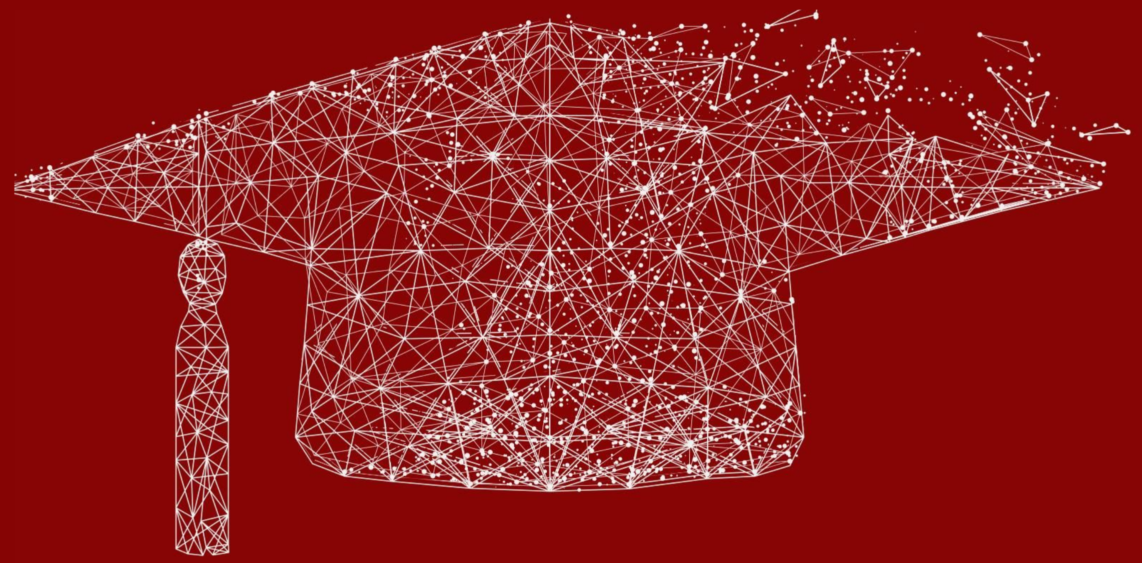
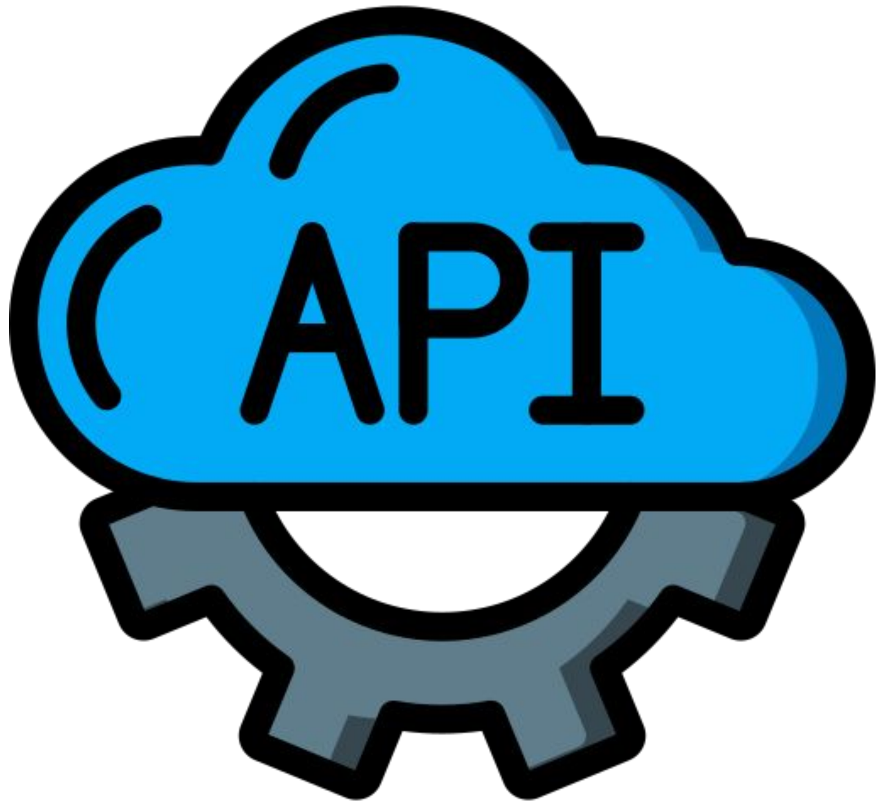
Club EH Root-Me - 12

Application Programming Interface (API) Testing



Sommaire

Introduction aux API

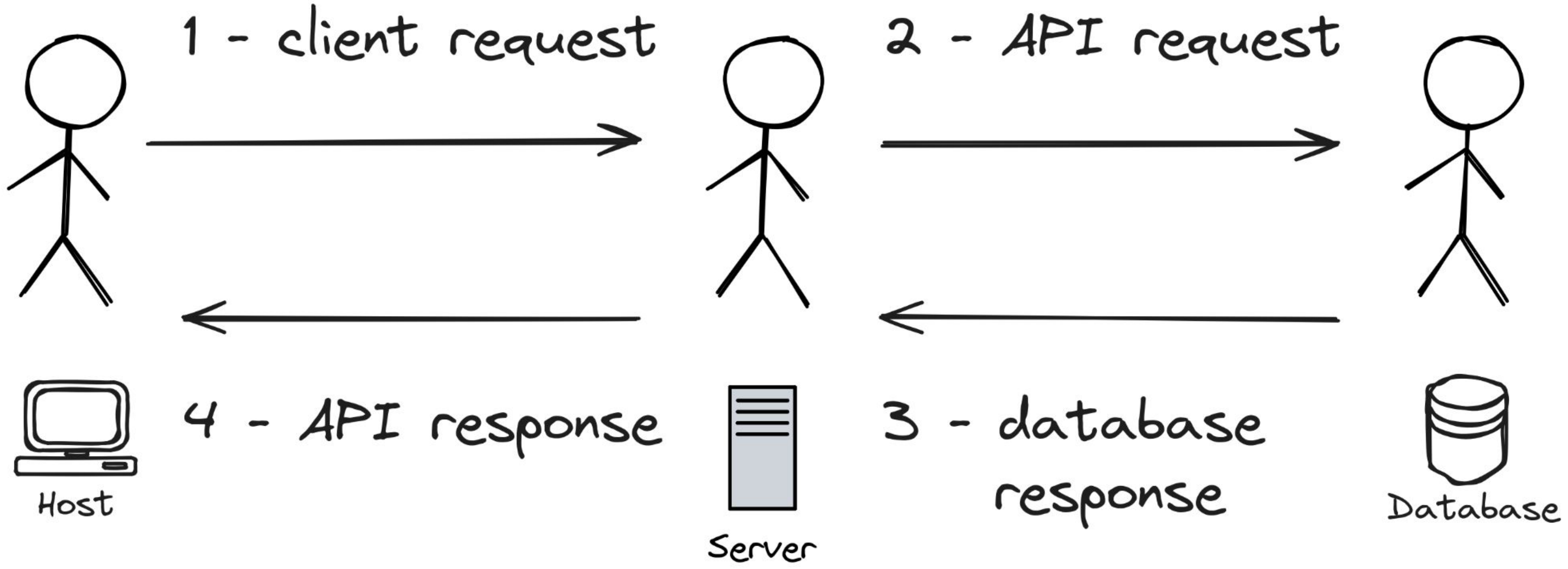


Introduction aux API

- Mécanisme défini pour uniformiser un échange de données entre deux entités.
- Peut être utilisé pour **l'authentification**, **l'autorisation** et **l'échange de données**.
- Différents types d'API (REST, SOAP, RPC, GraphQL…).
- Versioning, documentation et support multiplateforme.

