API Security

Securing GraphQL without going around in circles

Kirk Jackson

Lightspeed

6 Sept 2024

OWASP NZ Day 2024

Thank You to Our Sponsors and Hosts!

AppSec NZ appsec.org.nz

OWASP NEW ZEALAND owasp.org.nz

AUT UNIVERSITY TE WĀNANGA ARONUI O TAMAKI MAKAU RAU

BASTION SECURITY GROUP

2°

DATACOM

aws

84.

PentesterLab

plexure

VERACODE

Without them, this Conference couldn't happen.

API Security

Securing GraphQL without going around in circles

Kirk Jackson

Lightspeed

6 Sept 2024

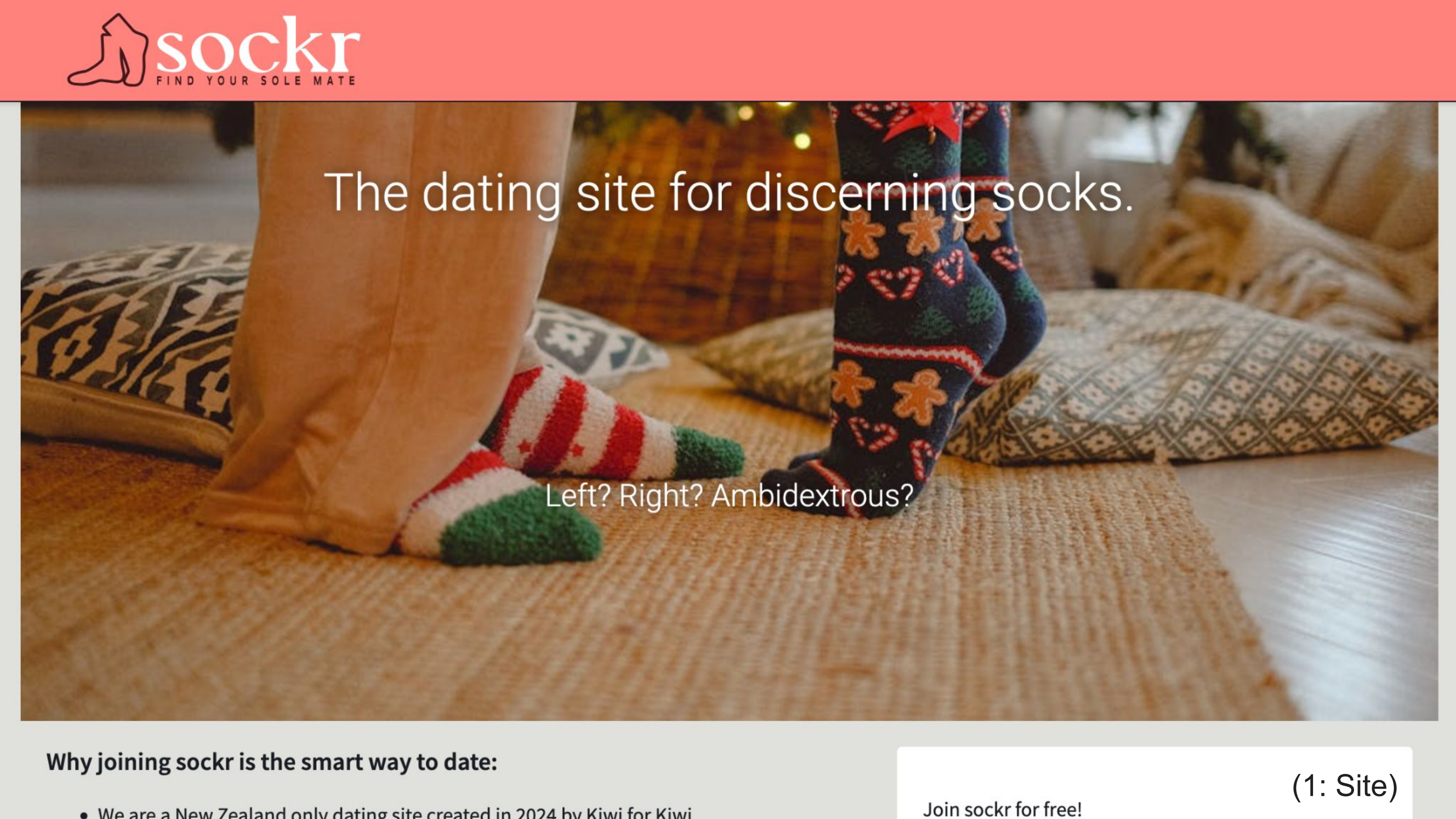
OWASP NZ Day 2024

[redacted]
[redacted]

sockr

FIND YOUR SOLE MATE

https://sockr.net


sockr
FIND YOUR SOLE MATE

The dating site for discerning socks.

Left? Right? Ambidextrous?

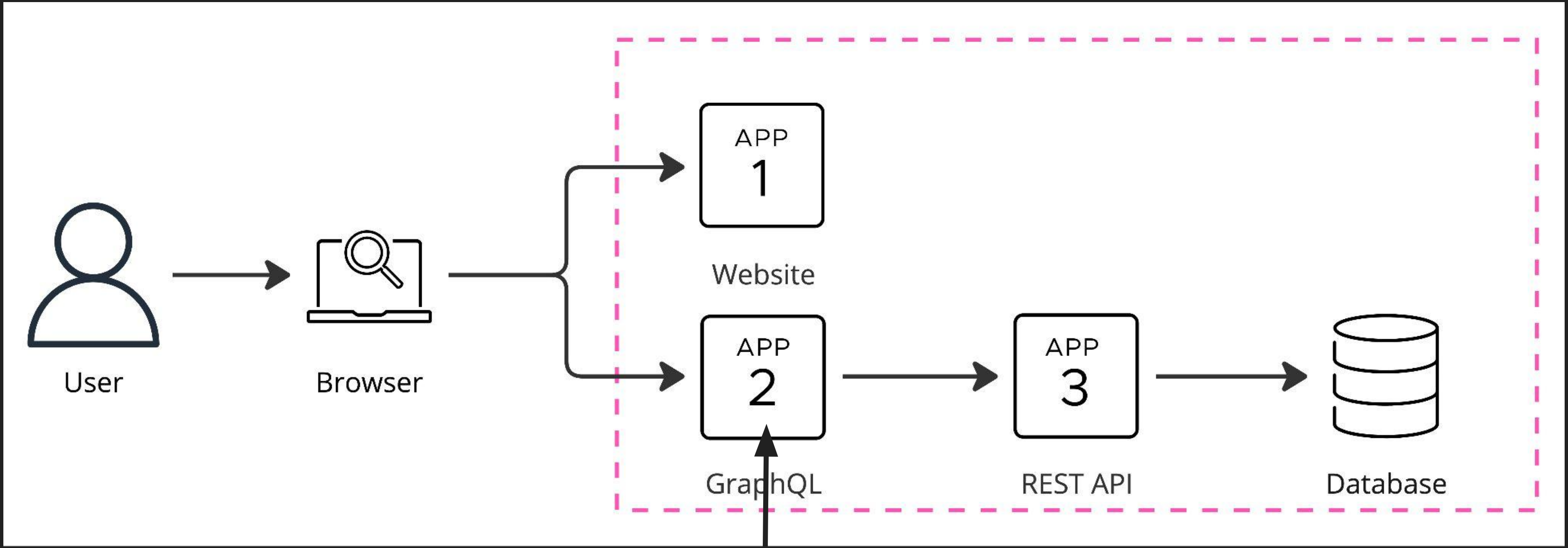
Why joining sockr is the smart way to date:

- We are a New Zealand only dating site created in 2024 by Kiwi for Kiwi

Join sockr for free!

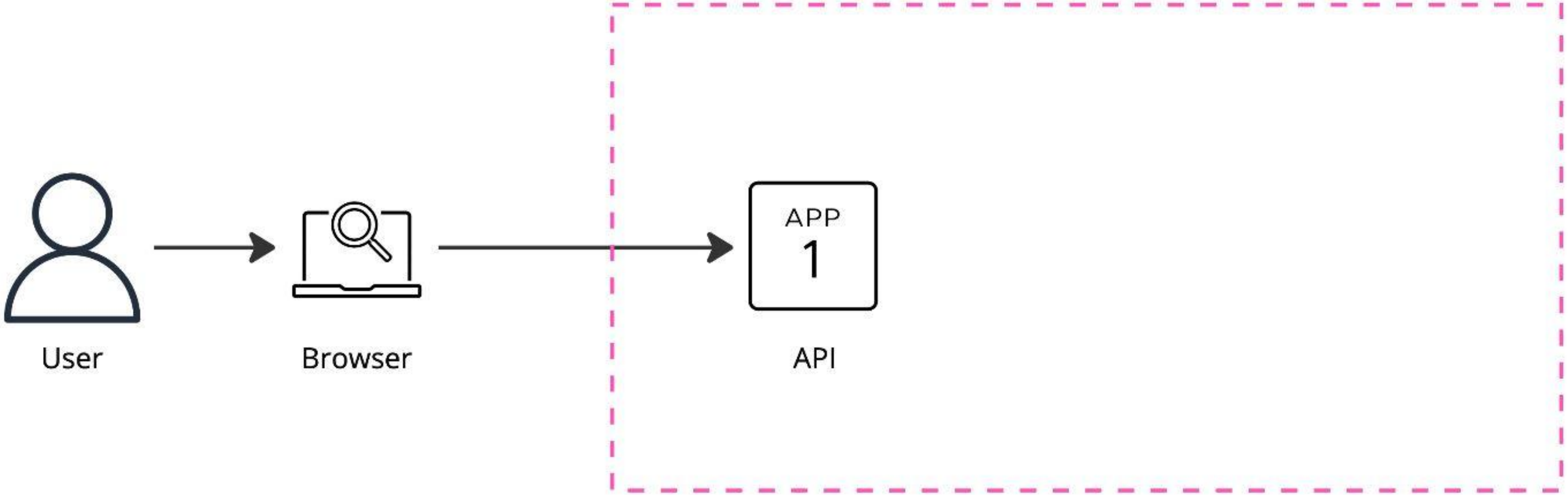
(1: Site)

