escape

GraphQL Vulnerabilities in the Wild

What scanning real-life GraphQL apps told us
about exploiting GraphQL in Production

Table of content

01 About Escape

02 Why is GraphQL vulnerable?

03 Why is is hard to test production GraphQL apps?

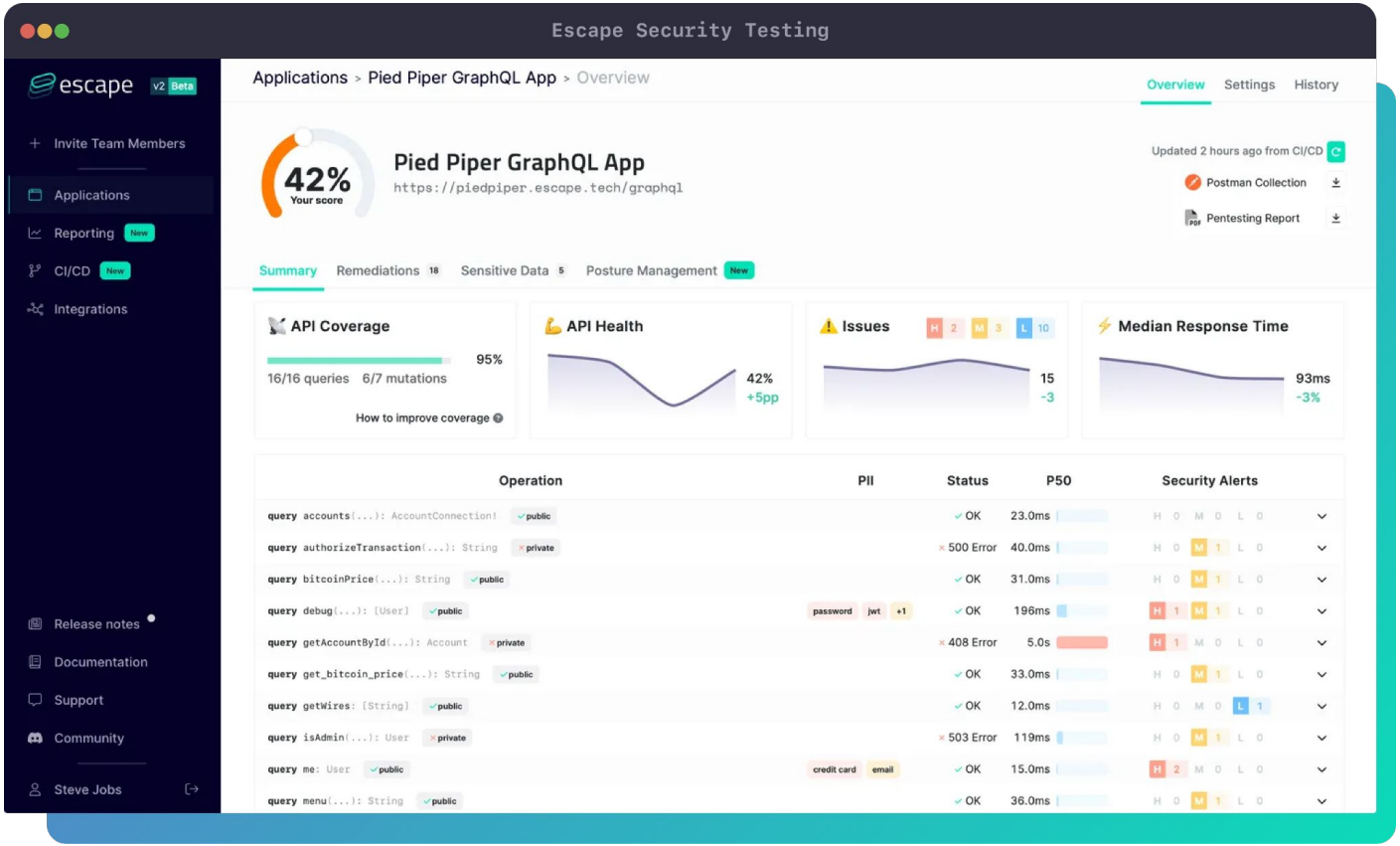
04 The findings

05 Sensitive data made public!

escape BSIDES MILANO

About Escape.tech

First automated Security Testing platform (DAST) able to test the business logic of APIs and GraphQL.



escape

BSIDES MILANO

1599

production APIs scanned

escape

BSIDES

1599

production APIs scanned

416

cumulative scan hours

escape BSIDES

1599

production APIs scanned

416

cumulative scan hours

46,809

security alerts found

escape BSIDES

30

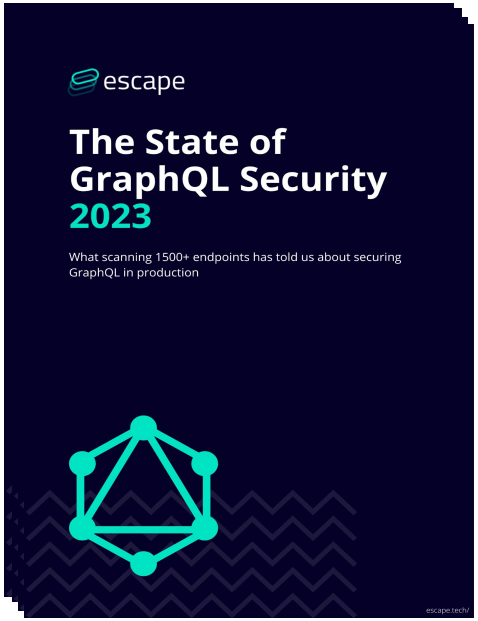
Security alerts per GraphQL app on average

escape

BSIDES

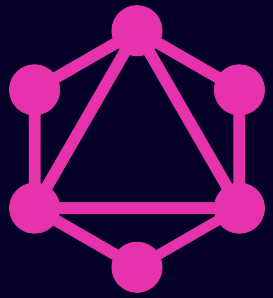
Tristan Kalos
Co-founder & CEO
tristan@escape.tech
@tristankalos

Antoine Carossio
Co-founder & CTO
antoine@escape.tech
@iCarossio



https://escape.tech/resources/state-of-graphql-security-2023/

escape BSIDES MILANO



Why is GraphQL vulnerable?

escape

BSIDES

GraphQL is a query language for APIs

PELOTON NETFLIX The New York Times Walmart PayPal

nerdwallet expedia group glassdoor priceline

escape BSIDES MILANO

Why is GraphQL vulnerable?

1 – GraphQL is a full-featured language...

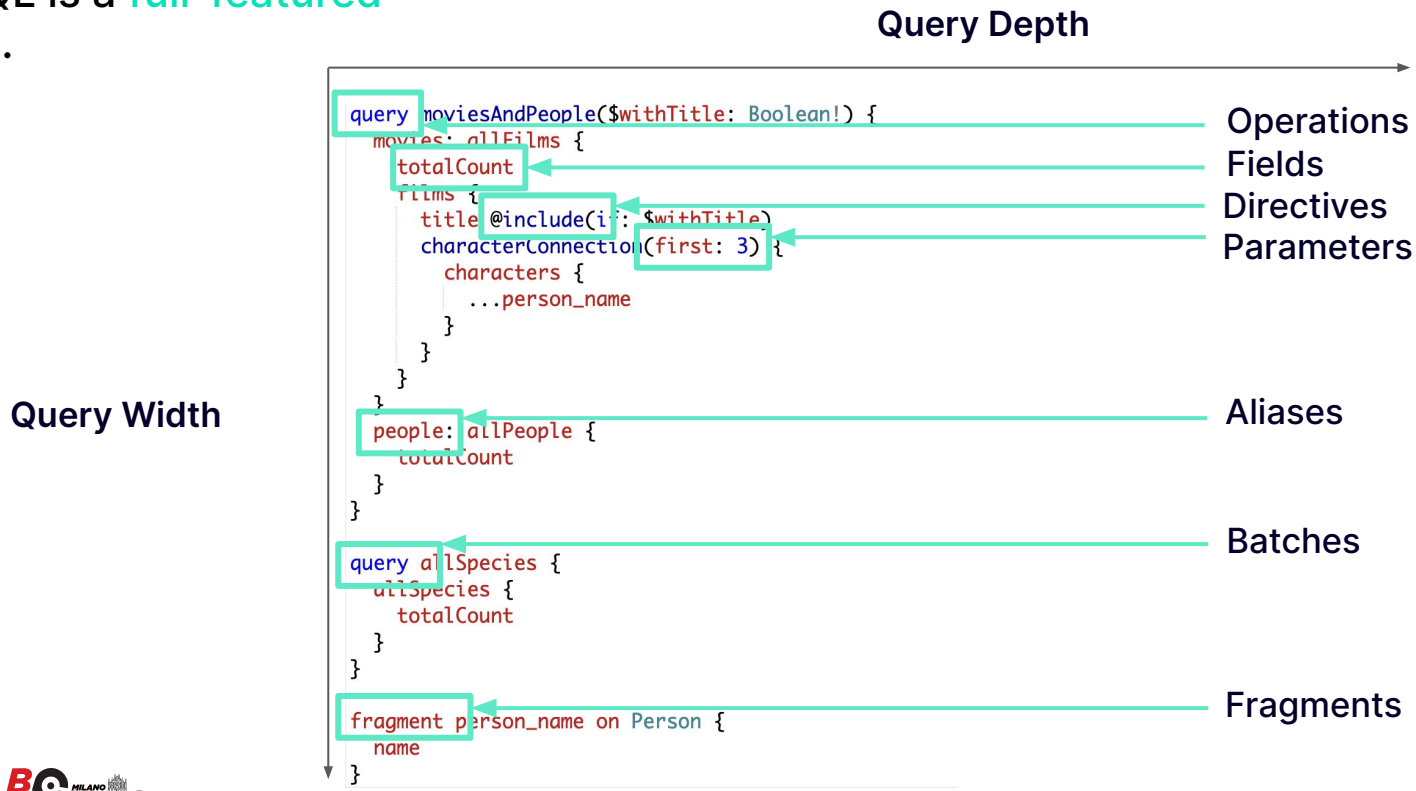
```
query moviesAndPeople($withTitle: Boolean!) {
  movies: allFilms {
    totalCount
    films {
      title @include(if: $withTitle)
      characterConnection(first: 3) {
        characters {
          ...person_name
        }
      }
    }
  }
  people: allPeople {
    totalCount
  }
}

query allSpecies {
  allSpecies {
    totalCount
  }
}

fragment person_name on Person {
  name
}
```

