OWASP API Security: Top 10

Introduction

Open Worldwide Application Security Project (OWASP) is a non-profit and collaborative online community that aims to improve application security via a set of security principles, articles, documentation etc. Back in 2019, OWASP released a list of the top 10 API vulnerabilities, which will be discussed in detail, along with its potential impact and a few effective mitigation measures.

We have split this room into two parts. In Part 1, you will study the top 5 principles, and in Part 2, you will learn the remaining principles.

Learning Objectives

- Best practices for API authorization & authentication
- Identification of authorization level issues
- Handling excessive data exposure
- Lack of resources and rate-limiting issues

Learning Pre-requisites

An understanding of the following topics is recommended before starting the room:

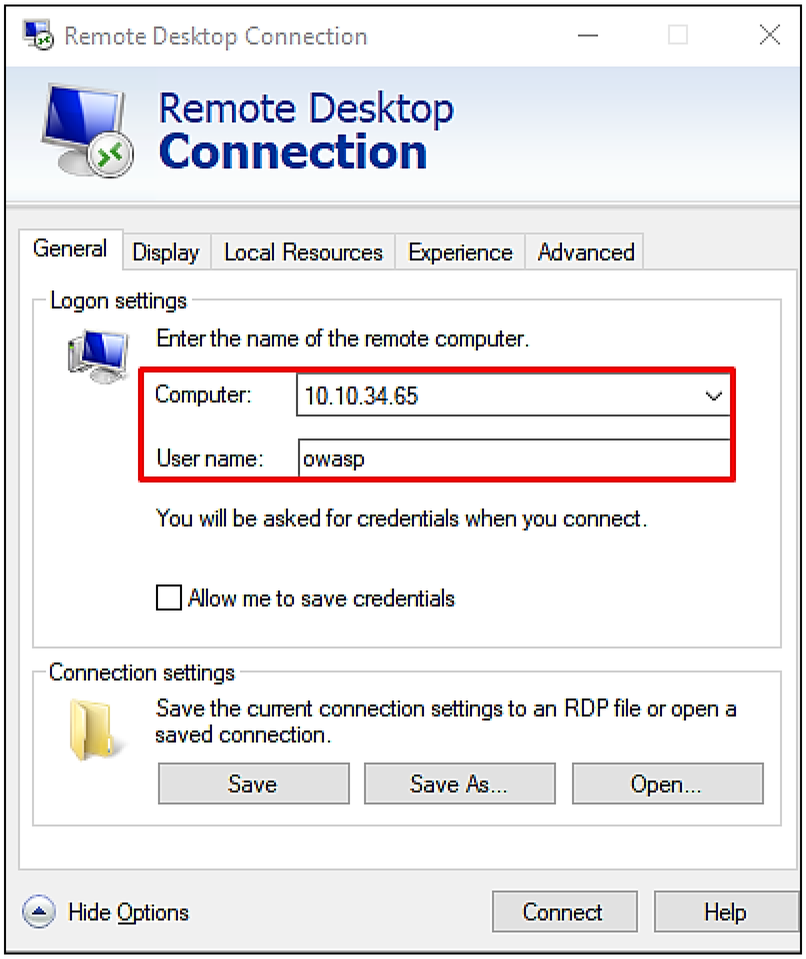
- How websites work
- HTTP protocols & methods
- Principles of security
- OWASP top 10 web vulnerabilities

Connecting to the Machine

We will be using Windows as a development/test machine along with Talend API Tester - free edition throughout the room with the following credentials:

- Machine IP: 10.10.57.114
- Username: Administrator
- Password: Owasp@123

You can start the virtual machine by clicking Start Machine. The machine will start in a split-screen view. In case the VM is not visible, use the blue Show Split View button at the top-right of the page. Alternatively, you can connect with the VM through Remote Desktop using the above credentials. Please wait 1-2 minutes after the system boots completely to let the auto scripts run successfully that will execute Talend API Tester and Laravel-based web application automatically.



Enter the name of the remote computer.

Computer: 10.10.34.65

User name: owasp

You will be asked for credentials when you connect.

☐ Allow me to save credentials

Save the current connection settings to an RDP file or open a saved connection.

Understanding APIs - A Refresher

What is an API & Why is it important?

API stands for Application Programming Interface. It is a middleware that facilitates the communication of two software components utilising a set of protocols and definitions. In the API context, the term 'application' refers to any software having specific functionality, and 'interface' refers to the service contract between two apps that make communication possible via requests and responses. The API documentation contains all the information on how developers have structured those responses and requests. The significance of APIs to app development is in just a single sentence, i.e., API is a building block for developing complex and enterprise-level applications.

Recent Data Breaches through APIs

- LinkedIn data breach: In June 2021, the data of over 700 million LinkedIn users were offered for sale on one of the dark web forums, which was scraped by exploiting the LinkedIn API. The hacker published a sample of 1 million records to confirm the legitimacy of the LinkedIn breach, containing full names of the users, email addresses, phone numbers, geolocation records, LinkedIn profile links, work experience information, and other social media account details.

- Twitter data breach: In June 2022, data of more than 5.4 Million Twitter users was released for sale on the dark web. Hackers conducted the breach by exploiting a zero-day in the Twitter API that showed Twitter's handle against a mobile number or email.
- PIXLR data breach: In January 2021, PIXLR, an online photo editor app, suffered a data breach that impacted around 1.9 million users. All the data collected by the hackers was dumped on a dark web forum, which included usernames, email addresses, countries, and hashed passwords.

Now that we understand the threat and the damage caused due to non-adherence to mitigation measures - let's discuss developing a secure API through OWASP API Security Top 10 principles.
****

Answer the questions below:

In the LinkedIn breach (Jun 2021), how many million records (sample) were posted by a hacker on the dark web?
Answer: 1

Is the API documentation a trivial item and not used after API development (yea/nay)?
Answer: nay

I understand the APIs and am ready to learn OWASP Top 10 Principles.
No Answer Needed
****

Vulnerability I - Broken Object Level Authorization(BOLA)

How Does it Happen?

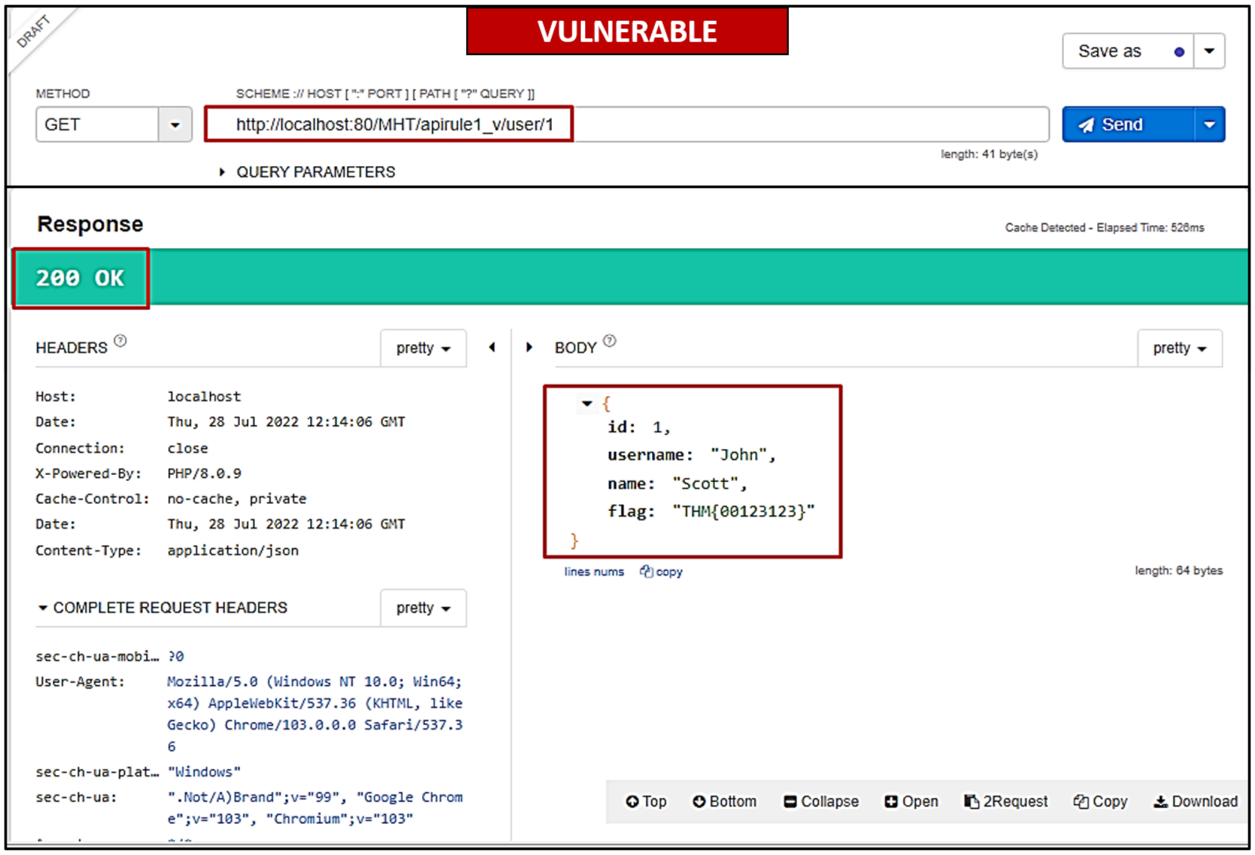
Generally, API endpoints are utilised for a common practice of retrieving and manipulating data through object identifiers. BOLA refers to Insecure Direct Object Reference (IDOR) - which creates a scenario where the user uses the input functionality and gets access to the resources they are not authorized to access. In an API, such controls are usually implemented through programming in Models (Model-View-Controller Architecture) at the code level.