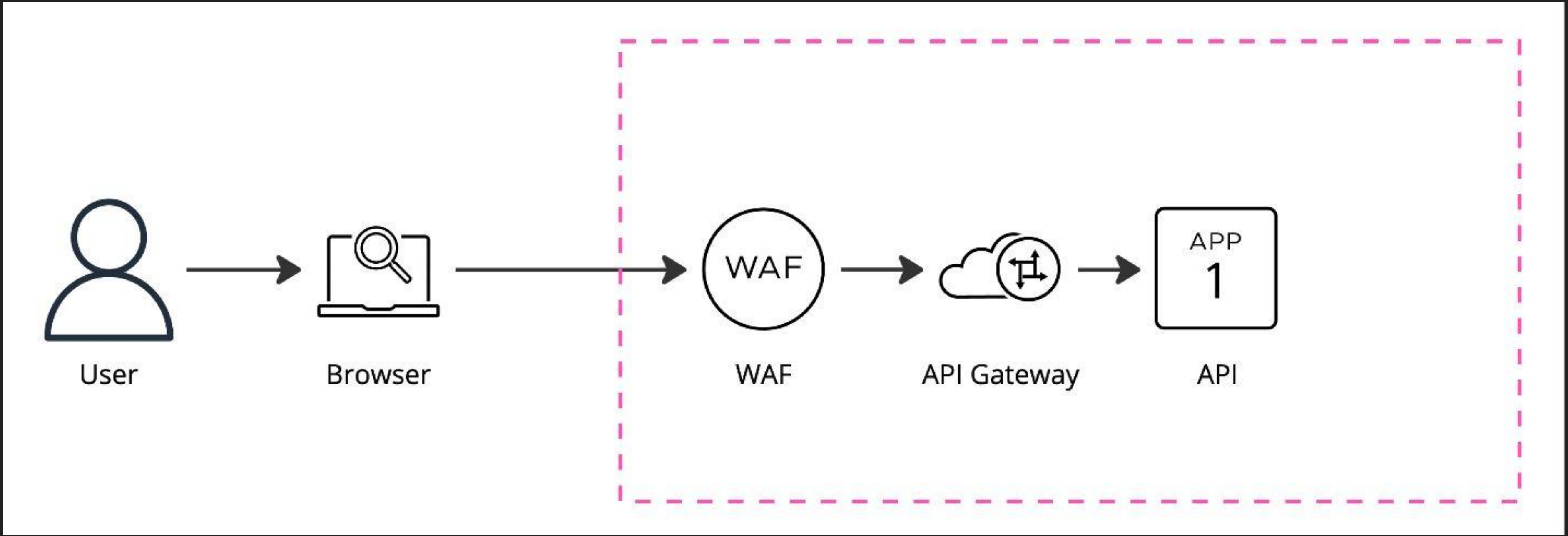
sockr architecture

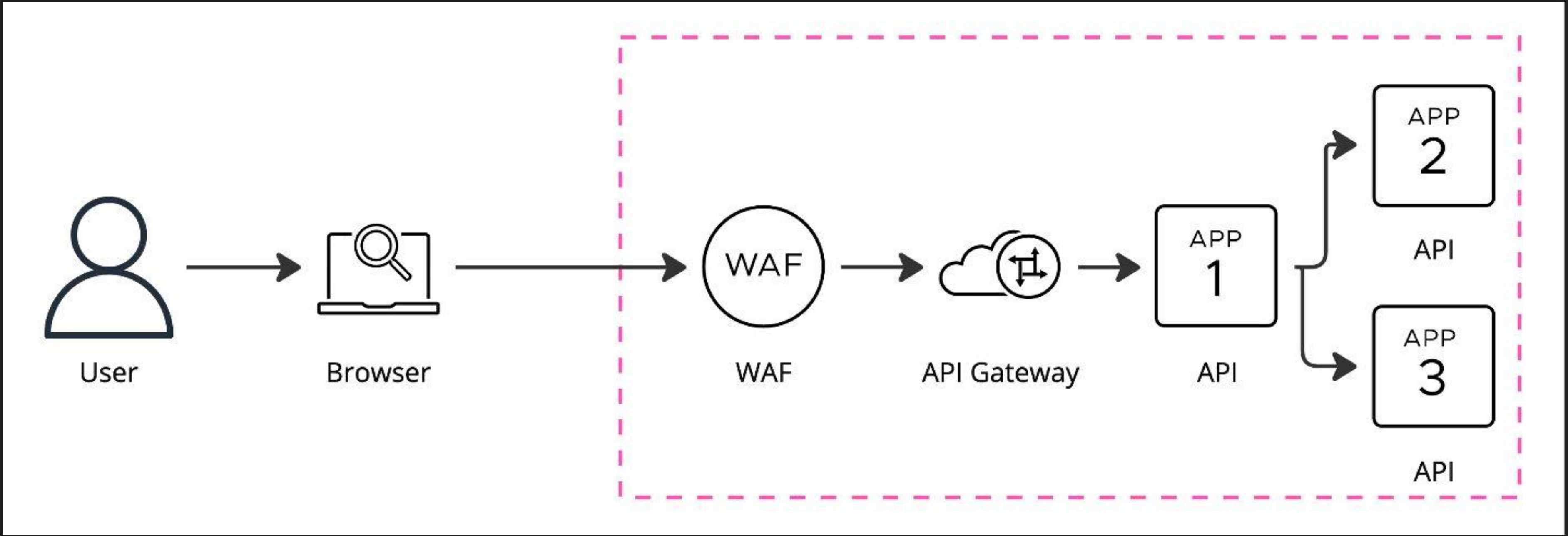


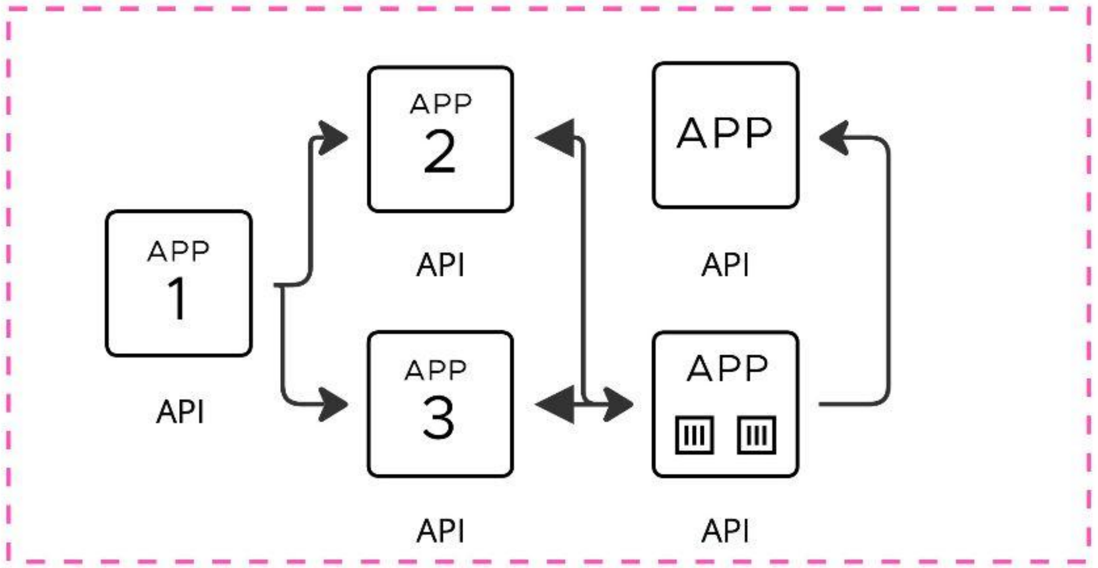
Auth

Why does API security matter?


APP
1
User
Browser
API


User
Browser
WAF
WAF
API Gateway
APP
1
API


User
Browser
WAF
WAF
API Gateway
APP 1
API
APP 2
API
APP 3
API


APP 1
API
APP 2
API
APP 3
API
APP
API
APP
API

Why does API security matter?