Why is GraphQL vulnerable?

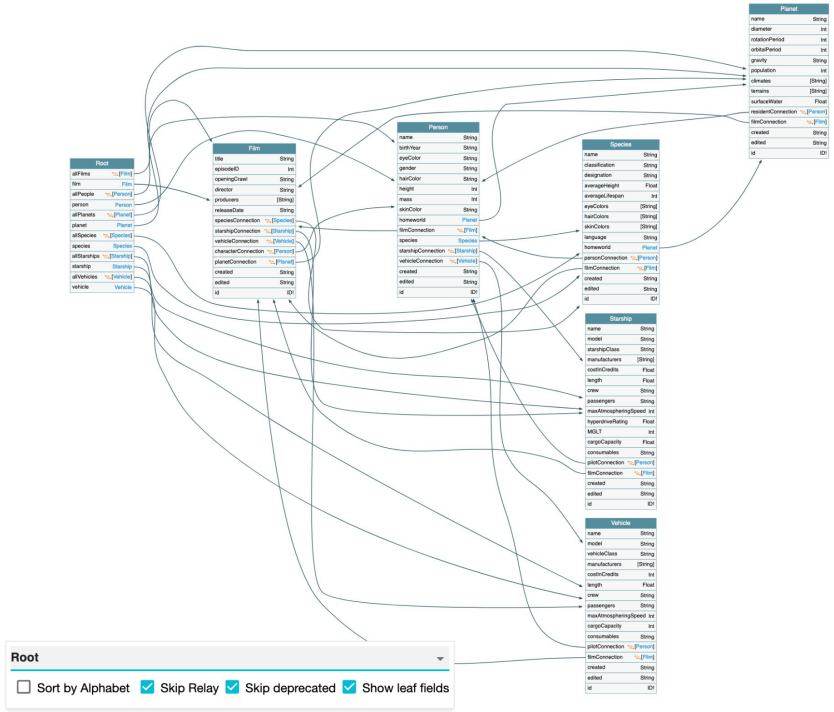
1 – GraphQL is a full-featured language...



escape BSIDES MILANO

Why is GraphQL vulnerable?

2 - GraphQL is a graph... with a lot of different paths leading to the same resource

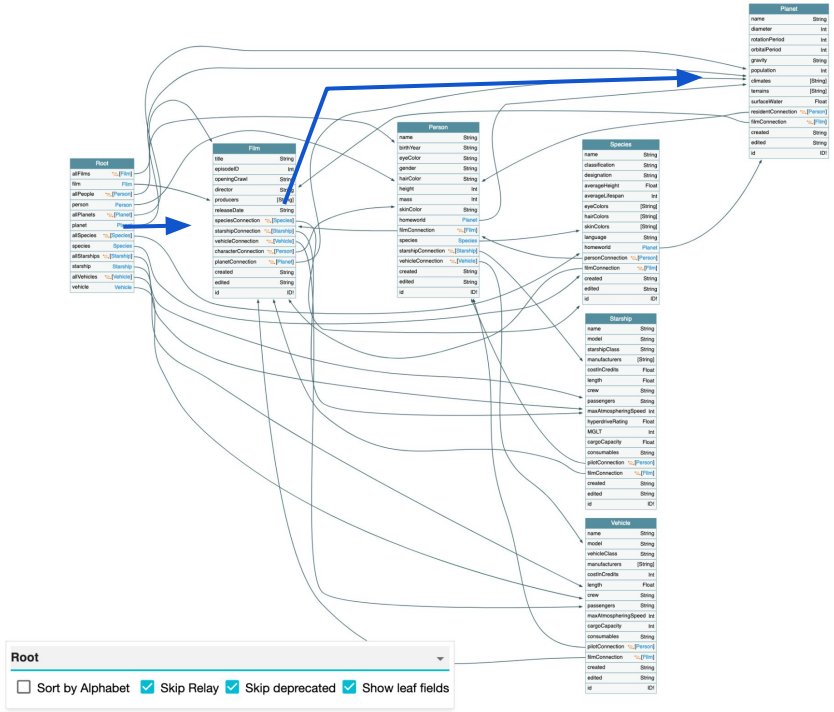


escape

BSIDES MILANO

Why is GraphQL vulnerable?

2 - GraphQL is a graph... with a lot of different paths leading to the same resource



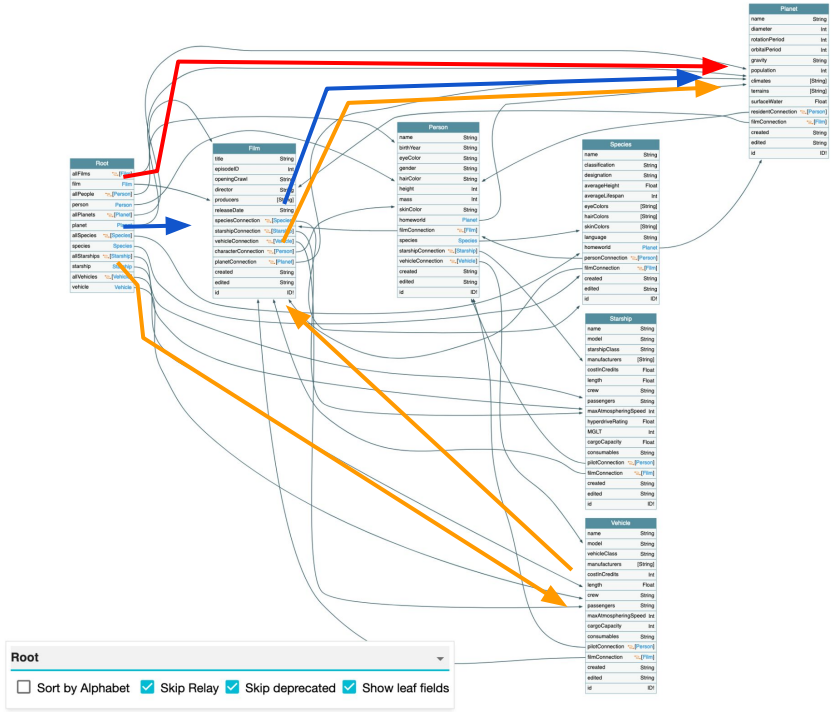
1. stripe_token

escape

BSIDES MILANO

Why is GraphQL vulnerable?

2 - GraphQL is a graph... with a lot of different paths leading to the same resource



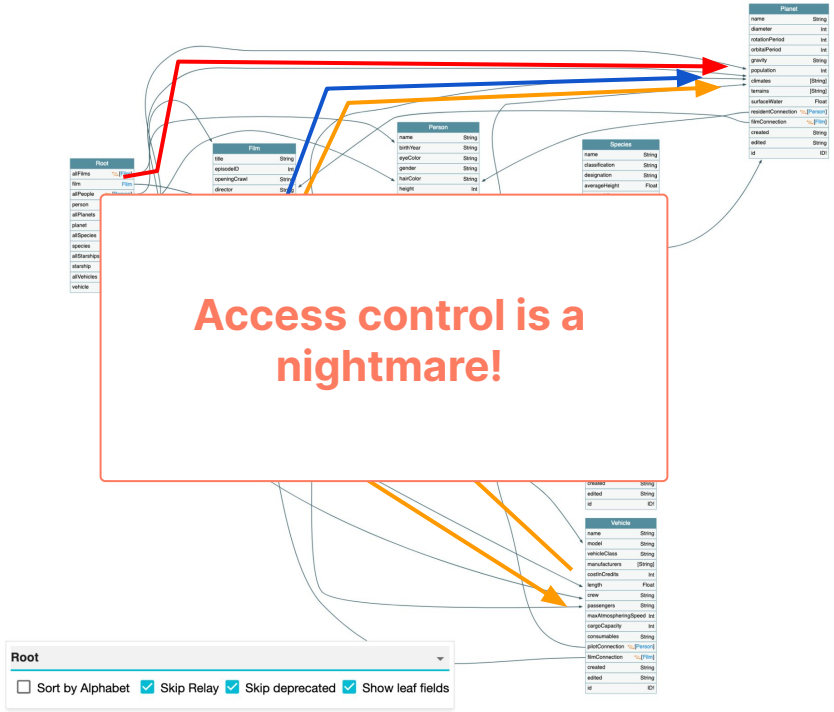
1. stripe_token

escape

BSIDES MILANO

Why is GraphQL vulnerable?