Likely Impact

The absence of controls to prevent unauthorized object access can lead to data leakage and, in some cases, complete account takeover. User's or subscribers' data in the database plays a critical role in an organization's brand reputation; if such data is leaked over the internet, that may result in substantial financial loss.

Practical Example

- Open the VM. You will find that the Chrome browser and Talend API Tester application are running automatically, which we will be using for debugging the API endpoints.
- Bob is working as an API developer in Company MHT and developed an endpoint /apirule1/users/{ID} that will allow other applications or developers to request information by sending an employee ID. In the VM, you can request results by sending GET requests to http://localhost:80/MHT/apirule1_v/user/1.



GET http://localhost:80/MHT/apirule1_v/user/1

length: 41 byte(s)

Cache Detected - Elapsed Time: 526ms

200 OK

```
Host:           localhost
Date:           Thu, 28 Jul 2022 12:14:06 GMT
Connection:     close
X-Powered-By:   PHP/8.0.9
Cache-Control:  no-cache, private
Date:           Thu, 28 Jul 2022 12:14:06 GMT
Content-Type:   application/json
```

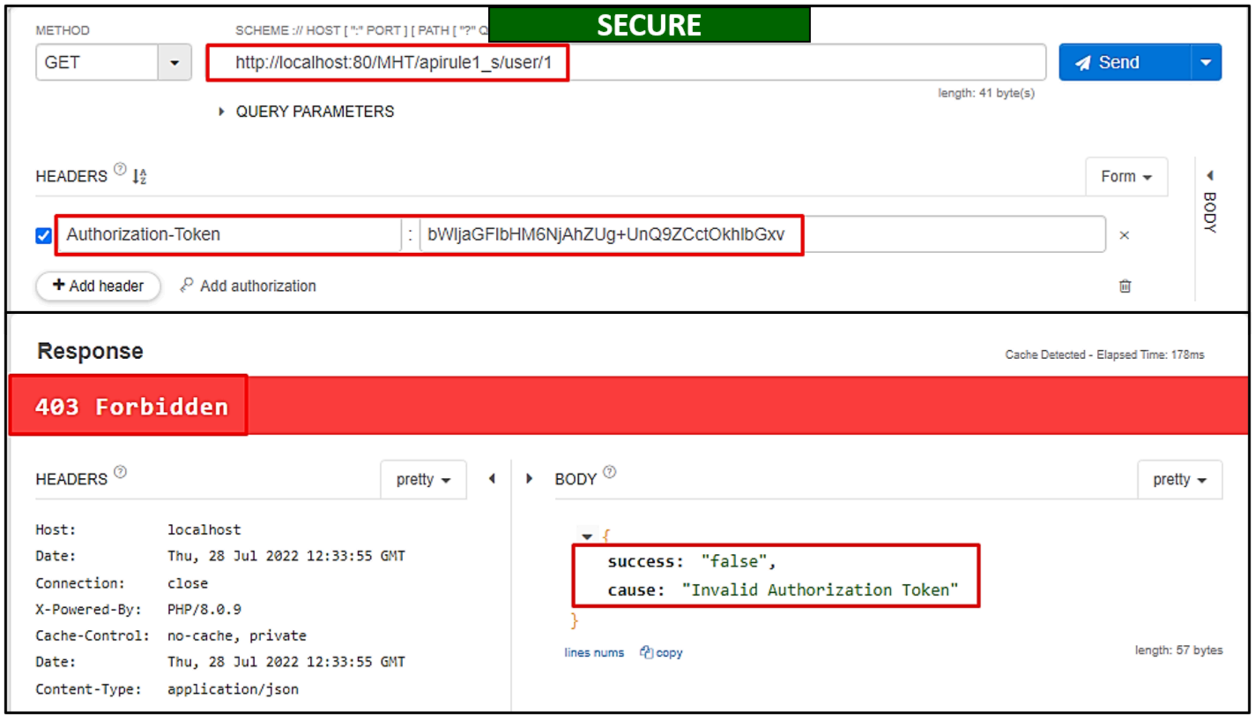
```
{
  id: 1,
  username: "John",
  name: "Scott",
  flag: "THM{00123123}"
}
```

length: 64 bytes

```
sec-ch-ua-mobi… ?0
User-Agent:     Mozilla/5.0 (Windows NT 10.0; Win64;
                x64) AppleWebKit/537.36 (KHTML, like
                Gecko) Chrome/103.0.0.0 Safari/537.3
                6
sec-ch-ua-plat… "Windows"
sec-ch-ua:      ".Not/A)Brand";v="99", "Google Chrom
                e";v="103", "Chromium";v="103"
```

- What is the issue with the above API call? The problem is that the endpoint is not validating any incoming API call to confirm whether the request is valid. It is not checking for any authorization whether the person requesting the API call can ask for it or not.

- The solution for this problem is pretty simple; Bob will implement an authorization mechanism through which he can identify who can make API calls to access employee ID information.
- The purpose is achieved through access tokens or authorization tokens in the header. In the above example, Bob will add an authorization token so that only headers with valid authorization tokens can make a call to this endpoint.
- In the VM, if you add a valid Authorization-Token and call http://localhost:80/MHT/apirule1_s/user/1, only then will you be able to get the correct results. Moreover, all API calls with an invalid token will show 403 Forbidden an error message (as shown below).



http://localhost:80/MHT/apirule1_s/user/1

length: 41 byte(s)

Authorization-Token : bWljaGFIbHM6NjAhZUg+UnQ9ZCctOkhlbGxv

Cache Detected - Elapsed Time: 178ms

403 Forbidden

```
Host:           localhost
Date:           Thu, 28 Jul 2022 12:33:55 GMT
Connection:     close
X-Powered-By:   PHP/8.0.9
Cache-Control:  no-cache, private
Date:           Thu, 28 Jul 2022 12:33:55 GMT
Content-Type:   application/json
```

```
{
  success:  "false",
  cause:  "Invalid Authorization Token"
}
```

length: 57 bytes

Mitigation Measures

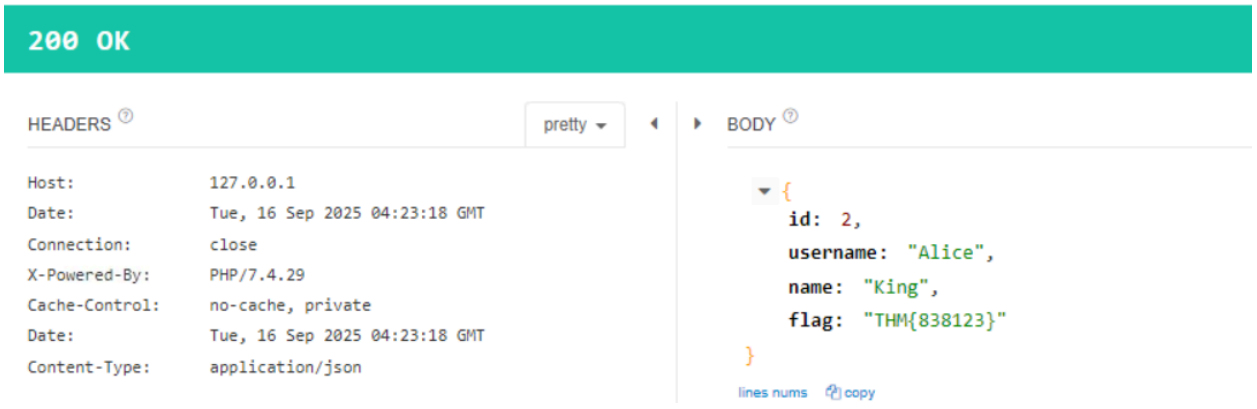
- An authorization mechanism that relies on user policies and hierarchies should be adequately implemented.
- Strict access controls methods to check if the logged-in user is authorized to perform specific actions.
- Promote using completely random values (strong encryption and decryption mechanism) for nearly impossible-to-predict tokens.

****

Answer the questions below:

Suppose the employee ID is an integer with incrementing value. Can you check through the vulnerable API endpoint the total number of employees in the company?
Answer: 3

What is the flag associated with employee ID 2?

