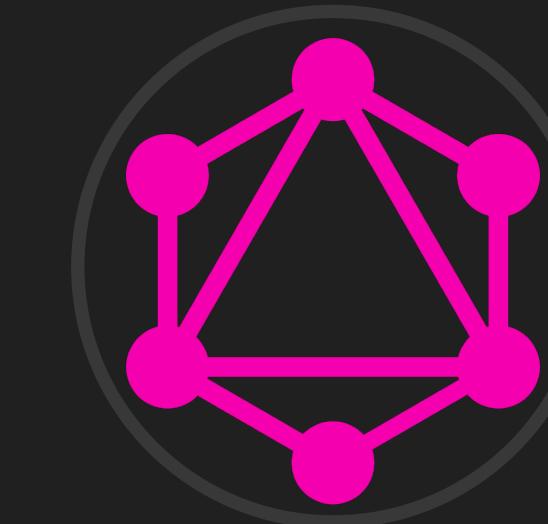
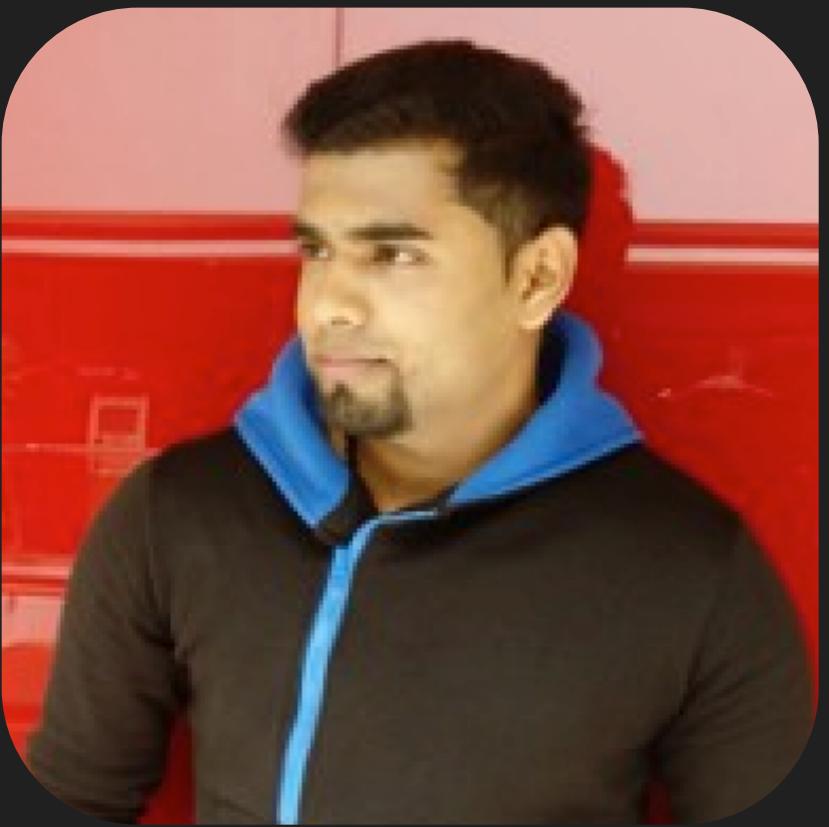


OFFENSIVE GRAPHQL EXPLOITATION

ARUN.S



● whoami



Query whoami {

Arun.S

Senior Security Consultant @ IBM India Software Labs.

Speaker at Conferences and Communities like c0c0n Bsides Delhi, Null/OWASP.

Null Bangalore Chapter Lead, Member of OWASP & BSides Bangalore.

Certified – OSCP, eWPT, ECSA etc.,

}



[Howtographql.com](https://www.howtographql.com)

GraphQL

- GraphQL is open sourced by Facebook.
- GraphQL is a query language for APIs - not databases.
- GraphQL is often confused with being a database technology.
- GraphQL server only exposes a single endpoint.