2 - GraphQL is a graph... with a lot of different paths leading to the same resource



1. stripe_token

escape

BSIDES MILANO

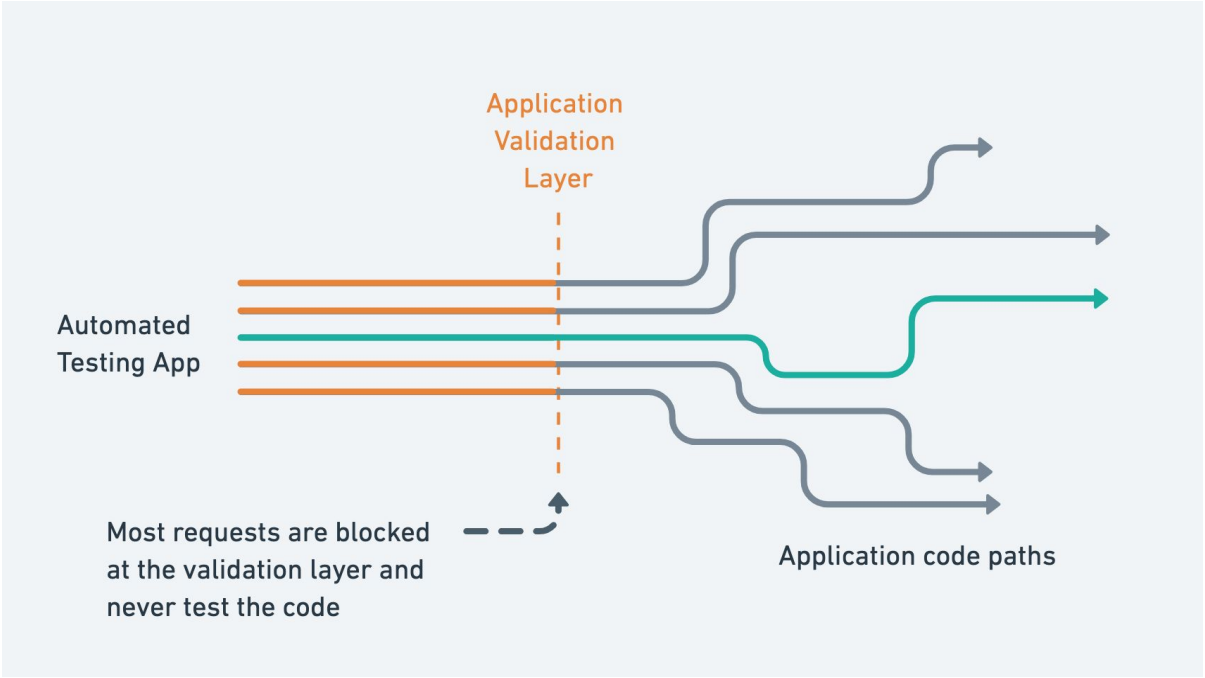
Why is is hard to test production GraphQL apps?

escape

BSIDES

The two problems when testing GraphQL APIs

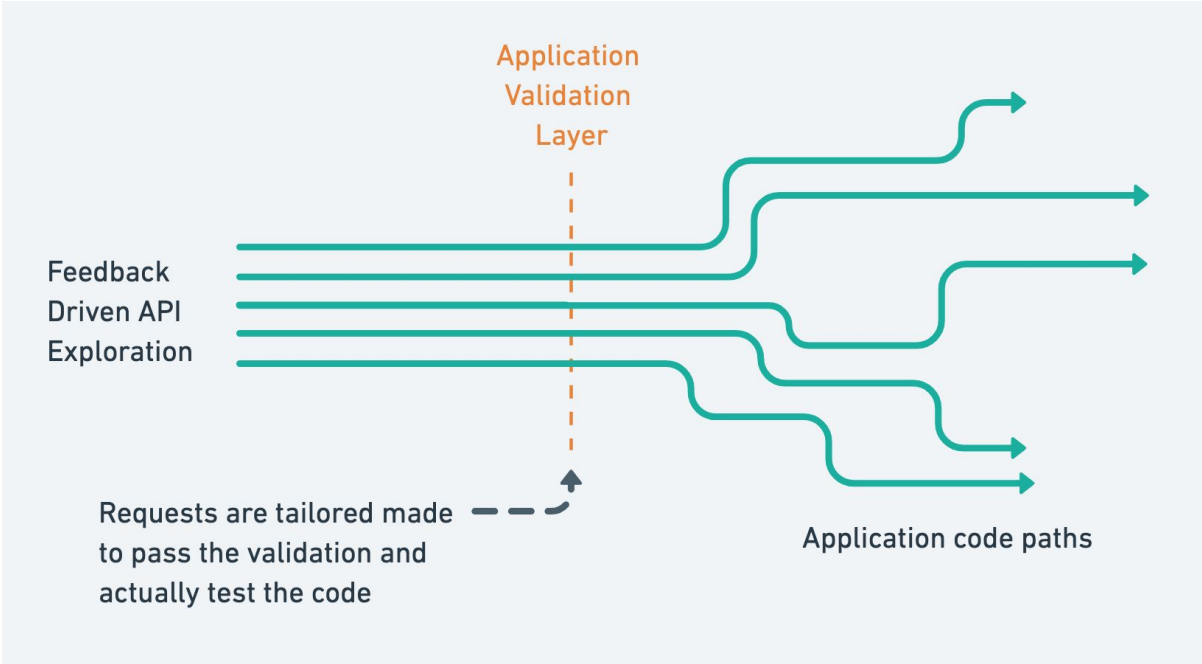
Problem 1 : Classic testing cannot assess the business logic of APIs



escape BSIDES MILANO

The two problems when testing GraphQL APIs

Solution 1 : Feedback-Driven API Exploration

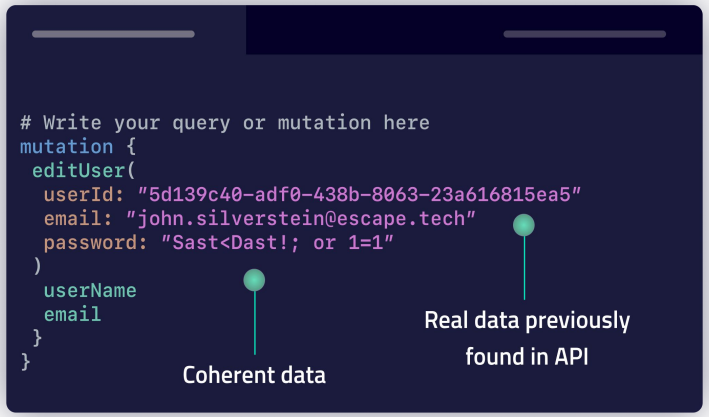


escape BSIDES MILANO

The two problems when testing GraphQL APIs

Solution 1 : Feedback-Driven API Exploration

Regular API Testing

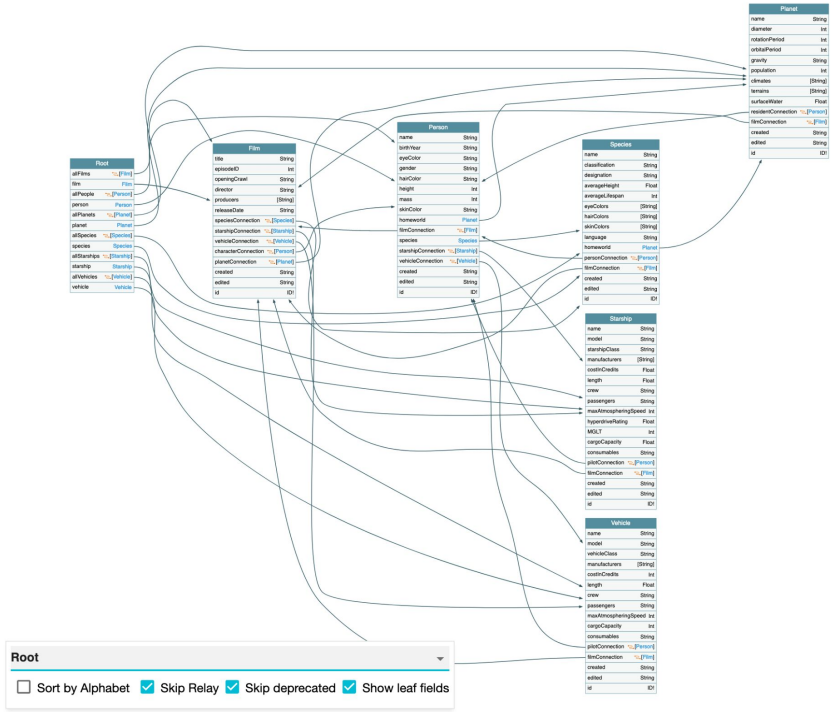


Feedback Driven API Exploration

escape BSIDES MILANO

The two problems when testing GraphQL APIs

Problem 2 : GraphQL is a Graph...



