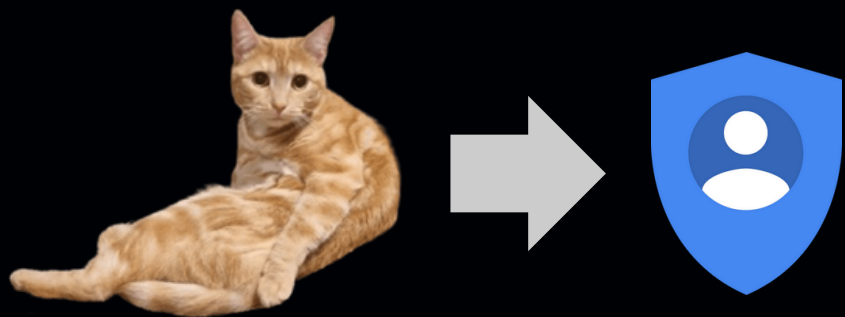
[https://oauth2.slack.com/signin/oauth/google/end?state={"provider":"google","origin":"signin","timestamp":1724683119,"visitor":"796e18fba4f59969ef46afe47b252c14"}|0e201ea7bbbed1ab29fba6000258397c02b391792d110740d86561bfadbf3c3a1&code=4/0AQlEd8ybjf6rEDd5FsglnCIFI9Pg0qF5vU1yTYBuuaG49WJnkukVIQB aVCdVFMgV7Ck2eg&scope=email profile openid https://www.googleapis.com/auth/userinfo.profile https://www.googleapis.com/auth/userinfo.email&authuser=0&prompt=consent](https://oauth2.slack.com/signin/oauth/google/end?state={)

*Authentication response*



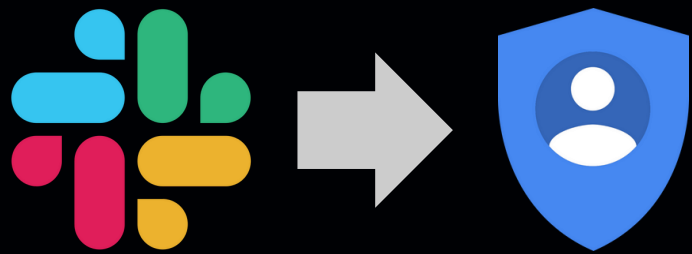
[https://accounts.google.com/o/oauth2/v2/auth/oauthchooseaccount?client\\_id=606092904014-s1u3idjanlbhr4ns5b1hcjgfn63cr9nh.apps.googleusercontent.com&redirect\\_uri=https://oauth2.slack.com/signin/oauth/google/end&scope=openid email profile&response\\_type=code&access\\_type=offline&state={"provider":"google","origin":"signin","timestamp":1724683119,"visitor":"796e18fba4f59969ef46afe47b252c14"}|0e201ea7bbed1ab29fba6000258397c02b391792d110740d86561bfadbf3c3a1&prompt=consent&service=lso&o2v=2&ddm=0&flowName=GeneralOAuthFlow](https://accounts.google.com/o/oauth2/v2/auth/oauthchooseaccount?client_id=606092904014-s1u3idjanlbhr4ns5b1hcjgfn63cr9nh.apps.googleusercontent.com&redirect_uri=https://oauth2.slack.com/signin/oauth/google/end&scope=openid email profile&response_type=code&access_type=offline&state={)

*Authentication request*



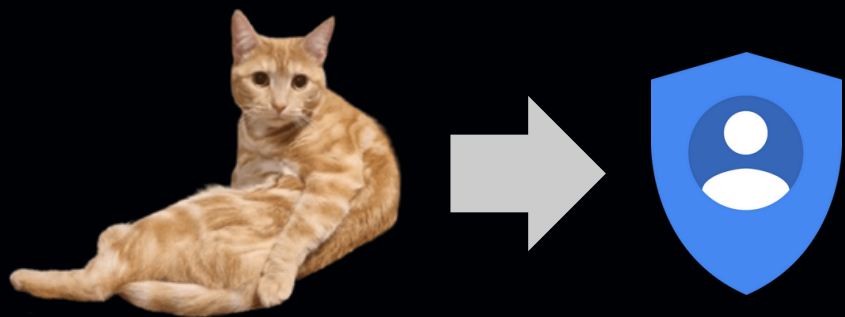
[https://oauth2.slack.com/signin/oauth/google/end?state={"provider":"google","origin":"signin","timestamp":1724683119,"visitor":"796e18fba4f59969ef46afe47b252c14"}|0e201ea7bbed1ab29fba6000258397c02b391792d110740d86561bfadbf3c3a1&code=4/0AQIEd8ybjf6rEDd5FsglnCIFI9Pg0qF5vU1yTYBuuaG49WJnkukVIQB aVCdVFMgV7Ck2eg&scope=email profile openid https://www.googleapis.com/auth/userinfo.profile https://www.googleapis.com/auth/userinfo.email&authuser=0&prompt=consent](https://oauth2.slack.com/signin/oauth/google/end?state={)

*Authentication response*



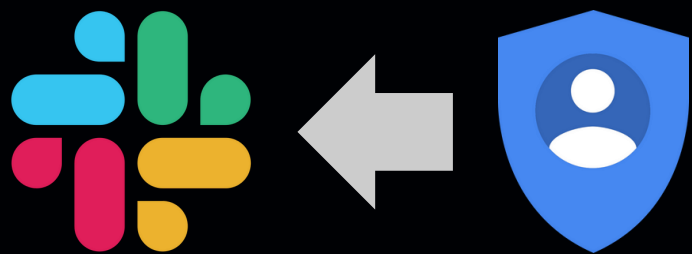
[https://accounts.google.com/o/oauth2/v2/auth/oauthchooseaccount?client\\_id=606092904014-s1u3idjanlbhr4ns5b1hcjgfn63cr9nh.apps.googleusercontent.com&redirect\\_uri=https://oauth2.slack.com/signin/oauth/google/end&scope=openid email profile&response\\_type=id\\_token&nonce=1234&state={"provider":"google","origin":"signin","timestamp":1724683119,"visitor":"796e18fba4f59969ef46afe47b252c14"}|0e201ea7bbbed1ab29fba6000258397c02b391792d110740d86561bfadbf3c3a1&prompt=consent&service=lso&o2v=2&ddm=0&flowName=GeneralOAuthFlow](https://accounts.google.com/o/oauth2/v2/auth/oauthchooseaccount?client_id=606092904014-s1u3idjanlbhr4ns5b1hcjgfn63cr9nh.apps.googleusercontent.com&redirect_uri=https://oauth2.slack.com/signin/oauth/google/end&scope=openid email profile&response_type=id_token&nonce=1234&state={)

*Authentication request*




[https://oauth2.slack.com/signin/oauth/google/end#state={"provider":"google","origin":"signin","timestamp":1724764245,"visitor":"796e18fba4f59969ef46afe47b252c14"}|81cdfea6c1e8dacfe2b43e7672239ac847ef2f4f95008cb7f5e0b1cf6339c86a&id\\_token=eyJhbGciOiJSUzI1NiIsImtpZCI6ImE0OTM5MWJmNTJiNTljMWQ1NjAyNTVjMmYyYTA0ZTU5ZTIyYTdiNjUiLCJ0eXAiOiJKV1QiLCJpc3MiOiJodHRwczovL2FjY291bnRzLmdvb2dsZS5jb20iLCJhenAiOiI2M...&authuser=0&prompt=consent&version\\_info=CmxfU1ZJX0VPdmNlSm14bFlhREdCQWIQMDFCUIVSSVpsOXhRWGhNU1hsRWFUXUZZMVpLVGtrMGVITk9jVzlsSU2pWR2NFMTFVR2xoWmpWamN6UnhZVzljJTFVGMtE9XcEdTazFuV0UxcWVGtMphd18](https://oauth2.slack.com/signin/oauth/google/end#state={)

*Authentication response*



← → ↺ 🔒 🔓 🔍 https://oauth2.slack.com/signin/oauth/google/end#state={"provider":"google","orig . 🔵 ↕ ☆ 📄 ☰

## **Server Error**

Sorry! Something went wrong, but we're looking into it. 

If the problem continues, please check our Status page for updates: [slack-status.com](https://slack-status.com)



# “Non-happy path”

*-Frans Rosen, <https://labs.detectify.com/writeups/account-hijacking-using-dirty-dancing-in-sign-in-oauth-flows/#non-happy-paths-in-the-oauth-dance>*

# OAuth 2.0 ← built on → OIDC

← → ↻ datatracker.ietf.org/doc/html/rfc6749 ☆

Internet Engineering Task Force (IETF) Request for Comments: 6749  
Obsoletes: [5849](#)  
Category: Standards Track  
ISSN: 2070-1721

D. Hardt, Ed.  
Microsoft  
October 2012

## The OAuth 2.0 Authorization Framework

### Abstract

The OAuth 2.0 authorization framework enables a third-party application to obtain limited access to an HTTP service, either on behalf of a resource owner by orchestrating an approval interaction between the resource owner and the HTTP service, or by allowing the third-party application to obtain access on its own behalf. This specification replaces and obsoletes the OAuth 1.0 protocol described in [RFC 5849](#).

### Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in [Section 2 of RFC 5741](#).

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at <http://www.rfc-editor.org/info/rfc6749>.

### Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

← → ↻ openid.net/specs/openid-connect-core-1\_0.html 🔍 ☆ 🏠 📄 📄 📄 👤

Final	N. Sakimura
	NAT.Consulting (was at NRI)
	J. Bradley
	Yubico (was at Ping Identity)
	M. Jones
	Self-Issued Consulting (was at
	Microsoft)
	B. de Medeiros
	Google
	C. Mortimore
	Disney (was at Salesforce)
	December 15, 2023

**OpenID Connect Core 1.0 incorporating errata set 2**

### Abstract

OpenID Connect 1.0 is a simple identity layer on top of the OAuth 2.0 protocol. It enables Clients to verify the identity of the End-User based on the authentication performed by an Authorization Server, as well as to obtain basic profile information about the End-User in an interoperable and REST-like manner.

This specification defines the core OpenID Connect functionality: authentication built on top of OAuth 2.0 and the use of Claims to communicate information about the End-User. It also describes the security and privacy considerations for using OpenID Connect.

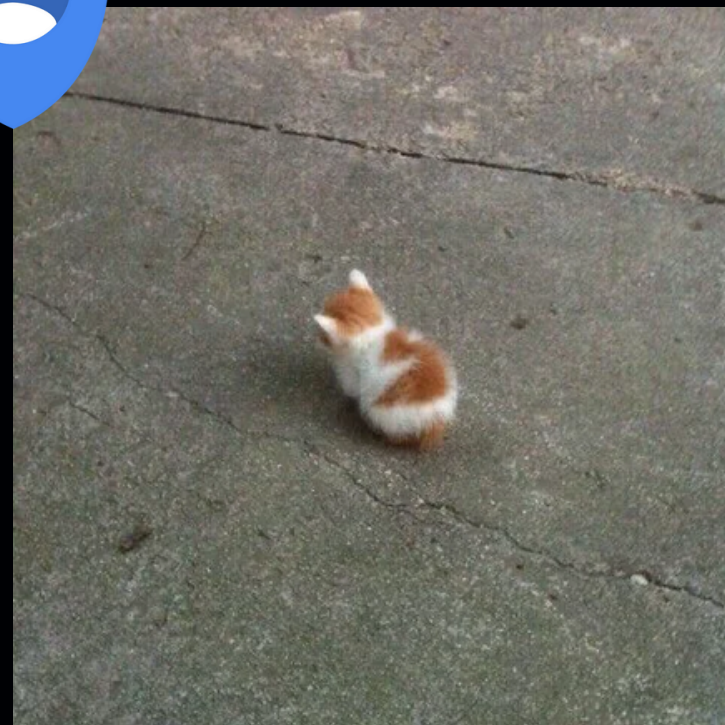
### Table of Contents

- 1. Introduction**
  - 1.1. Requirements Notation and Conventions**
  - 1.2. Terminology**
  - 1.3. Overview**
- 2. ID Token**
- 3. Authentication**
  - 3.1. Authentication using the Authorization Code Flow**
    - 3.1.1. Authorization Code Flow Steps**
    - 3.1.2. Authorization Endpoint**

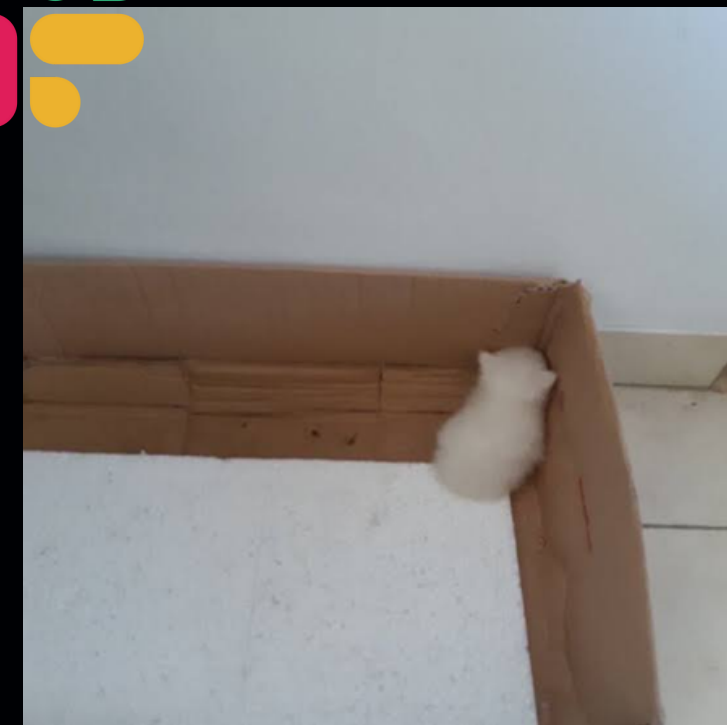


# RECAP

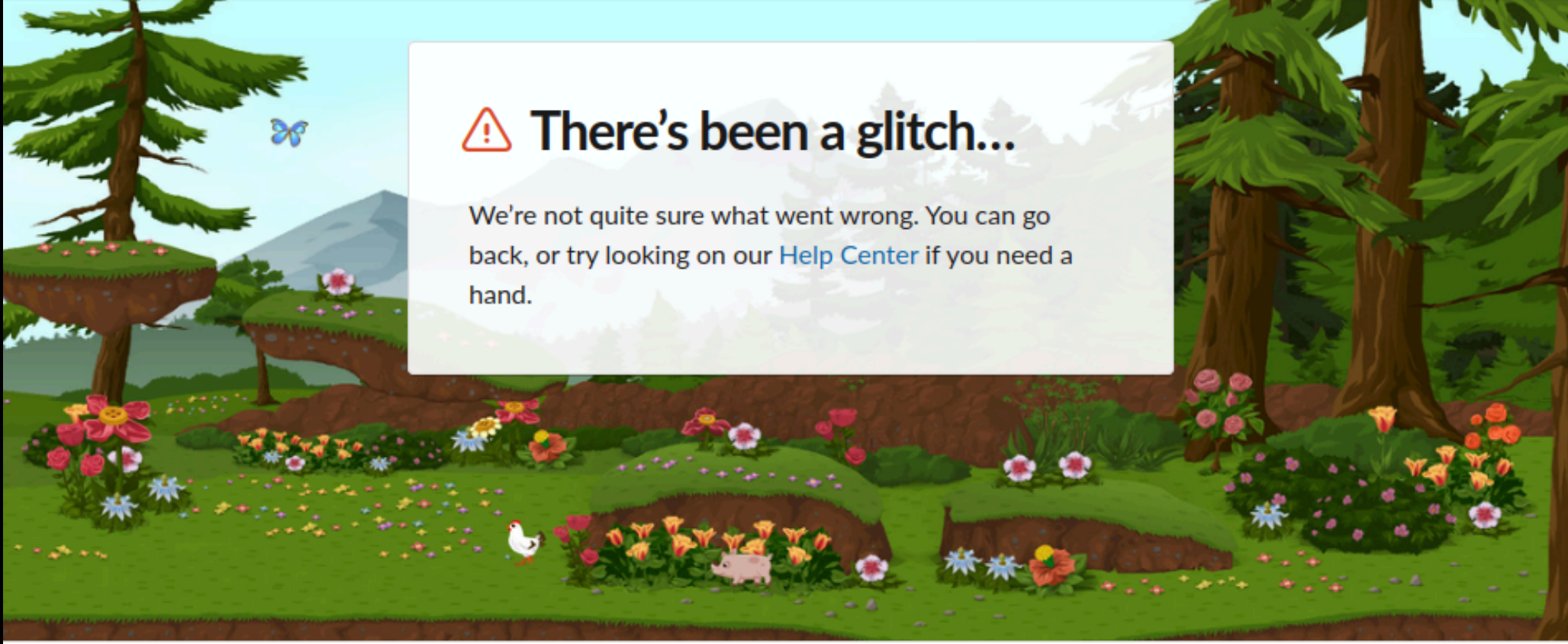
- OpenID Connect (OIDC) is an SSO protocol based on OAuth 2.0




Identity  
Provider



Relaying  
Party



 **There's been a glitch...**

We're not quite sure what went wrong. You can go back, or try looking on our [Help Center](#) if you need a hand.




New to Slack?  
[Create an account](#)


# Sign in to Slack

We suggest using the email address you use at work.

 **Sign In With Google**

 Something weird happened. Please try to sign up again.

