OWASP API Security Top 2: Part 2

Quick Recap

In the previous room, we studied the first five principles of OWASP API Security. Now in this room, we will briefly discuss the remaining principles and their potential impact and mitigation measures.

Learning Objectives

- Identification of security misconfigurations
- Preventing Denial of Service (DoS) against the API
- Ensuring appropriate logging and monitoring

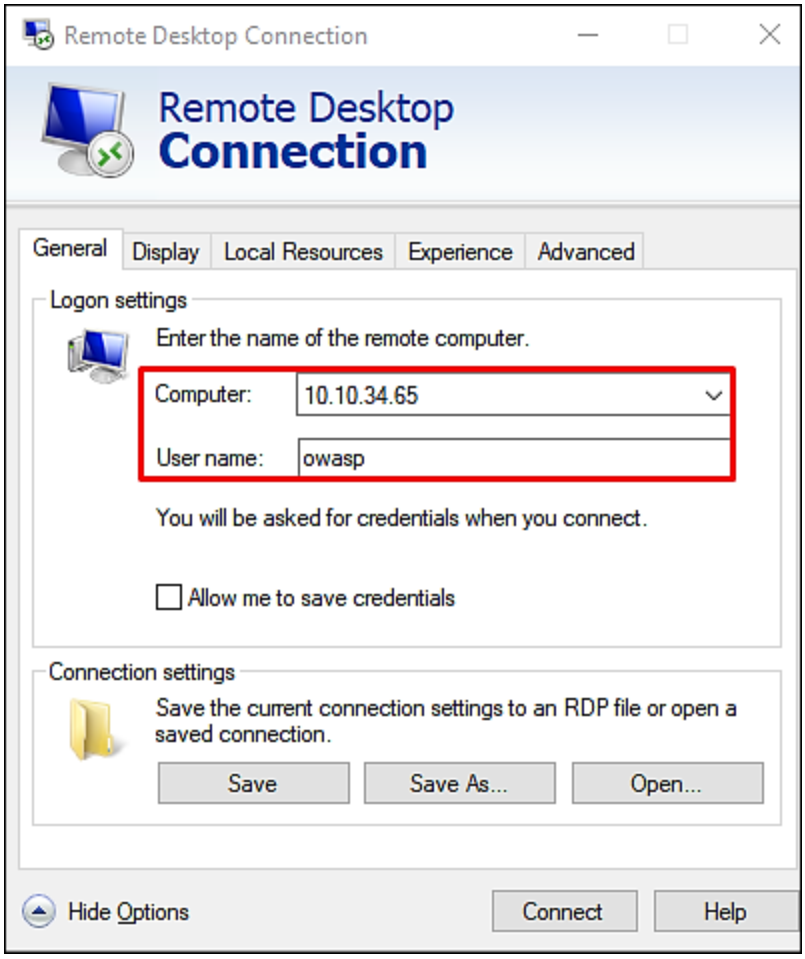
Connecting to the Machine

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: MACHINE_IP
- Username: Administrator
- Password: Owasp@123

You can start the virtual machine by clicking the Start Machine button. 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.



Vulnerability VI - Mass Assignment

How Does it Happen?

Mass assignment reflects a scenario where client-side data is automatically bound with server-side objects or class variables. However, hackers exploit the feature by first understanding the application's business logic and sending specially crafted data to the server, acquiring administrative access or inserting tampered data. This functionality is widely exploited in the latest frameworks like Laravel, Code Ignitor etc.