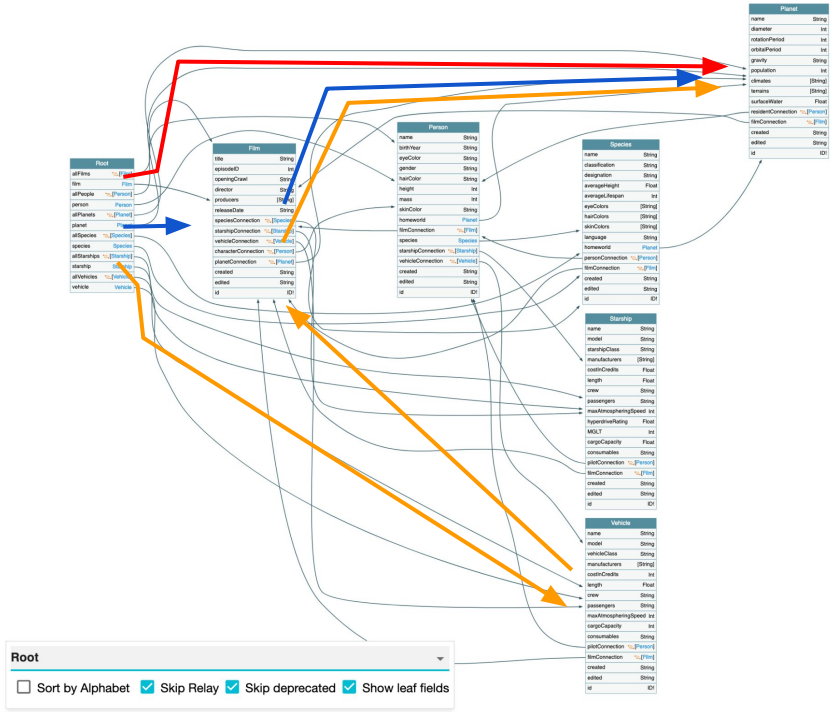
1. stripe_token

escape

BSIDES MILANO

The two problems when testing GraphQL APIs

Solution 2 : Recursively explore all paths in the graph



1. stripe_token

escape

BSIDES MILANO

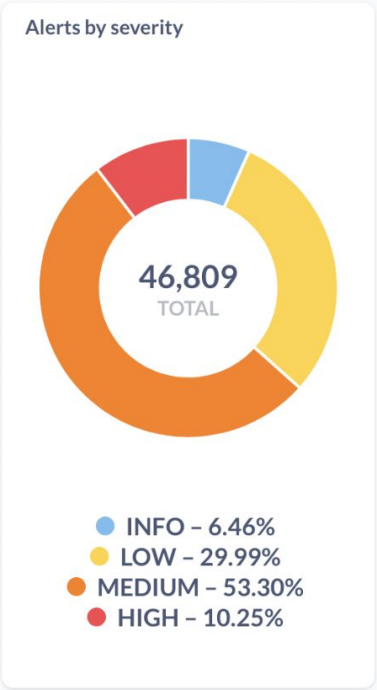
The findings

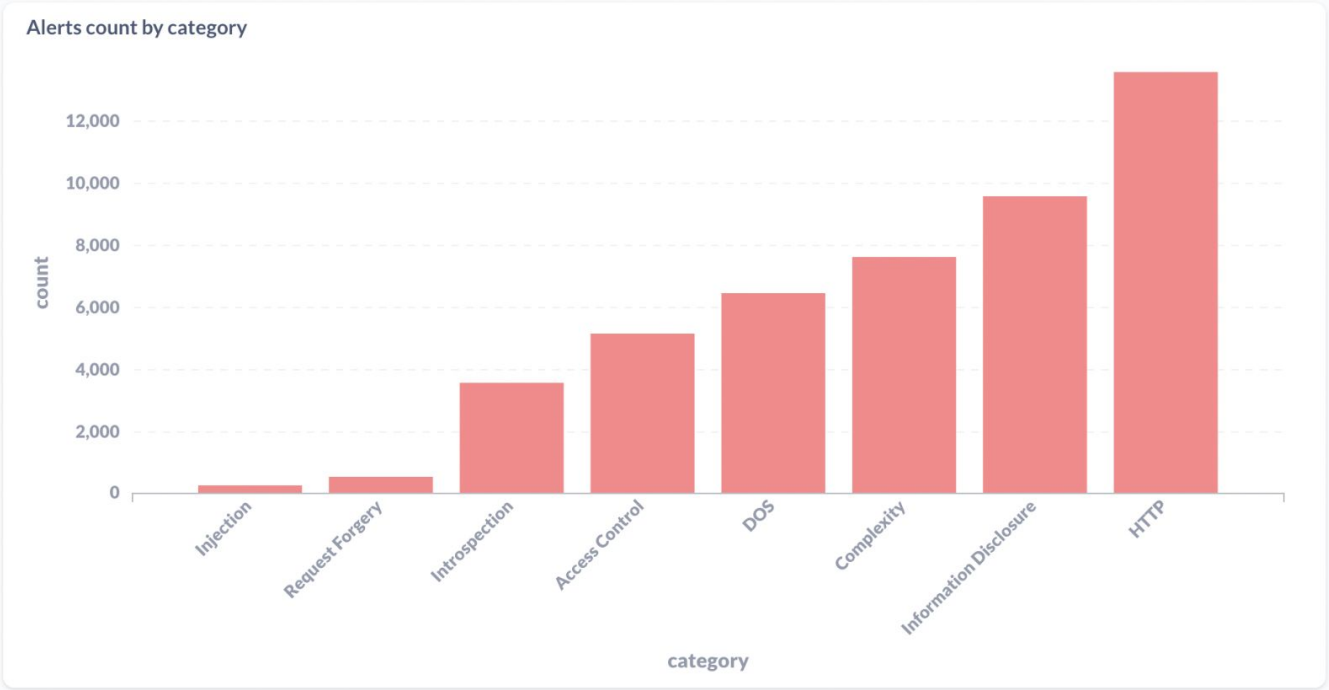
46,809

Security alerts found

escape

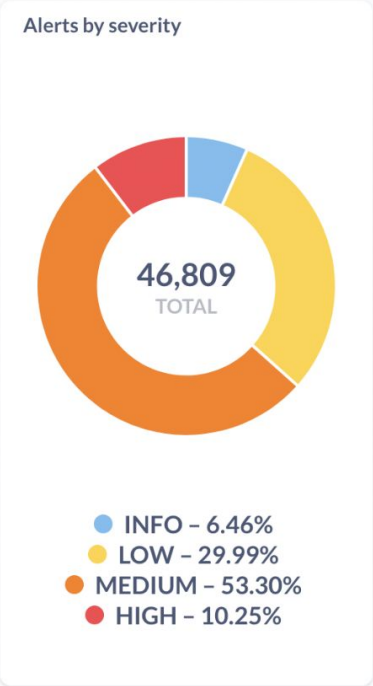
Which severity and category?