 **Sign In With Apple**

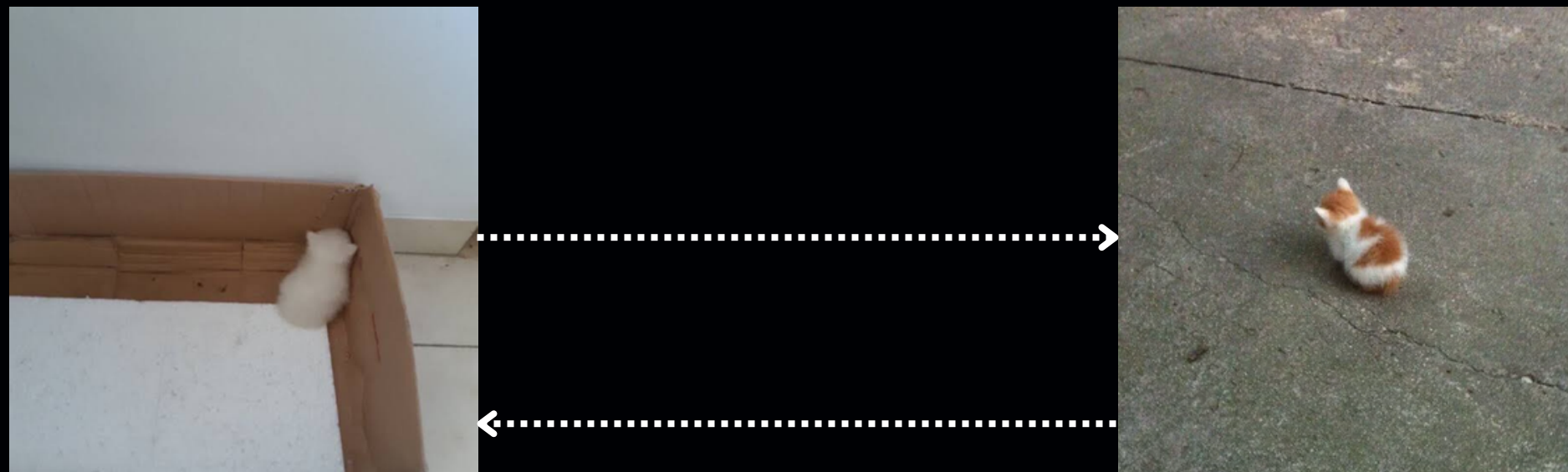
 Something weird happened. Please try to sign up again.

# Non-happy paths

- When codes or tokens issued in the OAuth flow are not consumed by the page they land on and remain in the website URL
- Can be accessed by JavaScript using `location.href`
- To XSS... and beyond?

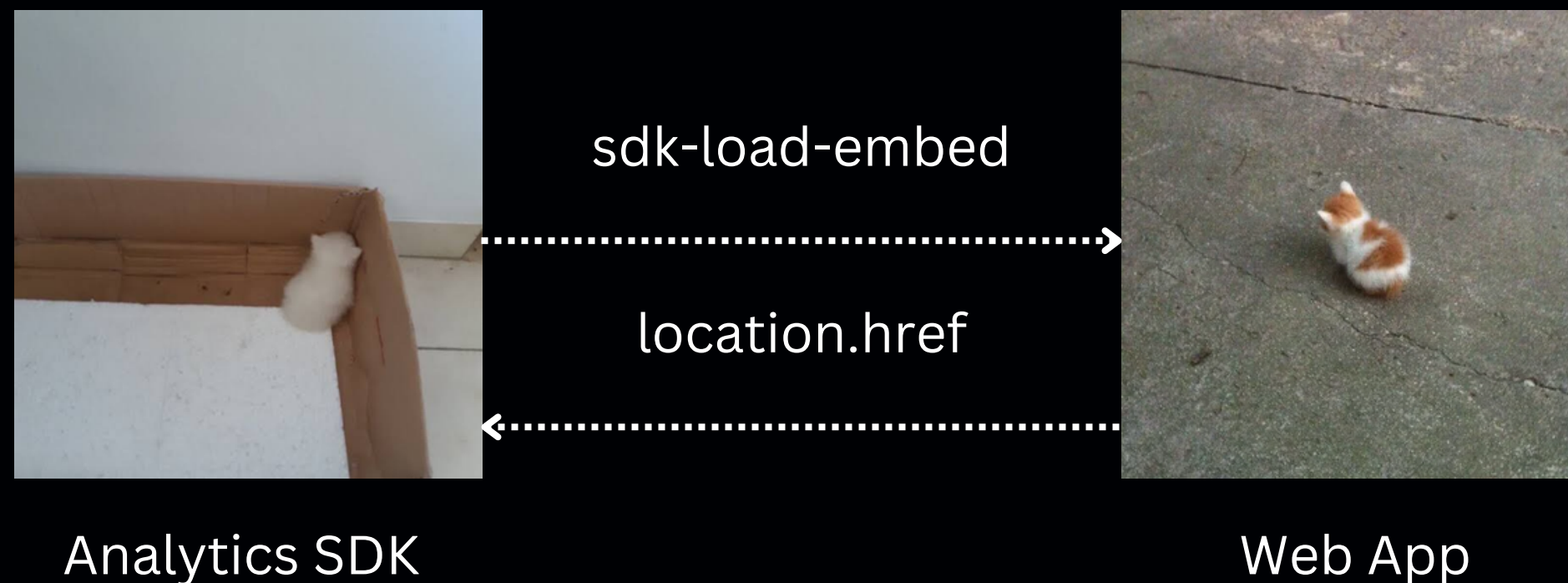
# postMessage()

- Allows cross origin communication between websites
- Should validate origin of sender for security
- Can be used to create URL-leaking “gadgets”



# postMessage()

- Allows cross origin communication between websites
- Should validate origin of sender for security
- Can be used to create URL-leaking “gadgets”



# Example attack

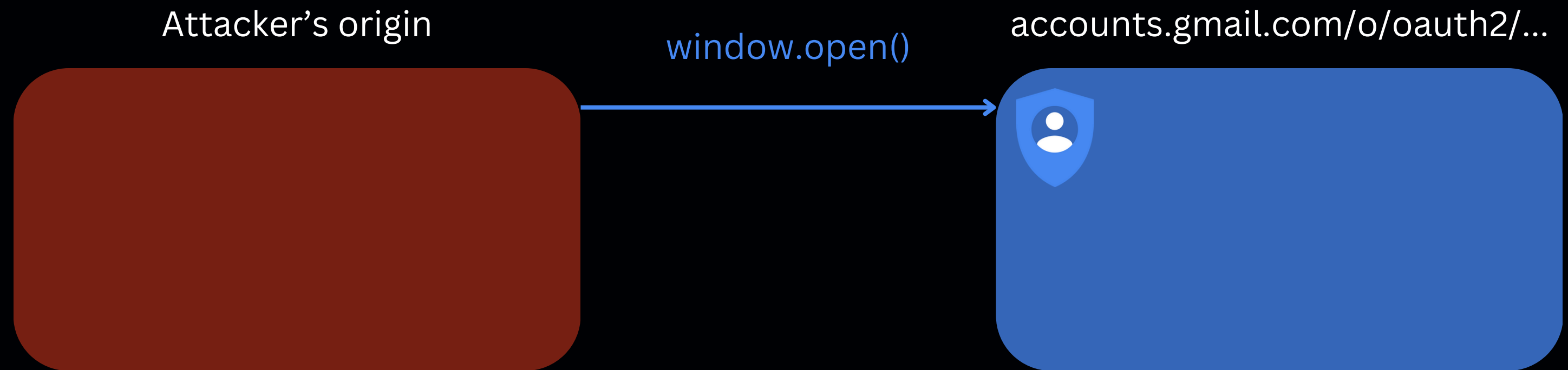
- Attacker creates a malicious page which is loaded in the target's browser

Attacker's origin



# Example attack

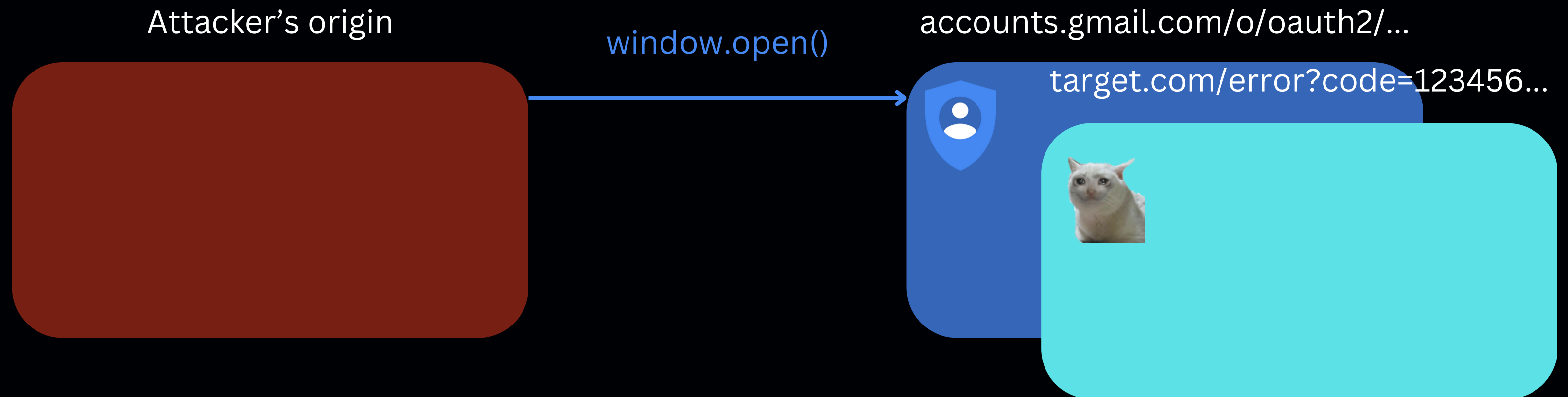
- Attacker creates a malicious page which is loaded in the target's browser