Introduction aux API

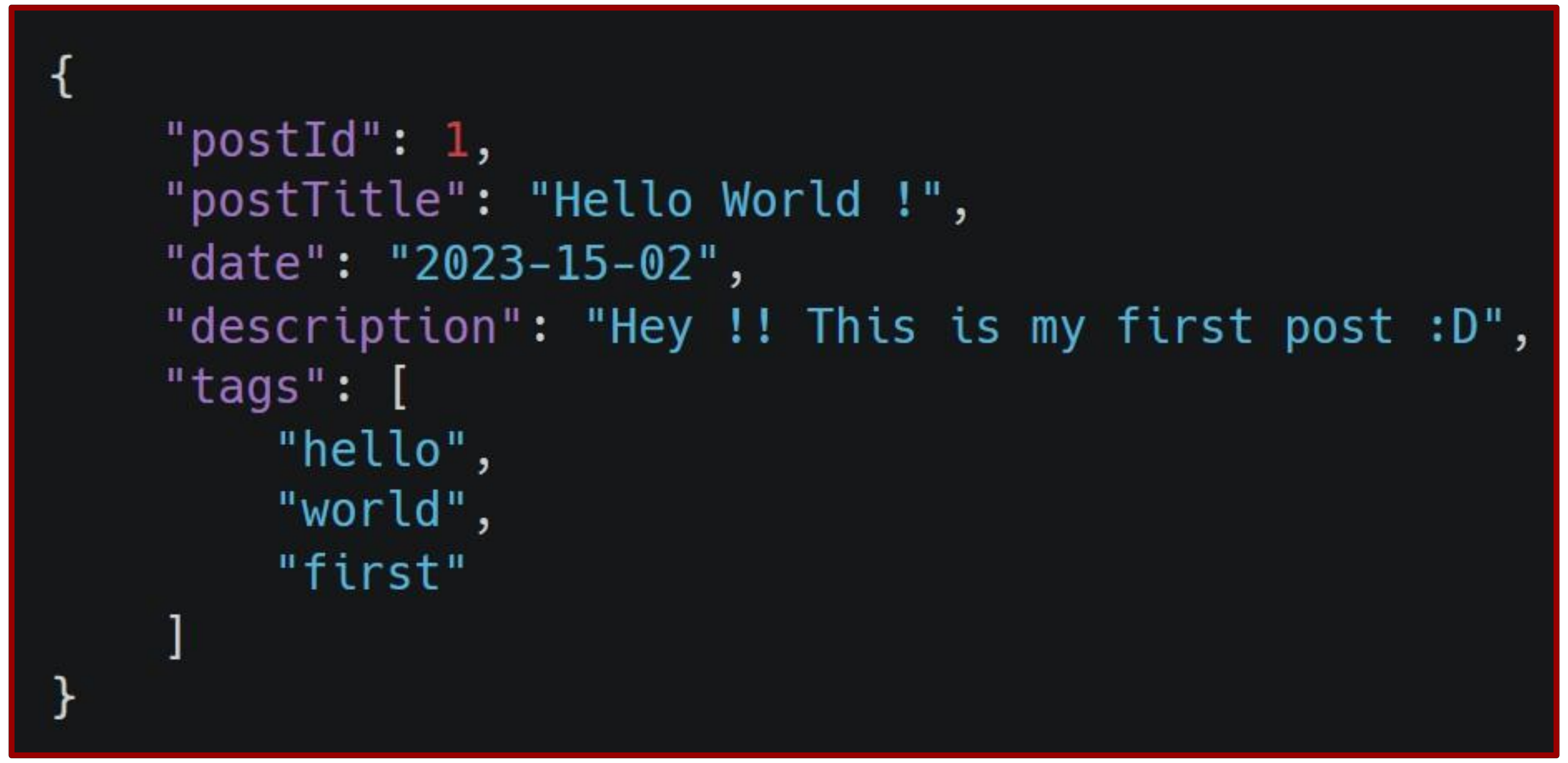
Introduction aux API



Introduction aux API

Qu'est-ce que JSON ?

- JavaScript Object Notation (JSON)
  - Format de données texte pour stocker des informations structurées ;
  - Syntaxe clé-valeur ;
  - Données typées (*string*, *int*, *booléen*, *array, etc.*)
  - Souvent utilisé dans les applications web (API) ;
  - Alternative au format XML ou YAML.

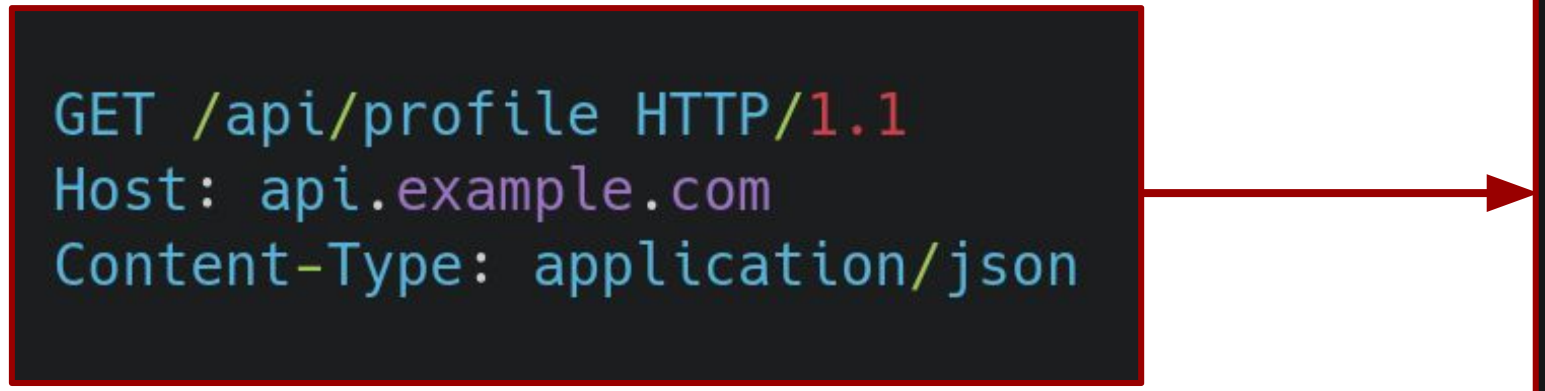


```
{
    "postId": 1,
    "postTitle": "Hello World !",
    "date": "2023-15-02",
    "description": "Hey !! This is my first post :D",
    "tags": [
        "hello",
        "world",
        "first"
    ]
}
```