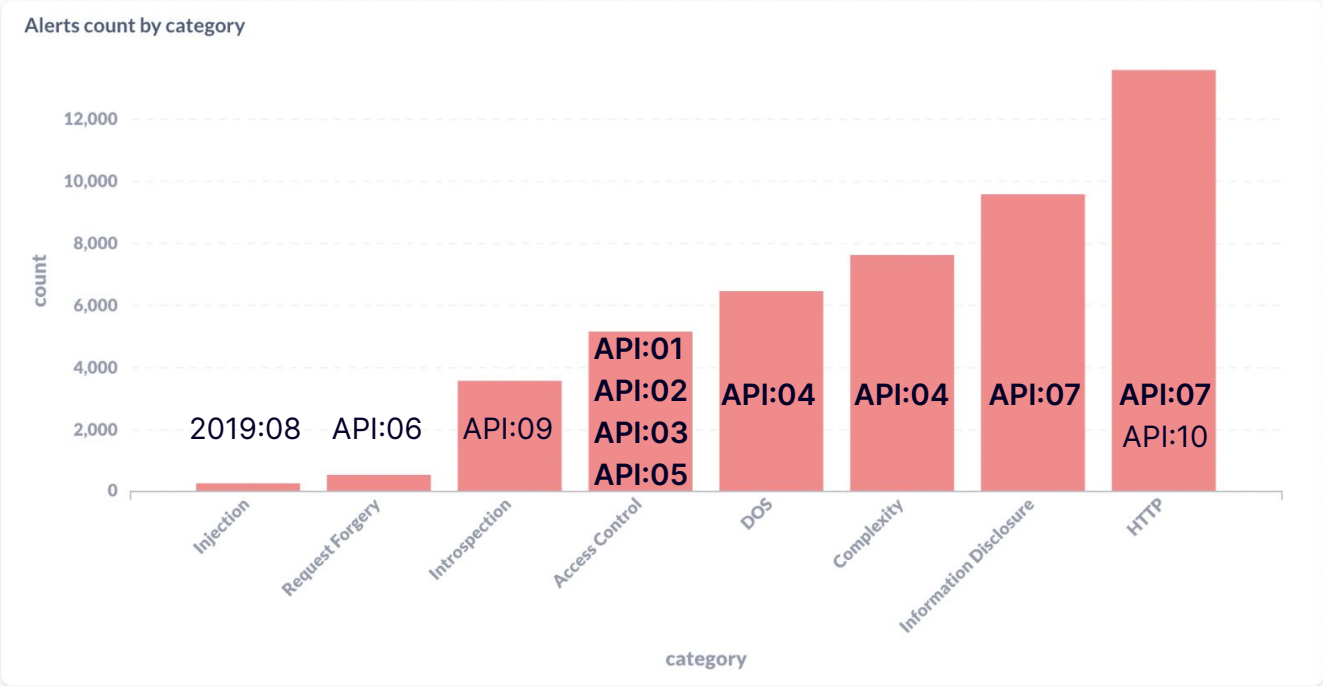




escape BSIDES MILANO

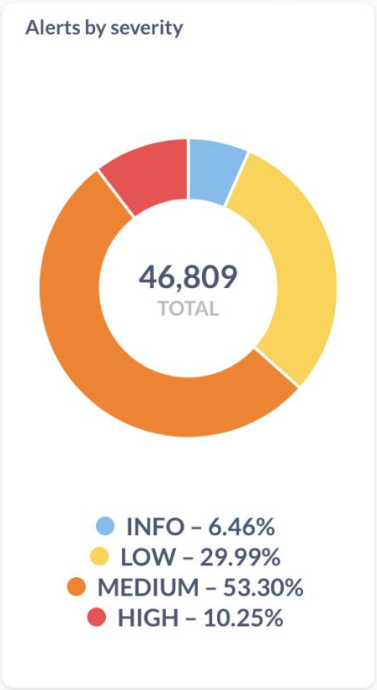
Which severity and category?

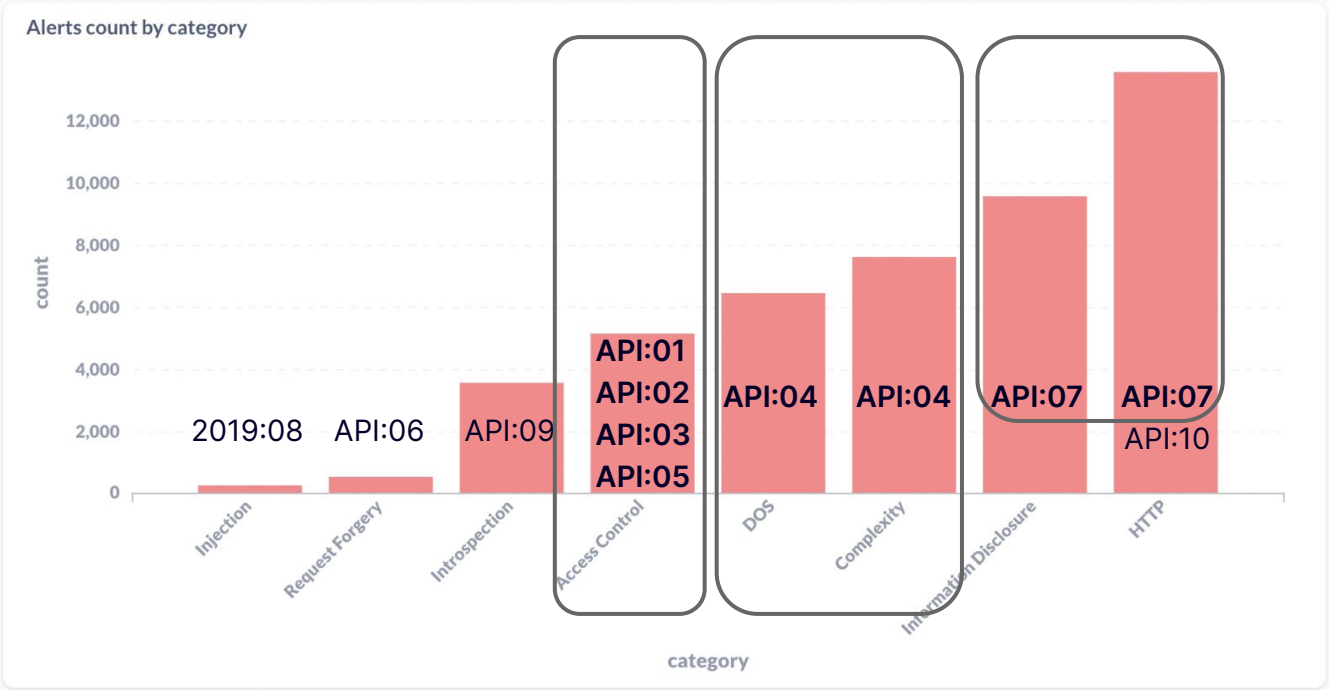




escape BSIDES MILANO

Which severity and category?





escape

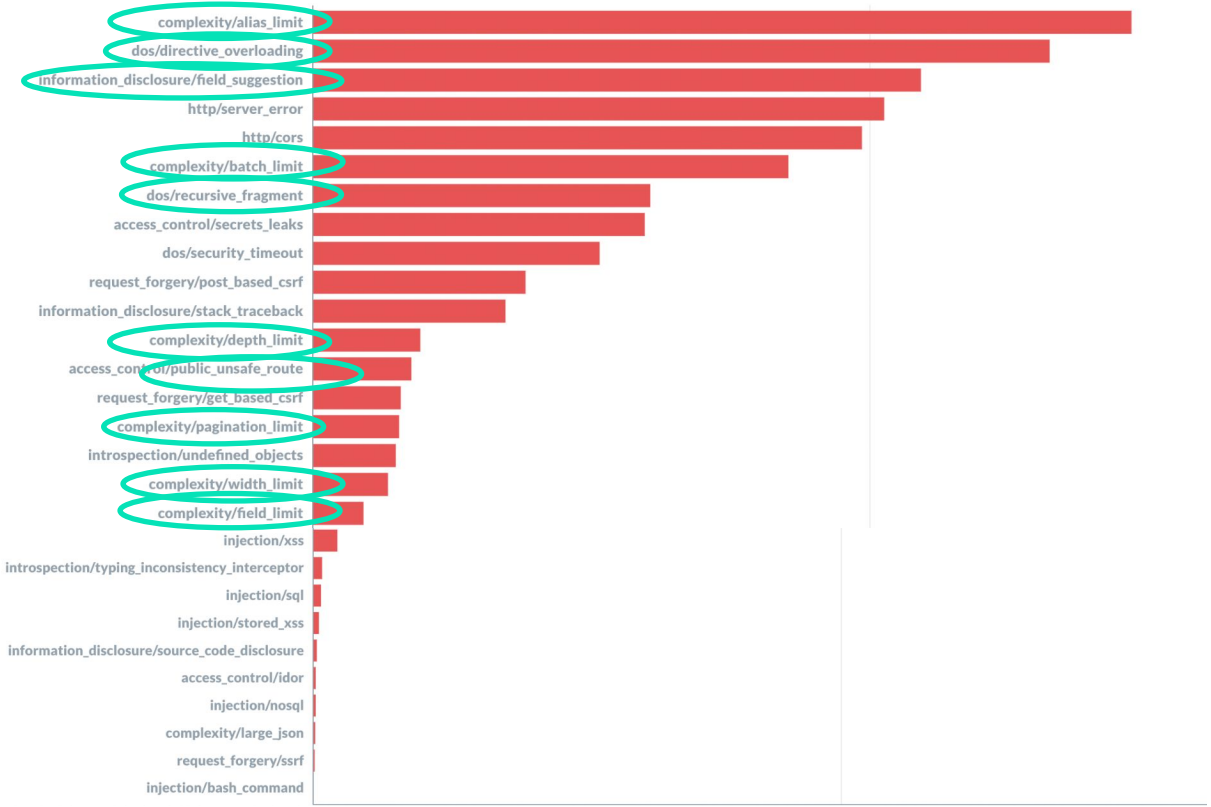
BSIDES MILANO

A lot are GraphQL-specific

escape

BSIDES

GraphQL-specific Vulnerabilities come first



escape

BSIDES MILANO

Top vulnerability 1: Bruteforcing API requests (API:04)