- API’s are everywhere
- They’re hidden behind our mobile apps, websites and in our organisations
- Microservices are abundant - each doing a small thing, in a different way
- Critical functionality
- Sensitive data
- Undocumented endpoints

What is GraphQL?

What is GraphQL?

Open sourced by Facebook in 2015

```
type Project {
  name: String
  tagline: String
  contributors: [User]
}
```

Schema:

How the data is structured

```
{
  project(name: "GraphQL") {
    tagline
  }
}
```

Request:

What data we want

```
{
  "project": {
    "tagline": "A query
language for APIs"
  }
}
```

Results:

Easy to consume format

GraphQL operations

```
query CurrentUser {
  currentUser {
    name
    age
  }
}
```

Query:

Read only data

```
mutation CreateUser(
  $name: String!, $age: Int!) {

  createUser(userName: $name,
             age: $age) {
    name
    age
  }
}
```

Mutation:

Write, change state, perform action

```
subscription {
  newPerson {
    name
    age
  }
}
```

Subscription:

Server-push on changes

GraphQL Benefits

Combine multiple API's into one

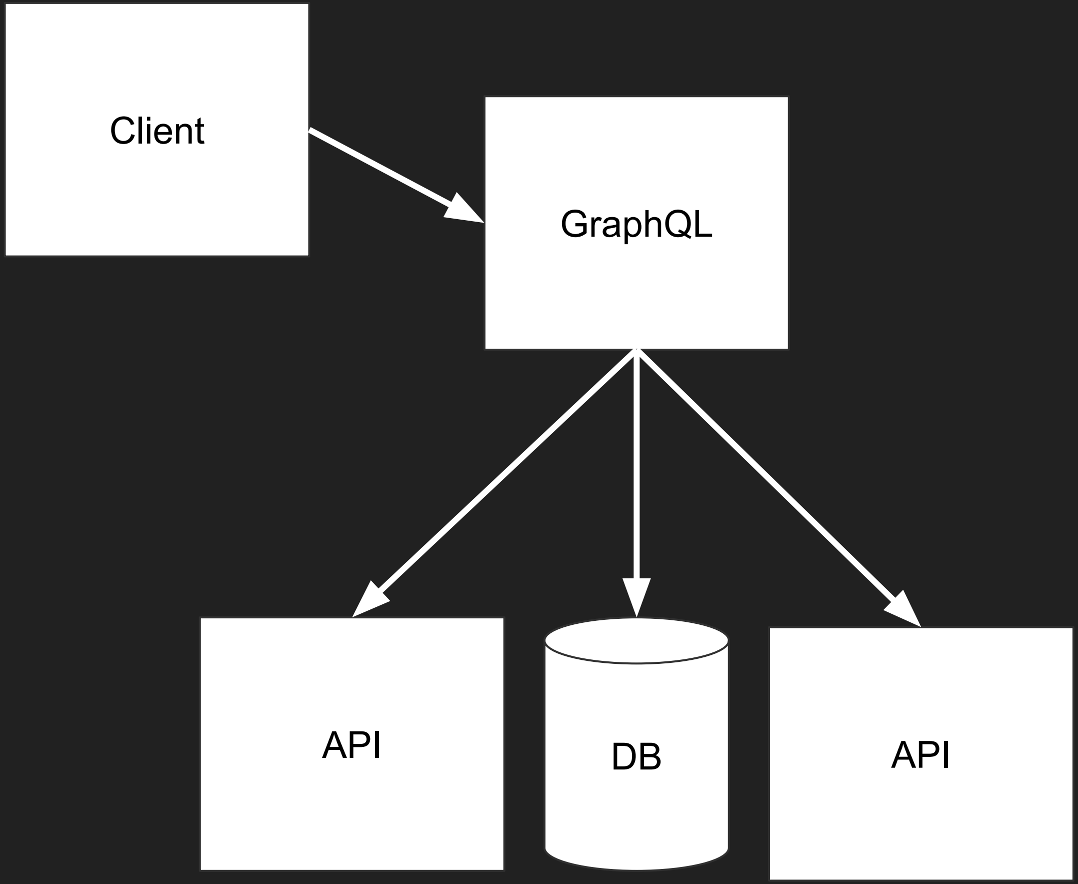
Consistent querying

Stable schema

Request just the data that's needed, in a single request

Caching

Self documenting



sockr it to me!

(2: Apollo Studio, GraphQL Voyager)

OWASP Top Ten API Security Risks - 2023

- Top Ten security risks that are specific to API’s
- Collaboration with API security practitioners
- Other Top Tens still apply

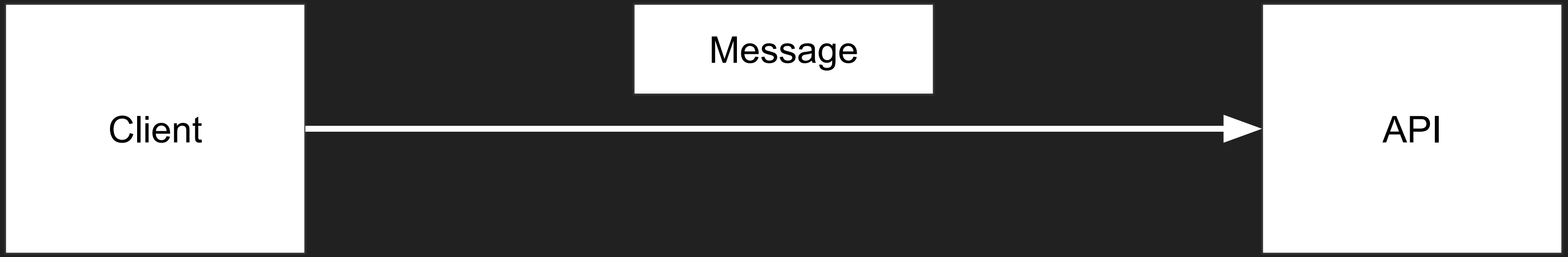
https://owasp.org/API-Security/

Also see the GraphQL cheat sheet:

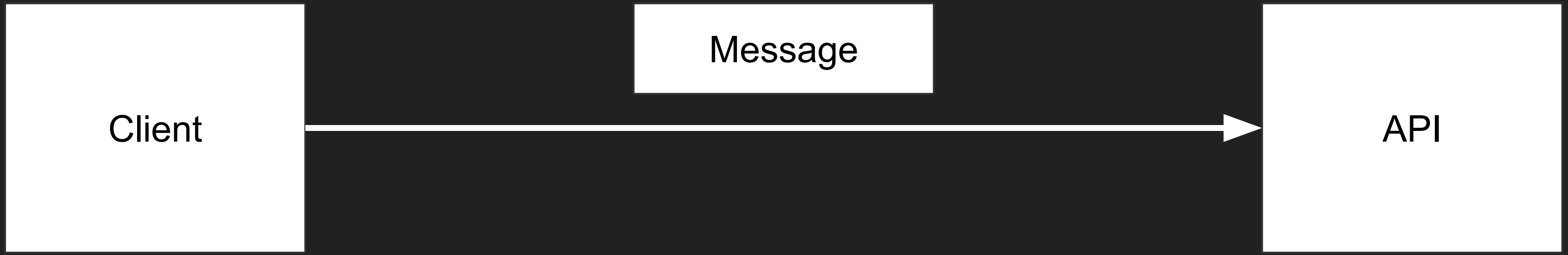
https://cheatsheetseries.owasp.org

OWASP Top Ten API Security Risks - 2023

1. Broken Object Level Authorization
2. Broken Authentication
3. Broken Object Property Level Authorization
4. Unrestricted Resource Consumption
5. Broken Function Level Authorization
6. Unrestricted Access to Sensitive Business Flows
7. Server Side Request Forgery
8. Security Misconfiguration
9. Improper Inventory Management
10. Unsafe Consumption of APIs


Client
Message
API

2. Broken Authentication



Who are you?

Username + password
API key
Client certificate
JWT token

2. Broken Authentication

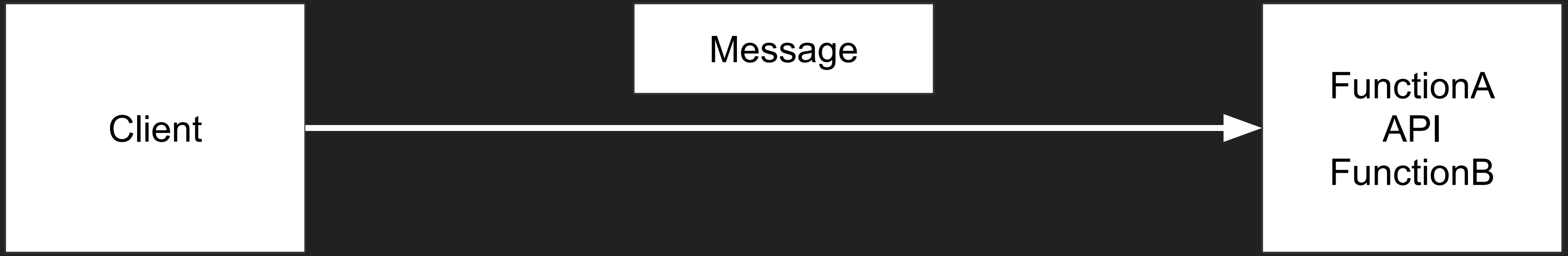
- Unauthenticated access
- Predictable credentials
- Weakly signed / validated tokens
- Allows brute-force attacks

sockr it to me!