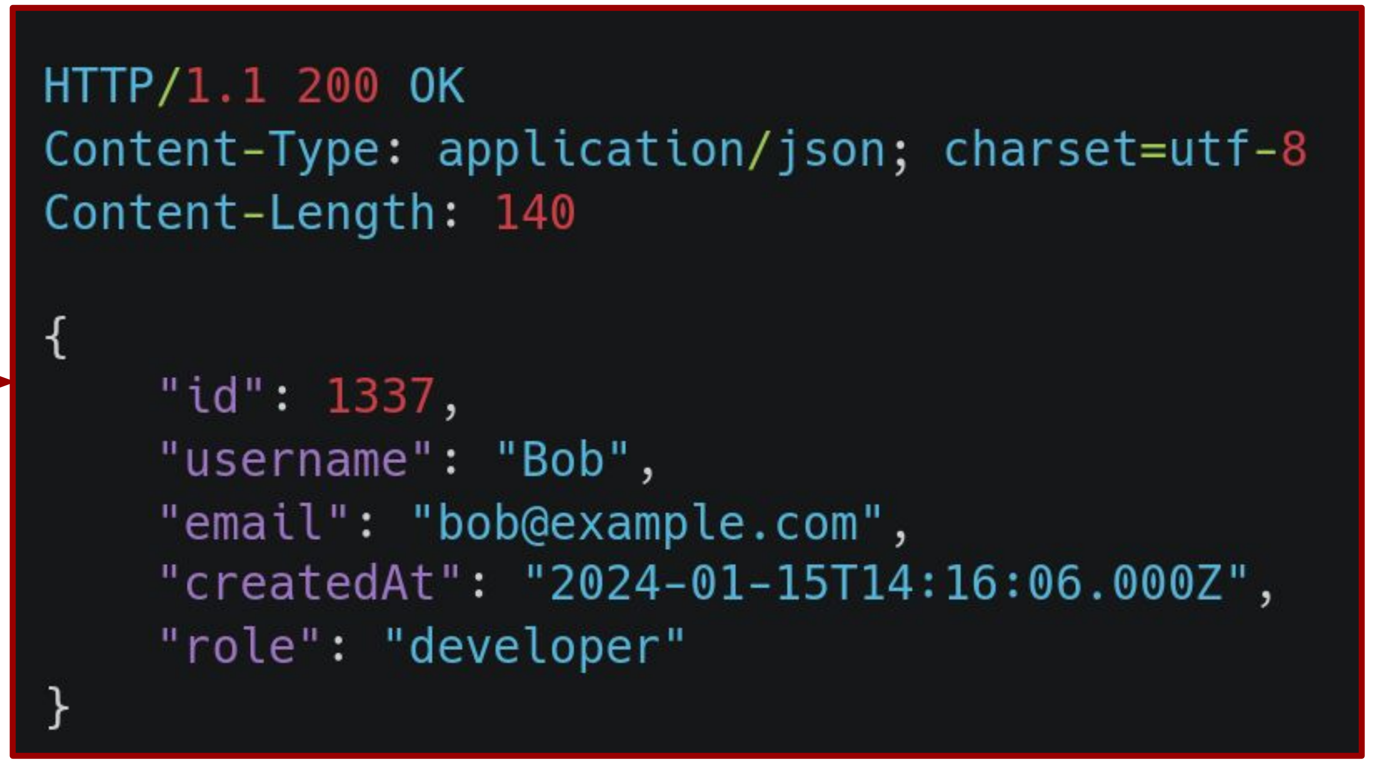
Introduction aux API

API - REST

- Points importants :
  - Multiples endpoints ;
  - Under-Fetching / Over-Fetching ;
  - Basées sur les conventions de communication HTTP (**GET**, **POST**, **PUT**, **DELETE**, …) ;
  - Largement supportées par les langages et framework existants.



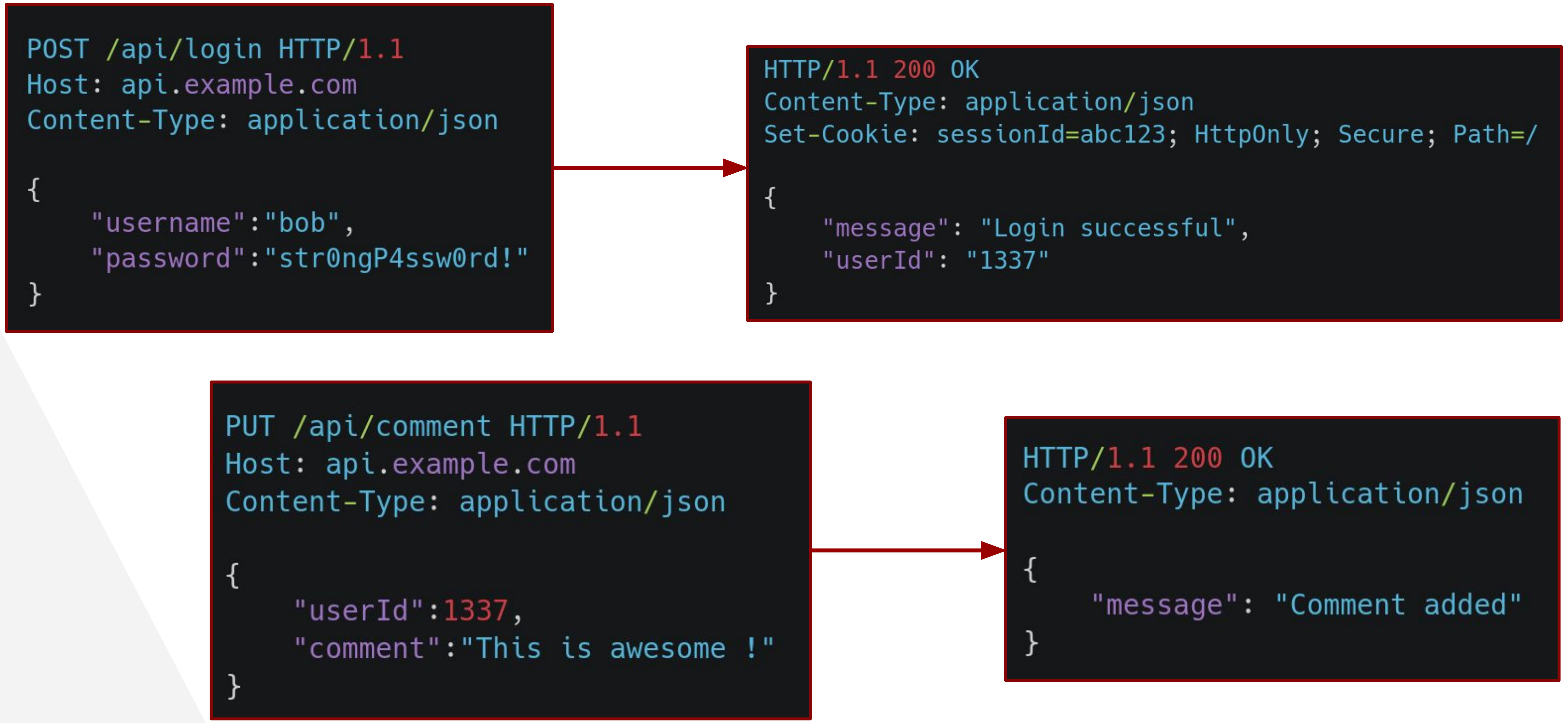
```
GET /api/profile HTTP/1.1
Host: api.example.com
Content-Type: application/json
```



```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Content-Length: 140

{
    "id": 1337,
    "username": "Bob",
    "email": "bob@example.com",
    "createdAt": "2024-01-15T14:16:06.000Z",
    "role": "developer"
}
```

Introduction aux API

API - REST



```
POST /api/login HTTP/1.1
Host: api.example.com
Content-Type: application/json

{
    "username":"bob",
    "password":"str0ngP4ssw0rd!"
}
```

```
HTTP/1.1 200 OK
Content-Type: application/json
Set-Cookie: sessionId=abc123; HttpOnly; Secure; Path=/

{
    "message": "Login successful",
    "userId": "1337"
}
```

```
PUT /api/comment HTTP/1.1
Host: api.example.com
Content-Type: application/json

{
    "userId":1337,
    "comment":"This is awesome !"
}
```

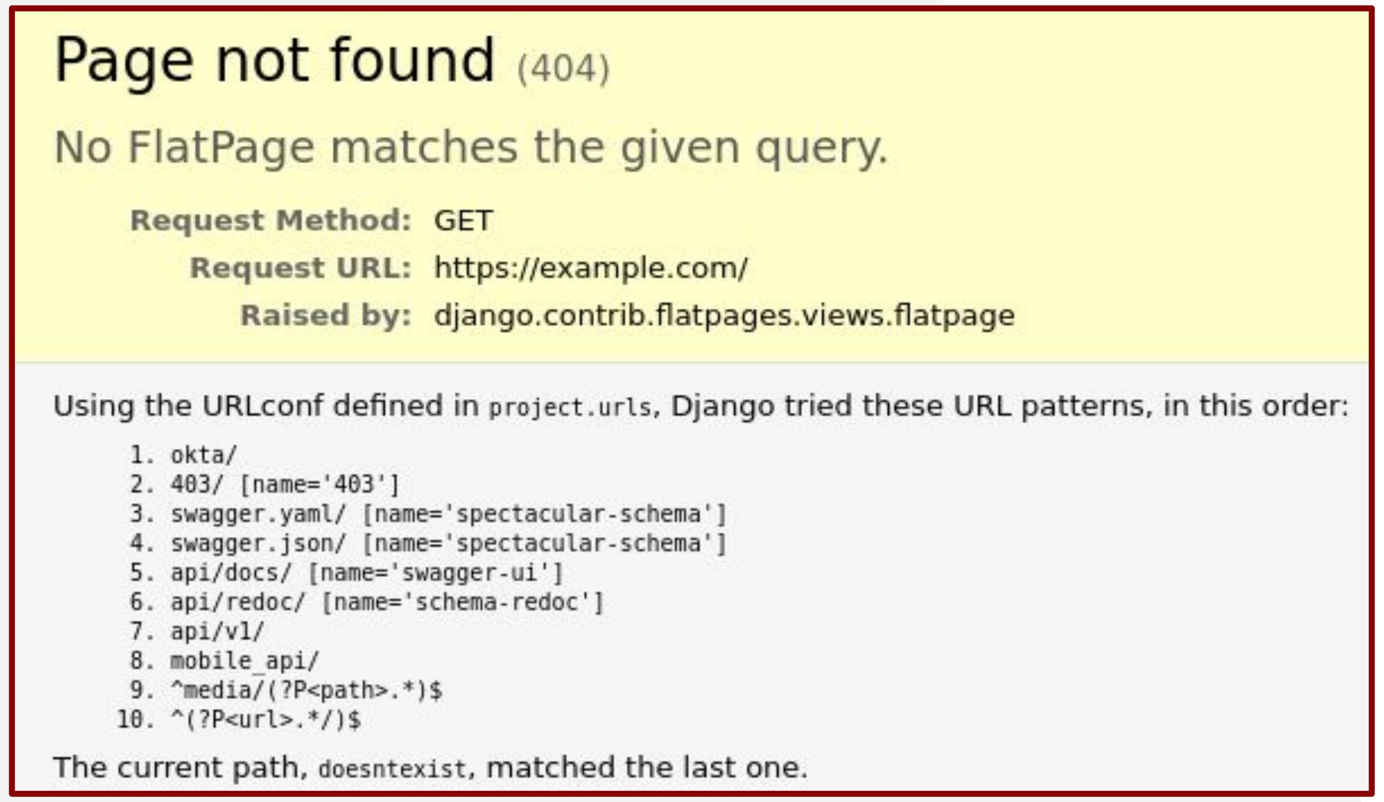
```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "message": "Comment added"
}
```