GraphQL Schema

SDL - GraphQL has its own type system that's used to define the *schema* of an API.

The syntax for writing schemas is called Schema Definition Language (SDL).

Queries

- Used for retrieving data/results.
- Similar to GET in REST.

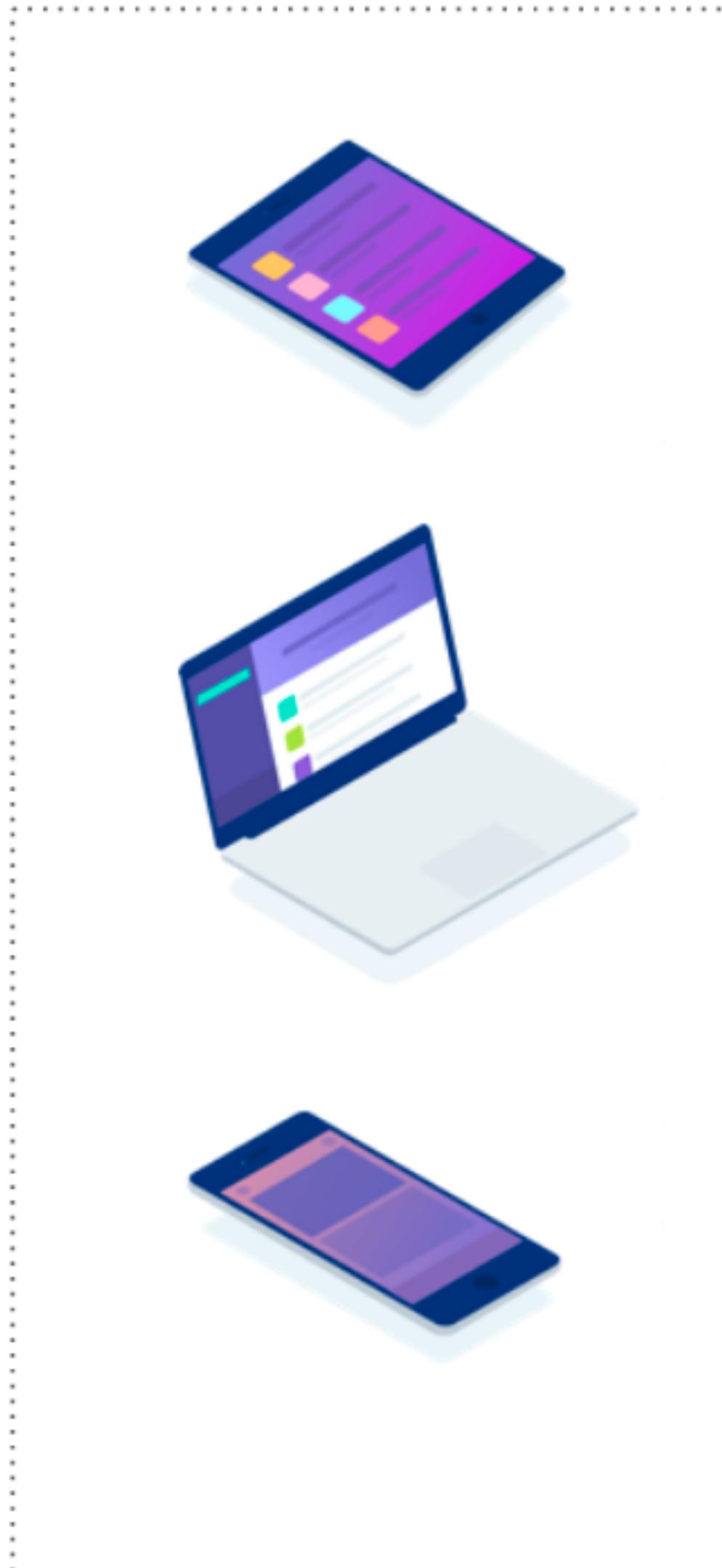
Mutation

- Used for some state changing activities.
- Similar to POST/PUT/DELETE.