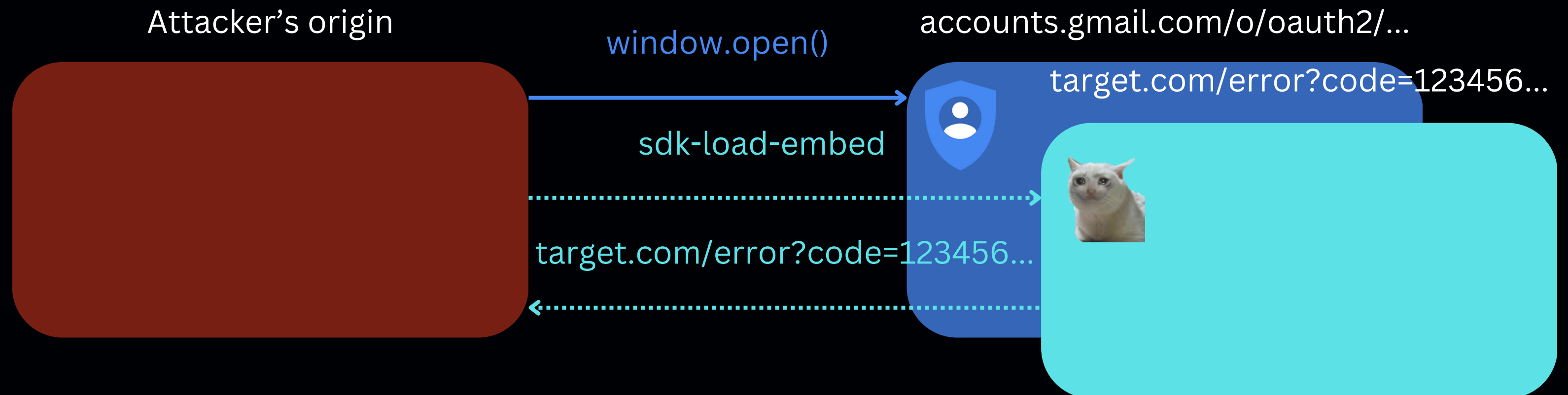
# Example attack

- Attacker creates a malicious page which is loaded in the target's browser



# Example attack

- Attacker creates a malicious page which is loaded in the target's browser



# WHAT IS OUT THERE



I wish  
vulnerabilities  
were real

# Custom flows?

- Saw an authentication request with the parameter `response_type=pi.flow`

# Custom flows?

- Saw an authentication request with the parameter `response_type=pi.flow`
- Custom flow defined by Ping identity



# Custom flows?

- Saw an authentication request with the parameter `response_type=pi.flow`
- Custom flow defined by Ping identity
- No need for redirects between origins
- Code or token is returned within response to API



# Custom flows?

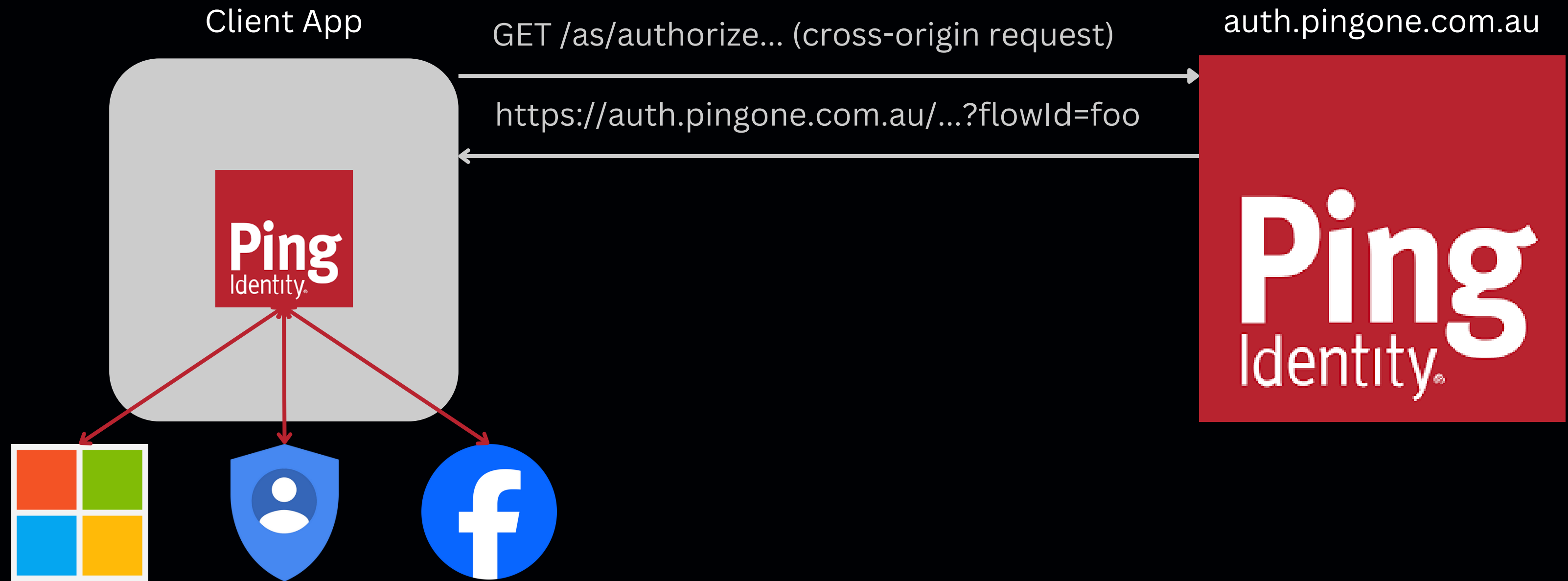


# Custom flows?

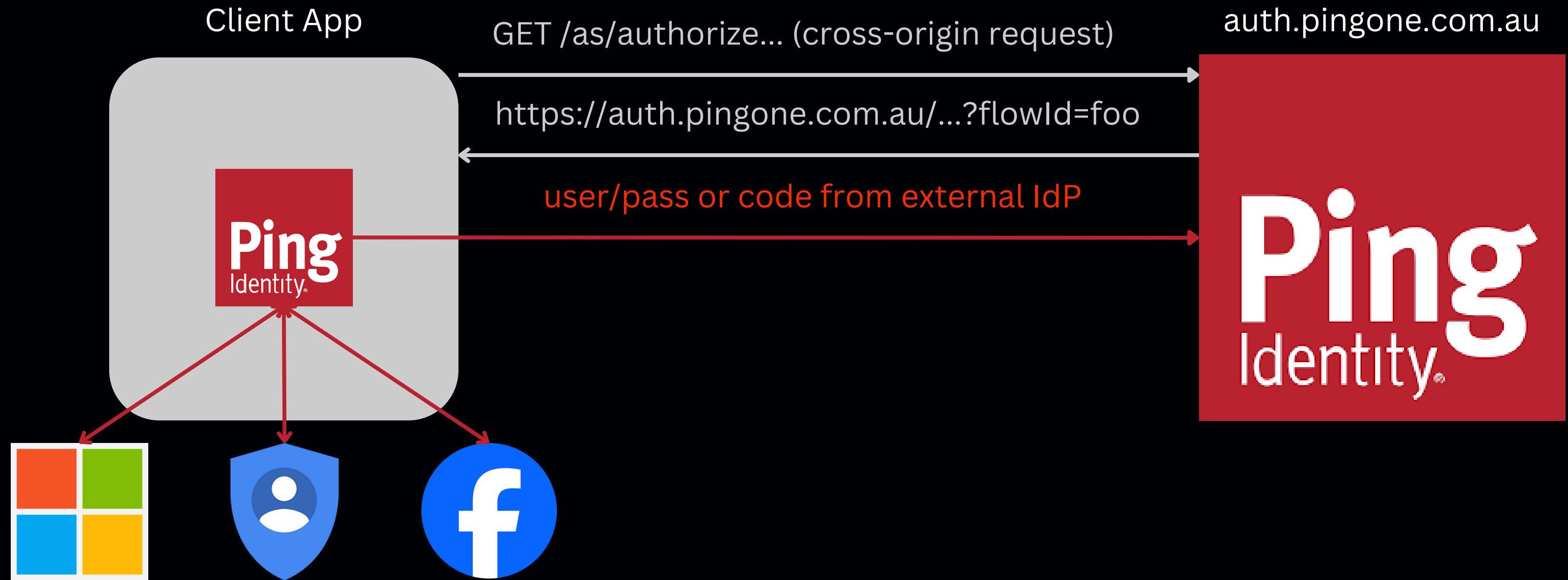




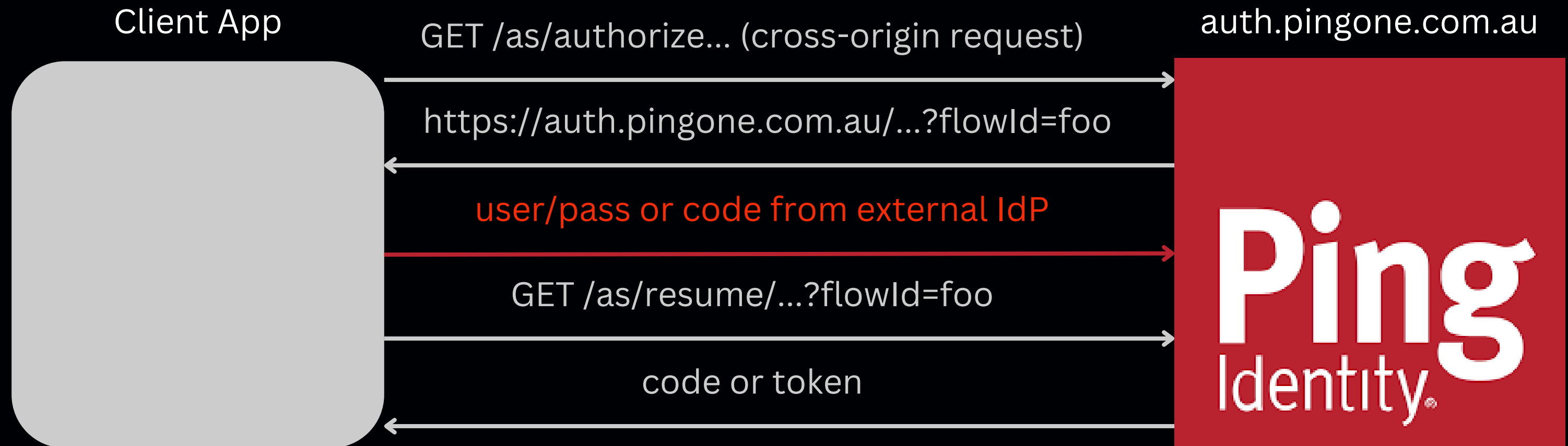
# Custom flows?



