

FWDSEC»

We Taught Burp How to Speak GraphQL

Jared Meit

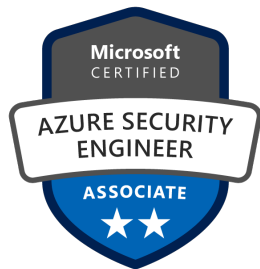
2022-09-27 | v0.1



Jared Meit



- Senior Application Security Consultant
- 16 years of web API design and development
- 5 years in security professionally + decades very unprofessionally
- Designations & Certifications: OSWE, Azure Security Engineer

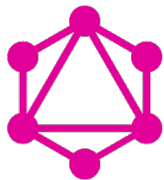


Quick Poll:

1. Who has heard of GraphQL?
2. Who uses Burp Suite?
3. Who has pentested GraphQL?
4. Who has pentested GraphQL with Burp?

- Intro (Already done)
- What is GraphQL
- GraphQL weaknesses + attacks
- Current security tools landscape
- Overview of my tool
- Demo
- Q&A

What even, like, is GraphQL?



GraphQL

“A query language for your API”

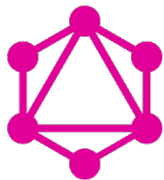
- ✓ Body format
- ✓ Reduces calls to API
- ✓ Reduces endpoints (sometimes to 1)

Request

```
{  
  hero {  
    name  
    height  
    mass  
  }  
}
```

Response

```
{  
  "hero": {  
    "name": "Luke Skywalker",  
    "height": 1.72,  
    "mass": 77  
  }  
}
```



GraphQL

“A query language for your API”

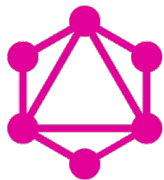
- ✓ Body format
- ✓ Reduces calls to API
- ✓ Reduces endpoints (sometimes to 1)
- ✓ Just as vulnerable as regular REST
- ✓ Bonus DoS vulns
- ✓ It self-documents (Introspection)

Request

```
{  
  hero {  
    name  
    height  
    mass  
  }  
}
```

Response

```
{  
  "hero": {  
    "name": "Luke Skywalker",  
    "height": 1.72,  
    "mass": 77  
  }  
}
```



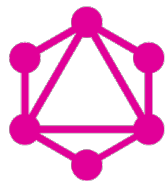
GraphQL

In your application code:

- Import GraphQL library
- Define GraphQL schema
- Define “resolvers” that respond to the queries

Request Types:

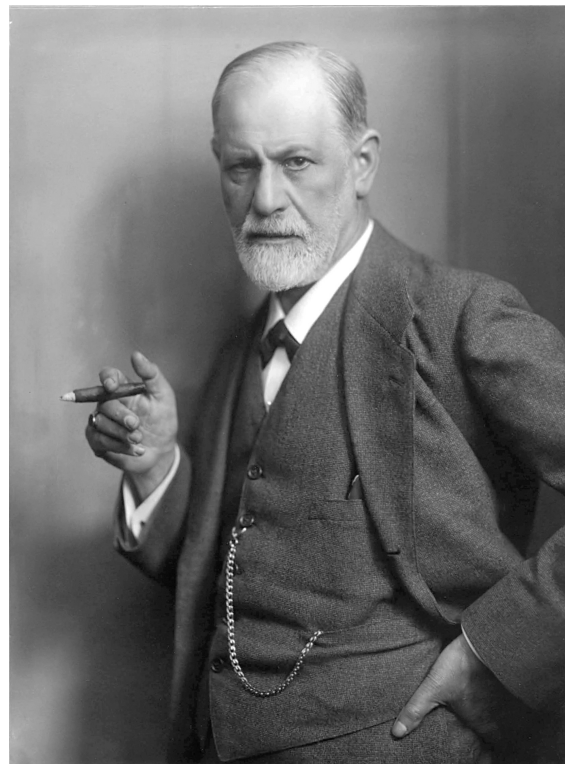
- Query
- Mutation
- Subscription



GraphQL

Introspection

- ▣ Special query
- ▣ Fetches entire schema
- ▣ Useful for frontend
- ▣ Vulnerability?