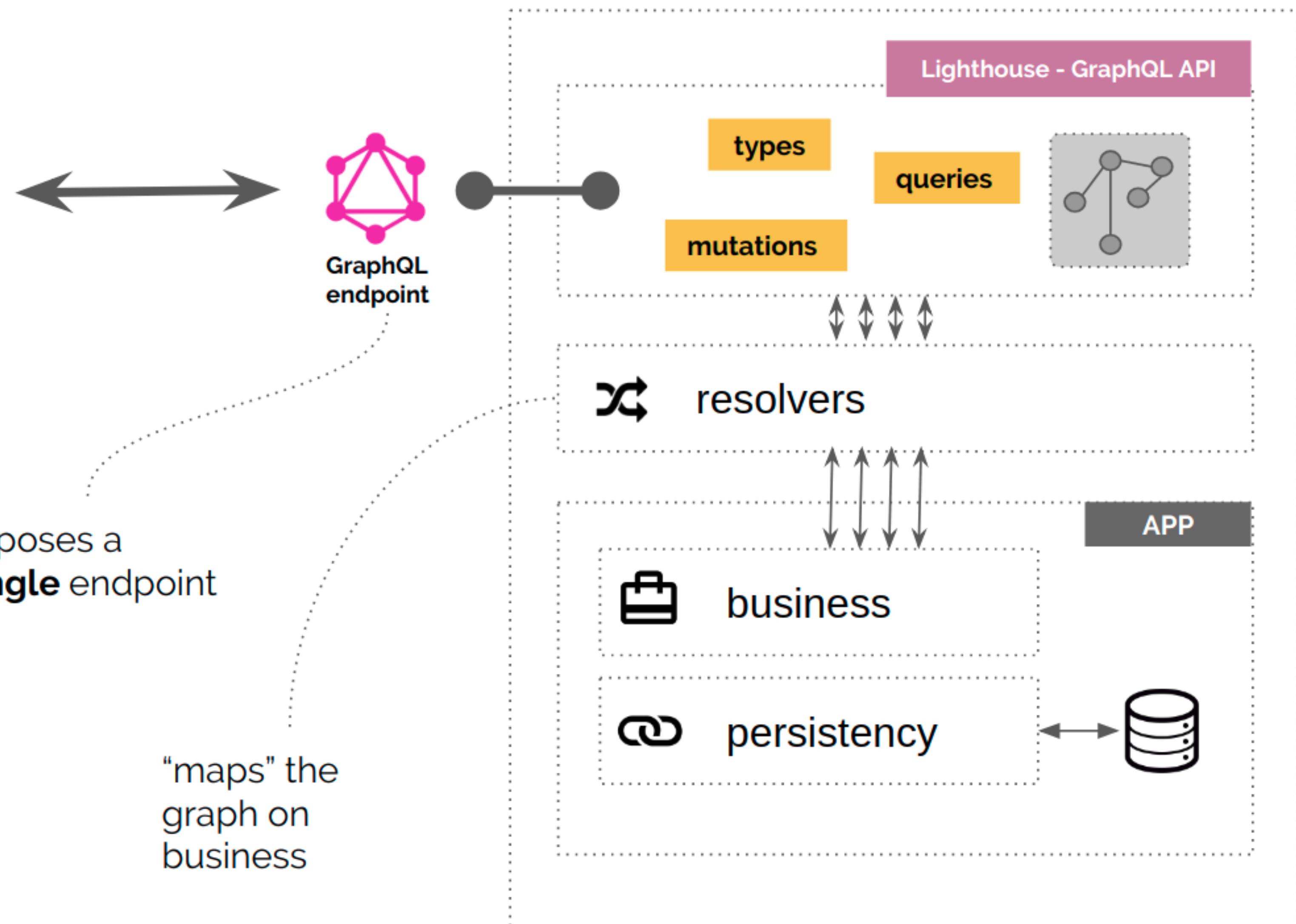
Subscriptions

- Used for Events/Realtime Updates.

