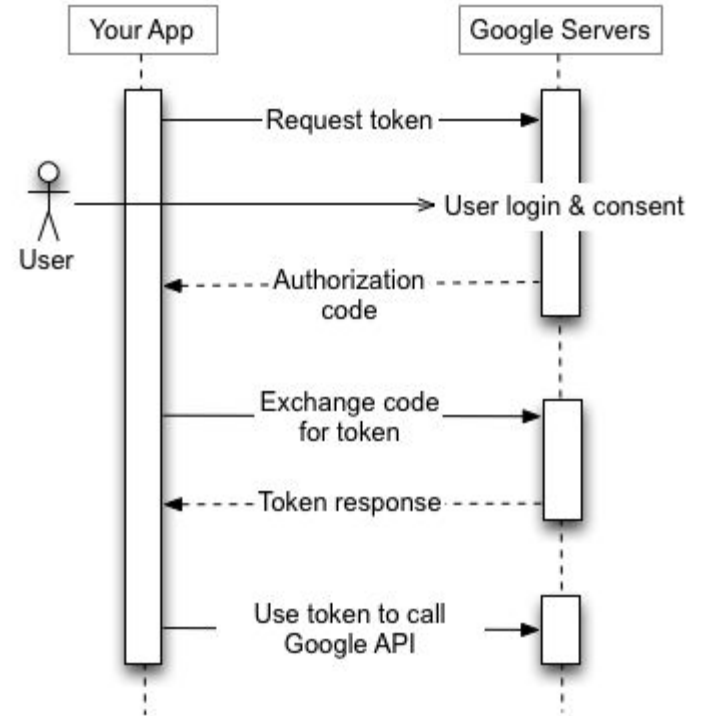


Fantastic OAuth tokens and where to find them

Aditya Saligrama + Glen Husman



Sign in with Google



<https://developers.google.com/identity/protocols/oauth2>

Some Vocabulary & Some Reading

Client Application / Relying Party: App which receives delegated authorized access from another server, or relies on another server for authentication. *e.g. Firebase apps!*

Authorization Server / Identity Provider (IdP): Server to which the user authenticates, and on which user consents to authorizing the app. *e.g. Google!*

OAuth 2.0: Authorization standard. Client apps prompt for authorization, receive an opaque *access token* to access a resource (identified by “scopes”).

OpenID Connect (OIDC): authentication standard built on OAuth 2 – used by (e.g.) Sign In With Google). Extends OAuth to return a verifiable *ID token* making claims (e.g. the user has X email) – in addition to the access token.

This is a summary – there are confusing subtleties!

<https://developer.okta.com/blog/2017/06/21/what-the-heck-is-oauth>
<https://oauth.net/articles/authentication/>

2-legged vs 3-legged

3-legged: user, client app, and authorization (authz) server are all different entities

2-legged: user and client app are 'the same' e.g. JS webpage

Why the difference?

- In 3-legged, client app can hold secrets hidden from (untrusted) users
- Authz server needs to have a way to know it's passing the token back to the real client app, not an imposter client app

2-legged vs 3-legged

How to authenticate the client *application*?

- 3-legged: client secret —> can be used to get token
- 2-legged: authz server will only redirect to defined web origins

What is an origin?

Origin defined as scheme://domain:port

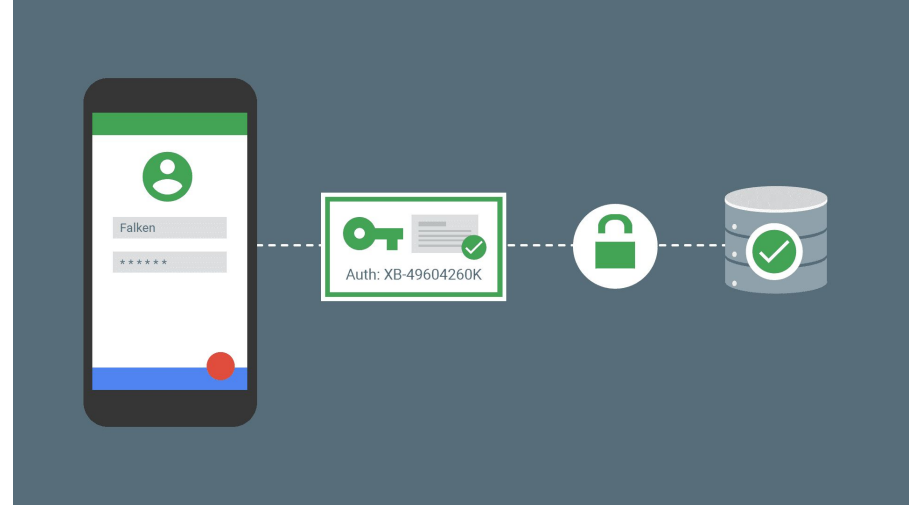
All of these are different origins — *cannot* access one another