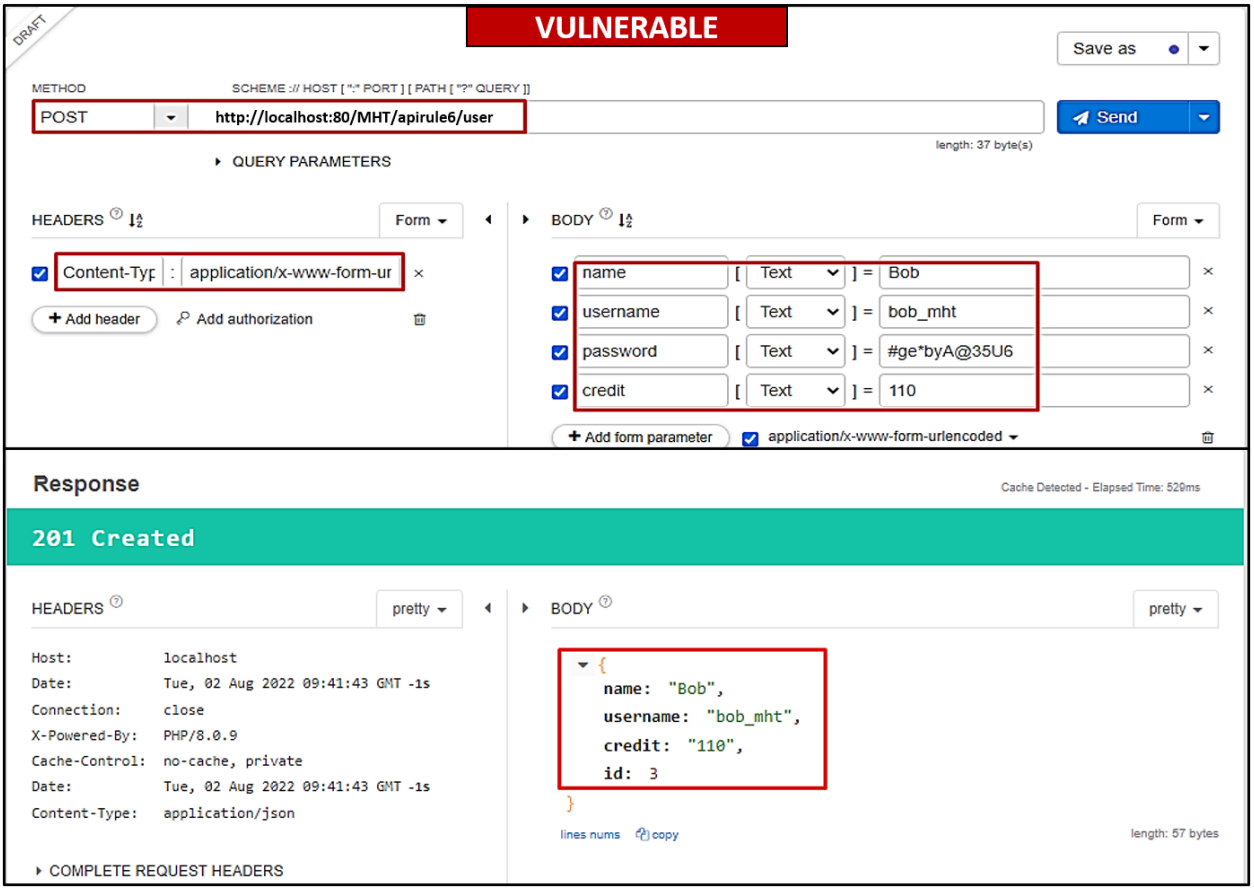
Consider a user's profiles dashboard where users can update their profile like associated email, name, address etc. The username of the user is a read-only attribute and cannot be changed; however, a malicious actor can edit the username and submit the form. If necessary filtration is not enabled on the server side (model), it will simply insert/update the data in the database.

Likely Impact

The attack may result in data tampering and privilege escalation from a regular user to an administrator.