frontend



backend



S

STRENGTHS

Strengths

GraphQL Architecture is gaining more popularity in the recent days.
Some of the reasons I can think of are;

- No More Over & UnderFetching.
- Simple & Efficient to Use.
- Evolve APIs without versioning issues.
- Schema Introspection
- Adapts to different requirements for different clients.
- No Wonder that big Facebook, Shopify, Pinterest, HackerOne etc.,

S

STRENGTHS

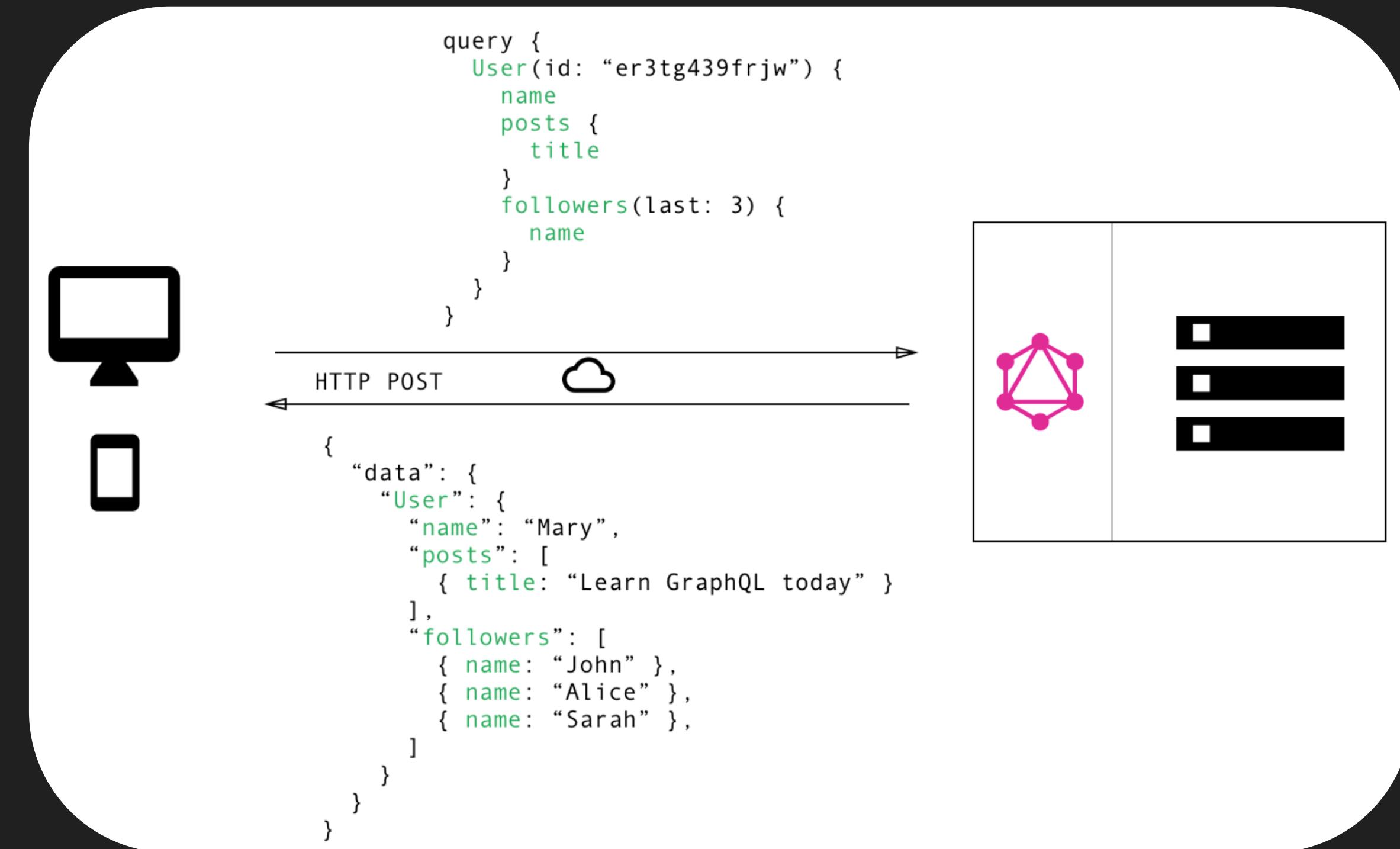
REST



S

STRENGTHS

GraphQL



REST

The screenshot shows the Swagger UI interface for a RESTful API. At the top, there's a dropdown for 'Schemes' set to 'HTTPS'. On the right, there's a green 'Authorize' button. Below the header, there are two main sections: 'pet' and 'store'. The 'pet' section contains endpoints for uploading images, adding pets, updating existing pets, finding pets by status or tags, finding a pet by ID, updating a pet in the store, and deleting a pet. The 'store' section contains endpoints for placing an order for a pet, finding a purchase order by ID, deleting a purchase order by ID, and returning pet inventories by status. Each endpoint is shown with its method (e.g., POST, GET, PUT, DELETE), URL, and a brief description.

STRENGTHS

GraphQL

The screenshot shows the GraphQL playground interface. On the left, there's a code editor window titled 'GraphQL' containing a query for 'allFilms':

```
1 {  
2   allFilms {  
3     films {  
4       title  