What is GraphQL?



Safe Injection Sites

Ohhhhhhhhhhh... It takes params.

- Standard and custom data types


```
{  
  human(id: "1000") {  
    name  
    height(unit: FOOT)  
  }  
}
```

```
{  
  "data": {  
    "human": {  
      "name": "Luke Skywalker",  
      "height": 5.6430448  
    }  
  }  
}
```

Old payloads

New locations

```
' UNION ALL SELECT LOAD_FILE('/etc/passwd') --
```



```
{  
  human(id: " ") {  
    name  
    height(unit: FOOT)  
  }  
}
```

```
{  
  "data": {  
    "human": {  
      "name": "root:*:0:0:System Administrator:/var/root:  
      "height": 0  
    }  
  }  
}
```

Beautified:

```
{  
  human(id: "1000") {  
    name  
    height(unit: FOOT)  
  }  
}
```

```
{  
  "data": {  
    "human": {  
      "name": "Luke Skywalker",  
      "height": 5.6430448  
    }  
  }  
}
```

Actual request body:

1. {"query": "query {\n\tthuman(id:\n\t\t\"1000\")\n\t\t{\n\t\t\tname\n\t\t\theight(unit:FOOT)\n\t\t}\n}"}

Burp is like:



Burp

```
query getPastes {  
  pastes(public:true) {  
    id  
    title  
    content  
    ipAddr  
    userAgent  
    owner {  
      name  
    }  
  }  
}' and 3778=3778'--
```

IS THIS AN INJECTION POINT?

**Gently coaching Burp without being
condescending**

Goals:

- Burp extension
- Auto find every possible request
- Leverage Active Scan

Solution

The screenshot displays the Burp Suite interface. The top-left panel shows '3. Extension driven active audit' with a 'Capturing' toggle switch and a summary of 'Issues: 6 (3 errors)' and '3319 requests (61 errors)'. The top-right panel lists various issues found, including 'Cross-site scripting (reflected)', 'OS command injection', 'SQL injection', and 'Unencrypted communications'. The bottom-right panel shows the details of 'Request 2', which is a POST request to '/graphql HTTP/1.1' with a 'Host: localhost:5013' and various headers. The request body is a JSON object containing a 'query' field with a payload designed to trigger a system debug message.

3. Extension driven active audit

Capturing: ☒

Issues: 6 (3 errors) 3319 requests (61 errors) View details >>

Issues found:

- Cross-site scripting (reflected)
- Cross-site scripting (reflected)
- OS command injection
- Cross-site scripting (reflected)
- SQL injection
- Input returned in response (reflected)
- Private IP addresses disclosed
- Unencrypted communications

Request 2

Pretty Raw Hex

```
1 POST /graphql HTTP/1.1
2 Host: localhost:5013
3 sec-ch-ua: "Chromium";v="105", "Not)A;Brand";v="8"
4 Accept: application/json
5 Content-Type: application/json
6 sec-ch-ua-mobile: ?0
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
  like Gecko) Chrome/105.0.5195.102 Safari/537.36
8 sec-ch-ua-platform: "macOS"
9 Origin: http://localhost:5013/graphql
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: cors
12 Sec-Fetch-Dest: empty
13 Accept-Encoding: gzip, deflate
14 Accept-Language: en-US,en;q=0.9
15 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss;
  continueCode=E30zQenePwoj4zk293aRX8KbBNYEa09GL5q01ZDwp6JyVxgQMmrLv7npKLvy; env=
  graphql:disable
16 Connection: close
17 Content-Length: 53
18
19 {
  "query": "query {\n\tsystemDebug(arg: \"code*&echo ad75ehbxrr r7ydx0jubu&\\")\n}"
}
```

1 highlight

3.1MB Disk (project): 1.0MB Disk (temp): 46.6MB

Goals:

- Burp extension
- Auto find every possible request
- Leverage Active Scan

Solution

The screenshot displays the Burp Suite interface. On the left, the '3. Extension driven active audit' panel shows 'Capturing' is enabled and 'Issues: 6' (with 3 errors). The main panel shows a list of issues found, including 'Cross-site scripting (reflected)', 'OS command injection', 'SQL injection', and 'Unencrypted communications'. On the right, the 'Request 2' tab is selected, showing a POST request to '/graphql HTTP/1.1'. The request body is highlighted, showing a JSON payload with a 'query' field containing a system debug command.