# Custom flows?



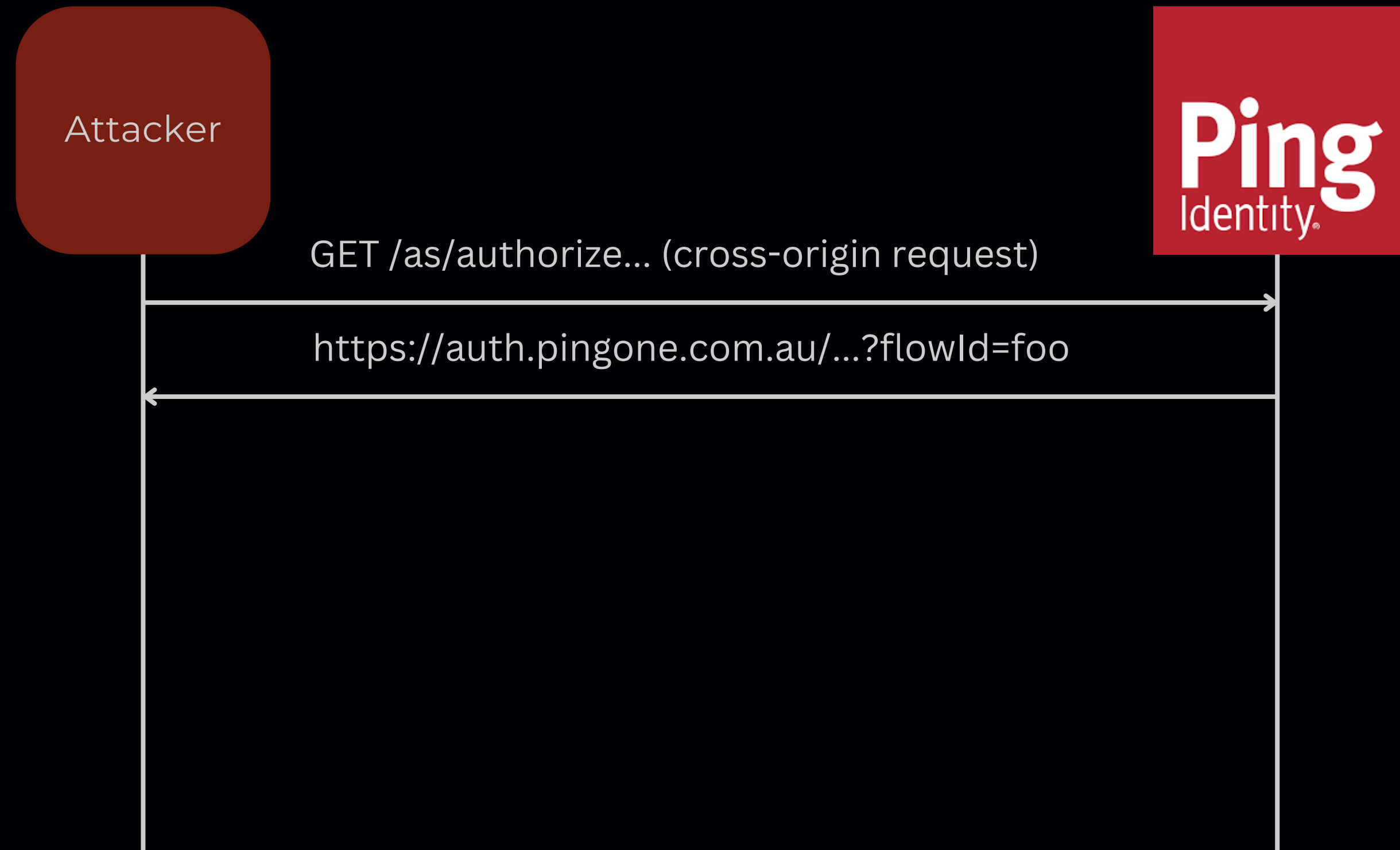
# Custom flows?



# Session fixation?

- Flow identifier acted as a session token
- Session fixation was a big issue in OAuth 1
- Attacker initiates a session then convinces target user into authenticating the session