- http://stanford.edu
- http://**www**.stanford.edu
- http://stanford.edu:**8080**
- **https**://stanford.edu

These origins are the same — *can* access one another

- http://stanford.edu
- http://stanford.edu:80
- http://stanford.edu/cs

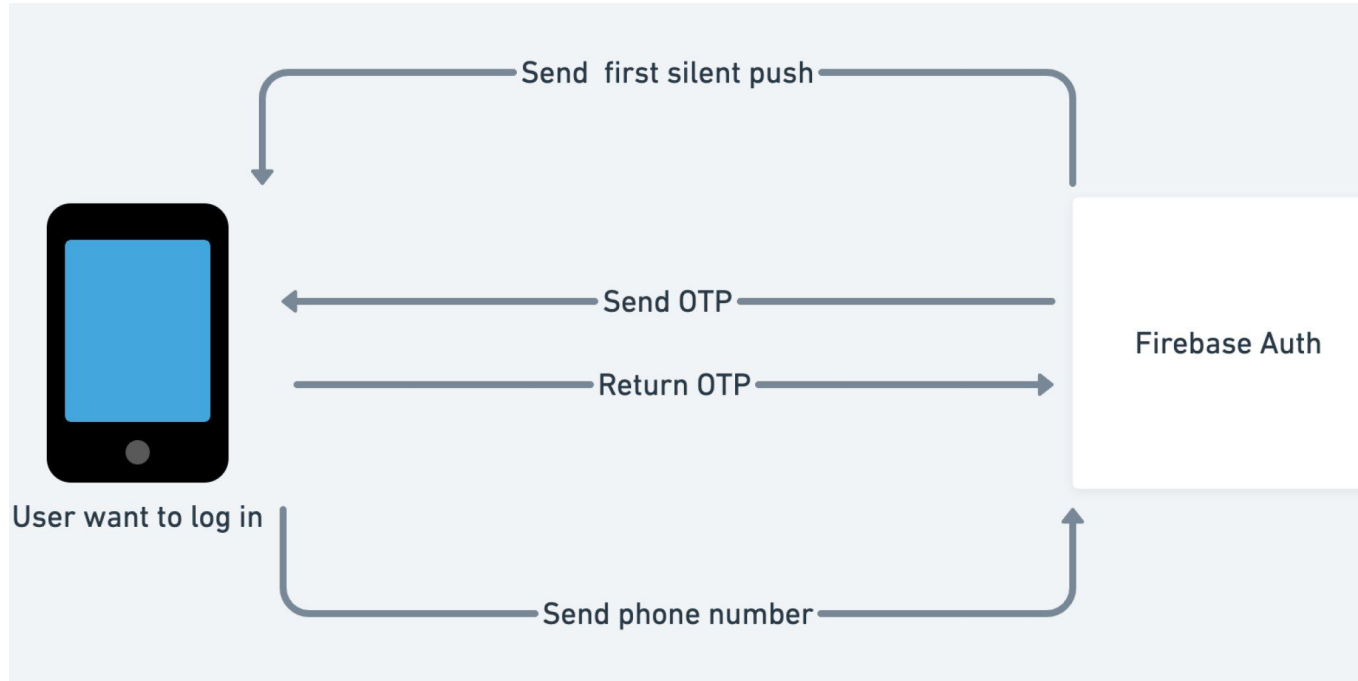
OAuth in practice: Firebase apps



Usually: user/pass auth in Firebase

- Firebase auth: need to pass Firebase instance tokens
 - API key: assigned by Firebase on project creation
 - Project ID
 - Storage Bucket
 - Messaging Sender ID
 - App ID: used by Firebase to ensure only the correct app accesses the project
- Then pass username/password. On successful login:
 - Firebase returns JWT token
 - Transmitted on further requests to the database