(3: JWT)

5. Broken Function Level Authorization

Are you allowed to call this function?



Roles, functions, capabilities and privileges

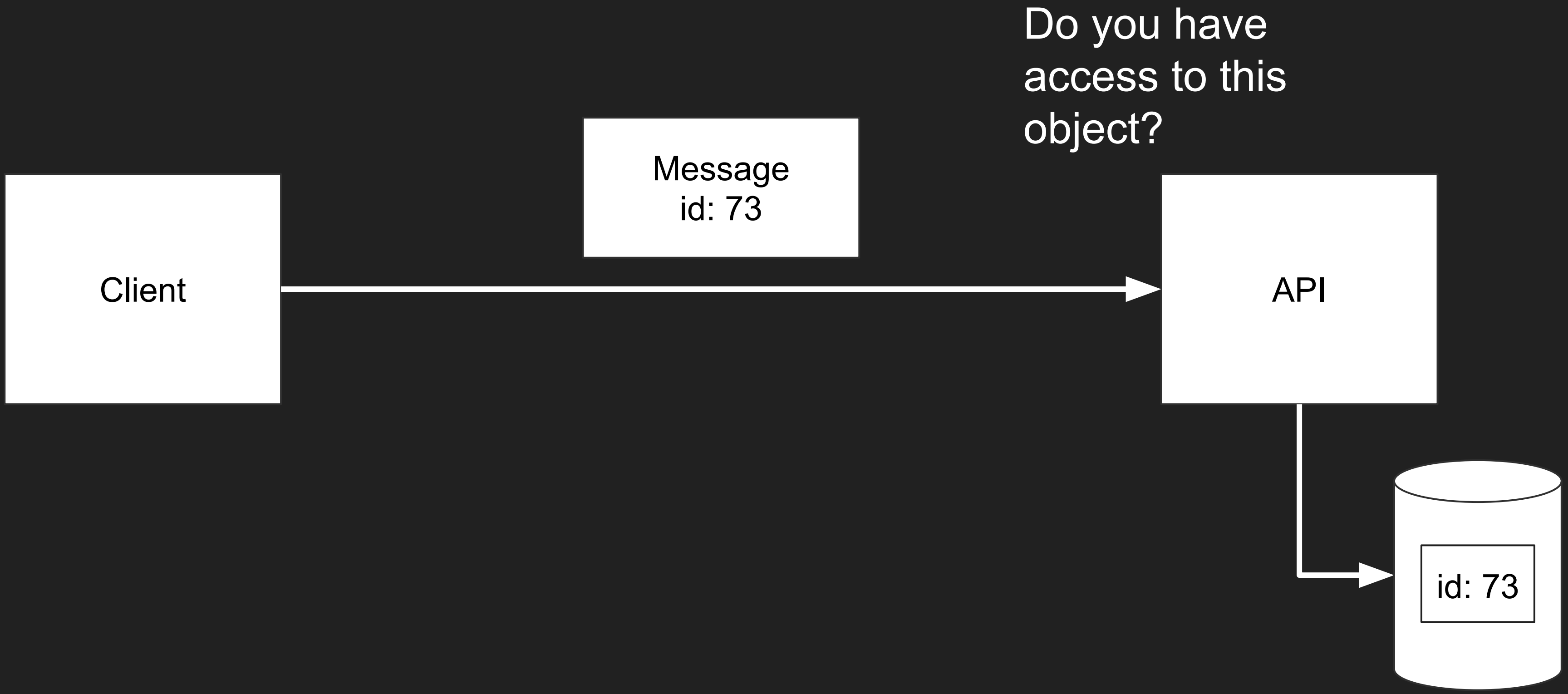
5. Broken Function Level Authorization

Should the authenticated user be authorized to access this function?

sockr it to me!

(4: Apollo Studio, GraphQL Voyager, introspection)

1. Broken Object Level Authorization



1. Broken Object Level Authorization

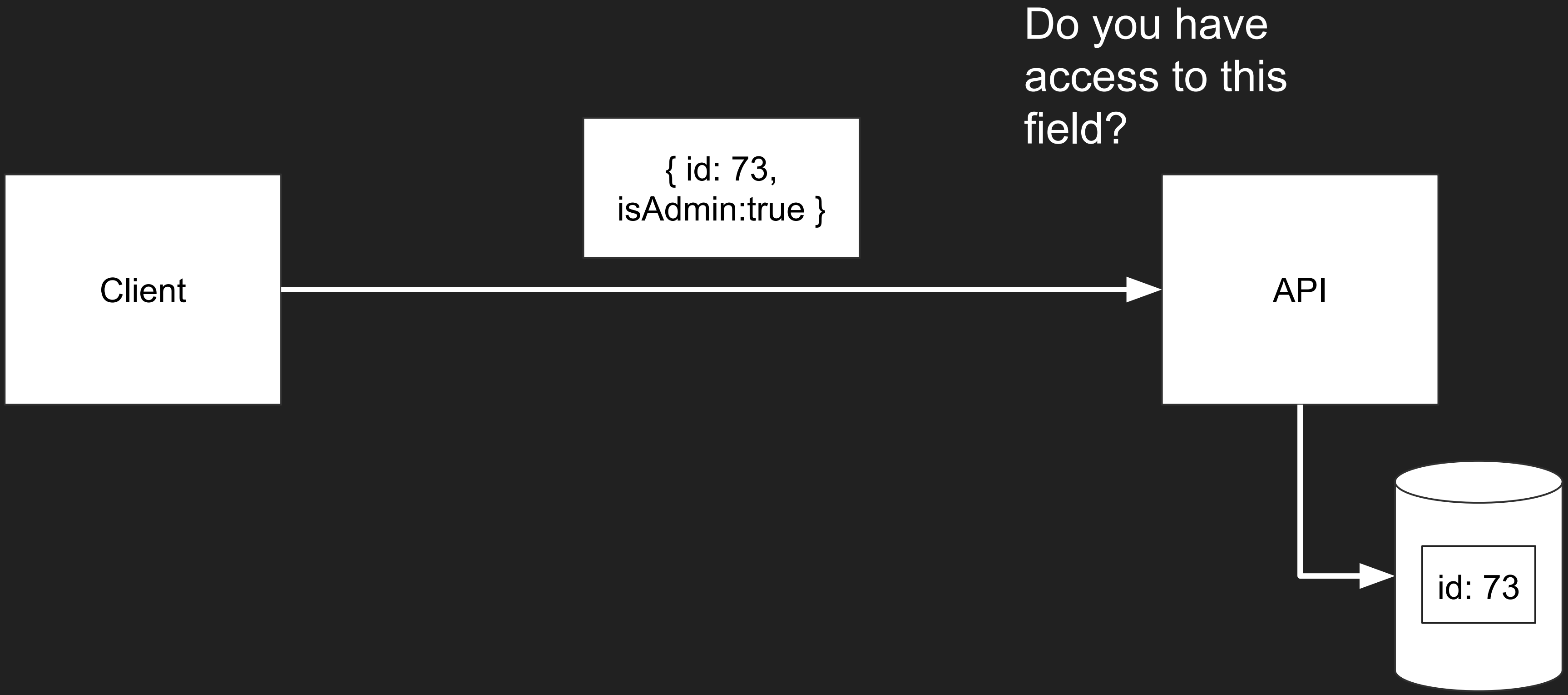
The user can access the function, but are they authorised to access that piece of data?

Are object ID’s exposed or predictable?

sockr it to me!

(5: Admin function)

3. Broken Object Property Level Authorization



3. Broken Object Property Level Authorization

Are their fields that shouldn’t be returned or modified?