5       openingCrawl  
6     }  
7   }  
8 }
```

On the right, the results pane displays the JSON response:

```
{  
  "data": {  
    "allFilms": {  
      "films": [  
        {  
          "title": "A New Hope",  
          "openingCrawl": "It is a period of...  
        },  
        {  
          "title": "The Empire Strikes Back",  
          "openingCrawl": "It is a dark time...  
        },  
        {  
          "title": "Return of the Jedi",  
          "openingCrawl": "Luke Skywalker has...  
        },  
        {  
          "title": "The Phantom Menace",  
          "openingCrawl": "Turmoil has engulfed...  
        },  
        {  
          "title": "Attack of the Clones",  
          "openingCrawl": "A long time ago in a galaxy far, far away...  
        }  
      ]  
    }  
  }  
}
```

Below the results, there's a 'Documentation Explorer' sidebar with a search bar and a note about root types. It also shows the 'query' field with the text 'query: Root'.



W

WEAKNESSES

Weaknesses

Every technology has some disadvantages, and Graphql do have few disadvantages.

- Query Complexity & Depths.
- Rate Limiting & Timeouts.
- Caching.

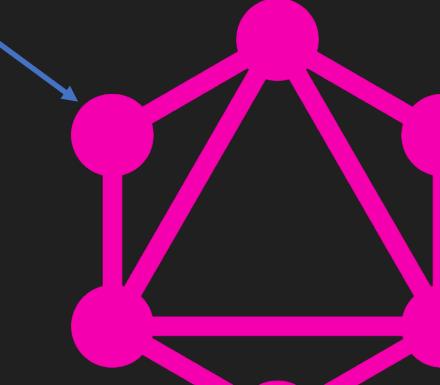
Opportunity for Hackers

- There is a wide scope for the pentesters & bug bounty hunters on GraphQL.
- The GraphQL Endpoints like /graphql or /graphiql are still publicly available out there due missing security best practices.