3. Extension driven active audit

Capturing: ☒

Issues: 6 (3 errors)

3319 requests (61 errors)

View details >>

Request 2

Request 2

Request 3

Response 3

Request 1

Response 1

Collaborator DNS interaction

Request 2

Response 2

Request 3

Response 3

Pretty Raw Hex

1 POST /graphql HTTP/1.1

2 Host: localhost:5013

3 sec-ch-ua: "Chromium";v="105", "Not)A;Brand";v="8"

4 Accept: application/json

5 Content-Type: application/json

6 sec-ch-ua-mobile: ?0

7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/105.0.5195.102 Safari/537.36

8 sec-ch-ua-platform: "macOS"

9 Origin: http://localhost:5013/graphql

10 Sec-Fetch-Site: same-origin

11 Sec-Fetch-Mode: cors

12 Sec-Fetch-Dest: empty

13 Accept-Encoding: gzip, deflate

14 Accept-Language: en-US,en;q=0.9

15 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; continueCode=E30zQenePwoj4zk293aRX8KbBNYEa09GL5q01ZDwp6JyVxgQMmrLv7npKLvy; env=graphql:disable

16 Connection: close

17 Content-Length: 53

18

19

"query": "query {\n systemDebug(arg: \"code*&echo ad75ehbxrr r7ydx0jubu&\")\n}"

1 highlight

3.1MB Disk (project): 1.0MB Disk (temp): 46.6MB

Goals:

- Burp extension
- Auto find every possible request
- Leverage Active Scan

Auto GQL™ (aka Burp GraphQL Auto Scanner)

- ❑ Give it a URL
- ❑ Customize headers (optional)
- ❑ Click “Go”

Auto GQL™ (aka Burp GraphQL Auto Scanner)

- ❑ Give it a URL
- ❑ Customize headers (optional)
- ❑ Click “Go”
 1. Runs Introspection Query
 2. Determines every possible request
 3. Finds insertion/injection points
 4. Sends them all to Active Scanner
 5. You: profit



Demo Time

Auto GQL™ (aka Burp GraphQL Auto Scanner)

- ❑ Give it a URL
- ❑ Customize headers (optional)
- ❑ Click “Go”
 1. Runs Introspection Query
 2. Determines every possible request
 3. Finds insertion/injection points
 4. Sends them all to Active Scanner
 5. You: profit

Solution

Burp Suite Professional v2022.8.5 - DVGA - licensed to Forward Security Inc. [5 user license]

Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extender Project options User options Logger++ **Auto GraphQL**

[] # Query Type Request Name Insertion Points

Auto GraphQL

1. Enter URL

GraphQL Endpoint URL: `http(s)://<host>/graphql`

1b. Edit/Add headers (optional)

```
Accept: application/json
Content-Type: application/json
User-Agent: Auto GraphQL via Burp
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close
```

2. Click "Fetch Queries"

Fetch Queries

3. Click "Run Scan"

Run Scan

Request

1

Search... 0 matches

Solution

Auto GraphQL

1 Runs Introspection Query

1. Enter URL

GraphQL Endpoint URL: `http://localhost:5013/graphql`

1b. Edit/Add headers (optional)

```
Accept: application/json
Content-Type: application/json
User-Agent: Auto GQL via Burp
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close
```