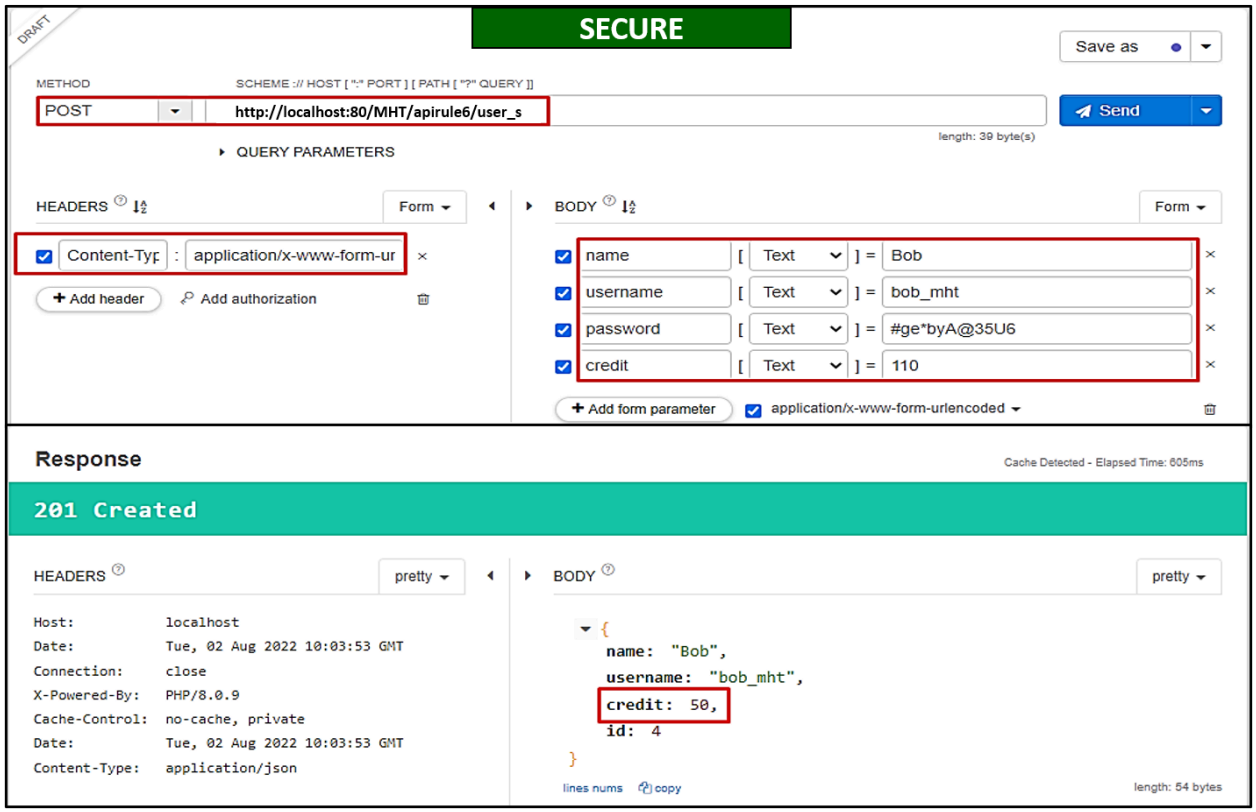
Practical Example

- 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 has been assigned to develop a signup API endpoint /apirule6/user that will take a name, username and password as input parameters (POST). The user's table has a credit column with a default value of 50. Users will upgrade their membership to have a larger credit value.
- Bob has successfully designed the form and used the mass assignment feature in Laravel to store all the incoming data from the client side to the database (as shown below).



- What is the problem here? Bob is not doing any filtering on the server side. Since using the mass assignment feature, he is also inserting credit values in the database (malicious actors can update that value).

- The solution to the problem is pretty simple. Bob must ensure necessary filtering on the server side (apirule6/user_s) and ensure that the default value of credit should be inserted as 50, even if more than 50 is received from the client side (as shown below).



Mitigation Measures

- Before using any framework, one must study how the backend insertions and updates are carried out. In the Laravel framework, fillable and guarded arrays mitigate the above-mentioned scenarios.
- Avoid using functions that bind an input from a client to code variables automatically.
- Allowlist those properties only that need to get updated from the client side.

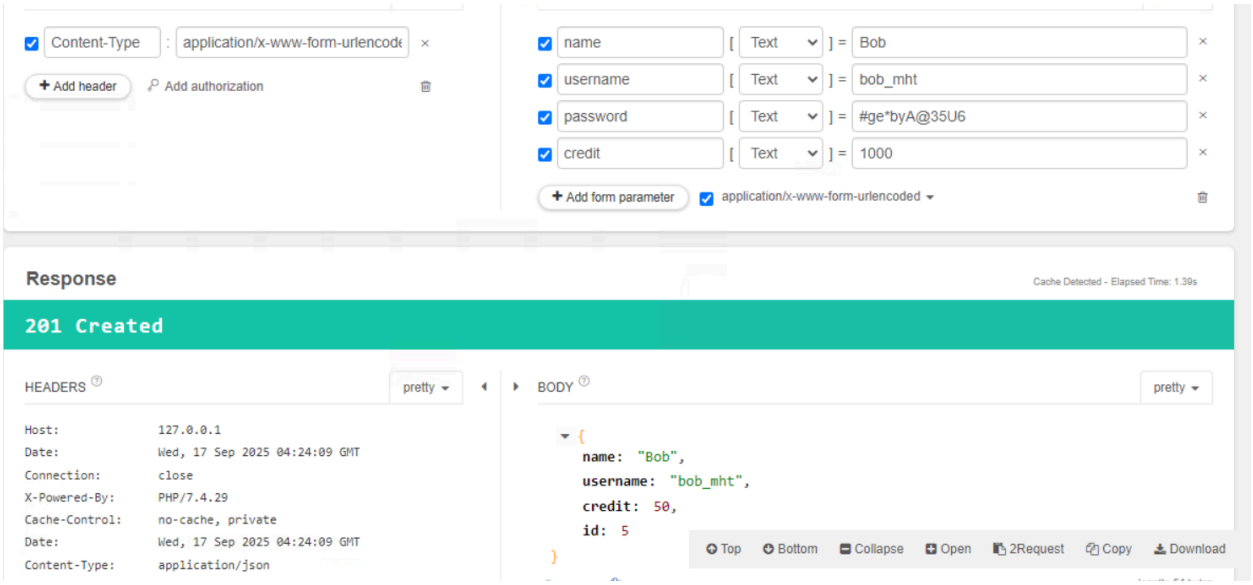
****

Answer the questions below:

Is it a good practice to blindly insert/update user-provided data in the database (yea/nay)?