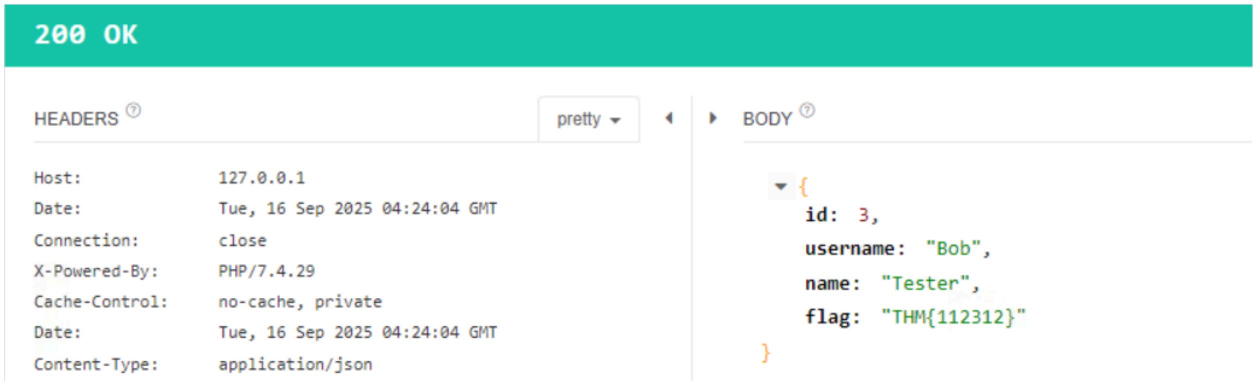


```
200 OK

HEADERS                                    BODY
Host:            127.0.0.1                 {
Date:            Tue, 16 Sep 2025 04:23:18 GMT   id: 2,
Connection:      close                       username: "Alice",
X-Powered-By:    PHP/7.4.29                  name: "King",
Cache-Control:   no-cache, private           flag: "THM{838123}"
Date:            Tue, 16 Sep 2025 04:23:18 GMT }
Content-Type:    application/json
```

Answer: THM{838123}

What is the username of employee ID 3?



```
200 OK

HEADERS                                    BODY
Host:            127.0.0.1                 {
Date:            Tue, 16 Sep 2025 04:24:04 GMT   id: 3,
Connection:      close                       username: "Bob",
X-Powered-By:    PHP/7.4.29                  name: "Tester",
Cache-Control:   no-cache, private           flag: "THM{112312}"
Date:            Tue, 16 Sep 2025 04:24:04 GMT }
Content-Type:    application/json
```

Answer: Bob
****

Vulnerability II - Broken User Authentication(BUA)

How Does it Happen?

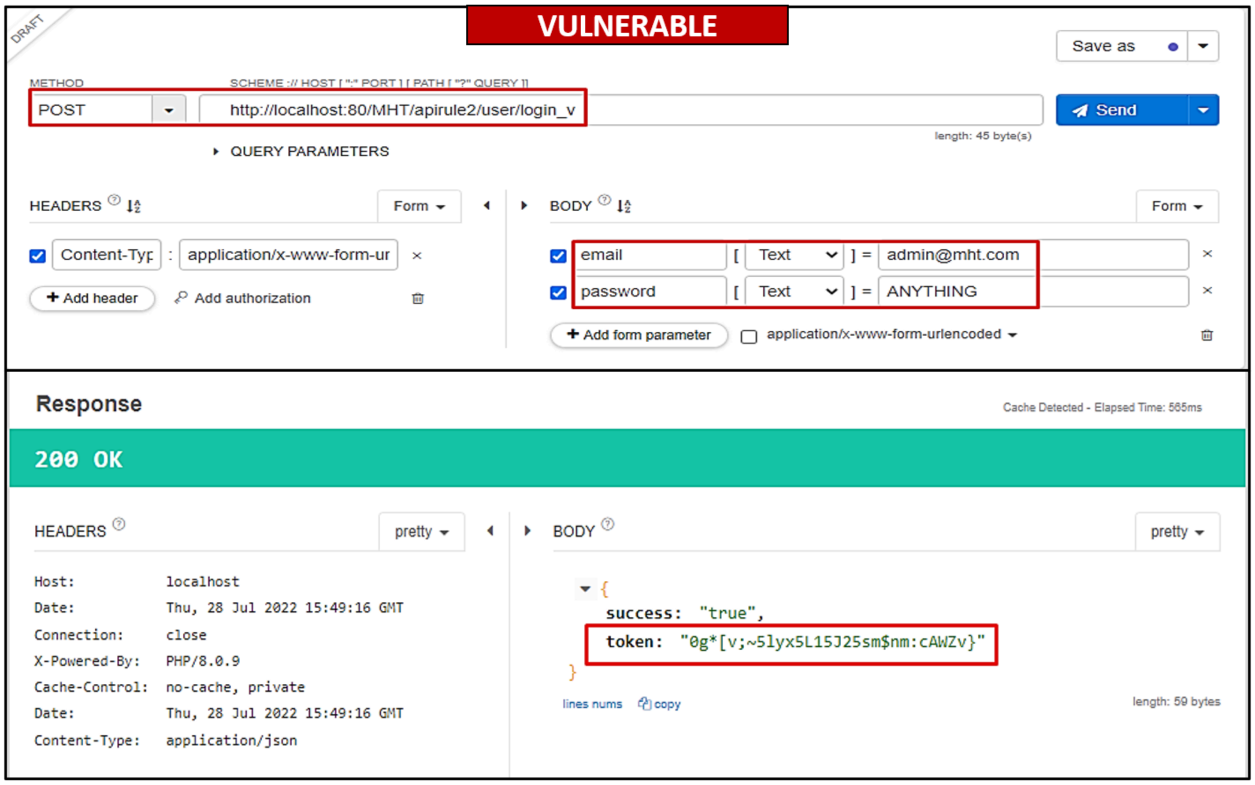
User authentication is the core aspect of developing any application containing sensitive data. Broken User Authentication (BUA) reflects a scenario where an API endpoint allows an attacker to access a database or acquire a higher privilege than the existing one. The primary reason behind BUA is either invalid implementation of authentication like using incorrect email/password queries etc., or the absence of security mechanisms like authorization headers, tokens etc.

Consider a scenario in which an attacker acquires the capability to abuse an authentication API; it will eventually result in data leaks, deletion, modification, or even the complete account takeover by the attacker. Usually, hackers have created special scripts to profile, enumerate users on a system and identify authentication endpoints. A poorly implemented authentication system can lead any user to take on another user's identity.

Likely Impact
In broken user authentication, attackers can compromise the authenticated session or the authentication mechanism and easily access sensitive data. Malicious actors can pretend to be someone authorized and can conduct an undesired activity, including a complete account takeover.

Practical Example

- Continue to use the Chrome browser and Talend API Tester for debugging in the VM.
- Bob understands that authentication is critical and has been tasked to develop an API endpoint apirule2/user/login_v that will authenticate based on provided email and password.
- The endpoint will return a token, which will be passed as an Authorization-Token header (GET request) to apirule2/user/details to show details of the specific employee. Bob successfully developed the login endpoint; however, he only used email to validate the user from the user table and ignored the password field in the SQL query. An attacker only requires the victim's email address to get a valid token or account takeover.
- In the VM, you can test this by sending a POST request to http://localhost:80/MHT/apirule2/user/login_v with email and password in the form parameters.



METHOD: POST
SCHEME :// HOST [":" PORT] [PATH ["?" QUERY]]: http://localhost:80/MHT/apirule2/user/login_v

length: 45 byte(s)

HEADERS: Form

✓	Content-Typ	application/x-www-form-ur

BODY: Form

✓	email	Text	admin@mht.com
✓	password	Text	ANYTHING

application/x-www-form-urlencoded

Response

Cache Detected - Elapsed Time: 565ms

200 OK

HEADERS: pretty

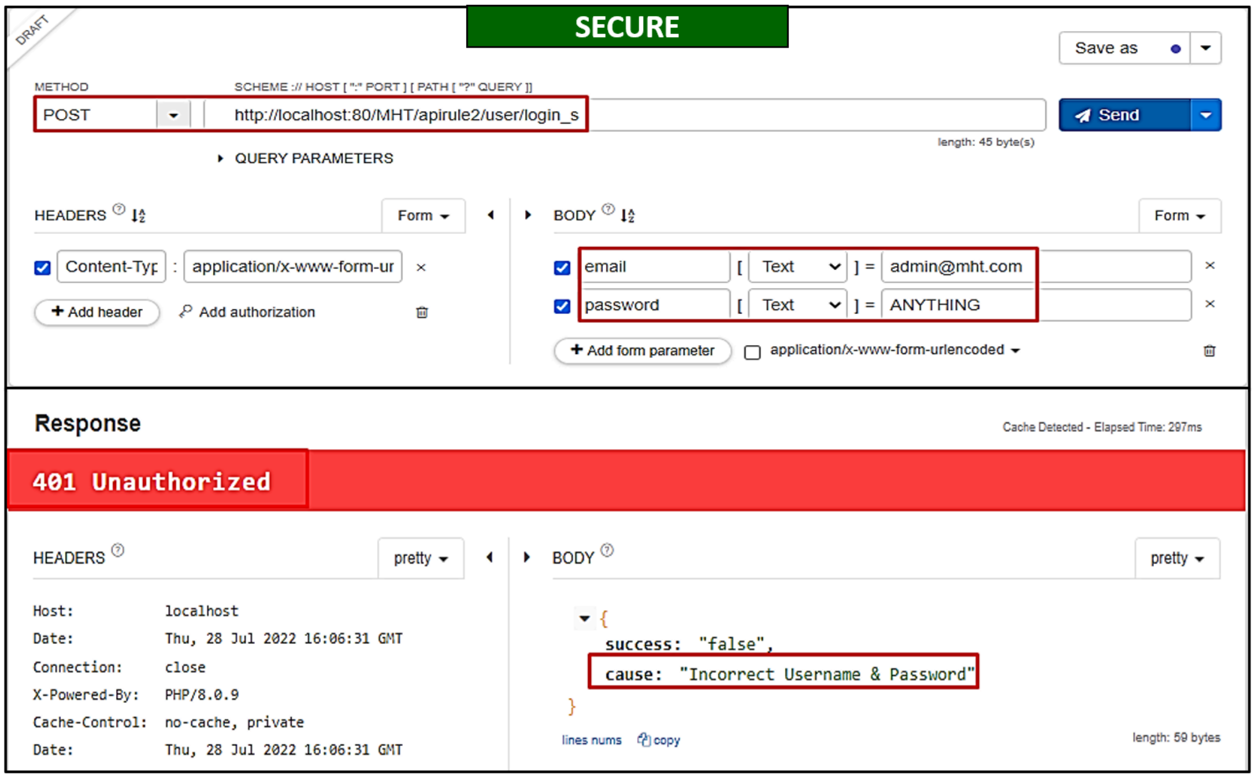
```
Host:          localhost
Date:          Thu, 28 Jul 2022 15:49:16 GMT
Connection:    close
X-Powered-By:  PHP/8.0.9
Cache-Control: no-cache, private
Date:          Thu, 28 Jul 2022 15:49:16 GMT
Content-Type:  application/json
```

BODY: pretty

```
{
  success:  "true",
  token:  "0g*[v;~5lyx5L15J25sm$nm:cAWZv}"
}
```

length: 59 bytes

- As we can see, the vulnerable endpoint received a token which can be forwarded to /apirule2/user/details to get detail of a user.
- To fix this, we will update the login query logic and use both email and password for validation. The endpoint /apirule2/user/login_s is a valid endpoint, as shown below, that authorises the user based on password and email both.