Alternatively: phone/OTP auth



Firebase security testing

- With username/password or phone/OTP auth, easy to get a JWT token and have it saved for any requests
- Just use Baserunner!

Log in with email & password

Log in

Log in with phone number (complete CAPTCHA first)

Log in

Firebase config

Set config

Query database

Run

Query templates (replace the text in ==s):

```
[Cloud Firestore] Read collection
[Cloud Firestore] Add to collection
[Cloud Firestore] Modify document in collection
[Cloud Firestore] Delete from collection

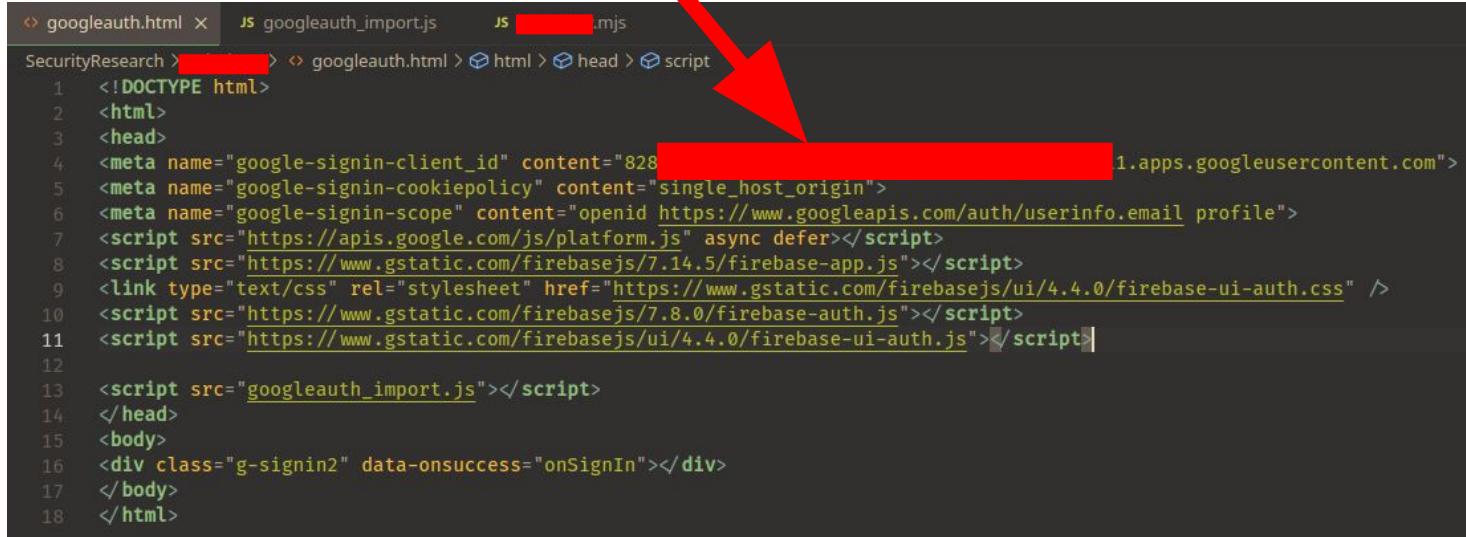
[Realtime Database] Read collection
[Realtime Database] Add to collection
[Realtime Database] Modify document in collection
[Realtime Database] Delete from collection
```