Phase de reconnaissance

Reconnaissance – Où ?

- Endpoint
  - /api/
  - /api/v1/
  - /api?wsdl
  - /openapi.json
  - /swagger
  - /swagger.json
  - SecLists
- Documentation
  - Swagger
  - Github
  - PDF
  - Endpoints
    - /docs
    - /api-docs
    - ...

- Identifier les méthodes HTTP acceptées
- Identifier les formats de requête acceptés (Content-Type)



Page not found (404)

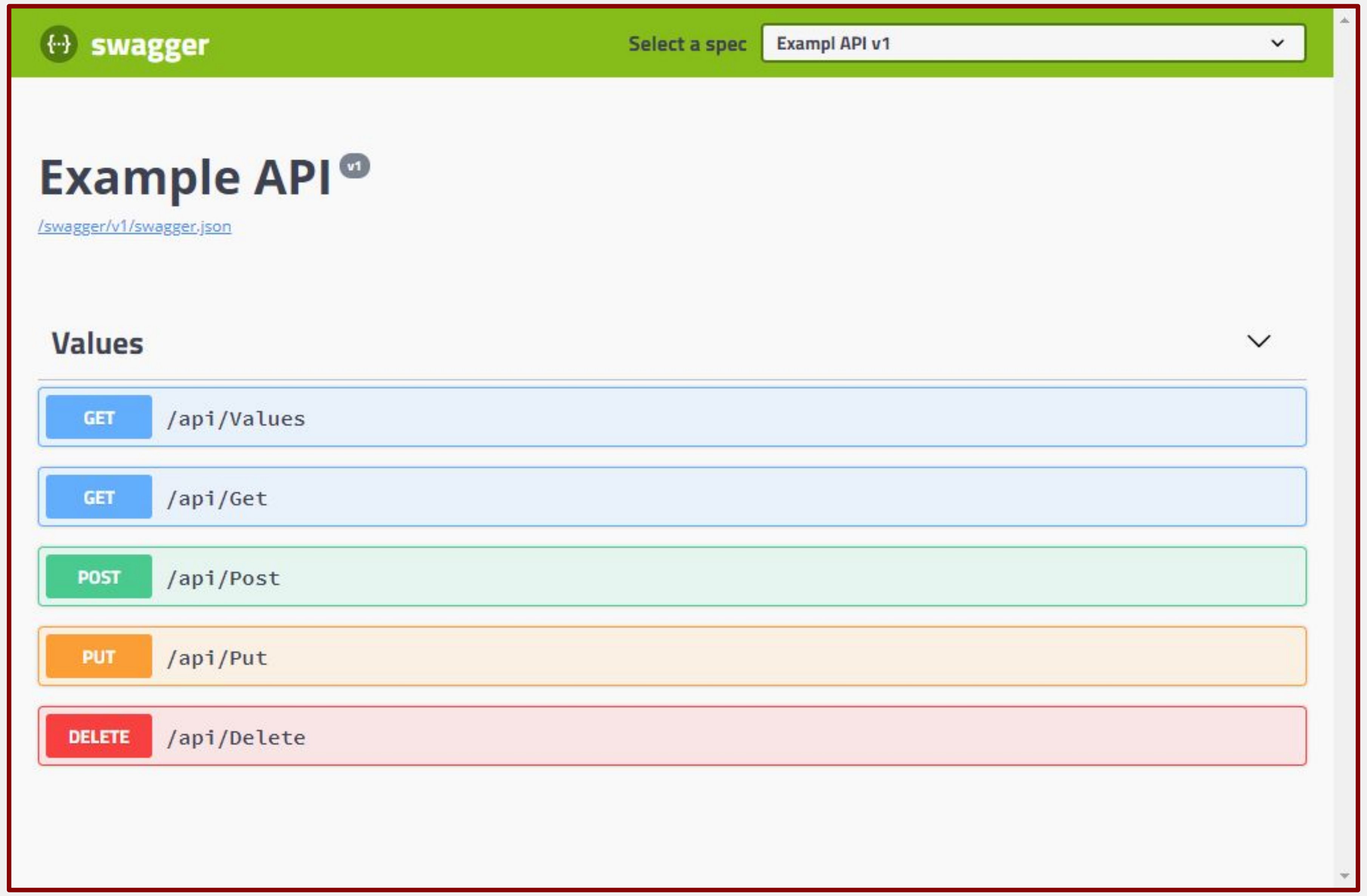
No FlatPage matches the given query.

```
Request Method:  GET
   Request URL:  https://example.com/
     Raised by:  django.contrib.flatpages.views.flatpage
```

Using the URLconf defined in `project.urls`, Django tried these URL patterns, in this order:

```
 1. okta/
 2. 403/ [name='403']
 3. swagger.yaml/ [name='spectacular-schema']
 4. swagger.json/ [name='spectacular-schema']
 5. api/docs/ [name='swagger-ui']
 6. api/redoc/ [name='schema-redoc']
 7. api/v1/
 8. mobile_api/
 9. ^media/(?P<path>.*)$
10. ^(?P<url>.*/)$
```

The current path, `doesntexist`, matched the last one.



Example API v1

/swagger/v1/swagger.json

Values

```
GET     /api/Values
GET     /api/Get
POST    /api/Post
PUT     /api/Put
DELETE  /api/Delete
```

Phase d’exploitation

Exploitation – Vulnérabilités

- Injections
  - SQL / NoSQL
  - OS Command Injection
  - Server Side Request Forgery (SSRF)
- Information Disclosure
- Path Traversal
- Broken Access Control (BAC)
- Race Condition
- Parameter Pollution
- Cryptographic Failure