METHOD: POST
SCHEME :// HOST [":" PORT] [PATH ["?" QUERY]]: http://localhost:80/MHT/apirule2/user/login_s
length: 45 byte(s)

HEADERS: Content-Typ : application/x-www-form-ur

BODY:
email [Text] = admin@mht.com
password [Text] = ANYTHING
application/x-www-form-urlencoded

Response
Cache Detected - Elapsed Time: 297ms

401 Unauthorized

HEADERS
```
Host:          localhost
Date:          Thu, 28 Jul 2022 16:06:31 GMT
Connection:    close
X-Powered-By:  PHP/8.0.9
Cache-Control: no-cache, private
Date:          Thu, 28 Jul 2022 16:06:31 GMT
```

BODY
```
{
  success:  "false",
  cause:  "Incorrect Username & Password"
}
```
length: 59 bytes

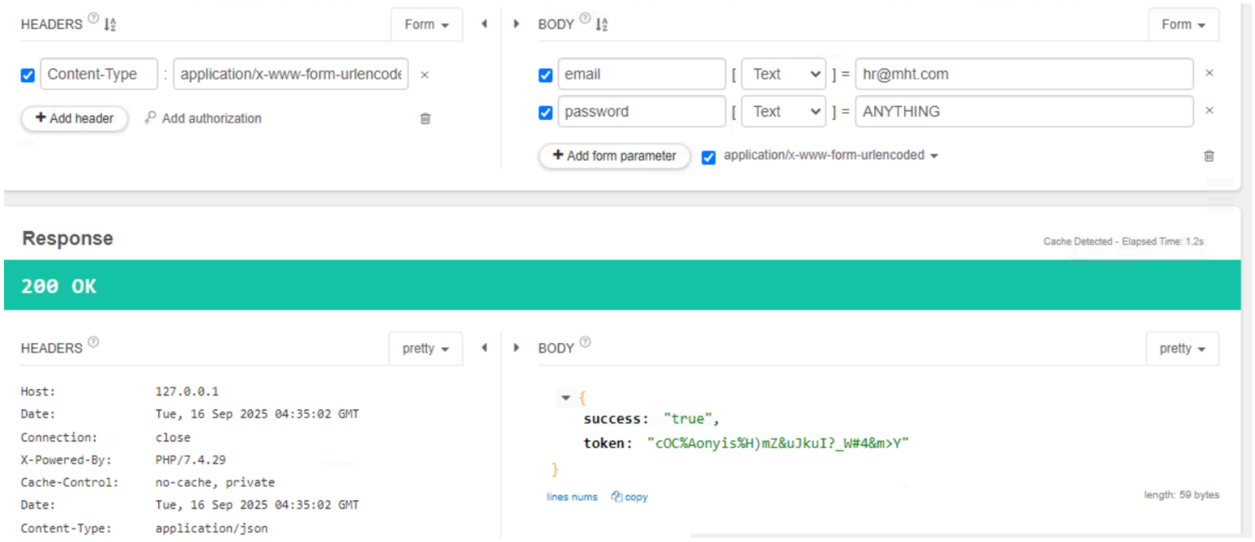
Mitigation Measures

- Ensure complex passwords with higher entropy for end users.
- Do not expose sensitive credentials in GET or POST requests.
- Enable strong JSON Web Tokens (JWT), authorization headers etc.
- Ensure the implementation of multifactor authentication (where possible), account lockout, or a captcha system to mitigate brute force against particular users.
- Ensure that passwords are not saved in plain text in the database to avoid further account takeover by the attacker.

****

Answer the questions below:

Can you find the token of hr@mht.com?



```
Content-Type : application/x-www-form-urlencode

email     [ Text ] = hr@mht.com
password  [ Text ] = ANYTHING
application/x-www-form-urlencoded
```

Response

Cache Detected - Elapsed Time: 1.2s

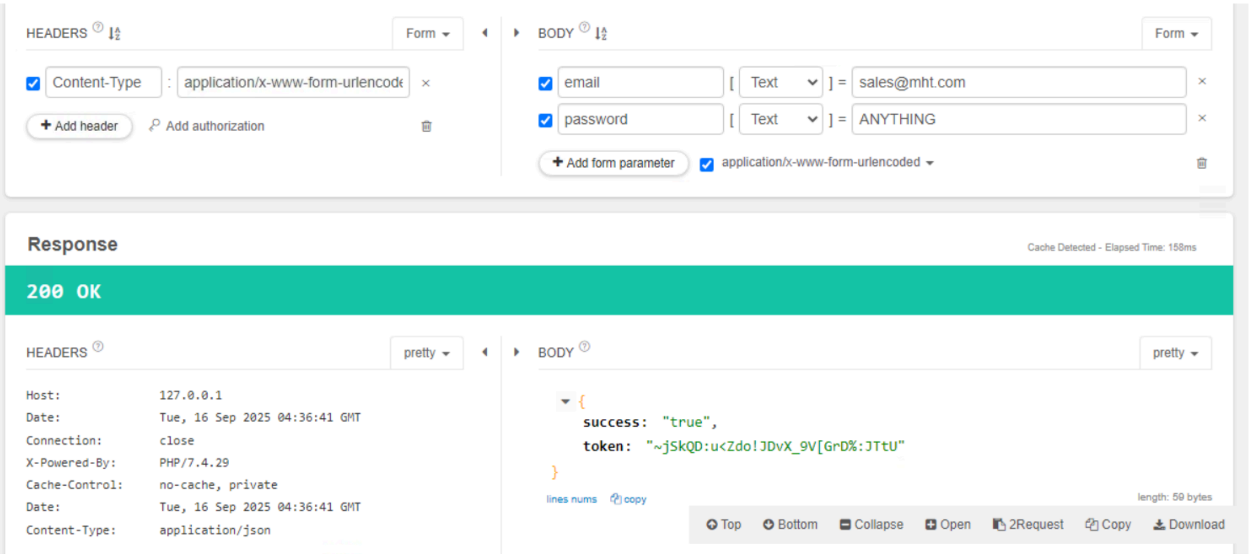
200 OK

```
Host:           127.0.0.1
Date:           Tue, 16 Sep 2025 04:35:02 GMT
Connection:     close
X-Powered-By:   PHP/7.4.29
Cache-Control:  no-cache, private
Date:           Tue, 16 Sep 2025 04:35:02 GMT
Content-Type:   application/json
```

```
{
  success:  "true",
  token:  "cOC%Aonyis%H)mZ&uJkuI?_W#4&m>Y"
}
```

Answer: cOC%Aonyis%H)mZ&uJkuI?_W#4&m>Y

To which country does sales@mht.com belong?



```
Content-Type : application/x-www-form-urlencode

email     [ Text ] = sales@mht.com
password  [ Text ] = ANYTHING
application/x-www-form-urlencoded
```

Response

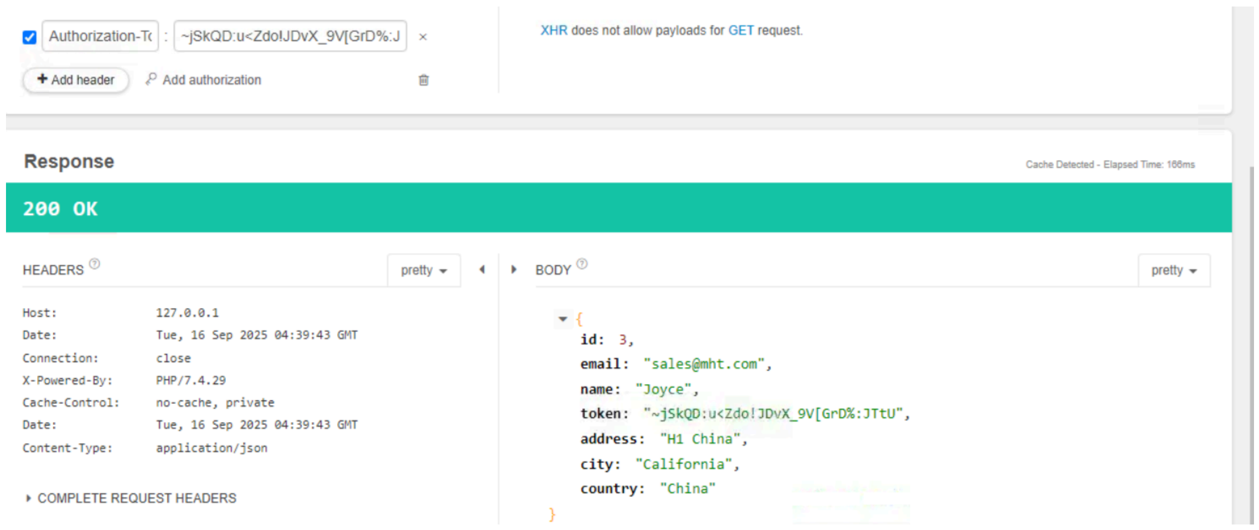
Cache Detected - Elapsed Time: 158ms

200 OK

```
Host:           127.0.0.1
Date:           Tue, 16 Sep 2025 04:36:41 GMT
Connection:     close
X-Powered-By:   PHP/7.4.29
Cache-Control:  no-cache, private
Date:           Tue, 16 Sep 2025 04:36:41 GMT
Content-Type:   application/json
```

```
{
  success:  "true",
  token:  "~jSkQD:u<Zdo!JDvX_9V[GrD%:JTtU"
}
```

Get the token for the sales account then switch to a GET request and pass it into the Authorization-Token header of the apirule2/user/details API.