2. Click "Fetch Queries"

3. Click "Run Scan"

Run Scan

Request

Pretty

Raw

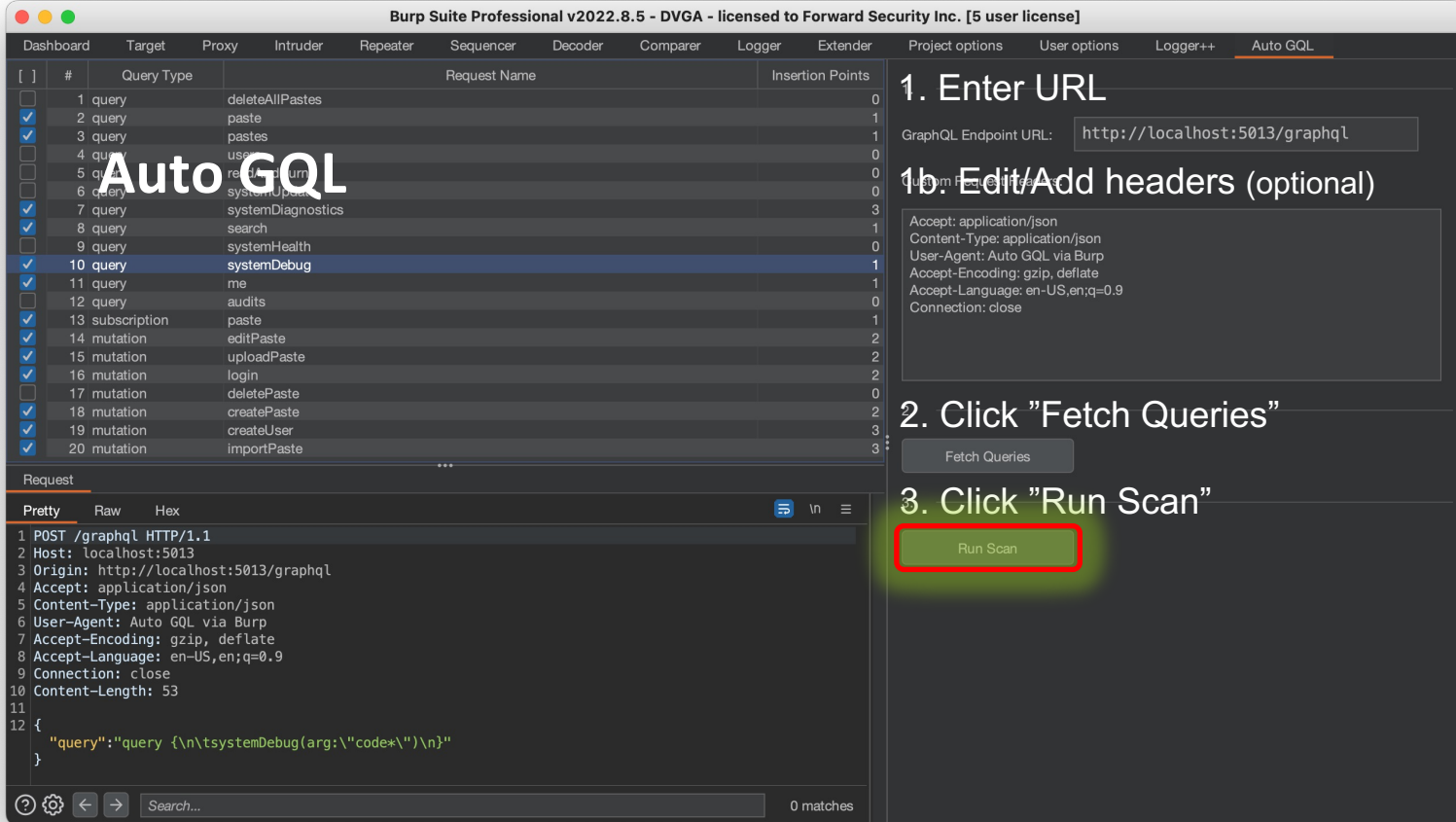
Hex


```
1 POST /graphql HTTP/1.1
2 Host: localhost:5013
3 Origin: http://localhost:5013/graphql
4 Accept: application/json
5 Content-Type: application/json
6 User-Agent: Auto GQL via Burp
7 Accept-Encoding: gzip, deflate
8 Accept-Language: en-US,en;q=0.9
9 Connection: close
10 Content-Length: 53
11
12 {
13   "query": "query {n\\systemDebug(arg:\\\"codex\\\")\\n}"
14 }
```