OPPORTUNITIES



Endpoint
Fuzzing



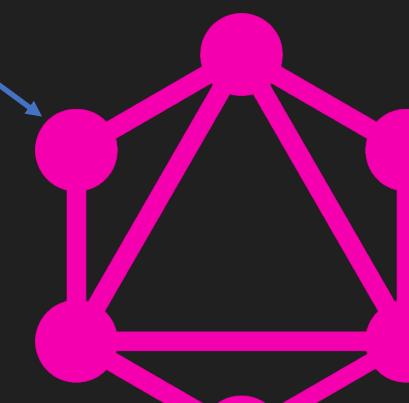
queries/
Mutation
via
schema

&debug=1



Crawl
WebApp

Parse JS
Files



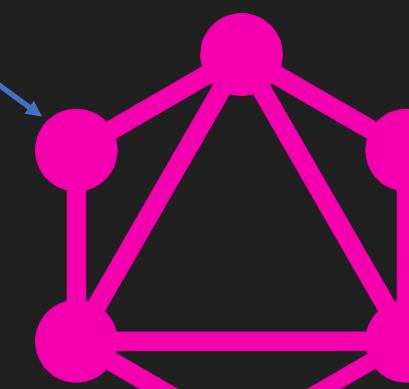
Build
Queries &
Mutations

O

LOOK FOR THE
AUTHENTICATION

JWT

ACL

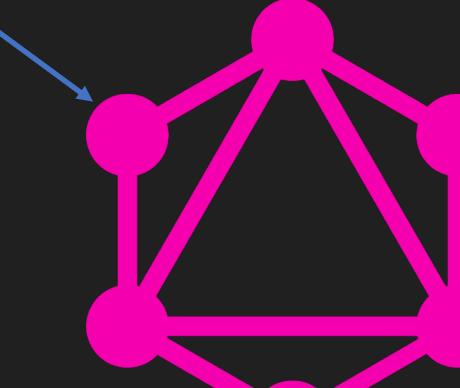


No-
Auth/Pro-
tections
on
Resolvers



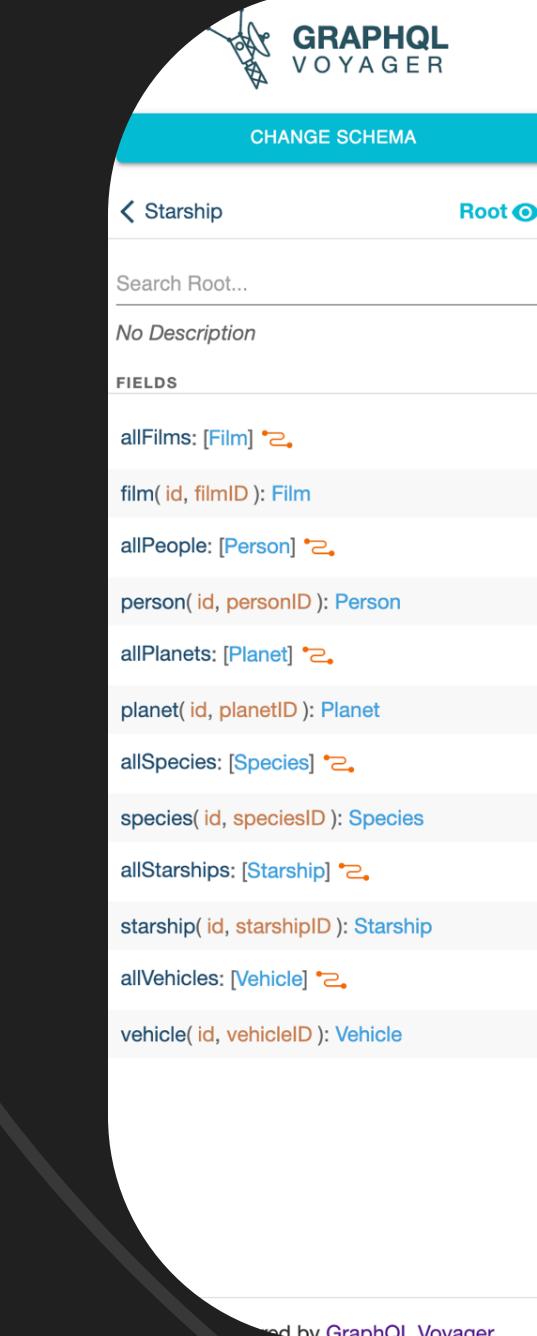
INQL

GRAPHQL
VOYAGER



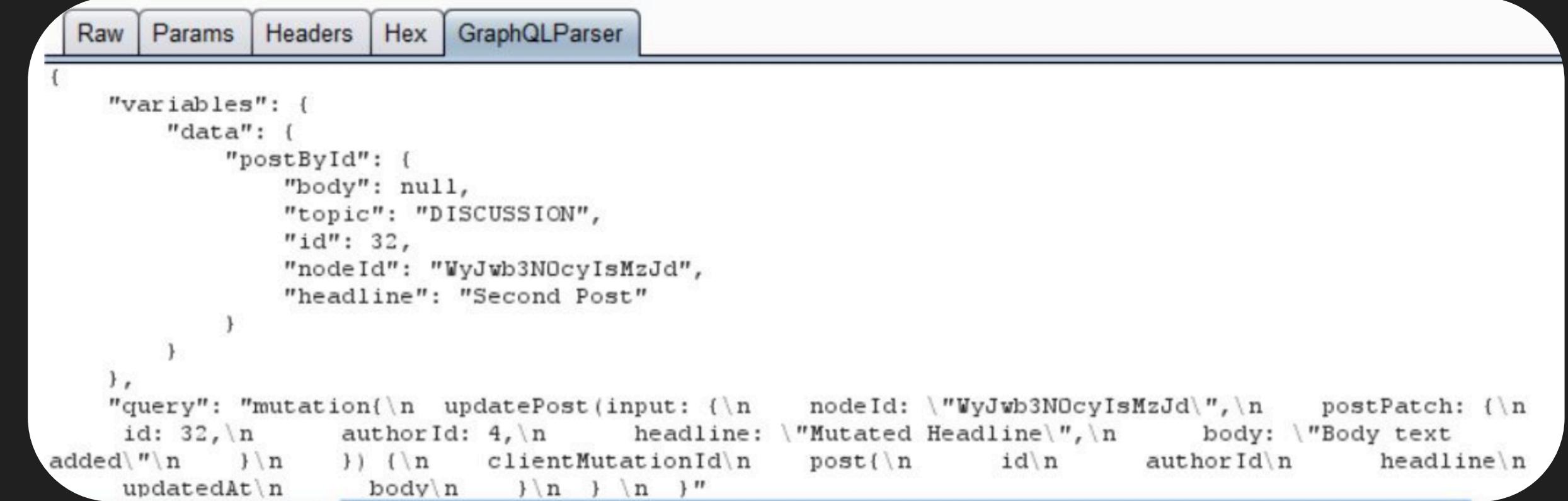
GQL
PARSER

TOOLS TO RESCUE



TOOLS TO RESCUE

O



The screenshot shows a GraphQL playground interface with the following tabs at the top: Raw, Params, Headers, Hex, and GraphQLParser. The GraphQLParser tab is selected. Below the tabs, there is a JSON-like configuration for a mutation:

```
{  
  "variables": {  
    "data": {  
      "postById": {  
        "body": null,  
        "topic": "DISCUSSION",  
        "id": 32,  
        "nodeId": "WyJwb3N0cyIsMzJd",  
        "headline": "Second Post"  
      }  
    }  
  },  
  "query": "mutation\n  updatePost(input: {\n    nodeId: \"WyJwb3N0cyIsMzJd\",  
    postPatch: {\n      id: 32,  
      authorId: 4,  
      headline: \"Mutated Headline\",  
      body: \"Body text added\"\n    }  
  }) {\n    clientMutationId  
    post {\n      id  
      authorId  
      headline  
      updatedAt  
      body  
    }  
  }  
"}  
The JSON object contains variables and a query. The variables object has a single entry for 'data' which points to a 'postById' object. The query is a mutation named 'updatePost' that takes an 'input' argument. The 'input' argument includes a 'nodeId' field set to 'WyJwb3N0cyIsMzJd' and a 'postPatch' field containing an object with 'id', 'authorId', 'headline', and 'body' fields. The mutation returns 'clientMutationId' and a 'post' object with fields 'id', 'authorId', 'headline', and 'updatedAt'.
```