Authorization-T(
~jSkQD:u<Zdo!JDvX_9V[GrD%:J
Add header
Add authorization
XHR does not allow payloads for GET request.
Response
Cache Detected - Elapsed Time: 166ms
200 OK
HEADERS
pretty
BODY
pretty
COMPLETE REQUEST HEADERS

```
Host:            127.0.0.1
Date:            Tue, 16 Sep 2025 04:39:43 GMT
Connection:      close
X-Powered-By:    PHP/7.4.29
Cache-Control:   no-cache, private
Date:            Tue, 16 Sep 2025 04:39:43 GMT
Content-Type:    application/json
```

```
{
  id: 3,
  email: "sales@mht.com",
  name: "Joyce",
  token: "~jSkQD:u<Zdo!JDvX_9V[GrD%:JTtU",
  address: "H1 China",
  city: "California",
  country: "China"
}
```

Answer: China

Is it a good practice to send a username and password in a GET request (yea/nay)?
Answer: nay
****

Vulnerability III - Excessive Data Exposure

How Does it Happen?

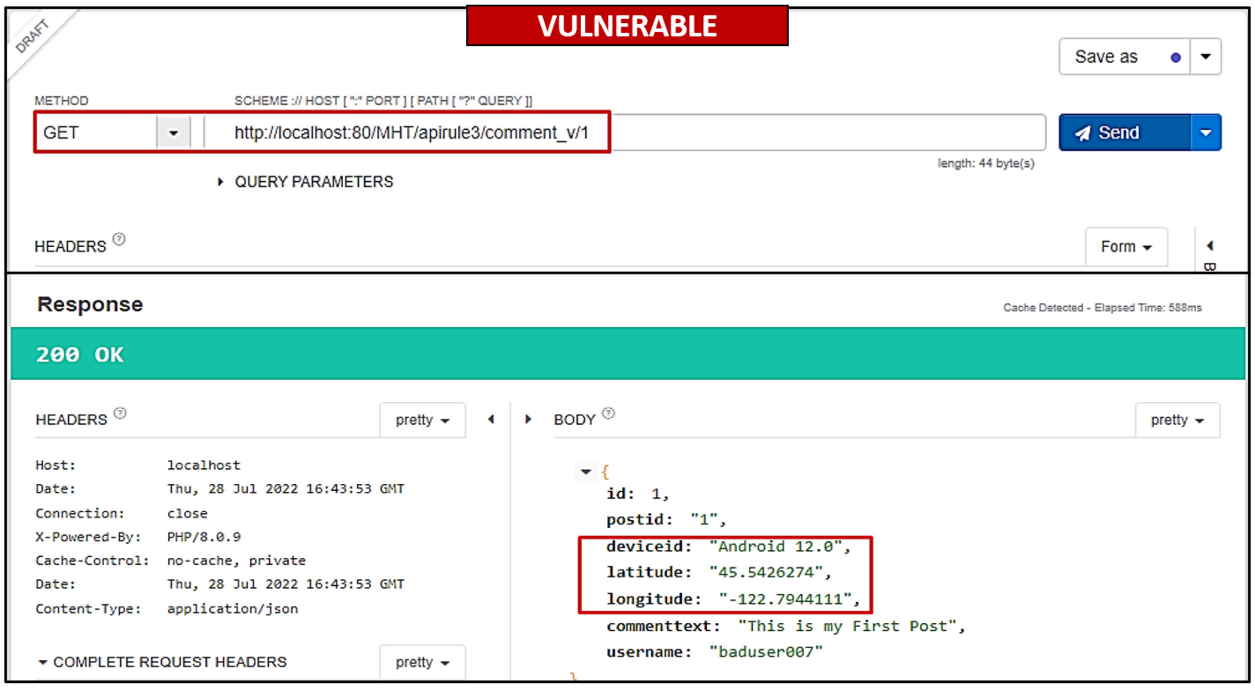
Excessive data exposure occurs when applications tend to disclose more than desired information to the user through an API response. The application developers tend to expose all object properties (considering the generic implementations) without considering their sensitivity level. They leave the filtration task to the front-end developer before it is displayed to the user. Consequently, an attacker can intercept the response through the API and quickly extract the desired confidential data. The runtime detection tools or the general security scanning tools can give an alert on this kind of vulnerability. However, it cannot differentiate between legitimate data that is supposed to be returned or sensitive data.

Likely Impact

A malicious actor can successfully sniff the traffic and easily access confidential data, including personal details, such as account numbers, phone numbers, access tokens and much more. Typically, APIs respond with sensitive tokens that can be later on used to make calls to other critical endpoints.

Practical Example

- Continue to use the Chrome browser and Talend API Tester for debugging in the VM.
- The company MHT launched a comment-based web portal that takes users' comments and stores them in the database and other information like location, device info, etc., to improve the user experience.
- Bob was tasked to develop an endpoint for showing users' comments on the company's main website. He developed an endpoint apirule3/comment_v/{id} that fetches all information available for a comment from the database. Bob assumed that the front-end developer would filter out information while showing it on the company's main website.



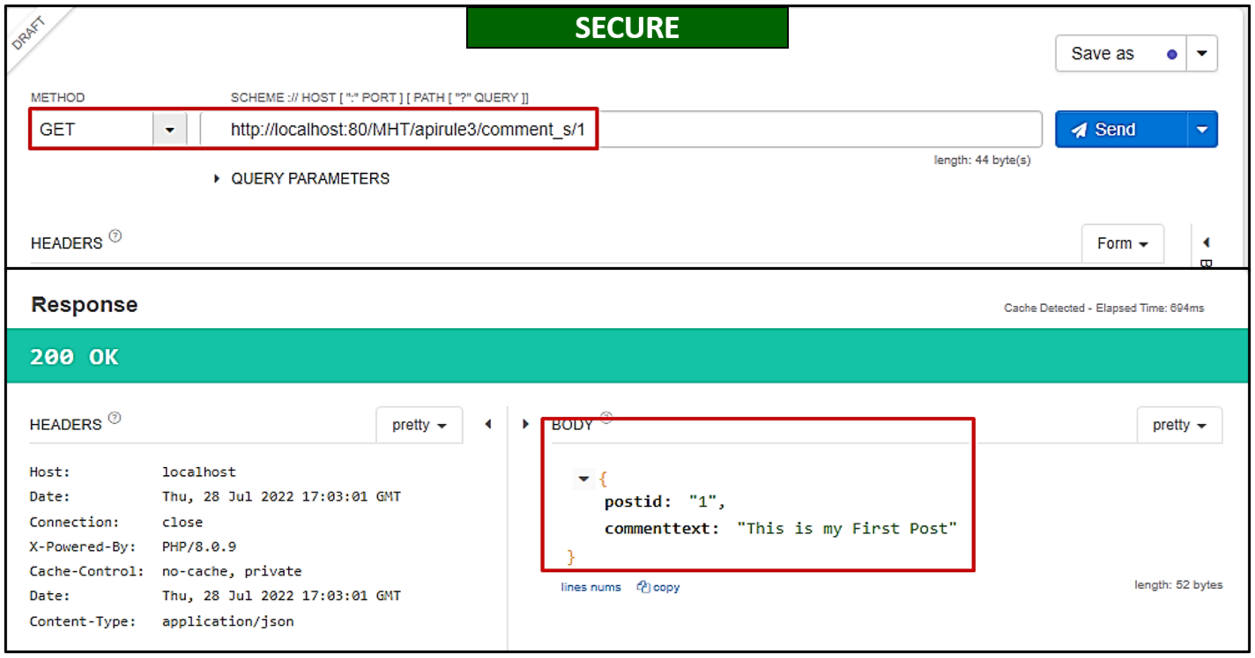
GET http://localhost:80/MHT/apirule3/comment_v/1

200 OK

```
Host:           localhost
Date:           Thu, 28 Jul 2022 16:43:53 GMT
Connection:     close
X-Powered-By:   PHP/8.0.9
Cache-Control:  no-cache, private
Date:           Thu, 28 Jul 2022 16:43:53 GMT
Content-Type:   application/json
```

```
{
  id:  1,
  postid:  "1",
  deviceid:  "Android 12.0",
  latitude:  "45.5426274",
  longitude:  "-122.7944111",
  commenttext:  "This is my First Post",
  username:  "baduser007"
}
```