Firestore security testing v2: Google auth only

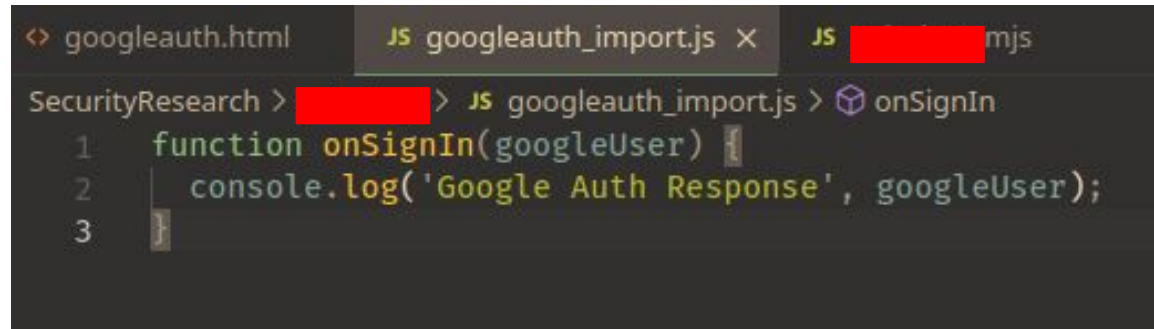
- Baserunner doesn't support Google auth
 - Due to origin restriction on logins
 - <https://github.com/iosiro/baserunner/issues/2>
- Goal: get an OAuth token where we can run:

```
GoogleAuthProvider.credential(id_token);
```

App Google client ID
Can find from "Network" tab in inspect element



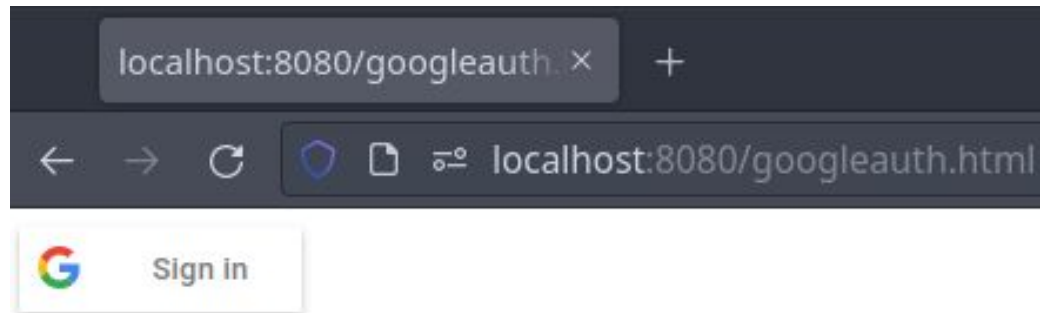
```
<? googleauth.html x JS googleauth_import.js JS [REDACTED].mjs
SecurityResearch > [REDACTED] > googleauth.html > html > head > script
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta name="google-signin-client_id" content="828[REDACTED]1.apps.googleusercontent.com">
5 <meta name="google-signin-cookiepolicy" content="single_host_origin">
6 <meta name="google-signin-scope" content="openid https://www.googleapis.com/auth/userinfo.email profile">
7 <script src="https://apis.google.com/js/platform.js" async defer></script>
8 <script src="https://www.gstatic.com/firebasejs/7.14.5/firebase-app.js"></script>
9 <link type="text/css" rel="stylesheet" href="https://www.gstatic.com/firebasejs/ui/4.4.0/firebase-ui-auth.css" />
10 <script src="https://www.gstatic.com/firebasejs/7.8.0/firebase-auth.js"></script>
11 <script src="https://www.gstatic.com/firebasejs/ui/4.4.0/firebase-ui-auth.js"></script>
12
13 <script src="googleauth_import.js"></script>
14 </head>
15 <body>
16 <div class="g-signin2" data-onsuccess="onSignIn"></div>
17 </body>
18 </html>
```



```
<? googleauth.html JS googleauth_import.js x JS [REDACTED].mjs
SecurityResearch > [REDACTED] > JS googleauth_import.js > onSignIn
1 function onSignIn(googleUser) {
2   console.log('Google Auth Response', googleUser);
3 }
```