GQL
PARSER

O

TOOLS TO RESCUE

INQL

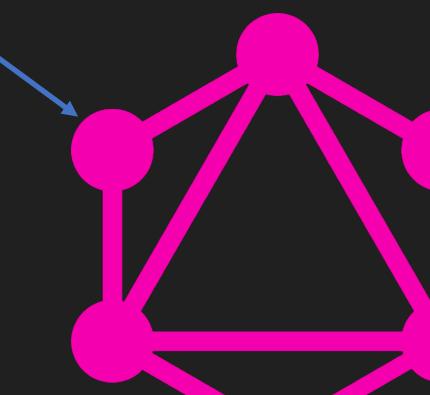




GQL
Goat

Pentester
Lab

Digi
Ninja
Lab





Threats!!!

GraphQL gives enormous power to clients.
But with great power come great responsibilities !!!





Vulnerabilities!!!

Most of all vulnerabilities related to REST APIs & WebApp
are applicable for GraphQL as well.



-  SQL Injection
-  NoSQL Injection
-  Access Control Related Issues.
-  Mass Assignment
-  IDOR
-  Bypassing 2FA/BruteForce Attacks.
-  DOS Attacks etc.,

STOP TALKING



SHOW ME A DEMO!!!!

Threats!!!

GraphQL is great, because it gives client so much more power. But if not used properly it could cost them more they can imagine.



Three screenshots from the HackerOne platform showing GraphQL-related vulnerabilities:

- RyotaK (ryotak) - #885539**: Private list members disclosure via GraphQL. State: Resolved (Closed). Disclosed: August 4, 2020 6:55am +0530. Reported To: Twitter. Asset: *twitter.com (Domain). Weakness: Improper Access Control - Generic. Bounty: \$2,940. Severity: Low (0.1 ~ 3.9). Participants: 4. Signal: 3.96 (82nd Percentile). Impact: 12.67 (78th Percentile).
- Hiffley (hiffley) - #898528**: GraphQL AdminGenerateSessionPayload is leaked to staff with no permission. State: Resolved (Closed). Disclosed: July 16, 2020 2:44pm +0530. Reported To: Shopify. Asset: your-store.myshopify.com (Domain). Weakness: None. Bounty: \$2,000. Severity: High (7 ~ 8.9). Participants: 10. Signal: -1.97 (40th Percentile). Impact: 18.33 (88th Percentile).
- Yash Sodha (yashrs) - #489146**: Confidential data of users and limited metadata of programs and reports accessible via GraphQL. State: Resolved (Closed). Disclosed: February 3, 2019 4:27pm +0530. Reported To: HackerOne. Asset: https://hackerone.com (Domain). Weakness: Information Disclosure. Bounty: \$20,000. Severity: Critical (9.3). Participants: 10. Signal: 5.83 (92nd Percentile). Impact: 21.33 (92nd Percentile).



Threats!!!

There are many approaches to secure your GraphQL server against these queries, but none of them is bullet proof. It's important to know what options are available and know their limits and take best decisions out of it!.

- Disable well known `/graphql` & `/graphiql` endpoints from the domain.
- Craft your own schema and avoid using autogenerated SDL which creates queries and mutations.
- Input validation, Authentication & Authorization should never be forgotten.
- Never trust user inputs.
- Enforce a limit on maximum query depth($\sim<10$), Query complexity, timeouts & enabled throttling based on the server time & query complexity.



References



Labs:

- <https://github.com/CarveSystems/vulnerable-graphql-api>
- <https://graphqlab.digi.ninja/>
- <https://pentesterlab.com/exercises/graphql/online>
- <https://www.hackerone.com/blog/graphql-week-hacker101-capture-flag-challenges>

Tools:

- <https://github.com/doyensec/inql>
- <https://github.com/br3akp0int/GQLParser>

Credits



Slide Template Credit:

<https://hislide.io/>

Demo App Credit:

<https://carvesystems.com/>

Thank you