Phase d'exploitation

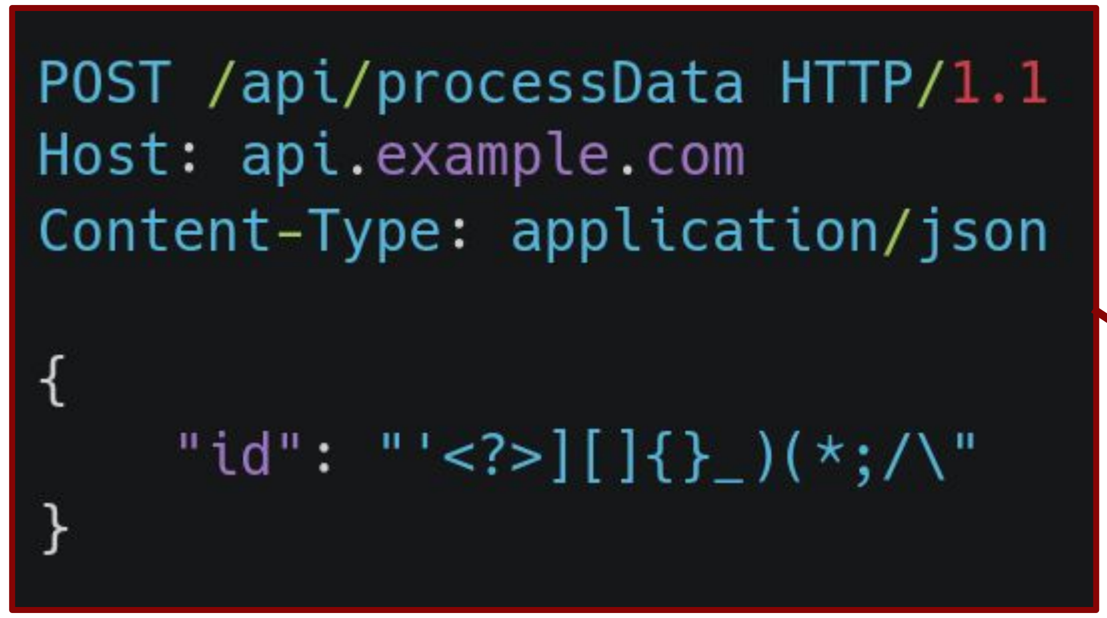
Exploitation – Information Disclosure

```
GET /api/admin/users/all HTTP/1.1
Host: api.example.com
Authorization: Bearer <token>
```

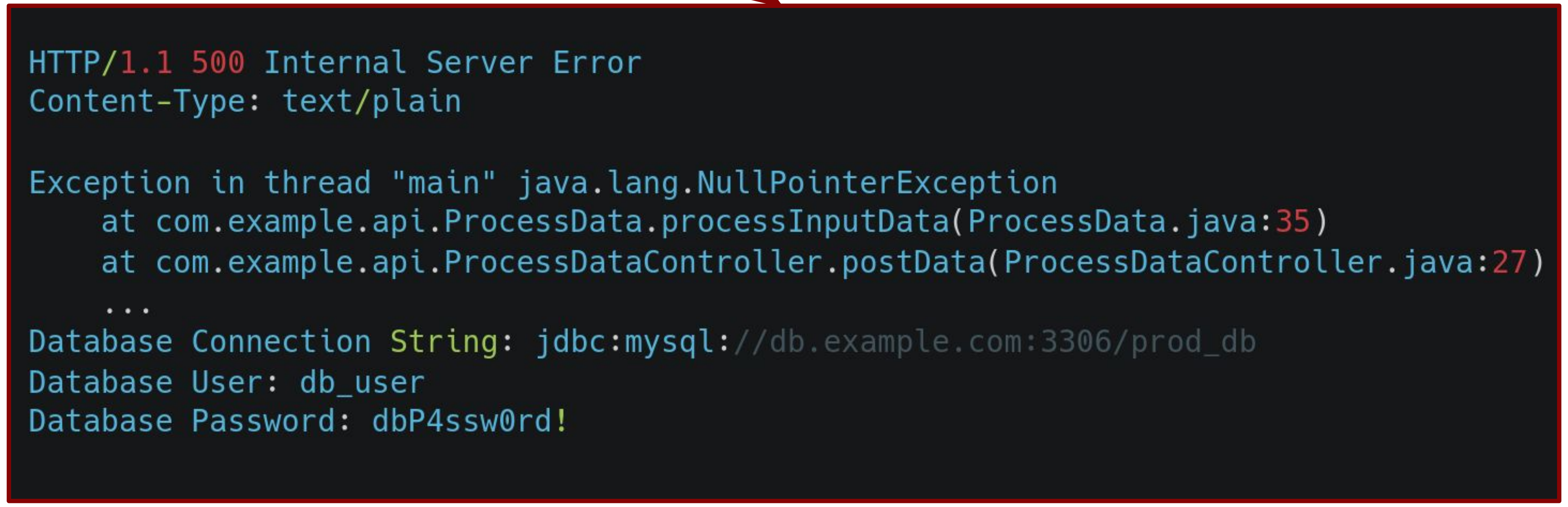
```
HTTP/1.1 200 OK
Content-Type: application/json

[
    {
        "userId": "12345",
        "username": "bob",
        "email": "bob@example.com",
        "password": "bobPass123!",
        "address": "123 Main St, Anytown, AN 12345",
        "phone": "123-456-7890"
    },
    {
        "userId": "67890",
        "username": "alice",
        "email": "alice@example.com",
        "password": "alicePass456!",
        "address": "456 Side St, Othertown, OT 67890",
        "phone": "987-654-3210"
    }
    // more
]
```

Phase d’exploitation

Exploitation – Information Disclosure



```
POST /api/processData HTTP/1.1
Host: api.example.com
Content-Type: application/json

{
    "id": "'<?>][]{}_)(*;/\"
}
```

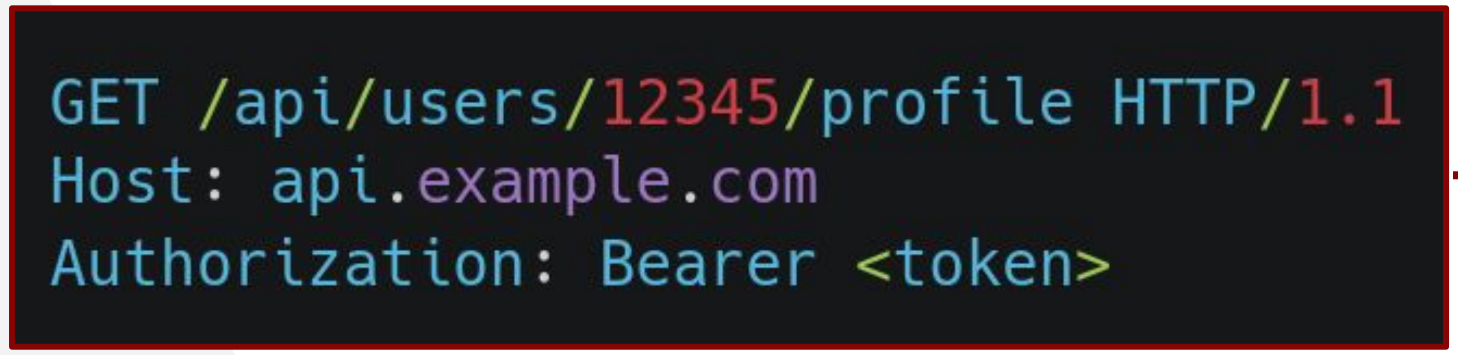


```
HTTP/1.1 500 Internal Server Error
Content-Type: text/plain

Exception in thread "main" java.lang.NullPointerException
    at com.example.api.ProcessData.processInputData(ProcessData.java:35)
    at com.example.api.ProcessDataController.postData(ProcessDataController.java:27)
    ...
Database Connection String: jdbc:mysql://db.example.com:3306/prod_db
Database User: db_user
Database Password: dbP4ssw0rd!
```