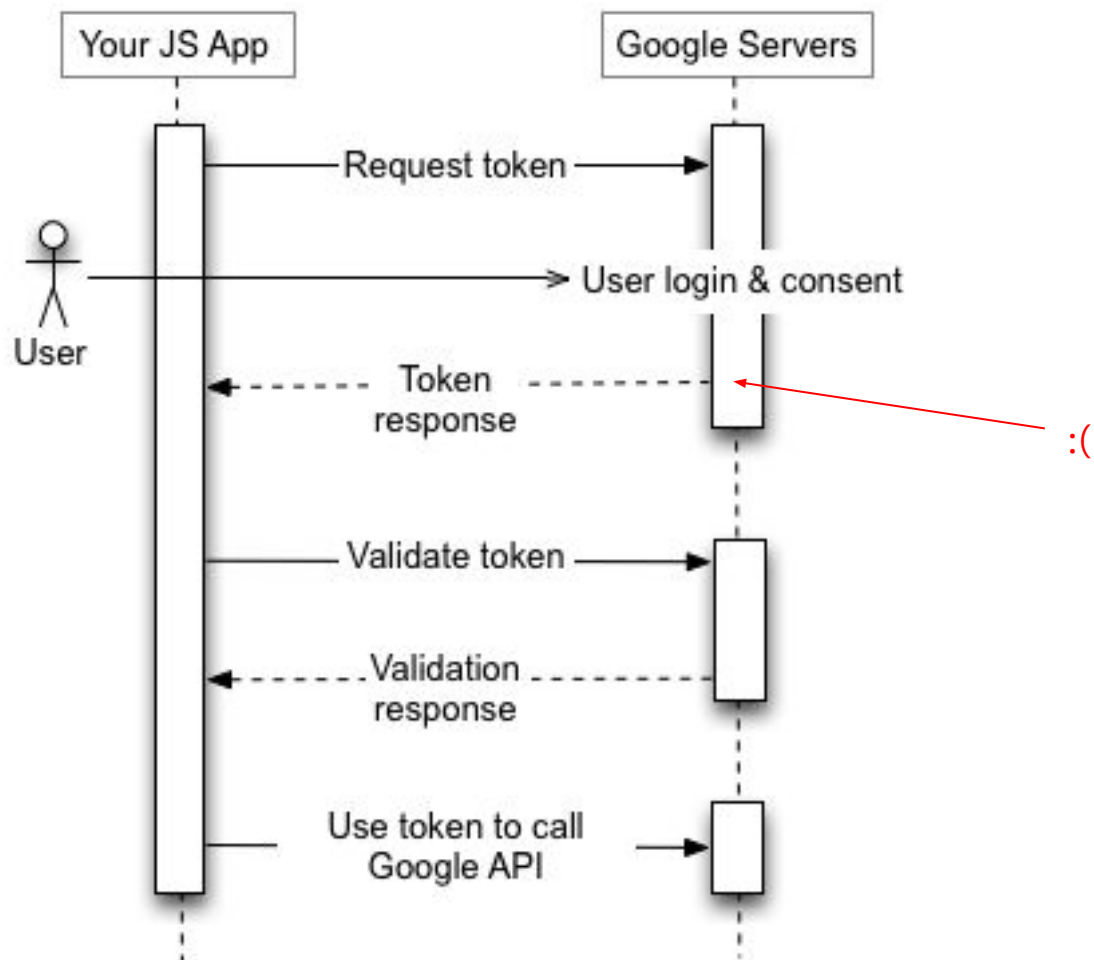
Attempt 1: http://localhost:8080

```
SecurityResearch/ » python3 -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
127.0.0.1 - - [01/May/2022 12:46:39] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [01/May/2022 12:46:39] code 404, message File not found
127.0.0.1 - - [01/May/2022 12:46:39] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [01/May/2022 12:46:41] "GET /googleauth.html HTTP/1.1" 200 -
127.0.0.1 - - [01/May/2022 12:46:41] "GET /googleauth_import.js HTTP/1.1" 200 -
127.0.0.1 - - [01/May/2022 12:47:03] "GET /googleauth.html HTTP/1.1" 200 -
127.0.0.1 - - [01/May/2022 12:47:04] "GET /googleauth_import.js HTTP/1.1" 200 -
127.0.0.1 - - [01/May/2022 12:47:04] code 404, message File not found
127.0.0.1 - - [01/May/2022 12:47:04] "GET /favicon.ico HTTP/1.1" 404 -
```