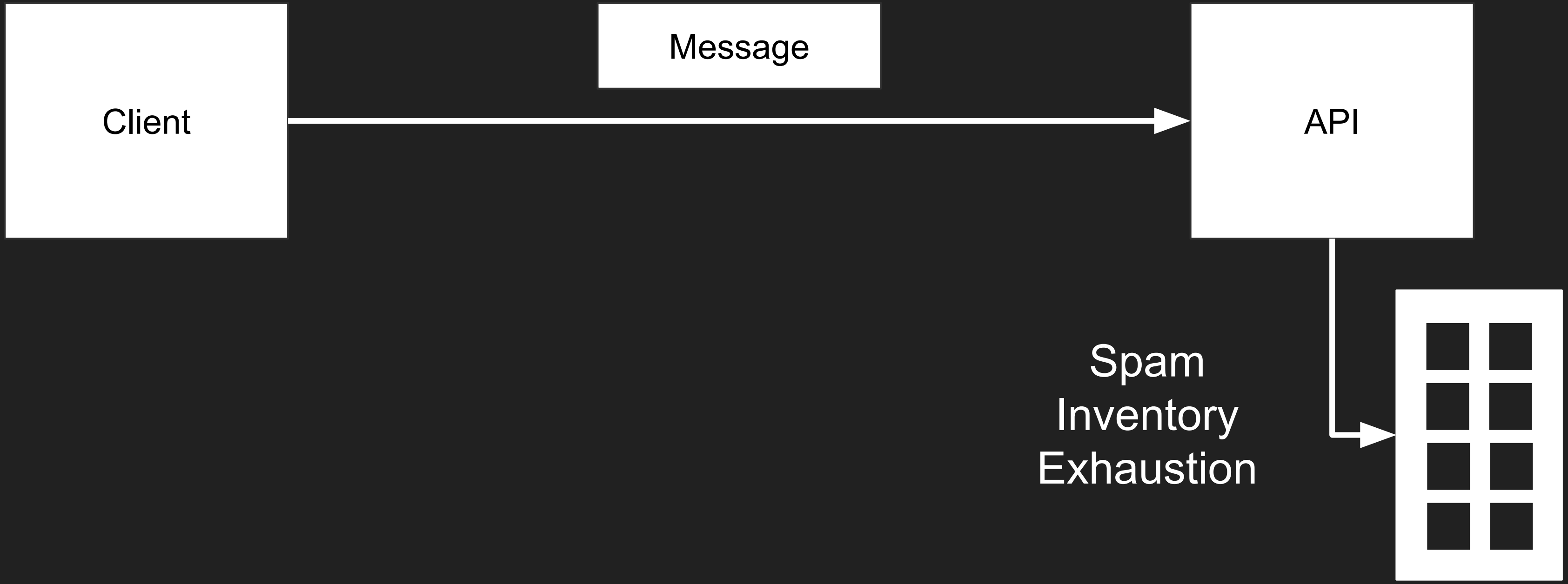
Can the user request different fields or guess field names?

sockr it to me!

(6: Author read query)

6. Unrestricted Access to Sensitive Business Flows

What is the impact on our business?

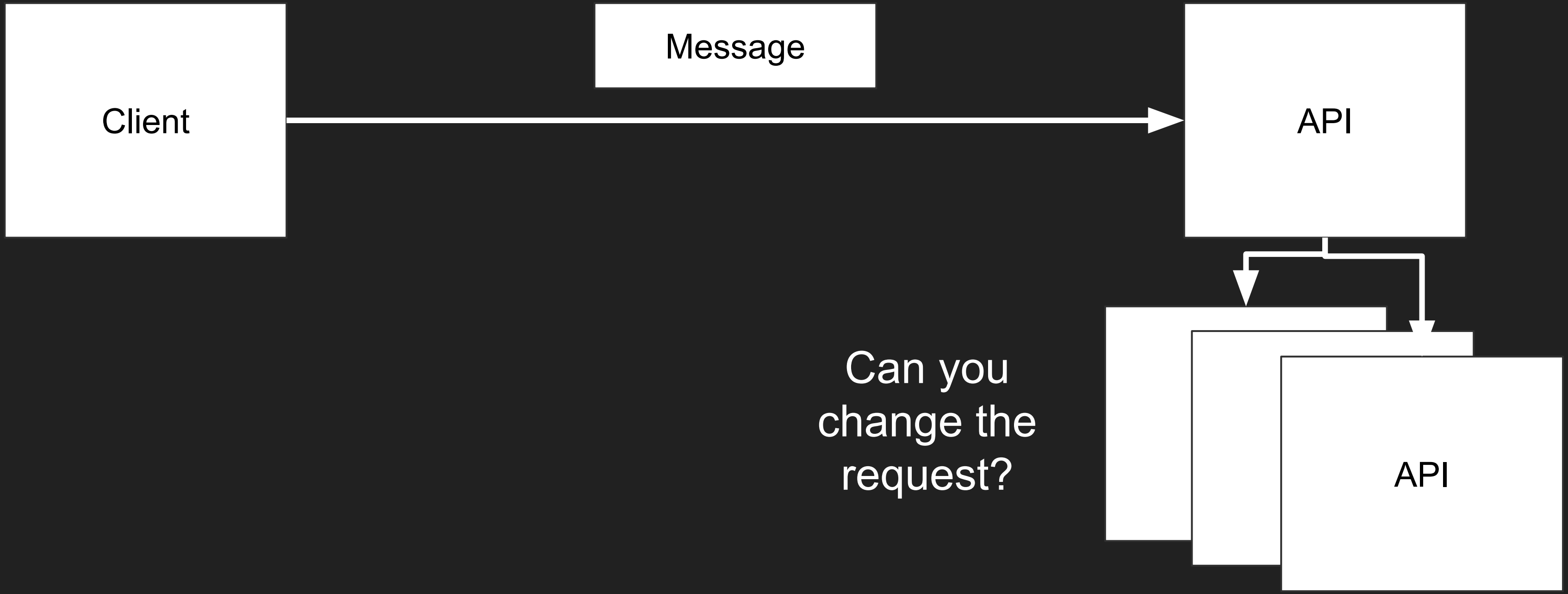


6. Unrestricted Access to Sensitive Business Flows

- Does the API expose a business function that can be abused?
- Will this have an impact on the business?
- Detection of humans vs bots
- Rate-limits

7. Server Side Request Forgery

Does this API request other resources?



Can you change the request?

7. Server Side Request Forgery

Can a user control the way the API connects to other resources?

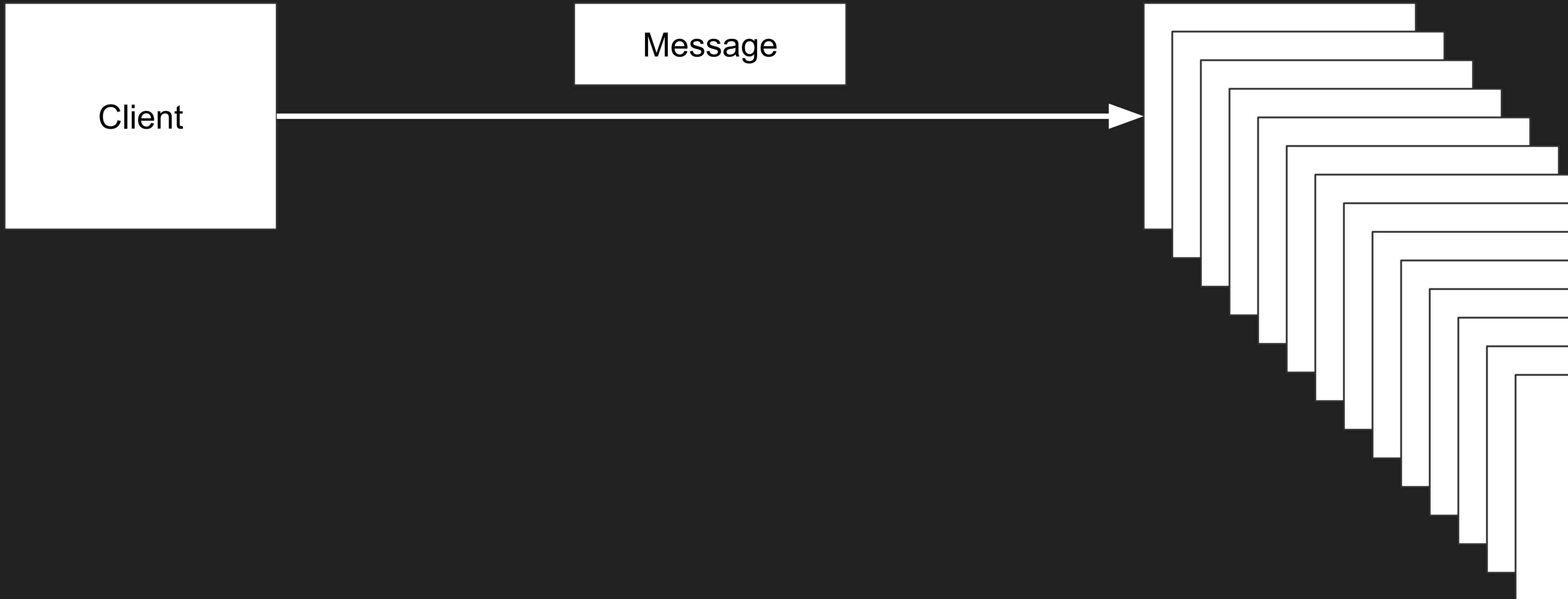
- Change the path of requests
- Access other hosts within or outside the network
- Admin interfaces or metadata resources

sockr it to me!