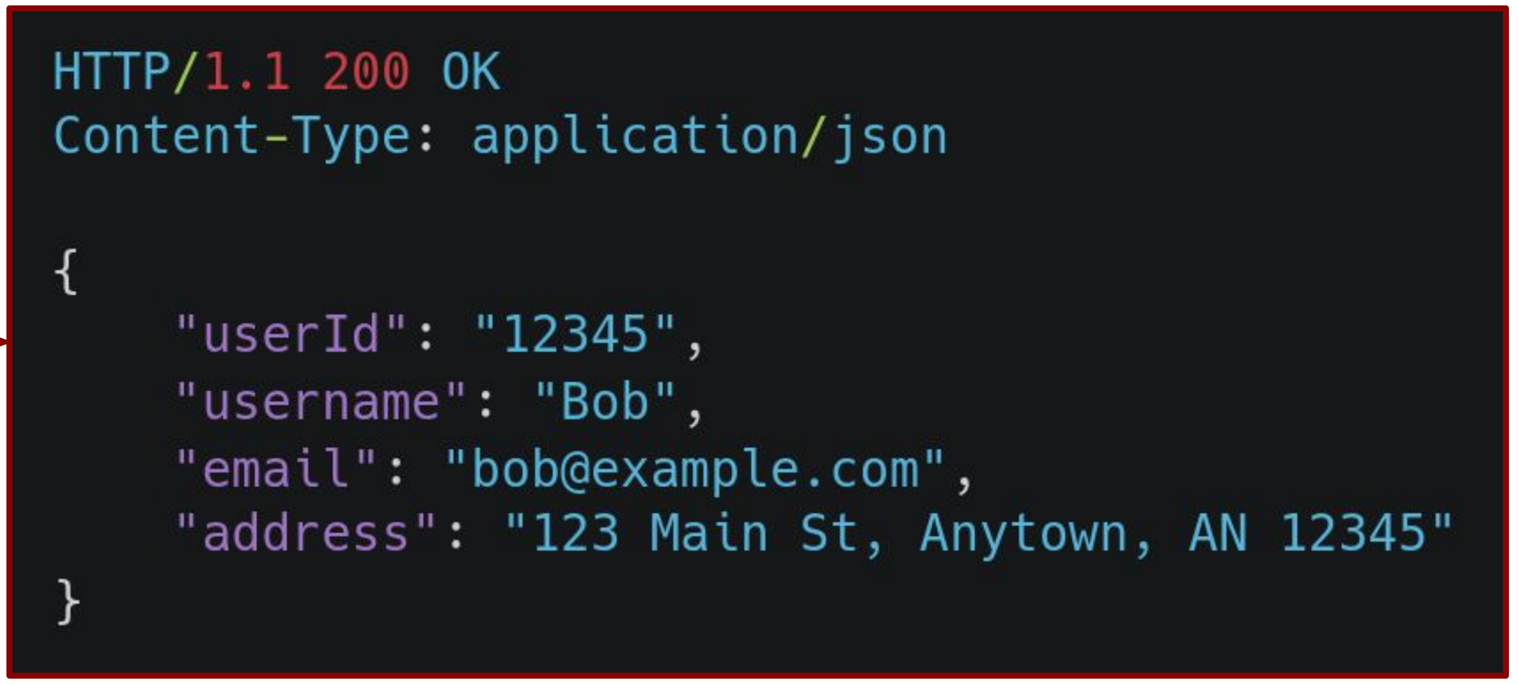
Phase d’exploitation

Exploitation – Insecure Direct Object Reference (IDOR)

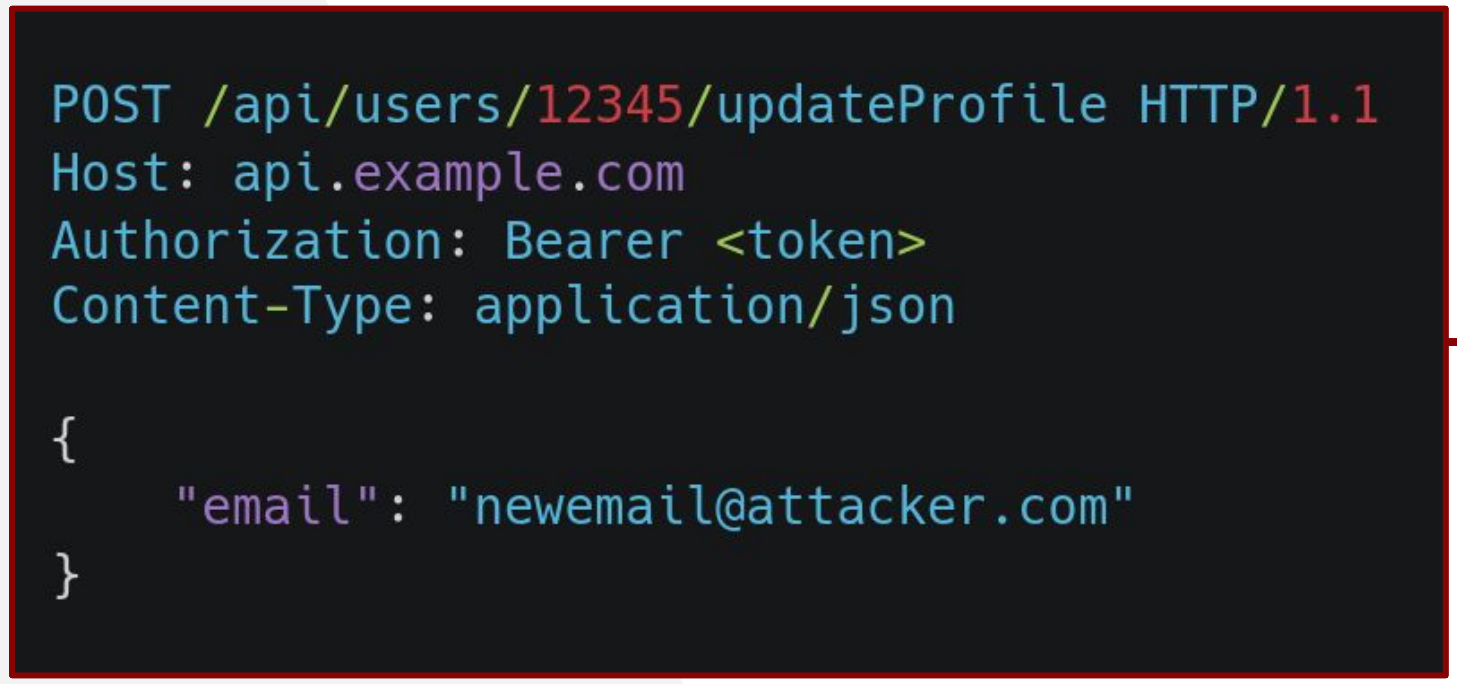
- Introduction aux API
- Phase de reconnaissance
- Phase d’exploitation
- Recommandations
- Mise en pratique



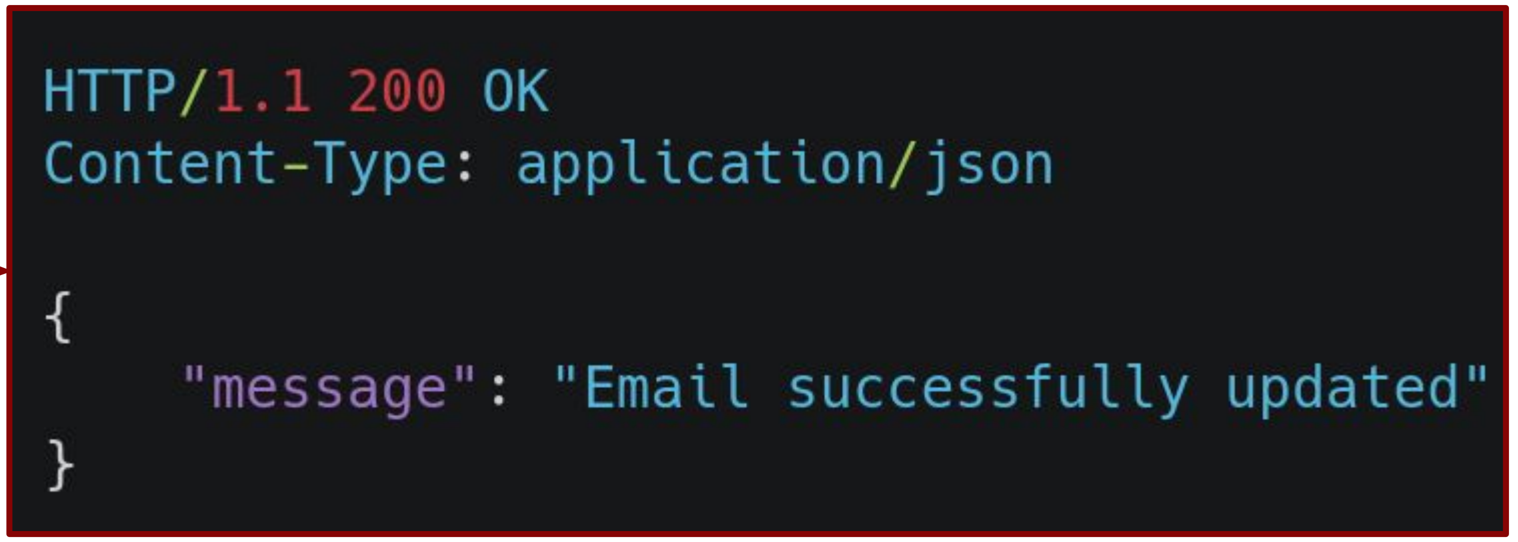
```
GET /api/users/12345/profile HTTP/1.1
Host: api.example.com
Authorization: Bearer <token>
```



```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "userId": "12345",
    "username": "Bob",
    "email": "bob@example.com",
    "address": "123 Main St, Anytown, AN 12345"
}
```



```
POST /api/users/12345/updateProfile HTTP/1.1
Host: api.example.com
Authorization: Bearer <token>
Content-Type: application/json

{
    "email": "newemail@attacker.com"
}
```



```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "message": "Email successfully updated"
}
```