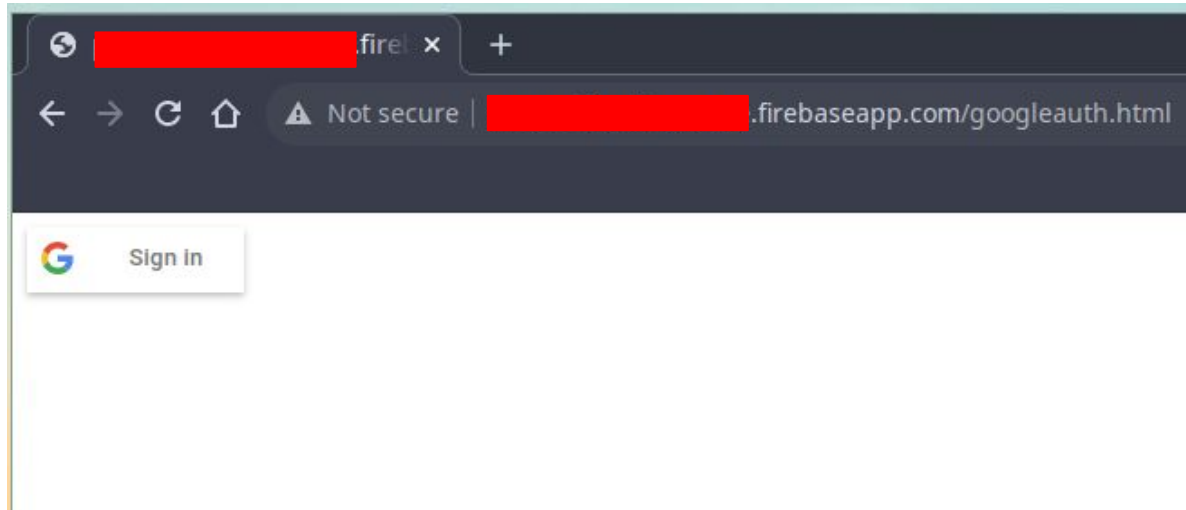
Attempt 1: http://localhost:8080

```
! Uncaught
▼ Object { error: "idpiframe_initialization_failed", details: "Not a valid origin for the client:
http://localhost:8080 has not been registered for client ID
828 [REDACTED]1.apps.googleusercontent.com. Please go to
https://console.developers.google.com/ and register this origin for your project's client ID." }
  details: "Not a valid origin for the client: http://localhost:8080 has not been registered for client ID
828 [REDACTED]1.apps.googleusercontent.com. Please go to
https://console.developers.google.com/ and register this origin for your project's client ID."
  error: "idpiframe_initialization_failed"
  ▶ <prototype>: Object { ... }
```