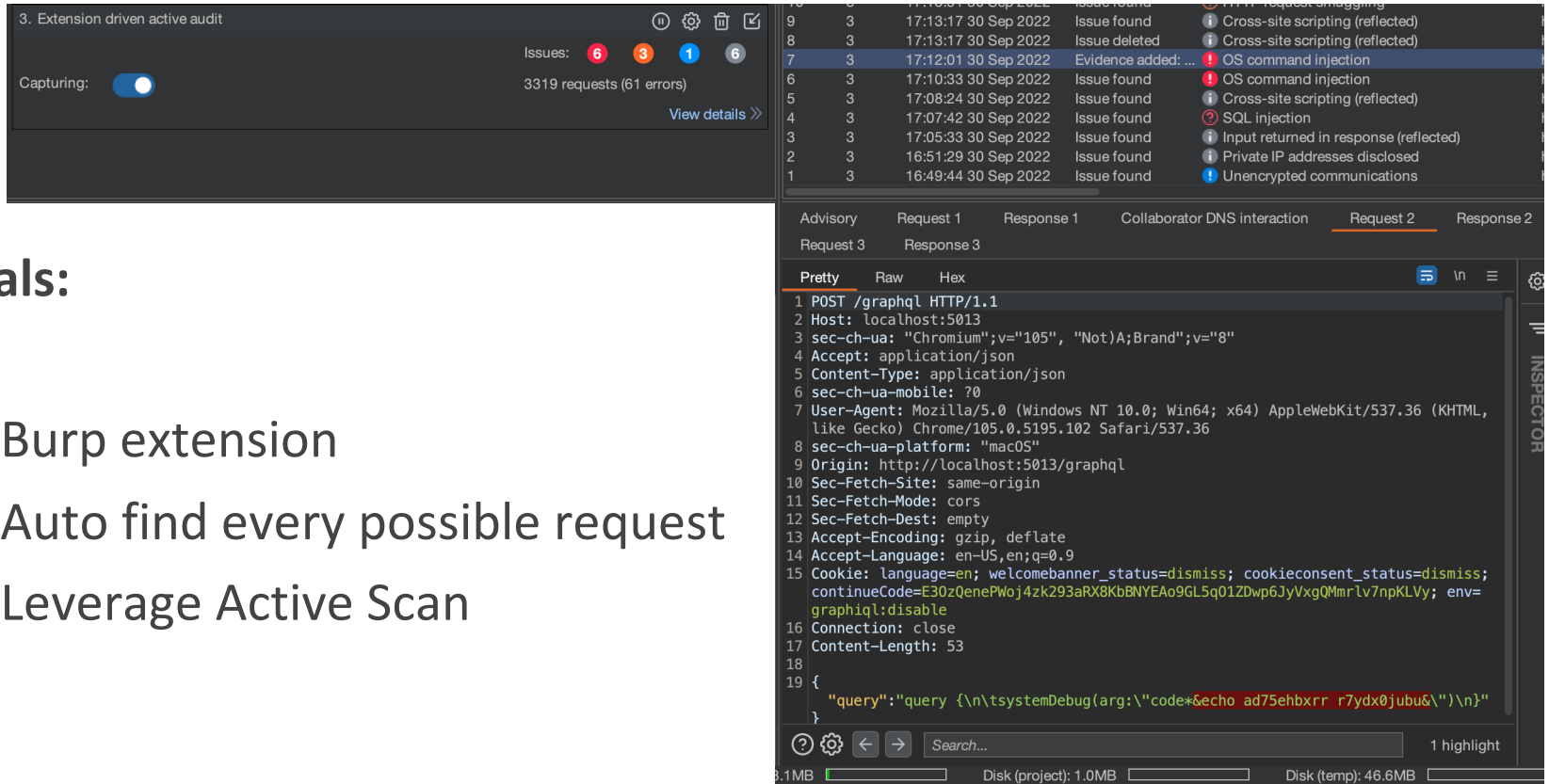
②

atches

Solution



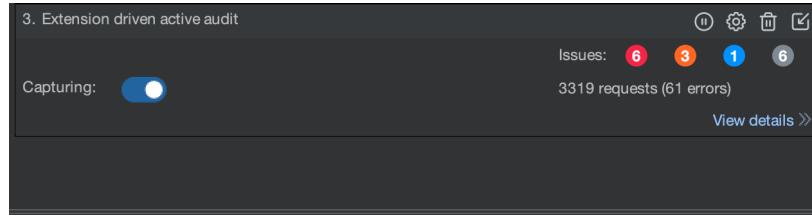
Solution



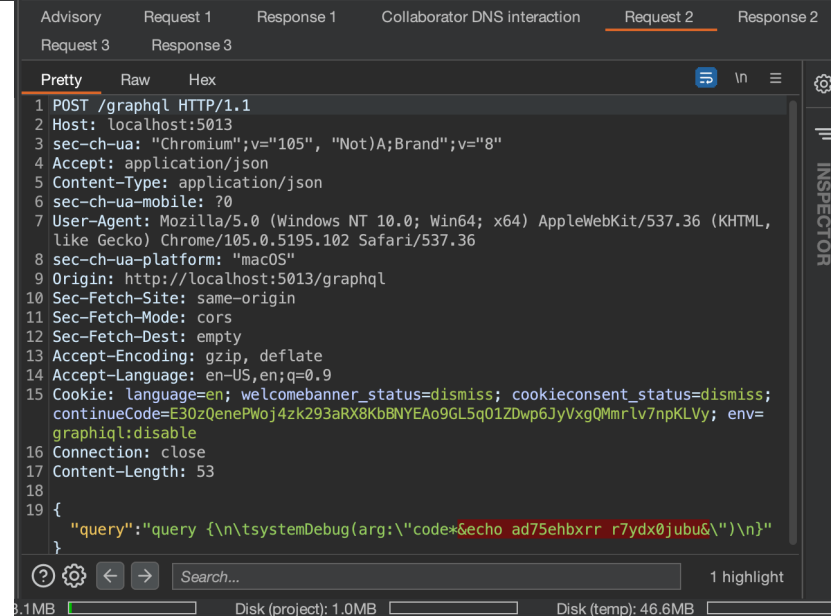
Goals:

- ☐ Burp extension
- ☐ Auto find every possible request
- ☐ Leverage Active Scan

Solution



9	3	17:10:13 30 Sep 2022	Issue found	! HTTP request smuggling
8	3	17:13:17 30 Sep 2022	Issue found	i Cross-site scripting (reflected)
7	3	17:13:17 30 Sep 2022	Issue deleted	i Cross-site scripting (reflected)
6	3	17:12:01 30 Sep 2022	Evidence added: ...	! OS command injection
5	3	17:10:33 30 Sep 2022	Issue found	! OS command injection
4	3	17:08:24 30 Sep 2022	Issue found	i Cross-site scripting (reflected)
3	3	17:07:42 30 Sep 2022	Issue found	? SQL injection
2	3	17:05:33 30 Sep 2022	Issue found	i Input returned in response (reflected)
1	3	16:51:29 30 Sep 2022	Issue found	i Private IP addresses disclosed
1	3	16:49:44 30 Sep 2022	Issue found	! Unencrypted communications



Goals:

- ✓ Burp extension
- ✓ Auto find every possible request
- ✓ Leverage Active Scan

- GraphQL is new and security is often lacking
- Burp's Active Scanner automates vulnerability hunting
- Auto GQL extends Active Scanner to find GraphQL insertion points ***and*** automatically grabs all possible requests.
- Get it here: <https://github.com/FWDSEC/burp-auto-gql>
- Save massive amounts of time

Q&A

Contact Me:

Jared Meit

Email: j.meit@fwdsec.com

Twitter: [@jaredmeit](https://twitter.com/jaredmeit)

<https://github.com/FWDSEC/burp-auto-gql>

Contact FWDSEC:

Twitter: [@FWD_SEC](https://twitter.com/FWD_SEC)