Phase d’exploitation

Exploitation – IDOR - UUID

```
GET /api/documents/f47ac10b-58cc-4372-a567-0e02b2c3d479 HTTP/1.1
Host: api.example.com
Authorization: Bearer <token>
```

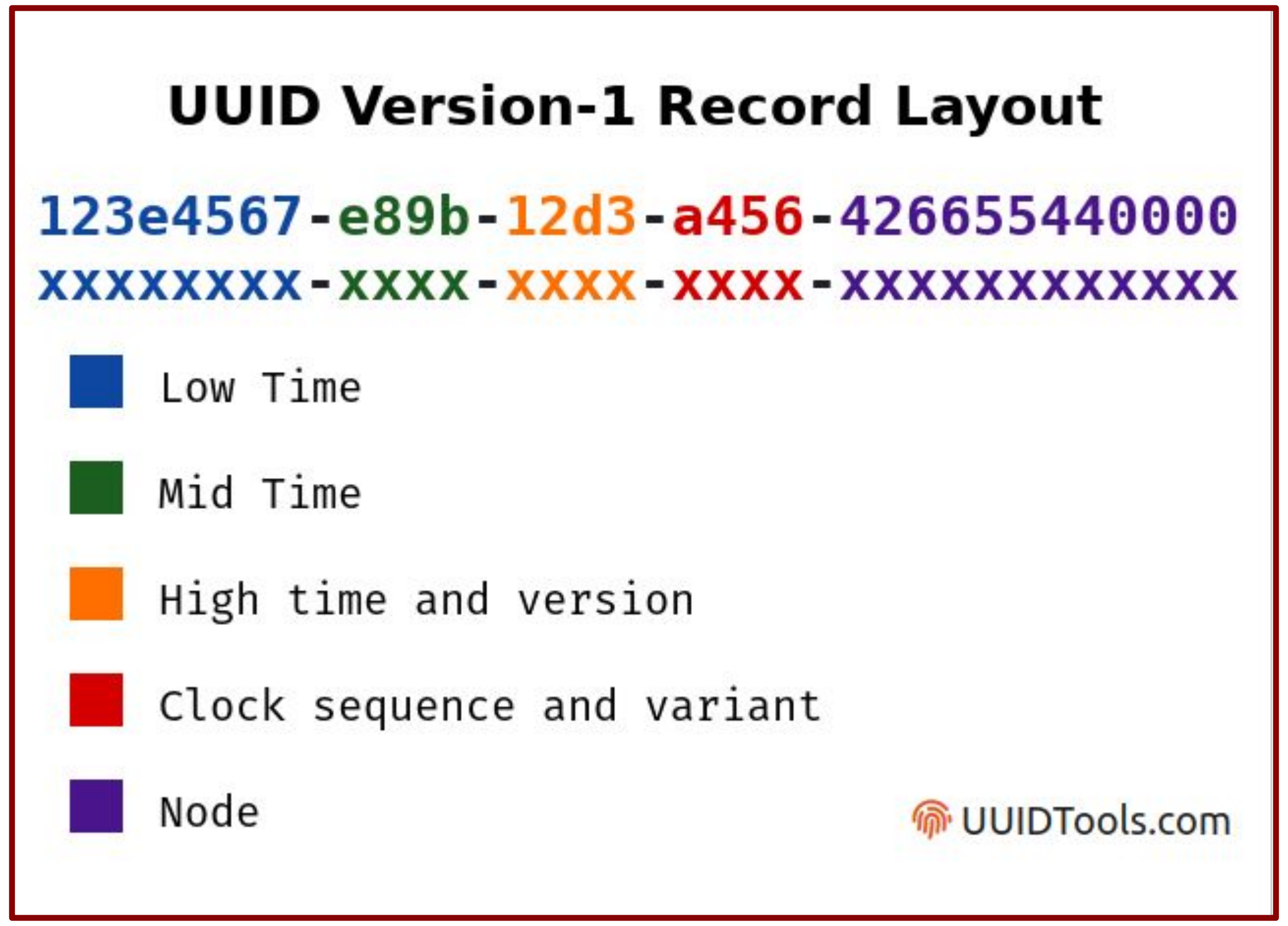
```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "documentId": "f47ac10b-58cc-4372-a567-0e02b2c3d479",
    "title": "Secret document",
    "content": "Content of the secret document..."
}
```

```
GET /api/documents/107a6980-722c-49a0-91f1-b0aedabf1c6b HTTP/1.1
Host: api.example.com
Authorization: Bearer <token>
```

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "documentId": "107a6980-722c-49a0-91f1-b0aedabf1c6b",
    "title": "My document",
    "content": "Content of my document..."
}
```

Phase d'exploitation

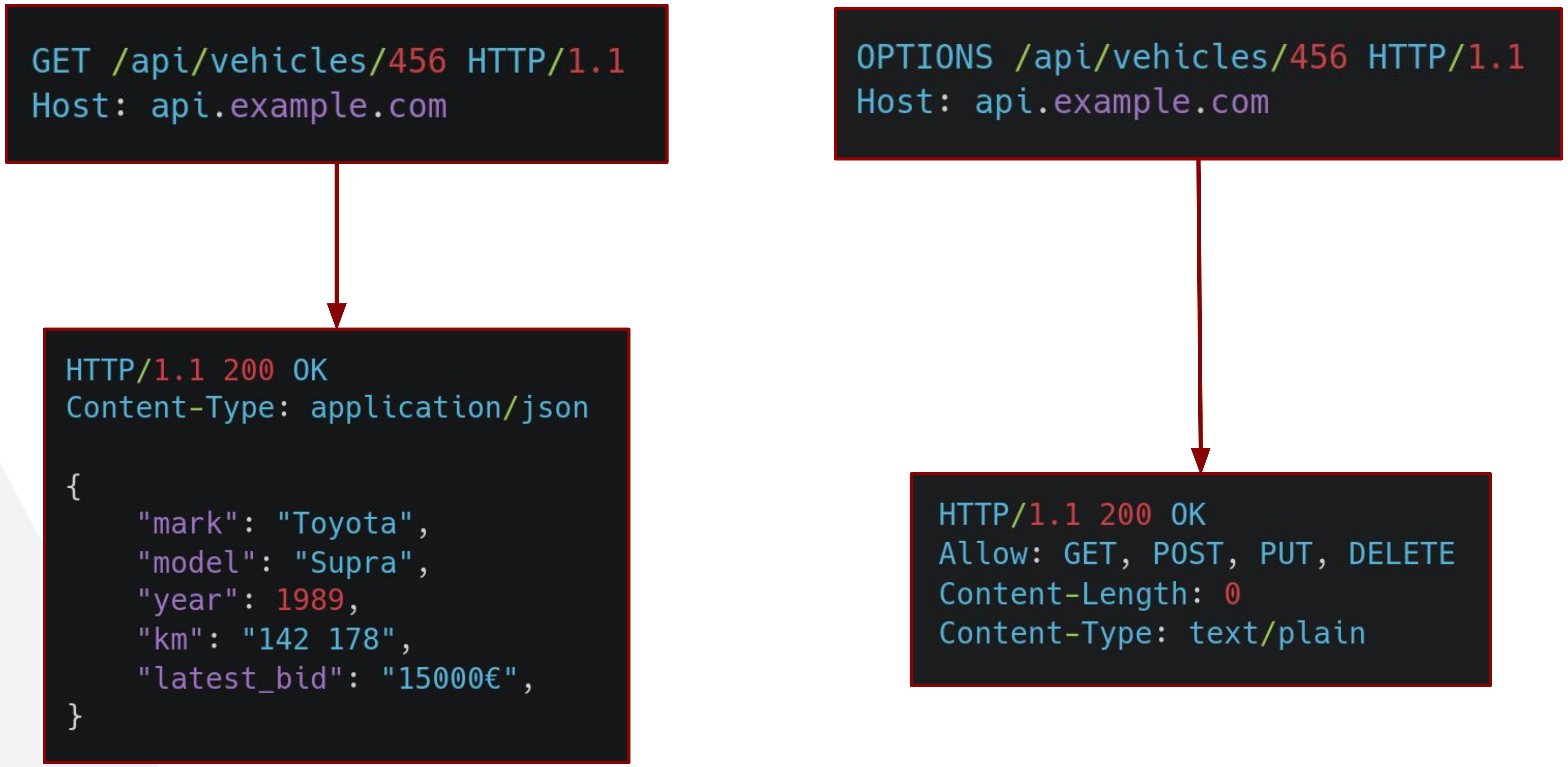
Exploitation – IDOR - UUID


UUID Version-1 Record Layout
123e4567-e89b-12d3-a456-426655440000
xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
🟦 Low Time
🟩 Mid Time
🟧 High time and version
🟥 Clock sequence and variant
🟪 Node
UUIDTools.com

Don't rely on UUIDs for security. Never use UUIDs for things like session identifiers. The standard itself warns implementors to "not assume that UUIDs are hard to guess; they should not be used as security capabilities (identifiers whose mere possession grants access, for example)."

Phase d’exploitation

Exploitation – Mass Assignment