(7: Listing SSRF)

9. Improper Inventory Management

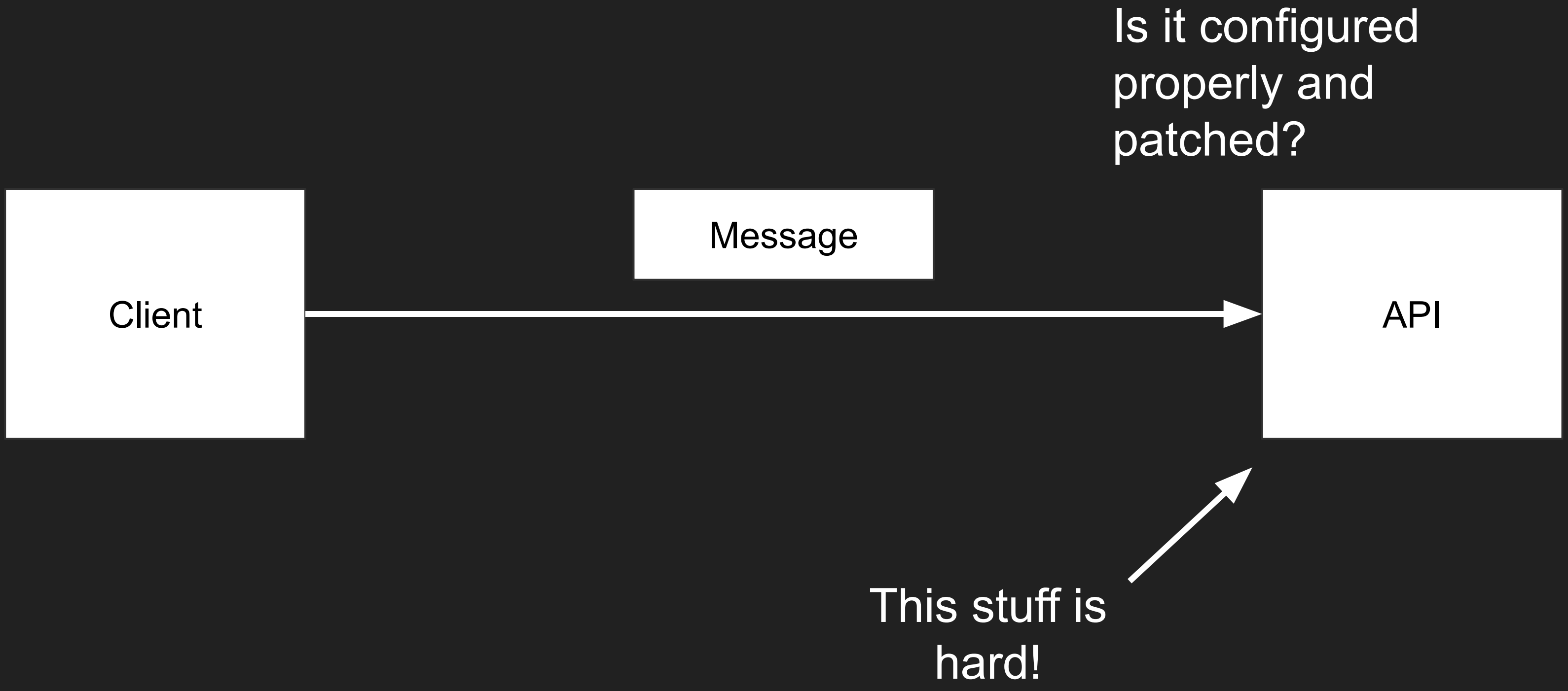
Do we know which API's we have?



9. Improper Inventory Management

- Old
- Unpatched
- Exposed by mistake
- Testing environment
- Temporary resources

8. Security Misconfiguration



8. Security Misconfiguration

Is everything configured, secured and patched correctly?

Extra features enabled?

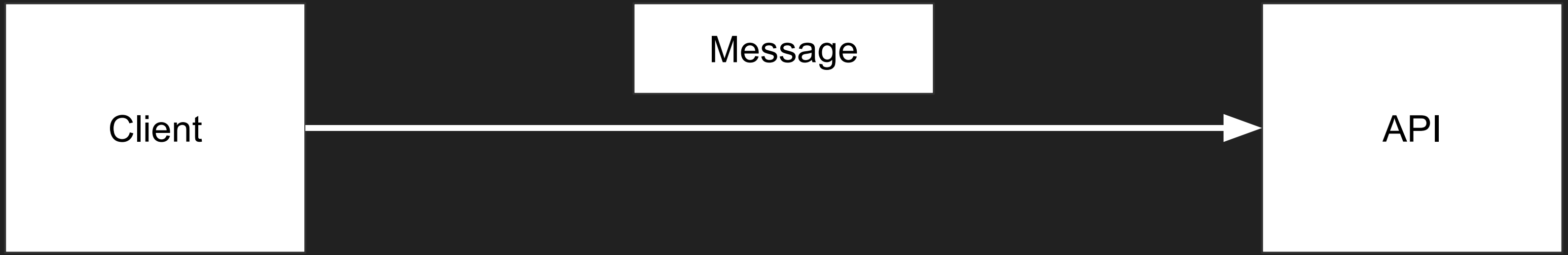
Debug enabled?

Stack traces disclosing sensitive information?

sockr it to me!

(8: Direct access)

10. Unsafe Consumption of APIs



Should we trust the data we get from APIs?

10. Unsafe Consumption of APIs

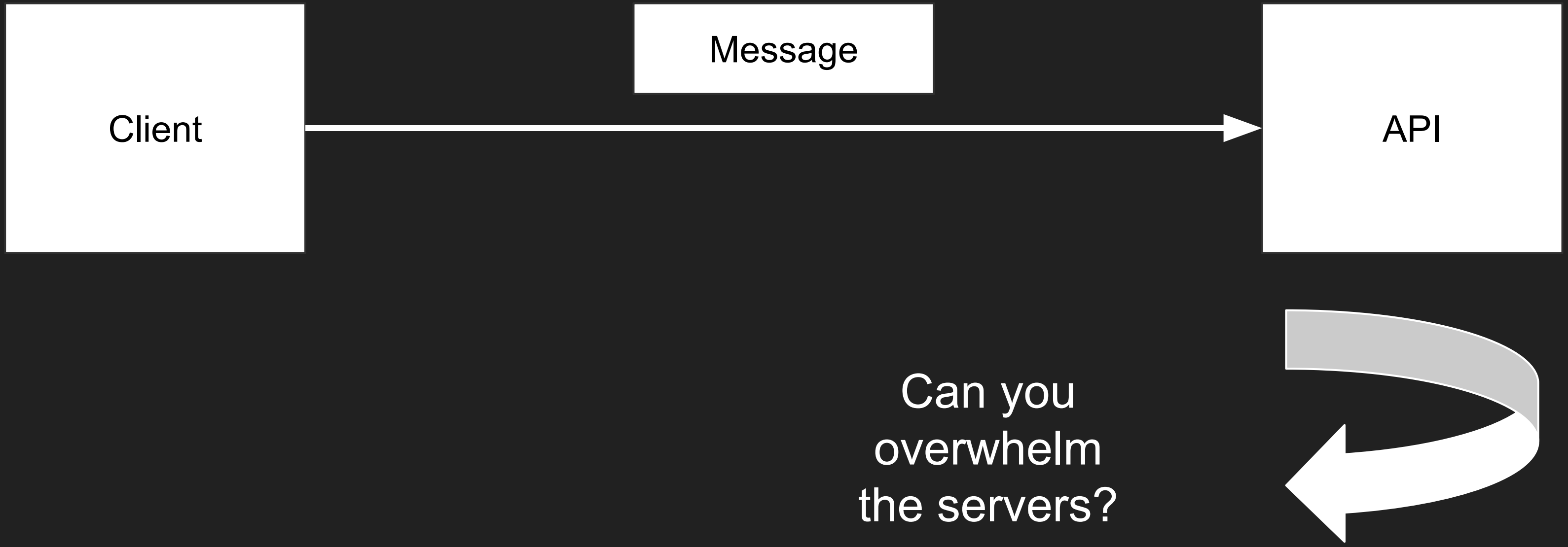
Do we sanitise data received from API’s?

Third party API’s?

Access over secure channels?

4. Unrestricted Resource Consumption

Does this API do heavy work?



Can you overwhelm the servers?

4. Unrestricted Resource Consumption

Swamping resources - CPU, bandwidth, storage

Incurring costs - SMS, per-request costs

sockr it to me!

(8: depth)