- What is the issue here? The API is sending more data than desired. Instead of relying on a front-end engineer to filter out data, only relevant data must be sent from the database.
- Bob realizing his mistake, updated the endpoint and created a valid endpoint /apirule3/comment_s/{id} that returns only the necessary information to the developer (as shown below).



```
GET  http://localhost:80/MHT/apirule3/comment_s/1
length: 44 byte(s)

200 OK

HEADERS
Host:            localhost
Date:            Thu, 28 Jul 2022 17:03:01 GMT
Connection:      close
X-Powered-By:    PHP/8.0.9
Cache-Control:   no-cache, private
Date:            Thu, 28 Jul 2022 17:03:01 GMT
Content-Type:    application/json

BODY
{
   postid:  "1",
   commenttext:  "This is my First Post"
}
length: 52 bytes
```

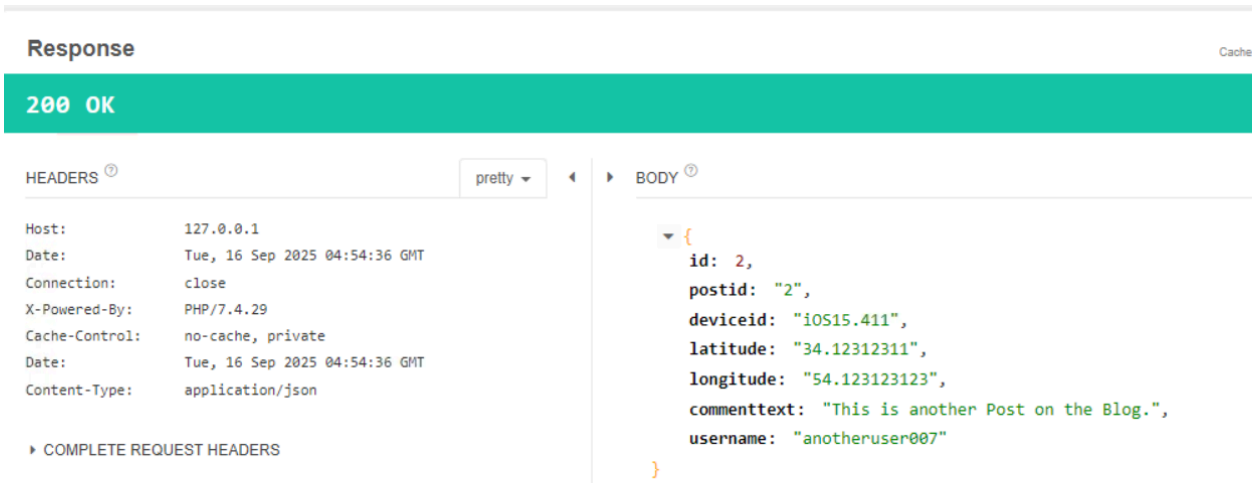
Mitigation Measures

- Never leave sensitive data filtration tasks to the front-end developer.
- Ensure time-to-time review of the response from the API to guarantee it returns only legitimate data and checks if it poses any security issue.
- Avoid using generic methods such as to_string() and to_json().
- Use API endpoint testing through various test cases and verify through automated and manual tests if the API leaks additional data.

****

Answer the questions below:

What is the device ID value for post-ID 2?



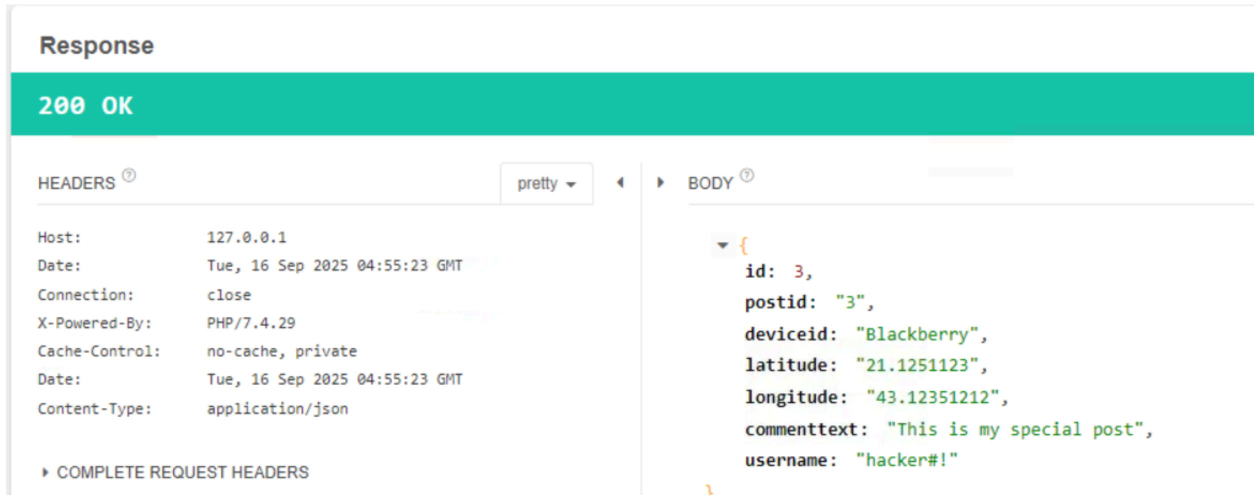
```
200 OK

HEADERS
Host:            127.0.0.1
Date:            Tue, 16 Sep 2025 04:54:36 GMT
Connection:      close
X-Powered-By:    PHP/7.4.29
Cache-Control:   no-cache, private
Date:            Tue, 16 Sep 2025 04:54:36 GMT
Content-Type:    application/json

BODY
{
   id:  2,
   postid:  "2",
   deviceid:  "iOS15.411",
   latitude:  "34.12312311",
   longitude:  "54.123123123",
   commenttext:  "This is another Post on the Blog.",
   username:  "anotheruser007"
}
```

Answer: ios15.411

What is the username value for post-ID 3?



```
Host:            127.0.0.1
Date:            Tue, 16 Sep 2025 04:55:23 GMT
Connection:      close
X-Powered-By:    PHP/7.4.29
Cache-Control:   no-cache, private
Date:            Tue, 16 Sep 2025 04:55:23 GMT
Content-Type:    application/json
```

COMPLETE REQUEST HEADERS

```
{
  id: 3,
  postid: "3",
  deviceid: "Blackberry",
  latitude: "21.1251123",
  longitude: "43.12351212",
  commenttext: "This is my special post",
  username: "hacker#!"
}
```

Answer: hacker#!

Should we use network-level devices for controlling excessive data exposure instead of managing it through APIs (programmatically) - (yea/nay)?
Answer: nay
****

Vulnerability IV - Lack of Resources and Rate Limiting

How Does it Happen?

Lack of resources and rate limiting means that APIs do not enforce any restriction on the frequency of clients' requested resources or the files' size, which badly affects the API server performance and leads to the DoS (Denial of Service) or non-availability of service. Consider a scenario where an API limit is not enforced, thus allowing a user (usually an intruder) to upload several GB files simultaneously or make any number of requests per second. Such API endpoints will result in excessive resource utilisation in network, storage, compute etc.

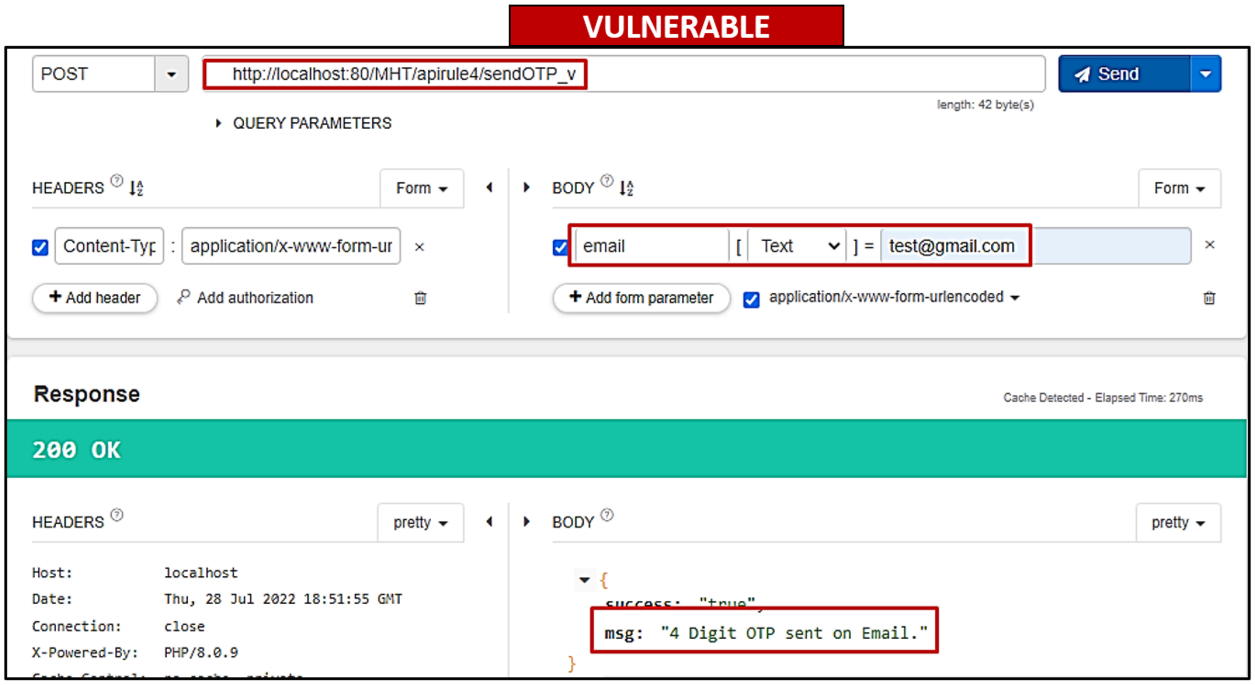
Nowadays, attackers are using such attacks to ensure the non-availability of service for an organization, thus tarnishing the brand reputation through increased downtime. A simple example is non-compliance with the Captcha system on the login form, allowing anyone to make numerous queries to the database through a small script written in Python.