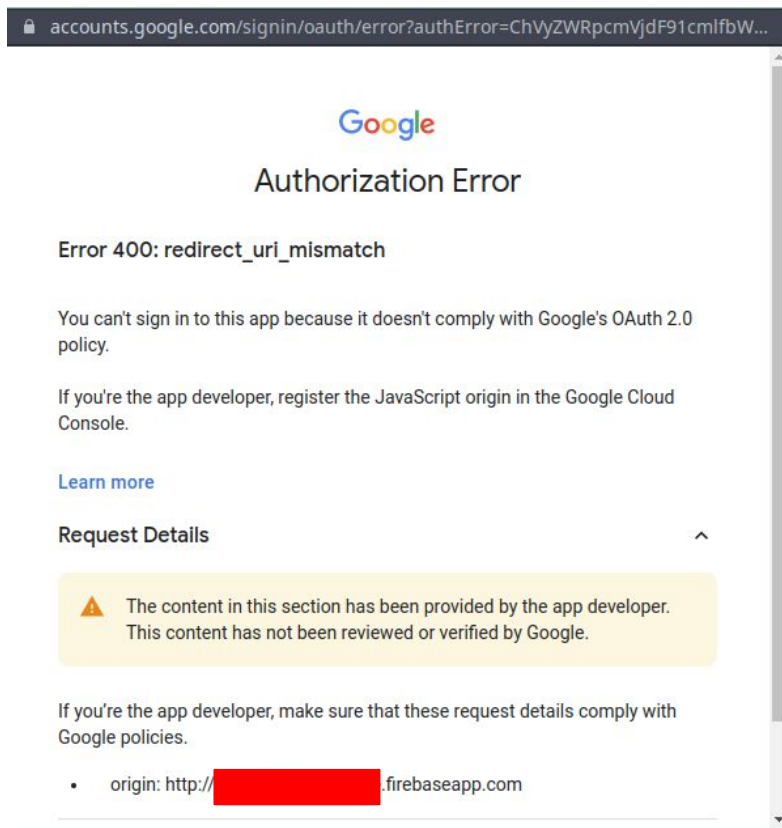


Attempt 2: <http://PROJECTID.firebaseio.com>

```
SecurityResearch/ [REDACTED] » cat /etc/hosts
# Static table lookup for hostnames.
# See hosts(5) for details
127.0.0.1 [REDACTED].firebaseapp.com
127.0.0.1 localhost
::1 localhost
127.0.1.1 ampere.localdomain ampere
```




Attempt 2: http://PROJECTID.firebaseio.com



The screenshot shows a web browser window with the address bar displaying a Google OAuth 2.0 error URL. The page content includes the Google logo, the title 'Authorization Error', and a specific error message: 'Error 400: redirect_uri_mismatch'. It explains that the app cannot sign in because it doesn't comply with Google's OAuth 2.0 policy and advises app developers to register the JavaScript origin in the Google Cloud Console. A 'Learn more' link is provided. Below this is a 'Request Details' section with an expandable arrow. A yellow warning box states that the content is provided by the app developer and has not been reviewed by Google. The details section instructs developers to ensure request details comply with Google policies and lists the origin as 'http://[redacted].firebaseapp.com'.

accounts.google.com/signin/oauth/error?authError=ChVyZWRpcmVjdF91cmIfbW...



Authorization Error


Error 400: redirect_uri_mismatch

You can't sign in to this app because it doesn't comply with Google's OAuth 2.0 policy.

If you're the app developer, register the JavaScript origin in the Google Cloud Console.

[Learn more](#)

Request Details

 The content in this section has been provided by the app developer. This content has not been reviewed or verified by Google.

If you're the app developer, make sure that these request details comply with Google policies.

- origin: http://[redacted].firebaseapp.com

Attempt 3: HTTPS

```
SecurityResearch/ » openssl req -new -x509 -keyout server.pem -out server.pem -days 365 -nodes
Generating a RSA private key
.....+++++
.....+++++
writing new private key to 'server.pem'

-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.

-----
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []: .firebaseapp.com
Email Address []:
```

```
1 # from https://gist.github.com/Alexufo/2303bfff77f0a16ba83568f0260b8cf47
2
3 import http.server
4 import socket
5 import ssl
6 import os
7 server_address = ('0.0.0.0', 443)
8
9 hostname = socket.gethostname()
10 local_ip = socket.gethostbyname(hostname)
11
12 class CORSHandler(http.server.SimpleHTTPRequestHandler):
13     extensions_map = {
14         '.': 'application/octet-stream',
15         '.manifest': 'text/cache-manifest',
16         '.html': 'text/html',
17         '.png': 'image/png',
18         '.jpg': 'image/jpeg',
19         '.svg': 'image/svg+xml',
20         '.css': 'text/css',
21         '.js': 'application/x-javascript',
22         '.wasm': 'application/wasm',
23         '.json': 'application/json',
24         '.xml': 'application/xml',
25     }
26
27     def end_headers(self):
28         # Include additional response headers here. CORS for example:
29         self.send_header('Access-Control-Allow-Origin', '*')
30         self.send_header('Cross-Origin-Opener-Policy', 'same-origin')
31         self.send_header('Cross-Origin-Embedder-Policy', 'require-corp')
32         http.server.SimpleHTTPRequestHandler.end_headers(self)
33
34 httpd = http.server.HTTPServer(server_address, CORSHandler)
35 ctx = ssl.SSLContext(ssl.PROTOCOL_TLS_SERVER)
36 ctx.check_hostname = False
37 ctx.load_cert_chain(certfile='server.pem') # with key inside
38 httpd.socket = ctx.wrap_socket(httpd.socket, server_side=True)
39 httpd.serve_forever()
```