```
GET /api/vehicles/456 HTTP/1.1
Host: api.example.com
```

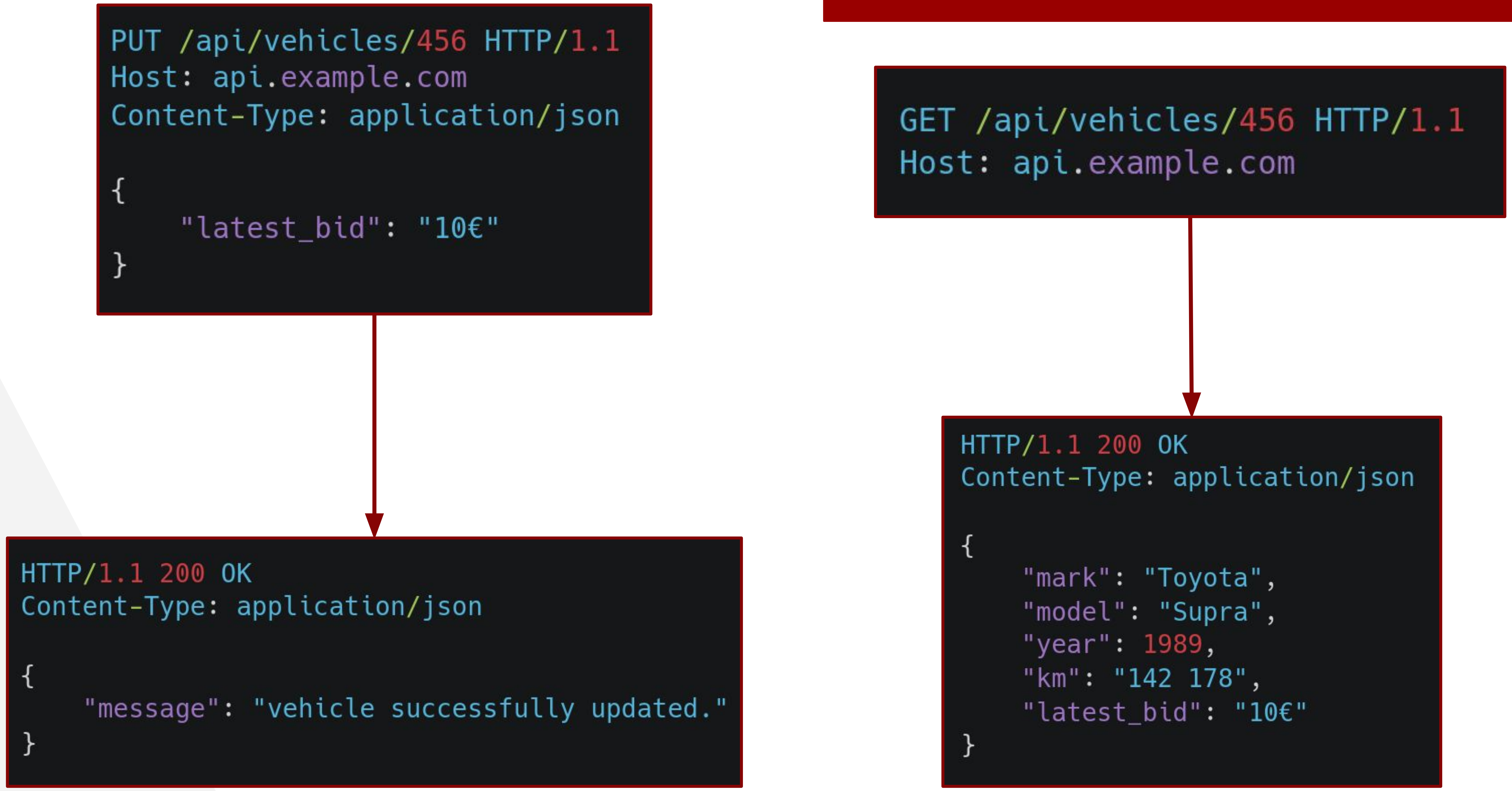
```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "mark": "Toyota",
    "model": "Supra",
    "year": 1989,
    "km": "142 178",
    "latest_bid": "15000€",
}
```

```
OPTIONS /api/vehicles/456 HTTP/1.1
Host: api.example.com
```

```
HTTP/1.1 200 OK
Allow: GET, POST, PUT, DELETE
Content-Length: 0
Content-Type: text/plain
```

Phase d’exploitation

Exploitation – Mass Assignment



```
PUT /api/vehicles/456 HTTP/1.1
Host: api.example.com
Content-Type: application/json

{
    "latest_bid": "10€"
}
```

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "message": "vehicle successfully updated."
}
```

```
GET /api/vehicles/456 HTTP/1.1
Host: api.example.com
```

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "mark": "Toyota",
    "model": "Supra",
    "year": 1989,
    "km": "142 178",
    "latest_bid": "10€"
}
```

Phase d'exploitation

Exploitation – Ressources

Pour aller plus loin :

- API Key Leaks
- API testing | Web Security Academy
- A collection of awesome API Security tools and resources.
- OWASP Top 10 API Security Risks – 2023
- Web API Pentesting - HackTricks
- API Security: Best Practices for Protecting APIs - NGINX
- What is API security?
- API-SecurityEmpire
- 0 Click ATO with the Sandwich Attack - Lupin & Holmes

Recommandations

Recommandations

- Quelques recommandations pour sécuriser l’implémentation d’une API :
  - Whitelist des méthodes HTTP autorisées ;
  - Whitelist du Content-Type supporté ;
  - Enlever les options de debug et de développement en production ;
  - Limiter les messages d’erreurs afin d’éviter des stacktraces ;
  - Supprimer ou limiter l’usage des anciennes versions de l’API ;
  - Mettre à jour la documentation ;
  - Cacher la documentation si l’API est privée ;
  - Vérifier les entrées des utilisateurs.

Mise en pratique

API - Introduction

API – Session

API – Broken Access

API – Hash

API – Mass Assignment

API – Broken Access 2

https://clusir.pro.root-me.org

Questions ?