Batching: Multiple GraphQL Queries in one HTTP Request

```
query Hero {
  hero {
    name
  }
}

query Vilain {
  vilain {
    name
  }
}
```

Aliasing: The same fields multiple times with aliases

```
{
  empireHero: hero(episode: EMPIRE) {
    name
  }
  jediHero: hero(episode: JEDI) {
    name
  }
}
```

Top vulnerability 1: Bruteforcing API requests (API:04)

The problem: can be used to bypass rate limiting on login mutations = Bruteforce attack

```
{
  login1: login(email: user0@graphql.org, password: "password1") {
    username
    jwt
  }
  login2: login(email: user0@graphql.org, password: "password22") {
    username
    jwt
  }
  login3: login(email: user0@graphql.org, password: "password2211") {
    username
    jwt
  }
}
...
```


escape BSIDES MILANO

Top vulnerability 2: Denial of Service (API:04)

Fragments: a piece of logic that can be shared between multiple queries

```
fragment NameParts on Person {
  firstName
  lastName
}
```

Top vulnerability 2: Denial of Service (API:04)

The problem: What if a fragment calls itself? Oops, you get an infinite recursion loop.

```
{
  ...X
}

fragment X on Query {
  ...Y
}

fragment Y on Query {
  ...X
}
```

```
{
  "errors": [
    {
      "message": "Maximum call stack size exceeded",
      "extensions": {
        "code": "INTERNAL_SERVER_ERROR",
        "exception": {
          "stacktrace": [
            "RangeError: Maximum call stack size exceeded",
            "    at Map.get (<anonymous>)",
            "    at getReferencedFieldsAndFragmentNames (/usr/s
/app/node_modules/graphql/validation/rules
/OverlappingFieldsCanBeMergedRule.js:468:45)",
```


escape BSIDES MILANO

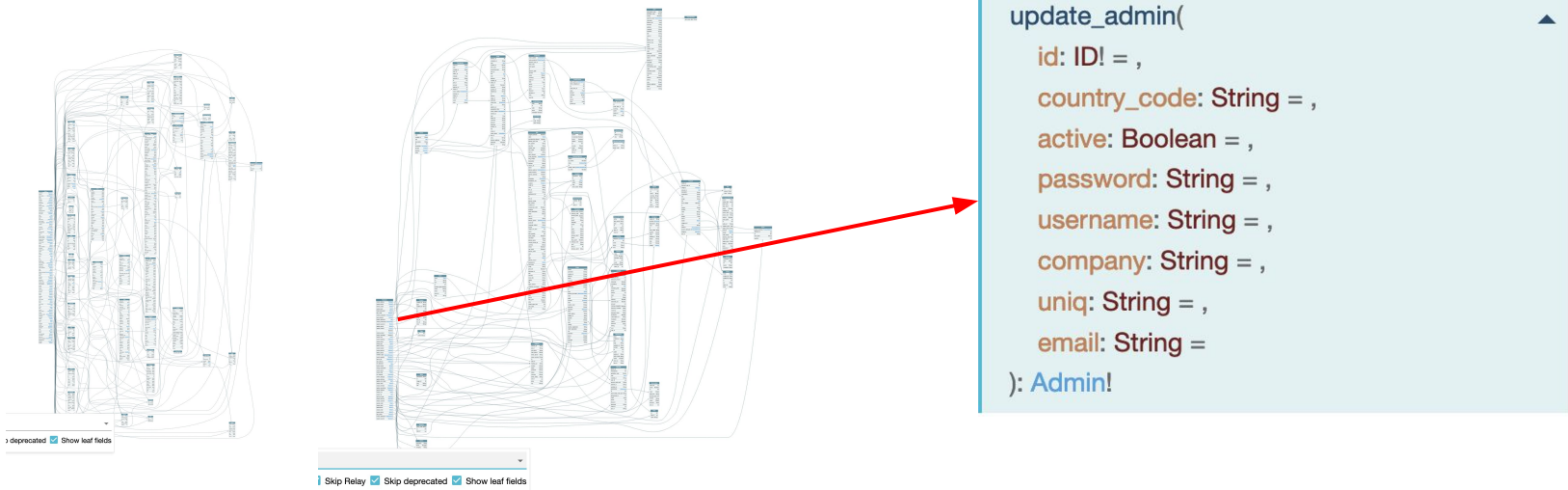
Top vulnerability 3: Internal API Schema Leak (API:07)

The problem: Endpoints with disabled introspection still leaks underlying schema through field suggestion using

```
{
  "errors": [
    {
      "message": "Cannot query field \"createSesion\" on
        type \"RootMutation\". Did you mean
        \"createSession\", \"createUser\",
        \"createFile\", or \"createImage\"?",
      "category": "graphql"
    }
  ]
}
```

Top vulnerability 3: Internal API Schema Leak (API:07)

The problem: Endpoints with disabled introspection still leaks underlying API Schema through field suggestion using open source tool Clairvoyance, anybody can build back the full Schema.

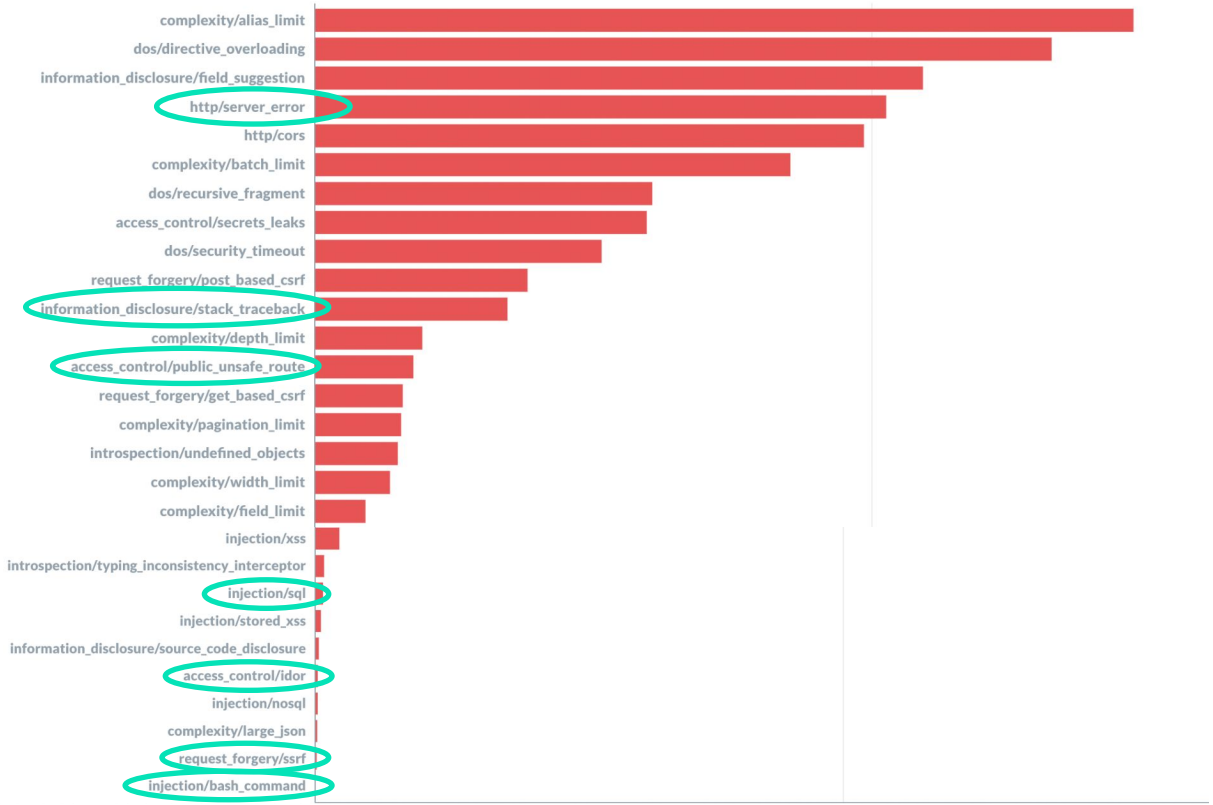


Classic API vulnerabilities are still there

escape

BSIDES

HTTP Errors, Access Control Issues, Injections



escape BSIDES MILANO

API Errors and Stacktraces disclosing internal data (API:07)

The problem: Stack Traces give too much information on what's vulnerable in your app 👀

```
      "extensions": {
        "code": "GRAPHQL_VALIDATION_FAILED",
        "exception": {
          "stacktrace": [
            "GraphQLError: Boolean cannot represent a non
boolean value: \"'\"",
            "    at GraphQLScalarType.parseLiteral
(/usr/local/app/common/temp/node_modules/.pnpm/@apollo
/gateway@0.6.1/node_modules/graphql
/type/scalars.js:271:13)",
            "    at isValidValueNode (/usr/local/app/common
/temp/node_modules/.pnpm/graphql@16.5.0/node_modules
```