Likely Impact

The attack primarily targets the Availability principles of security; however, it can tarnish the brand's reputation and cause financial loss.

Practical Example

- Continue to use the Chrome browser and Talend API Tester for debugging in the VM.
- The company MHT purchased an email marketing plan (20K emails per month) for sending marketing, password recovery emails etc. Bob realized that he had successfully developed a login API, but there must be a "Forgot Password" option that can be used to recover an account.
- He started building an endpoint /apirule4/sendOTP_v that will send a 4-digit numeric code to the user's email address. An authenticated user will use that One Time Password (OTP) to recover the account.



http://localhost:80/MHT/apirule4/sendOTP_v

Content-Typ : application/x-www-form-ur

email [Text] = test@gmail.com

application/x-www-form-urlencoded

200 OK

```
Host:          localhost
Date:          Thu, 28 Jul 2022 18:51:55 GMT
Connection:    close
X-Powered-By:  PHP/8.0.9
```

```
{
  success: "true",
  msg: "4 Digit OTP sent on Email."
}
```

- What is the issue here? Bob has not enabled any rate limiting in the endpoint. A malicious actor can write a small script and brute force the endpoint, sending many emails in a few seconds and using the company's recently purchased email marketing plan (financial loss).
- Finally, Bob came up with an intelligent solution (/apirule4/sendOTP_s) and enabled rate limiting such that the user has to wait 2 minutes to request an OTP token again.

Mitigation Measures

- Ensure using a captcha to avoid requests from automated scripts and bots.
- Ensure implementation of a limit, i.e., how often a client can call an API within a specified time and notify instantly when the limit is exceeded.
- Ensure to define the maximum data size on all parameters and payloads, i.e., max string length and max number of array elements.

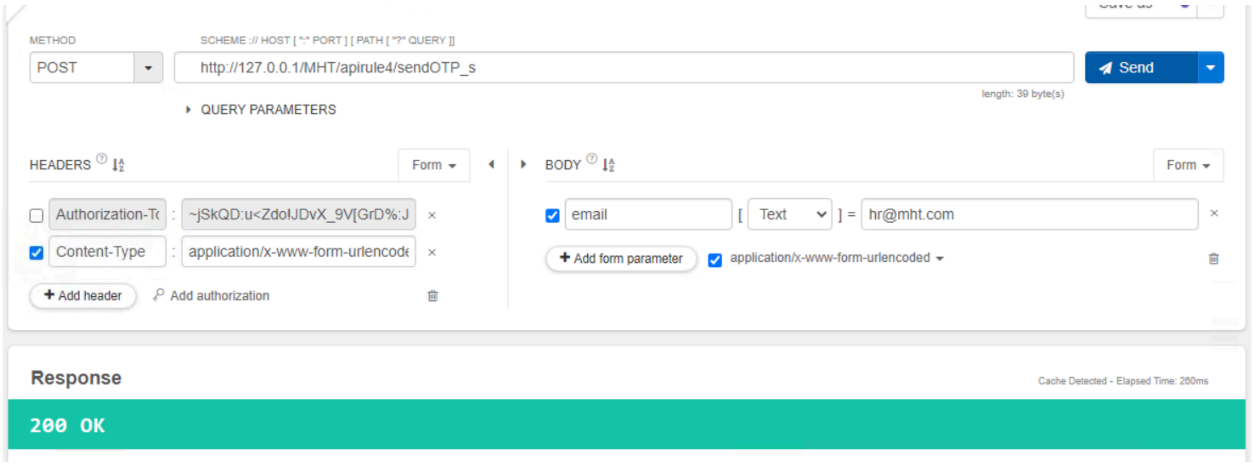
****

Answer the questions below:

Can rate limiting be carried out at the network level through firewall etc. (yea/nay)?

Answer: yea

What is the HTTP response code when you send a POST request to /apirule4/sendOTP_s using the email address hr@mht.com



POST

http://127.0.0.1/MHT/apirule4/sendOTP_s

length: 39 byte(s)

☐	Authorization-Tc	~jSkQD:u<Zdo!JDvX_9V[GrD%:J
✓	Content-Type	application/x-www-form-urlencode

✓	email	Text	hr@mht.com

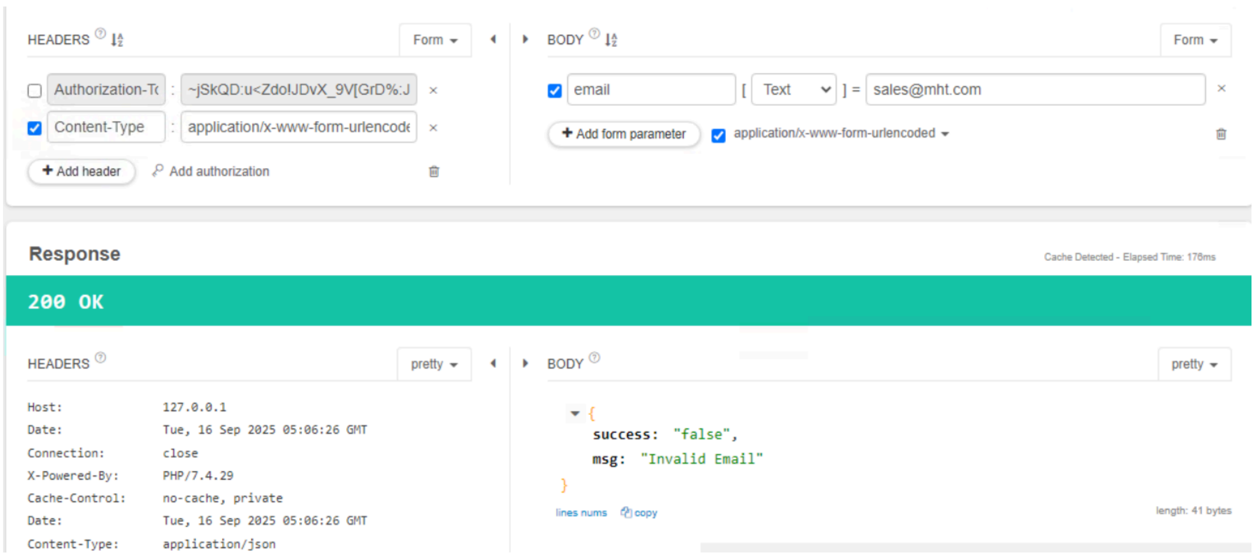
✓ application/x-www-form-urlencoded

Cache Detected - Elapsed Time: 280ms

200 OK

Answer: 200

What is the "msg key" value after an HTTP POST request to /apirule4/sendOTP_s using the email address sale@mht.com?


HEADERS | Form

- [ ] Authorization-Tc : ~jSkQD:u<ZdolJDvX_9V[GrD%:J
- [✓] Content-Type : application/x-www-form-urlencode

BODY | Form

- [✓] email [Text] = sales@mht.com
- [✓] application/x-www-form-urlencoded

Response

Cache Detected - Elapsed Time: 176ms

200 OK

HEADERS | pretty

```
Host:           127.0.0.1
Date:           Tue, 16 Sep 2025 05:06:26 GMT
Connection:     close
X-Powered-By:   PHP/7.4.29
Cache-Control:  no-cache, private
Date:           Tue, 16 Sep 2025 05:06:26 GMT
Content-Type:   application/json
```

BODY | pretty

```
{
   success:  "false",
   msg:  "Invalid Email"
}
```

length: 41 bytes

Answer: Invalid Email
****

Vulnerability V - Broken Function Level Authorization

How Does it Happen?

Broken Function Level authorization reflects a scenario where a low privileged user (e.g., sales) bypasses system checks and gets access to confidential data by impersonating a high privileged user (Admin). Consider a scenario of complex access control policies with various hierarchies, roles, and groups and a vague separation between regular and administrative functions leading to severe authorization flaws. By taking advantage of these issues, the intruders can easily access the unauthorized resources of another user or, most dangerously – the administrative functions.