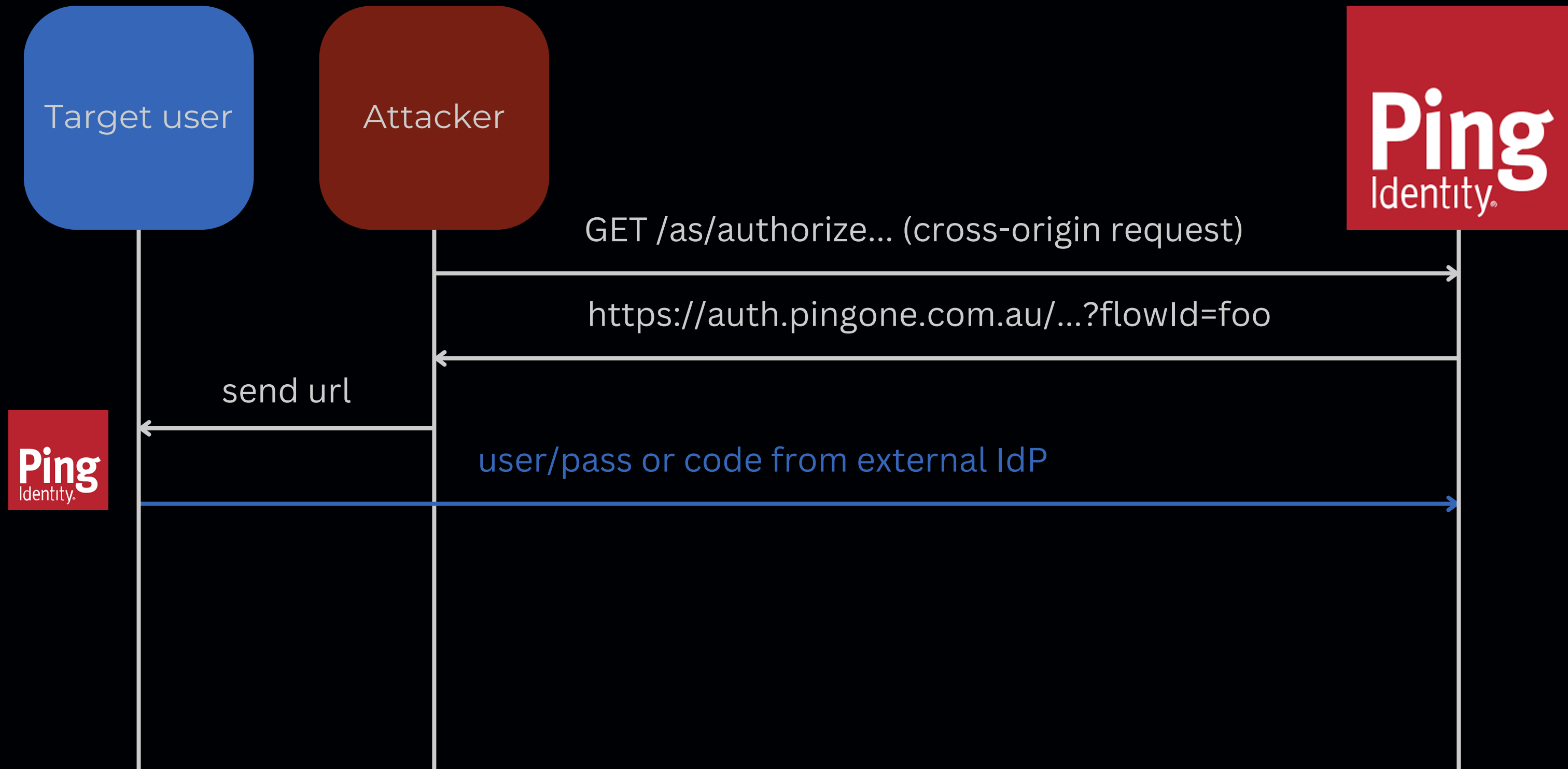
# Attack attempt 1



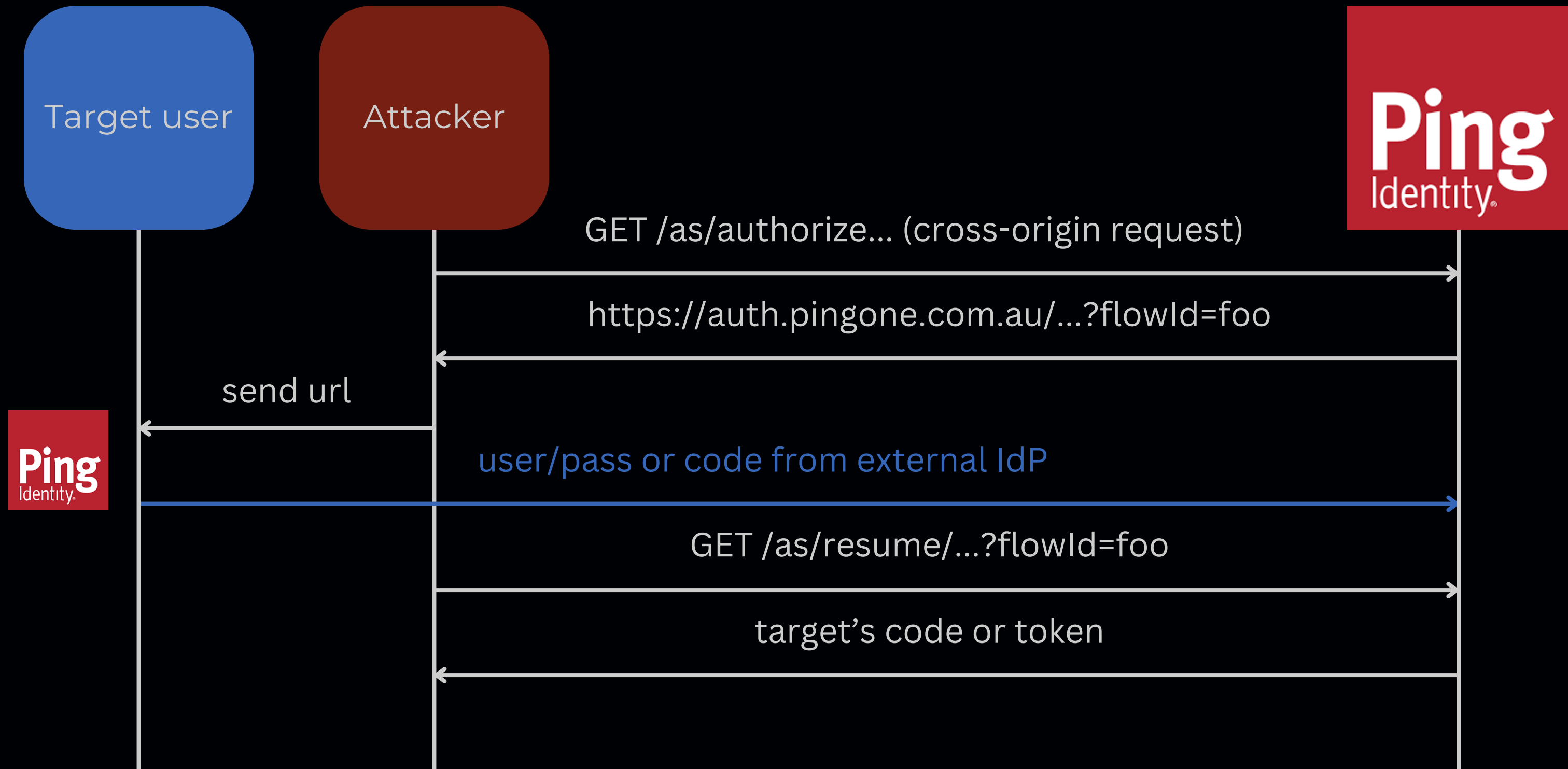
# Attack attempt 1



# Attack attempt 1

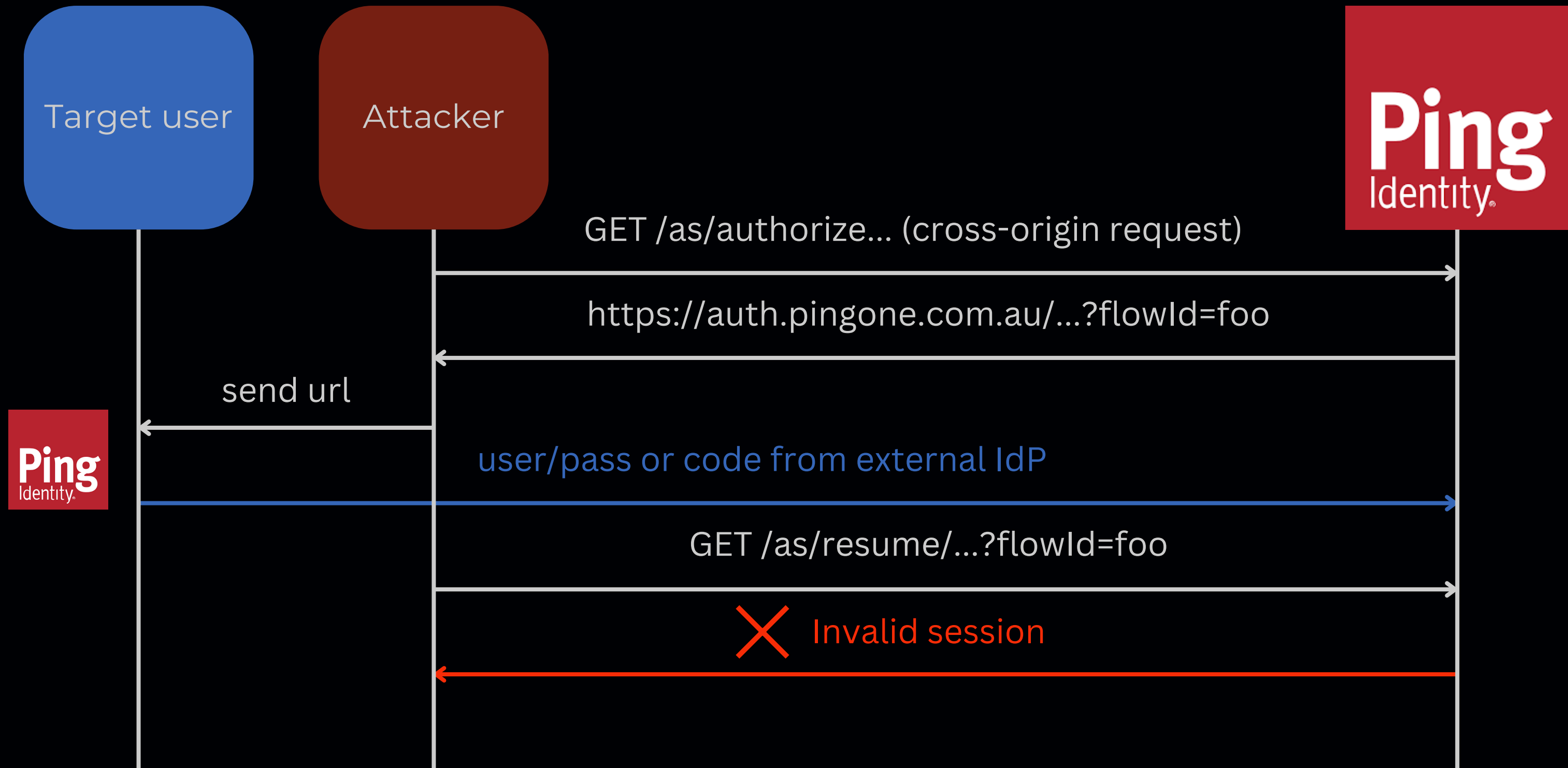


# Attack attempt 1

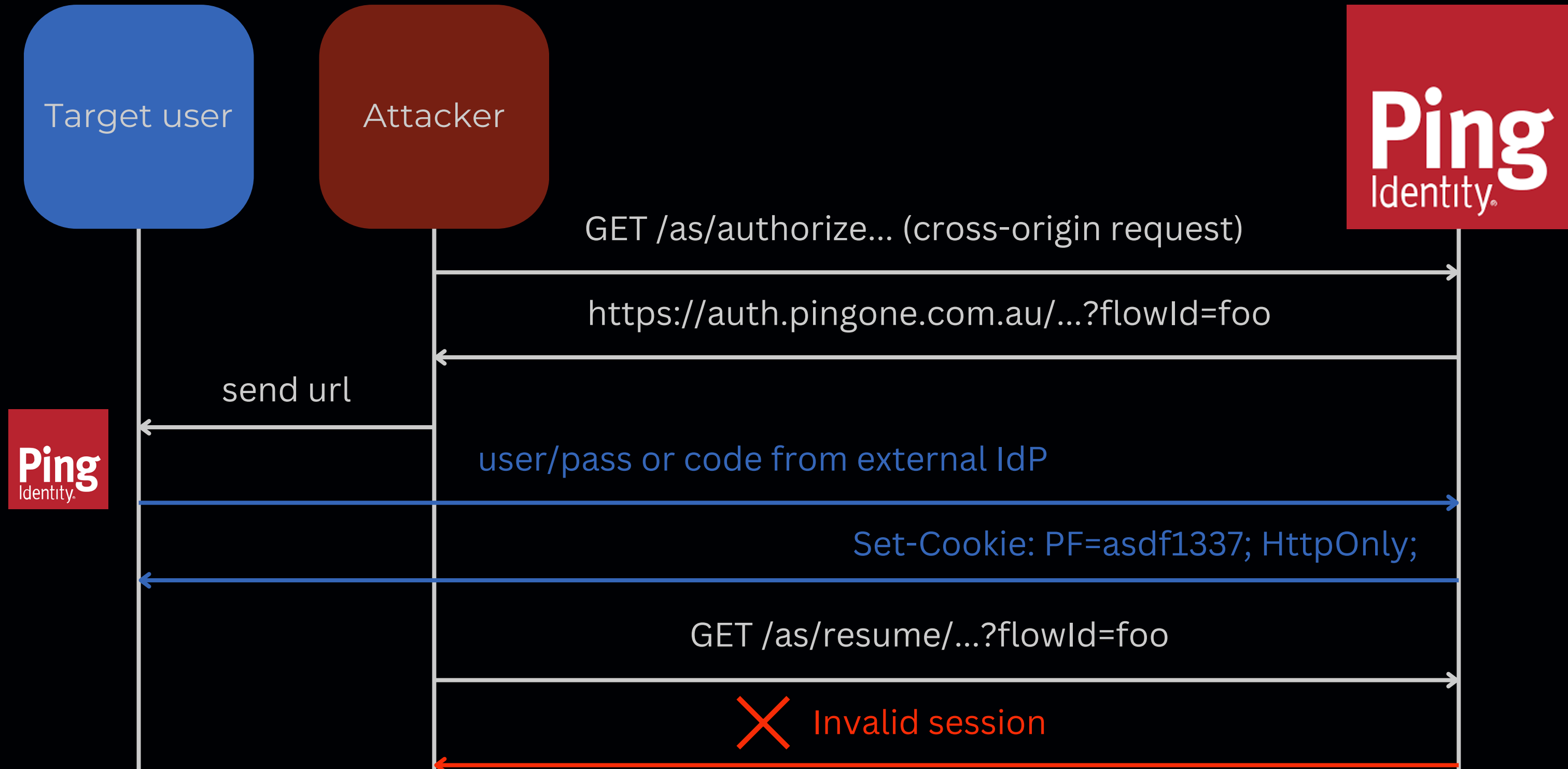




# Attack attempt 1



# Attack attempt 1



# Attack attempt 2

Attacker's origin

<https://auth.pingone.com.au/...?flowId=foo>



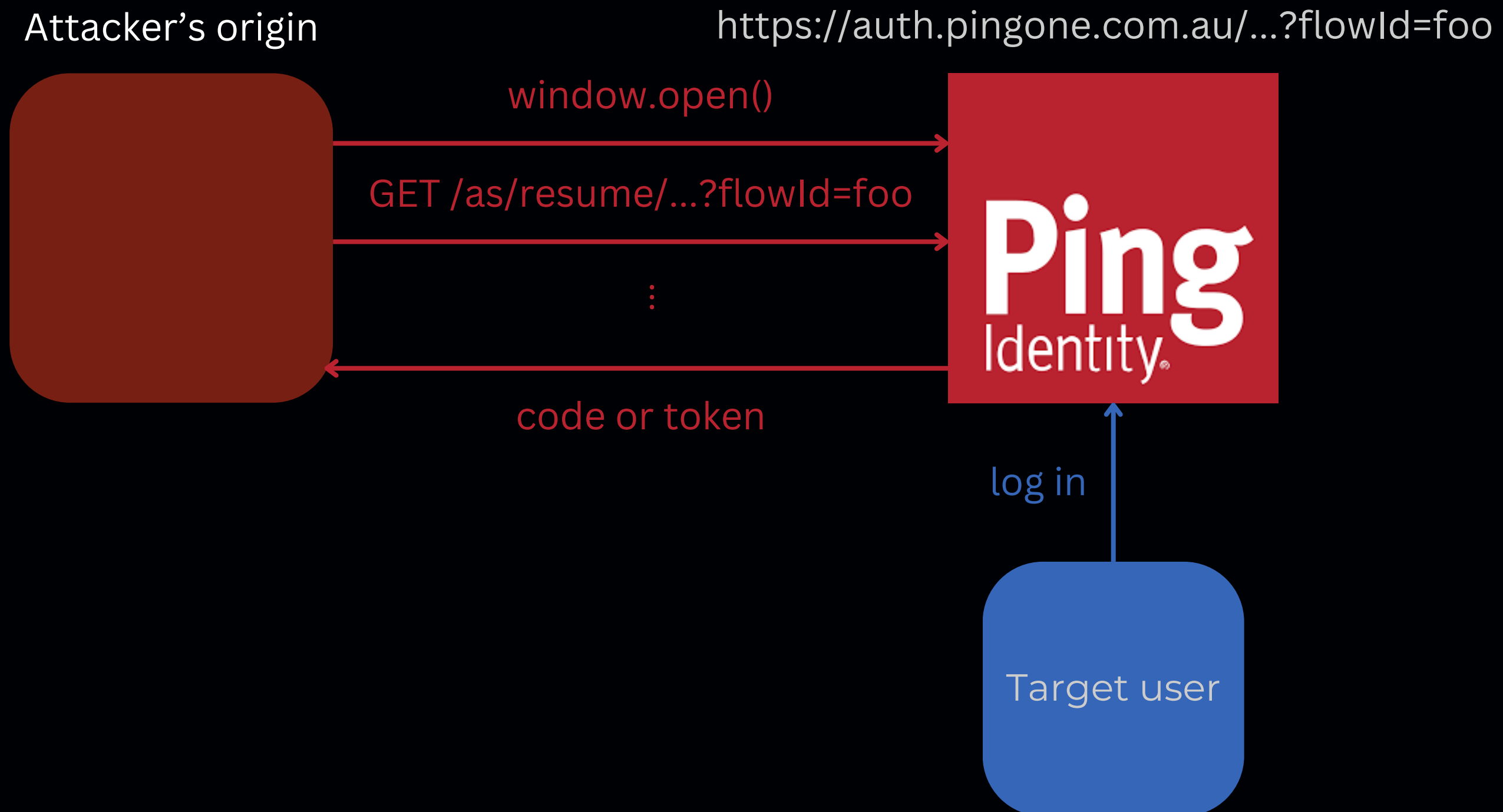
`window.open()`



# Attack attempt 2



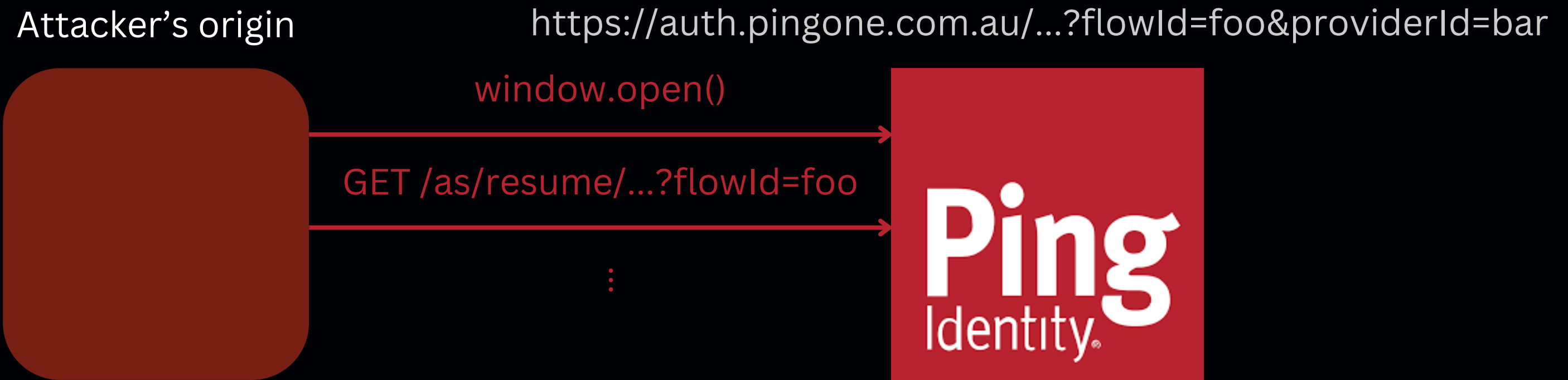
# Attack attempt 2



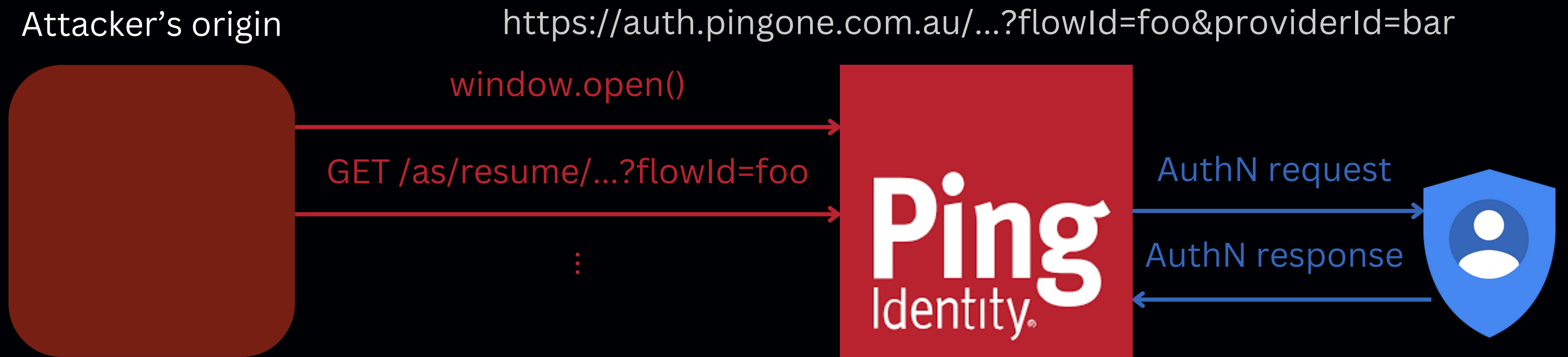
# Attack attempt 2



# Attack attempt 3

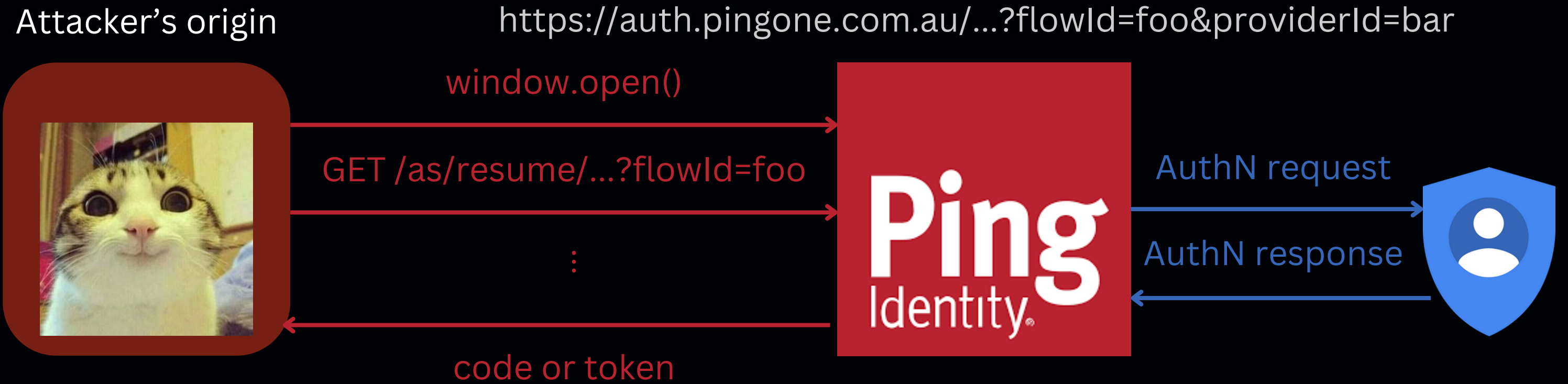


# Attack attempt 3





# Attack attempt 3



# RECAP

- pi.flow is a custom flow defined by Ping identity where code or token is returned within API response instead of redirect uri
- First attack (session fixation) didn't work because of a cookie set by pingone
- Created a website to perform attack with more steps
- Google SSO is your friend

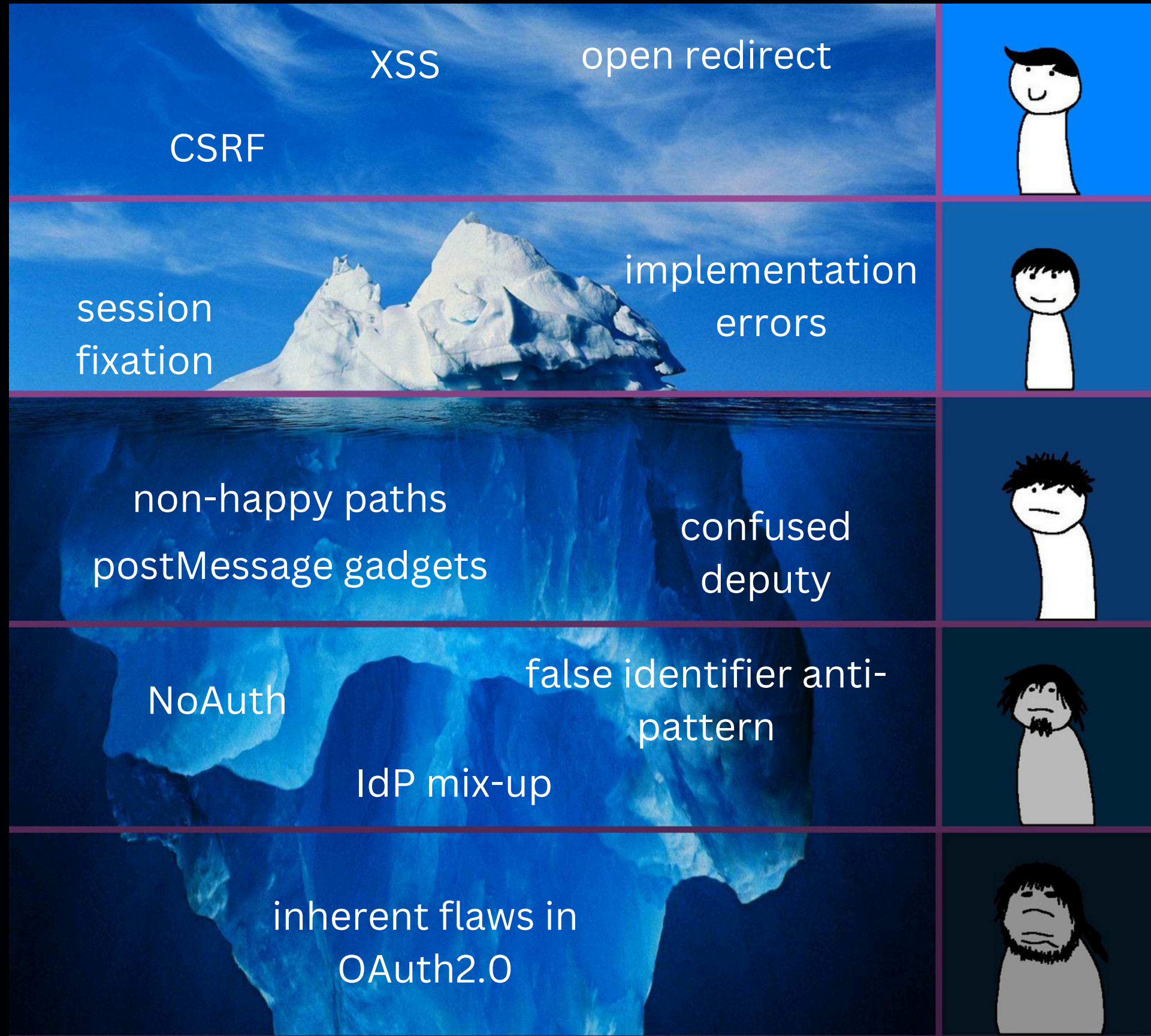