Prototype Pollution In

@Apollo/gateway

HIGH 7.3

Updated at: 11 months ago

Published: 3 years ago

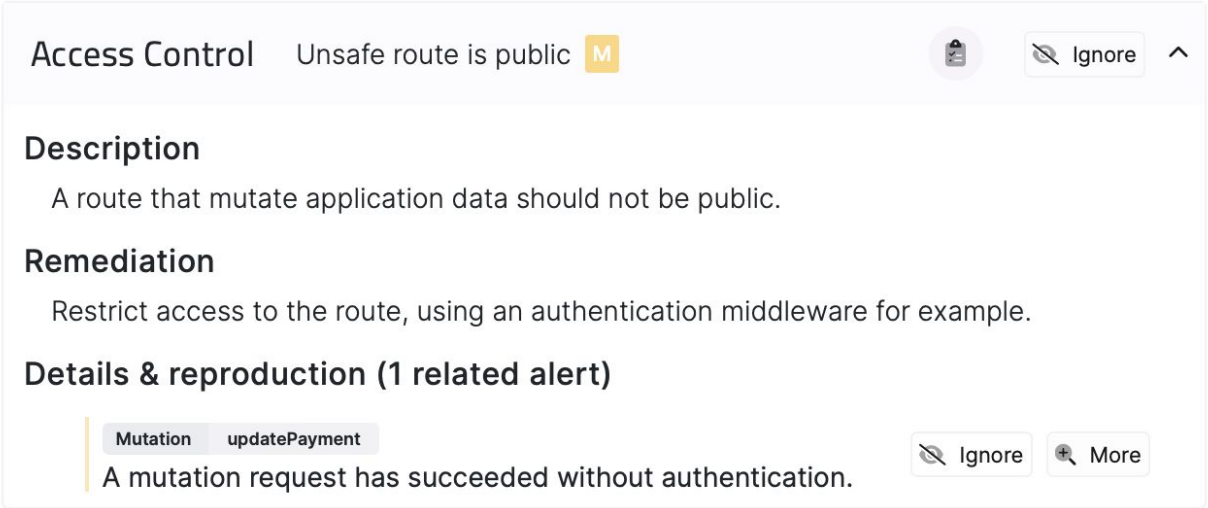
Package: **@apollo/gateway** (npm)
Affected Version: **< 0.6.2**
Patch Version: **0.6.2**

(12% endpoints misconfigured)

escape BSIDES MILANO

Access Control issues (API:01,02,03,05)

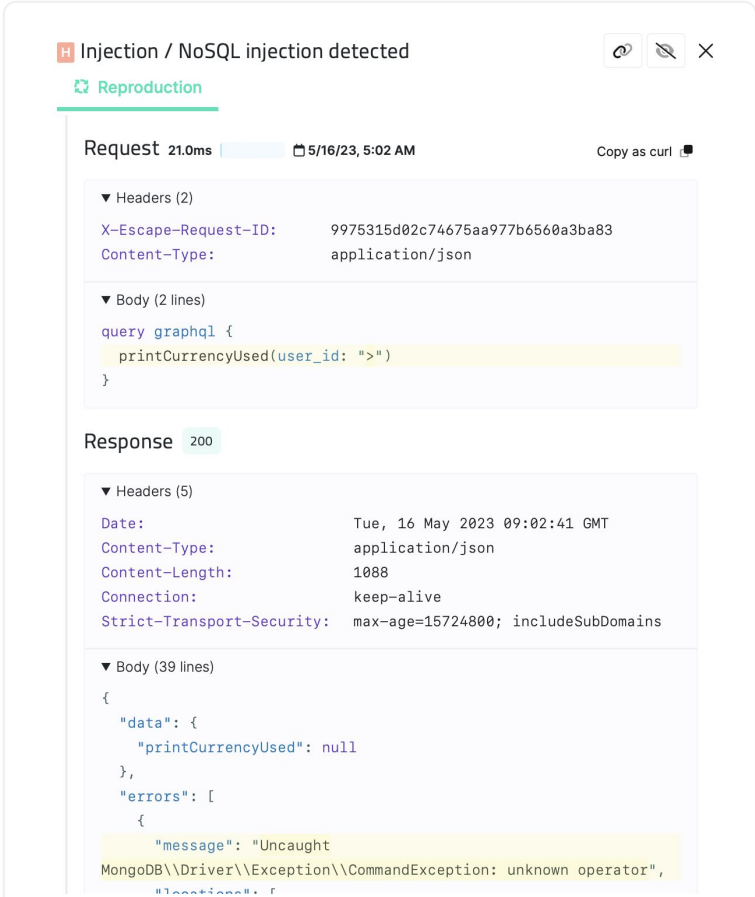
The problem: Most routes writing data (mutations) are not supposed to be accessible without authentication



escape BSIDES MILANO

Injections are still there (API 2019:08)

- 16 SQL or NoSQL injections
- 1 bash command injection
- 3 Server Side Request Forgery



escape BSIDES MILANO

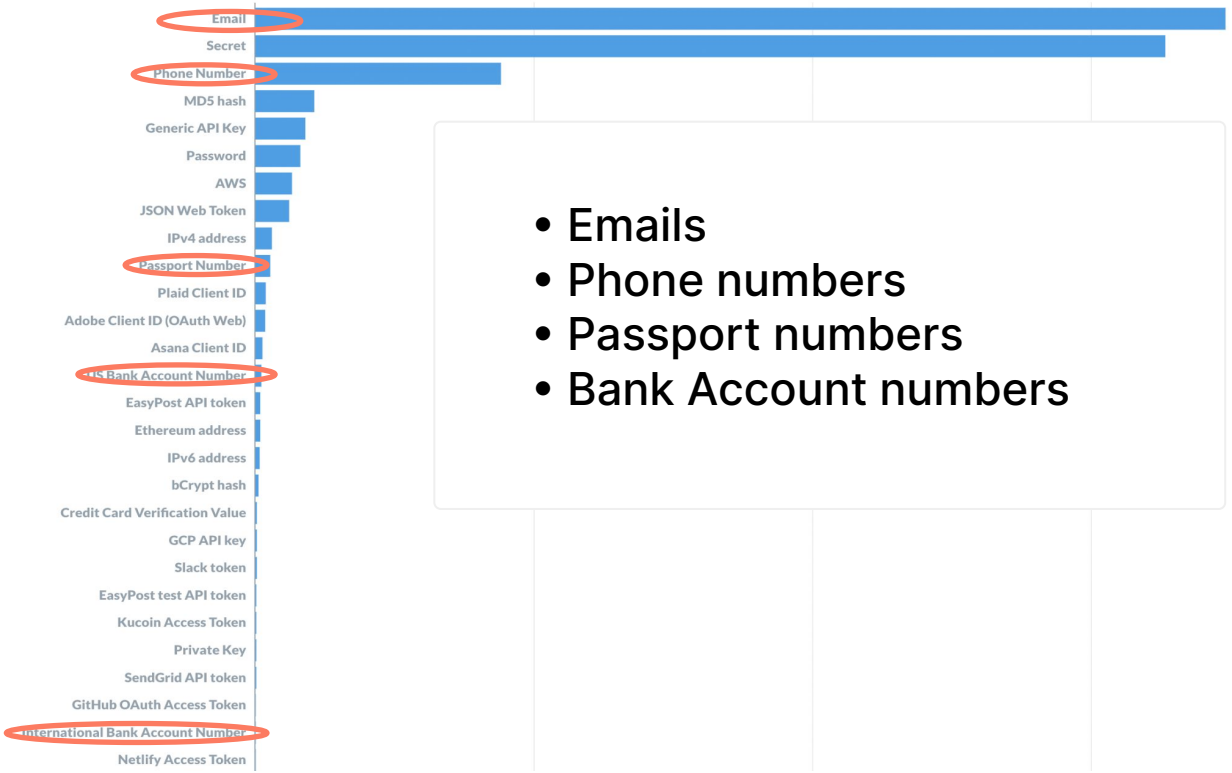
Sensitive data made public!

4,493 potential PII and token leaks found.

- Email
- Secret
- Phone Number
- MD5 hash
- Generic API Key
- Password
- AWS
- JSON Web Token
- IPv4 address
- Passport Number
- Plaid Client ID
- Adobe Client ID (OAuth Web)
- Asana Client ID
- US Bank Account Number
- EasyPost API token
- Ethereum address
- IPv6 address
- bCrypt hash
- Credit Card Verification Value
- GCP API key
- Slack token
- EasyPost test API token
- Kucoin Access Token
- Private Key
- SendGrid API token
- GitHub OAuth Access Token
- International Bank Account Number
- Netlify Access Token

escape BSIDES MILANO

4,493 potential PII and token leaks found.