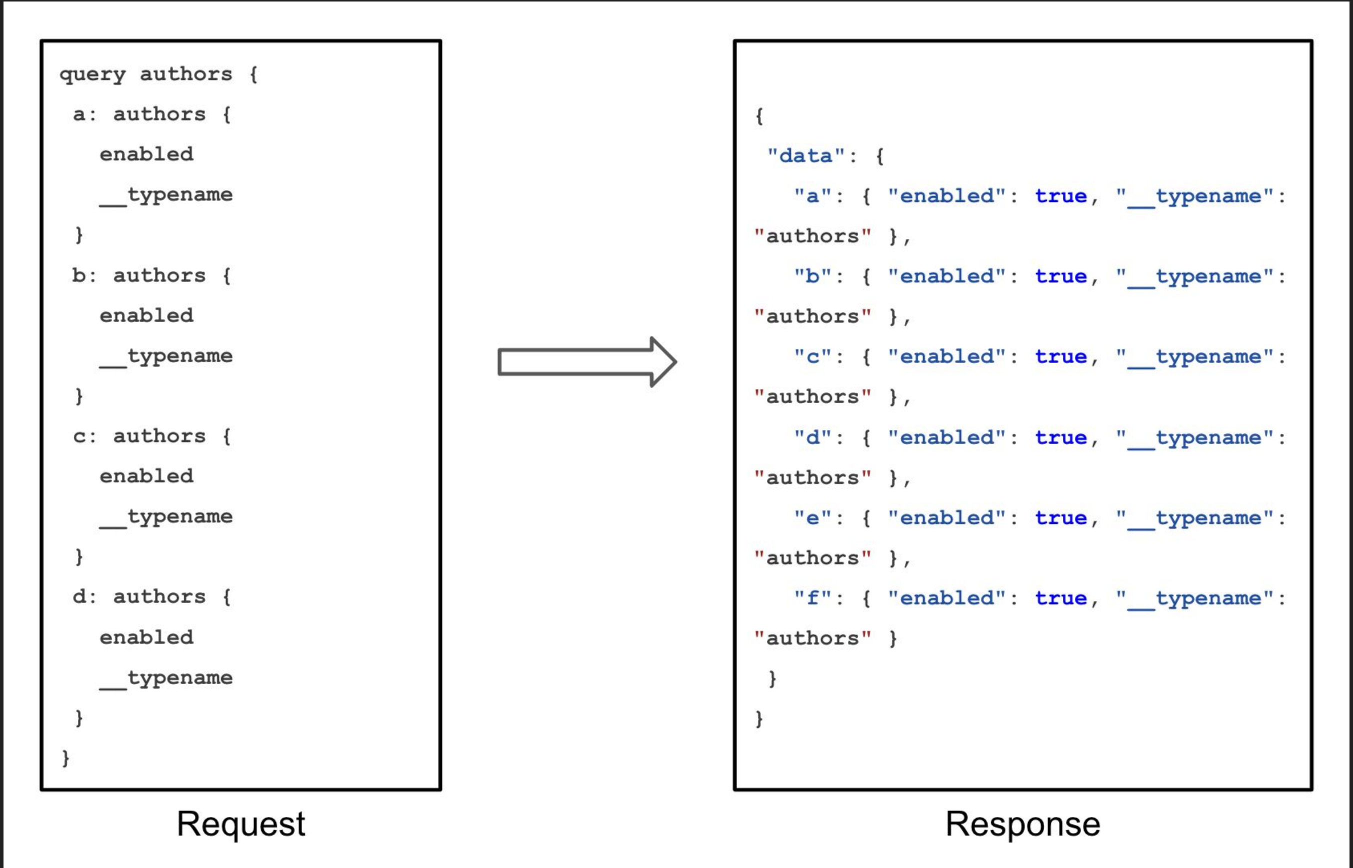
n+1

```
query ListingsForHome {
  listingsForHome {
     id                                          ←  l
     title
     author {
          id                                          ←  x m
          name
          photo
          description
          listings {
               id                                          ←  x n
               title
               thumbnail
               description
               alignment
               soughtalignment
               numberofholes
               location
               Numberofviews
          }
     }
  }
}
```

Batching

```
[
     {"operationName":"a","query":"query GetListings {listingsForHome{ id }}"},
     {"operationName":"b","query":"query GetListings {listingsForHome{ id }}"},
     {"operationName":"c","query":"query GetListings {listingsForHome{ id }}"},
     {"operationName":"d","query":"query GetListings {listingsForHome{ id }}"},
     {"operationName":"e","query":"query GetListings {listingsForHome{ id }}"},
     {"operationName":"f","query":"query GetListings {listingsForHome{ id }}"},
     {"operationName":"g","query":"query GetListings {listingsForHome{ id }}"},
     {"operationName":"h","query":"query GetListings {listingsForHome{ id }}"}
]
```

Batching without batching



```
query authors {
 a: authors {
   enabled
   __typename
 }
 b: authors {
   enabled
   __typename
 }
 c: authors {
   enabled
   __typename
 }
 d: authors {
   enabled
   __typename
 }
}
```

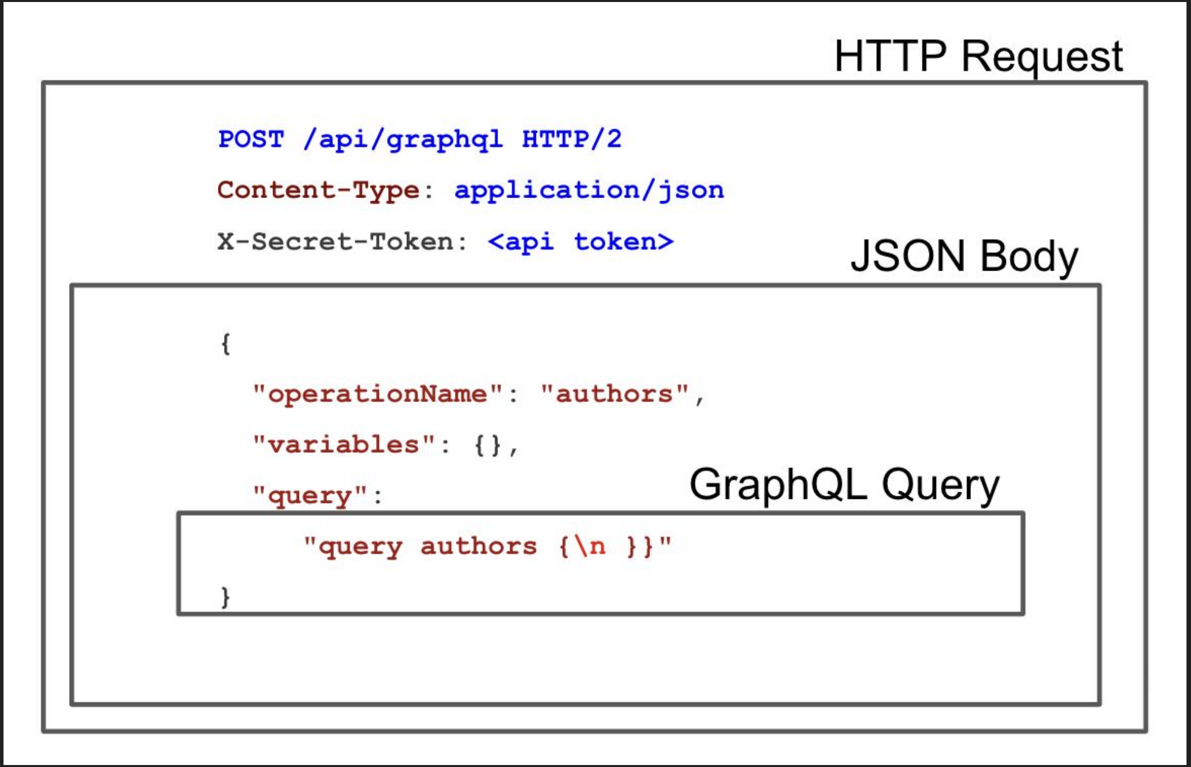
Request

```
{
 "data": {
   "a": { "enabled": true, "__typename":
"authors" },
   "b": { "enabled": true, "__typename":
"authors" },
   "c": { "enabled": true, "__typename":
"authors" },
   "d": { "enabled": true, "__typename":
"authors" },
   "e": { "enabled": true, "__typename":
"authors" },
   "f": { "enabled": true, "__typename":
"authors" }
 }
}
```

Response

Smuggling

Encoding and nested contexts



```
POST /api/graphql HTTP/2
Content-Type: application/json
X-Secret-Token: <api token>

{
  "operationName": "authors",
  "variables": {},
  "query":
     "query authors {\n }}"
}
```

Malformed JSON

```
POST /api/graphql HTTP/2
Content-Type: application/json
X-Secret-Token: <api token>

{
  "operationName": "authors",
  "variables": {},
  "query":
    "query authors {\n}"
}
```

Original Request

```
POST /api/graphql HTTP/2
Content-Type: application/json
X-Secret-Token: <api token>

{
  "operationName": "authors",
  "variables": {},
  "query": "",
  "query":
    "query authors {\n}"
}
```

Additional “query” name/value

Encoding

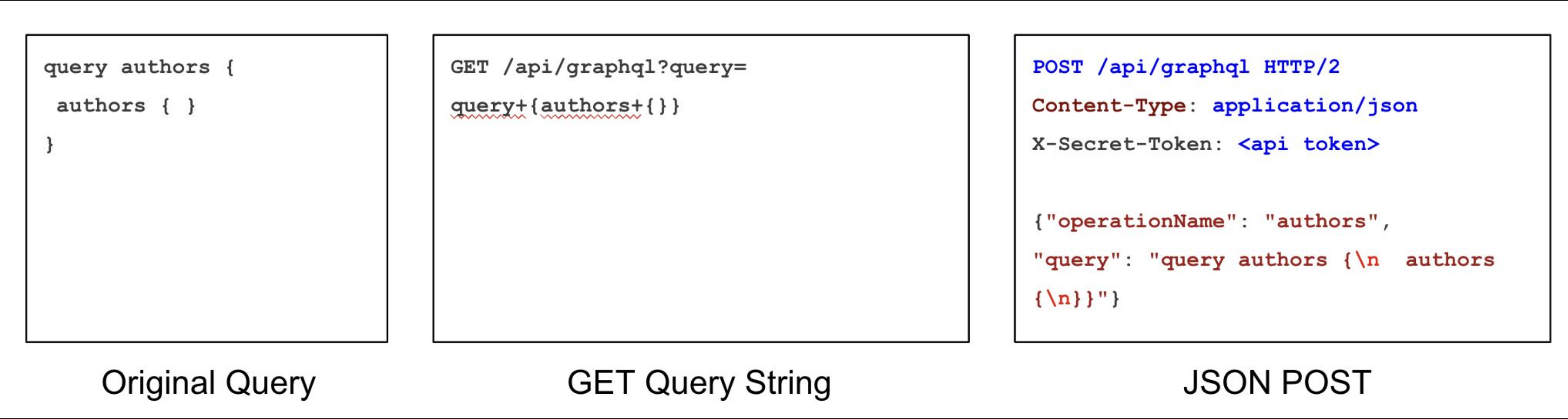
```
query authors {
 authors {
   enabled
   __typename
 }
}
```