# What is up with that Ping flow

**Note:** A browser-based application that interacts with the authorization endpoint using either `form_post` or `pi.flow` must protect itself against [Cross Site Scripting \(XSS\)](#) and [Cross Site Request Forgery \(CSRF\)](#) vulnerabilities. If a user has already authenticated using the browser, a malicious script running within the browser context that makes an authorization request using `form_post` or `pi.flow` may receive the user's access/id tokens directly in the response. To mitigate this risk, ensure your application follows these recommendations:

1. Disable support for the [OAuth 2.0 Implicit Grant Type](#).
2. Choose a Token Endpoint Authentication Method other than `NONE`.
3. Configure the application's `corsSettings.behavior` property to `ALLOW_SPECIFIC_ORIGINS` and in the `corsSettings.origins` property **only list your application's origins**. For more information, see [Application Operations](#).
4. Ensure that your application is written with [security best practices](#) to prevent script injection.

# **WRAPUP**

- **Learnt something new about OIDC protocol**
- **Go and look for OAuth bugs (fun)**



XSS

open redirect

CSRF



session fixation

implementation errors



non-happy paths  
postMessage gadgets

confused deputy



NoAuth

false identifier anti-pattern

IdP mix-up



inherent flaws in  
OAuth2.0



A series of approximately ten thin, wavy, light blue lines that originate from the top edge of the frame and curve downwards and to the right, creating a sense of motion and depth against the black background.

**THANKS FOR  
LISTENING**

# RESOURCES

- Frans Rosen's blog post: <https://labs.detectify.com/writeups/account-hijacking-using-dirty-dancing-in-sign-in-oauth-flows>
- Salt Security has done a LOT of OAuth research in the past year: <https://salt.security/blog/over-1-million-websites-are-at-risk-of-sensitive-information-leakage---xss-is-dead-long-live-xss>
- NoAuth Lore: <https://techcommunity.microsoft.com/t5/microsoft-entra-blog/the-false-identifier-anti-pattern/ba-p/3846013>
- Egor Homakov the GOAT: <https://homakov.blogspot.com/2014/02/how-i-hacked-github-again.html>
- Yep: <https://datatracker.ietf.org/doc/html/rfc6749#section-10.1>
- The Facebook 1click I mentioned: <https://ysamm.com/?p=763>
- For beginners: <https://portswigger.net/web-security/oauth#improper-implementation-of-the-implicit-grant-type>

# Contact me



@0xchsh



work: [chuanshu@sheasecurity.com.au](mailto:chuanshu@sheasecurity.com.au)  
personal: [jchuanshu@gmail.com](mailto:jchuanshu@gmail.com)