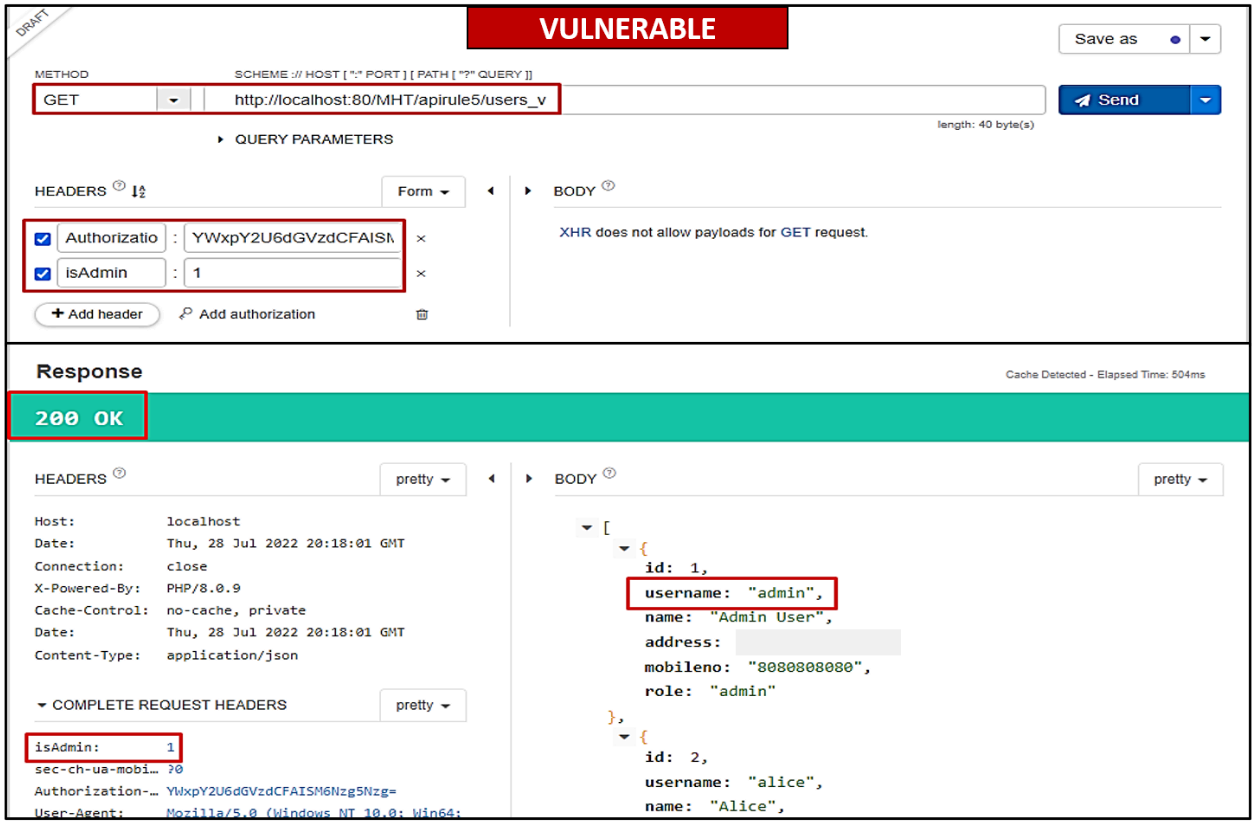
Broken Function Level authorization reflects IDOR permission, where a user, most probably an intruder, can perform administrative-level tasks. APIs with complex user roles and permissions that can span the hierarchy are more prone to this attack.

Likely Impact

The attack primarily targets the authorization and non-repudiation principles of security. Broken Functional Level authorization can lead an intruder to impersonate an authorized user and let the malicious actor get administrative rights to perform sensitive tasks.

Practical Example

- Continue to use the Chrome browser and Talend API Tester for debugging in the VM.
- Bob has been assigned another task to develop an admin dashboard for company executives so that they can view all employee's data and perform specific tasks.
- Bob developed an endpoint /apirule5/users_v to fetch data of all employees from the database. To add protection, he added another layer to security by adding a special header isAdmin in each request. The API only fetches employee information from the database if isAdmin=1 and Authorization-Token are correct. The authorization token for HR user Alice is YWxpY2U6dGVzdCFAISM6Nzg5Nzg=.



METHOD: GET
SCHEME :// HOST [":" PORT] [PATH ["?" QUERY]]: http://localhost:80/MHT/apirule5/users_v
length: 40 byte(s)
QUERY PARAMETERS

HEADERS
Authorizatio : YWxpY2U6dGVzdCFAISN
isAdmin : 1

BODY
XHR does not allow payloads for GET request.

Response
Cache Detected - Elapsed Time: 504ms
200 OK

HEADERS

```
Host:           localhost
Date:           Thu, 28 Jul 2022 20:18:01 GMT
Connection:     close
X-Powered-By:   PHP/8.0.9
Cache-Control:  no-cache, private
Date:           Thu, 28 Jul 2022 20:18:01 GMT
Content-Type:   application/json
```

COMPLETE REQUEST HEADERS

```
isAdmin:            1
sec-ch-ua-mobi…     ?0
Authorization-…     YWxpY2U6dGVzdCFAISM6Nzg5Nzg=
User-Agent:         Mozilla/5.0 (Windows NT 10.0; Win64;
```

BODY

```
[
  {
    id:  1,
    username:  "admin",
    name:  "Admin User",
    address:  [redacted]
    mobileno:  "8080808080",
    role:  "admin"
  },
  {
    id:  2,
    username:  "alice",
    name:  "Alice",
```

- We can see that Alice is a non-admin user (HR) but can see all employee's data by setting custom requests to the endpoint with isAdmin value = 1.
- The issue can be resolved programmatically by implementing correct authorization rules and checking the functional roles of each user in the database during the query. Bob implemented another endpoint /apirule5/users_s that validates each user's role and only shows employees' data if the role is Admin.

Mitigation Measures

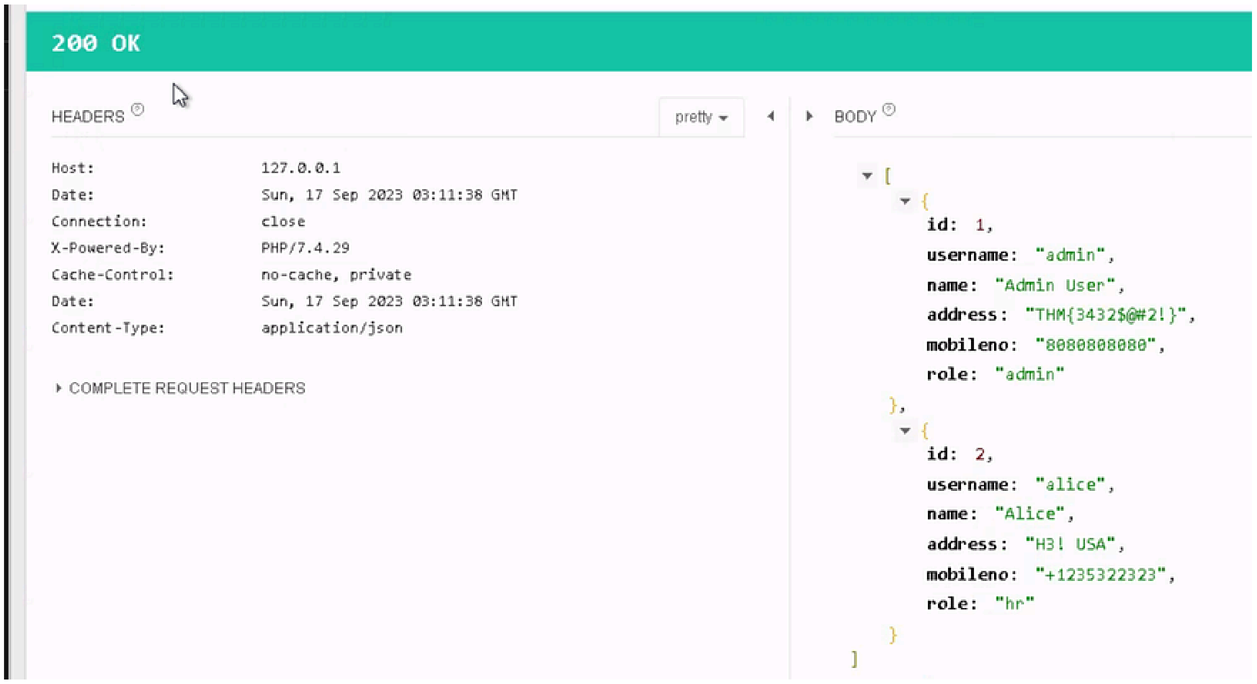
- Ensure proper design and testing of all authorization systems and deny all access by default.

- Ensure that the operations are only allowed to the users belonging to the authorized group.
- Make sure to review API endpoints against flaws regarding functional level authorization and keep in mind the apps and group hierarchy's business logic.

****

Answer the questions below:

What is the mobile number for the username Alice?



```
200 OK

HEADERS                                     BODY
Host:            127.0.0.1                  [
Date:            Sun, 17 Sep 2023 03:11:38 GMT   {
Connection:      close                        id: 1,
X-Powered-By:    PHP/7.4.29                   username: "admin",
Cache-Control:   no-cache, private            name: "Admin User",
Date:            Sun, 17 Sep 2023 03:11:38 GMT address: "THM{3432$@#2!}",
Content-Type:    application/json             mobileno: "8080808080",
                                              role: "admin"
▸ COMPLETE REQUEST HEADERS                  },
                                            {
                                              id: 2,
                                              username: "alice",
                                              name: "Alice",
                                              address: "H3! USA",
                                              mobileno: "+1235322323",
                                              role: "hr"
                                            }
                                            ]
```

Answer: +1235322323

Is it a good practice to send isAdmin value through the hidden fields in form requests - yea/nay?

Answer: nay

What is the address flag of username admin?

Answer: THM{3432$@#2!}

****