Original Query

```
query q {
 x: \u0061uthor\u0073 {
   y: enabled
   z: __typename
 }
}
```

With aliases and encoding

Multiple request types



```
query authors {
 authors { }
}
```

Original Query

```
GET /api/graphql?query=
query+{authors+{}}
```

GET Query String

```
POST /api/graphql HTTP/2
Content-Type: application/json
X-Secret-Token: <api token>

{"operationName": "authors",
"query": "query authors {\n  authors
{\n}}"}
```

JSON POST

Large request bodies

Cloudflare:

The `http.request.body.raw` variable is truncated at 128kb

AWS WAF:

AWS WAF can inspect at most the first 8 KB

F5:

Default buffer size is 10mb

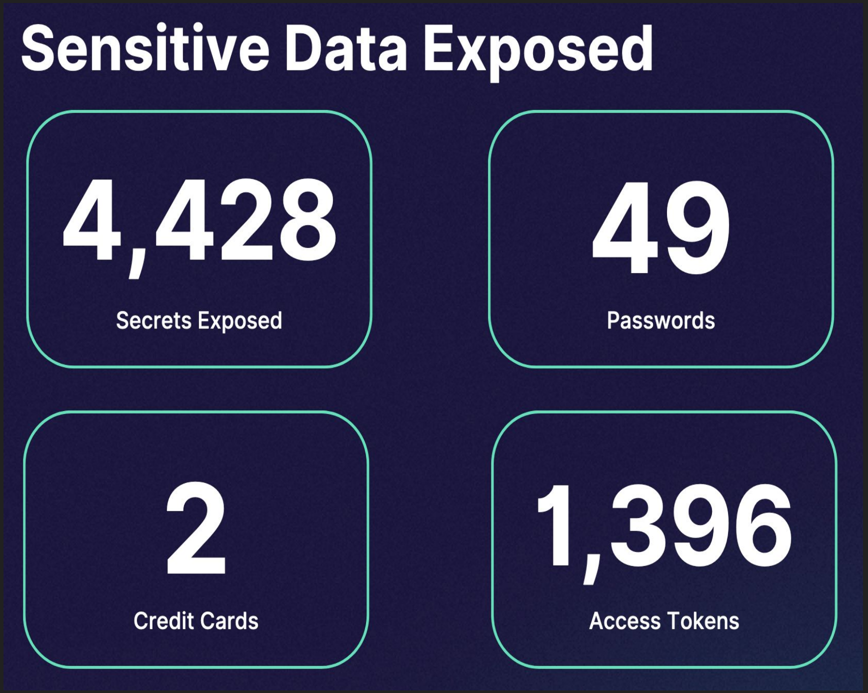
OWASP Top Ten API Security Risks - 2023

1. Broken Object Level Authorization
2. Broken Authentication
3. Broken Object Property Level Authorization
4. Unrestricted Resource Consumption
5. Broken Function Level Authorization
6. Unrestricted Access to Sensitive Business Flows
7. Server Side Request Forgery
8. Security Misconfiguration
9. Improper Inventory Management
10. Unsafe Consumption of APIs

Industry Survey

“The State of GraphQL Security 2024”

Analysed vulnerabilities in 160 public GraphQL API’s

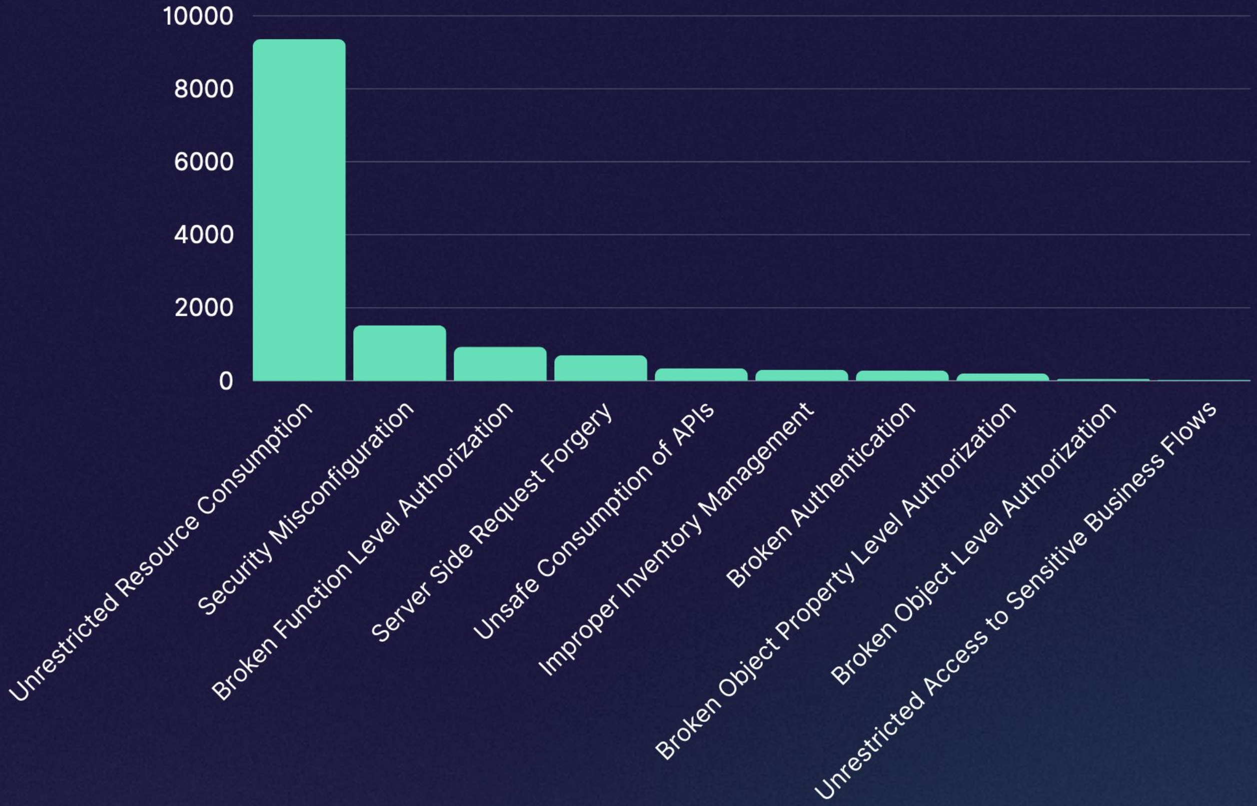

Sensitive Data Exposed

4,428 Secrets Exposed	49 Passwords
2 Credit Cards	1,396 Access Tokens

https://escape.tech/resources/the-state-of-graphql-security-2024

Analysis of Key Vulnerabilities

Breaking down the issues per category, we can quickly observe what the common risks for GraphQL APIs are:



GraphQL Security Checks

- Alias limit
- Automatic Persisted Queries
- Batch Limit
- Character limit
- Cyclic query
- Cyclic Recursive Query
- Debug mode
- Depth limit
- Directive overloading
- Duplicated object
- Error type inconsistency
- Field Duplication
- Field limit
- Field Suggestion
- File inclusion
- GET based CSRF
- GraphQL IDE
- GraphQL Response Format
- Improper Input Validation
- Injection
- Introspection enabled
- Large JSON input
- Pagination missing
- Partial SSRF
- Permissive JSON Input
- Positive integer validation
- Positive integer validation
- POST based CSRF
- Private fields
- Recursive Fragment
- Resource limiting bypass
- Response type mismatch
- Security timeout
- Server Side Request Forgery
- Stored Improper Input
- Validation Injection
- Typing misconfiguration
- Undefined objects
- Unreachable server
- Width limit
- Zombie object

https://docs.escape.tech/testing/vulnerabilities/

Securing GraphQL

WAFs aren’t much help:

- Too many json bypasses, protocol specifics
- Rate-limiting is hard due to single url, no string matches

API Gateways may have some checking

Harden your GraphQL server itself

- Middleware like GraphQL Armor might help

Handy dandy tools

GraphQL Shield - Permissions & Authorisation (MIT)

GraphQL Armor - Harden default GraphQL server config (MIT)

GraphQL.Security - Scanner (free trial)

GraphQL Voyager - Visualise schema (MIT)

Burp Suite - GraphQL scan and attack (paid) sockr

FIND YOUR SOLE MATE

Seeking investors: 10% for $NZD4.5m

API Security

Securing GraphQL without going around in circles

Kirk Jackson

Lightspeed

6 Sept 2024

OWASP NZ Day 2024