NORMAL simple-https-python-server.py
"simple-https-python-server.py" 39L, 1376B

python utf-8[unix] 5% ln:2/39eq:1

Attempt 3: HTTPS



Your connection is not private

Attackers might be trying to steal your information from [REDACTED]
[REDACTED].firebaseapp.com (for example, passwords, messages, or credit cards). [Learn more](#)

NET::ERR_CERT_AUTHORITY_INVALID

💡 To get Chrome's highest level of security, [turn on enhanced protection](#)

Advanced

Back to safety



This Connection is Not Secure

This does not look like the real [https://\[REDACTED\].firebaseapp.com](#).
Attackers might be trying to steal or alter information going to or from this site.

▼ Technical information

This website's identification was not issued by a trusted organization.

Accept Risk and Proceed

Go Back

Attempt 3: HTTPS

Google Auth Response

googleauth_import.js:2

▼ Lw {Ba: '10[REDACTED]', xc: {...}, Lu: Nw} ⓘ

Ba: "[REDACTED]"

▼ Lu: Nw

Bv: "aditya@saligrama.io"

TW: "109[REDACTED]"

iY: "Aditya"

rN: undefined

tf: "Aditya Saligrama"

wW: "Saligrama"

► [[Prototype]]: Object

▼ xc:

access_token: "[REDACTED]"

expires_at: 1651439302971

expires_in: 3599

first_issued_at: 1651435703971

id_token: "eyJhb[REDACTED]"

idpId: "google"

login_hint: "A[REDACTED]"

scope: "email profile openid https://www.googleapis.com/auth/userinfo.email https://www.googleapis.com/auth/userinfo.profi

► session_state: {extraQueryParams: {...}}

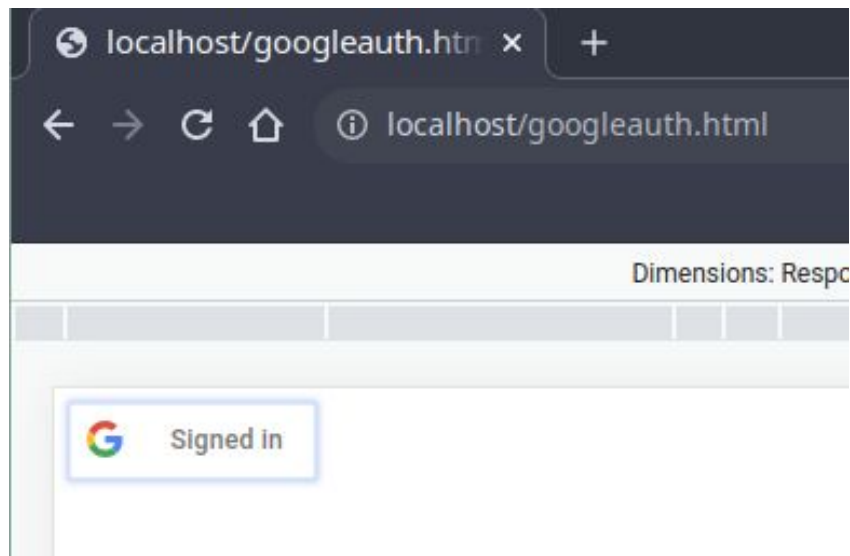
token_type: "Bearer"

► [[Prototype]]: Object

► [[Prototype]]: Object

↩ OAuth JWT token we want

Leaving localhost open as a possible origin



- This is not a security hole!
 - Origin restriction just prevents <https://evil.com> from getting a OAuth token for <https://legitapp.com>
 - If you're running something on <http://localhost>, you have root access to the machine
 - Bigger concerns than just getting an OAuth token