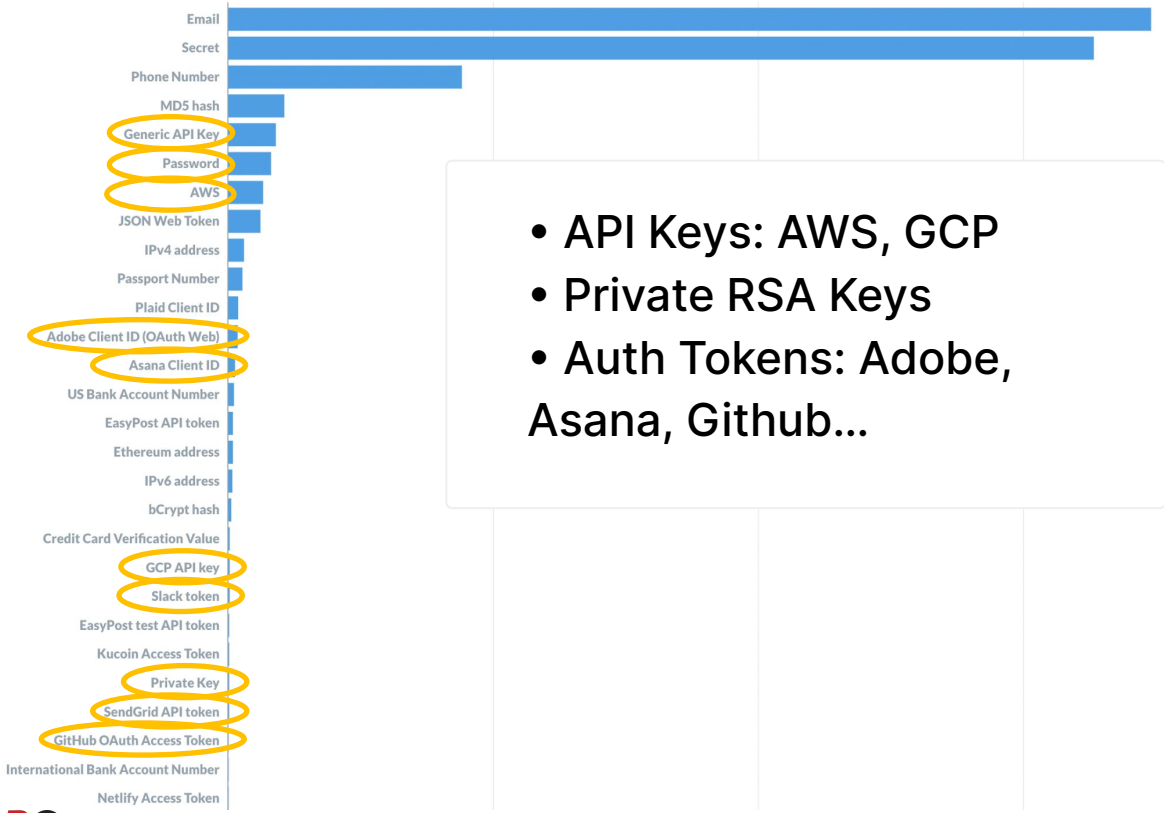


- Emails
- Phone numbers
- Passport numbers
- Bank Account numbers

escape

BSIDES MILANO

4,493 potential PII and token leaks found.



- API Keys: AWS, GCP
- Private RSA Keys
- Auth Tokens: Adobe, Asana, Github...

escape

BSIDES MILANO

How does OWASP TOP 10 reflect in prod-ready GraphQL apps?

DOS & Complexity issues
Due to GraphQL-specific features

API:04: Unrestricted Resource Consumption

escape BSIDES

How does OWASP TOP 10 reflect in prod-ready GraphQL apps?

DOS & Complexity issues

Due to GraphQL-specific features

→ **API:04**: Unrestricted Resource Consumption

Access Control, Authentication and Authorization:

Very common because of the graph nature of GraphQL

→ **API:01**: Broken Object Level Authorization (BOLA)
API:02: Broken Authentication
API:03: Broken Object Property Level Authorization (BOPLA)
API:05: Broken Function Level Authorization (BFLA)

escape BSIDES

How does OWASP TOP 10 reflect in prod-ready GraphQL apps?

DOS & Complexity issues

Due to GraphQL-specific features

→ **API:04**: Unrestricted Resource Consumption

Access Control, Authentication and Authorization:

Very common because of the graph nature of GraphQL

→ **API:01**: Broken Object Level Authorization (BOLA)
API:02: Broken Authentication
API:03: Broken Object Property Level Authorization (BOPLA)
API:05: Broken Function Level Authorization (BFLA)

Security Misconfiguration and Information Disclosure

By default in GraphQL Server Engines

→ **API:07**: Security Misconfiguration

escape BSIDES

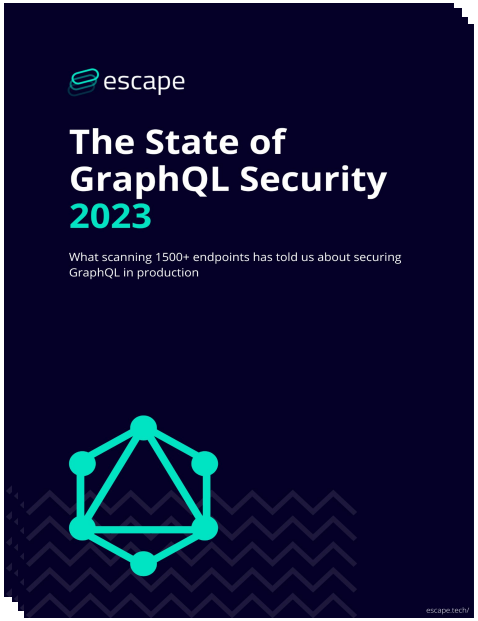
Conclusion, using only unauthenticated API calls:

- We found 130k+ public GraphQL APIs into the wild and scanned **1599** of them
- Highlighted numerous vulnerabilities, most frequently GraphQL Specific, but not only
- **Access control flaws** and **secret leaks** are *very* frequent in GraphQL

escape BSIDES

Tristan Kalos
Co-founder & CEO
tristan@escape.tech
@tristankalos

Antoine Carossio
Co-founder & CTO
antoine@escape.tech
@iCarossio



https://escape.tech/resources/state-of-graphql-security-2023/

escape BSIDES MILANO

Two more things...

GraphQL Armor

The missing free & open source security middleware for GraphQL applications

escape.tech/graphql-armor

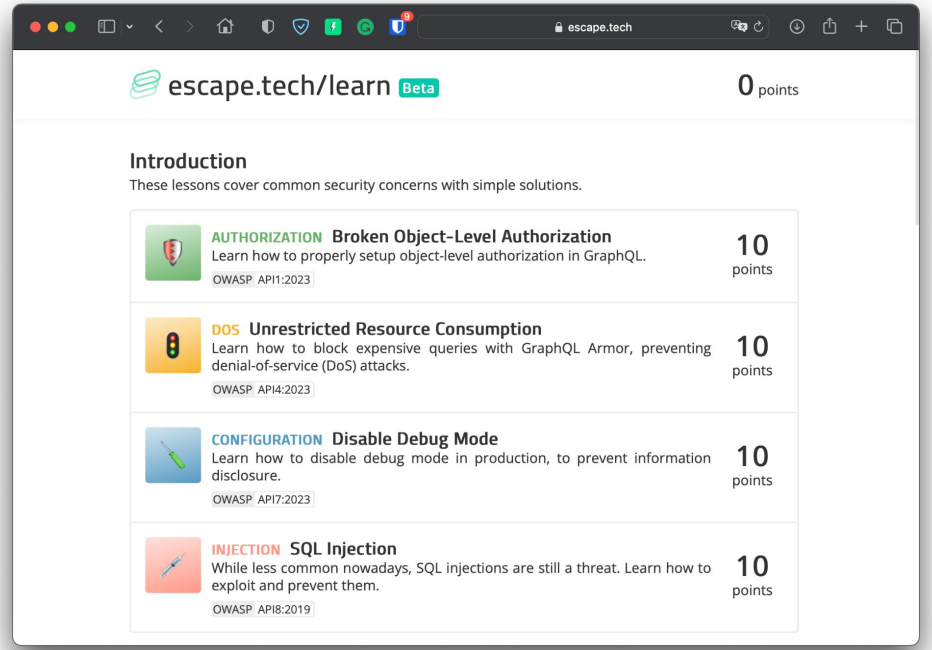
Provides out-of-the-box security best practices:

- Aliases Limit
- Character Limit
- Cost Limit
- Depth Limit
- Directives Limit
- Field Suggestion

escape BSIDES MILANO

Escape learning center

The first free & open-source security learning platform dedicated to GraphQL developers.



escape.tech/learn

escape BSIDES MILANO

We are hiring

to make DAST cool again.

jobs.escape.tech/

Thank you

We are hiring

to bring cybersecurity into all developers' hands.

jobs.escape.tech/