Answer: nay

Using /apirule6/user_s, insert a record in the database using the credit value as 1000.



What would be the returned credit value after performing Question#2?
Answer: 50
****

Vulnerability VII - Security Misconfiguration

How Does it Happen?

Security misconfiguration depicts an implementation of incorrect and poorly configured security controls that put the security of the whole API at stake. Several factors can result in security misconfiguration, including improper/incomplete default configuration, publically accessible cloud storage, Cross-Origin Resource Sharing (CORS), and error messages displayed with sensitive data. Intruders can take advantage of these misconfigurations to perform detailed reconnaissance and get unauthorized access to the system.

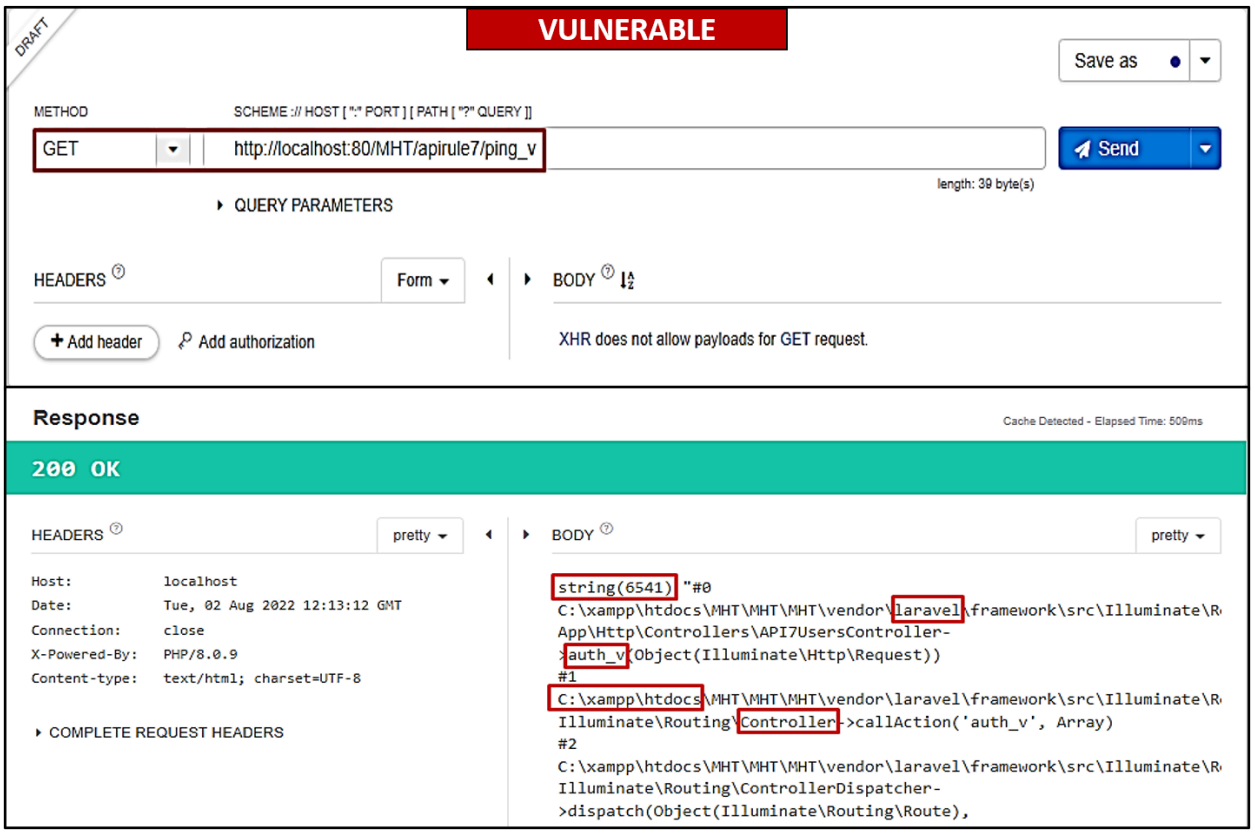
Security misconfigurations are usually detected by vulnerability scanners or auditing tools and thus can be curtailed at the initial level. API documentation, a list of endpoints, error logs etc., must not be publically accessible to ensure safety against security misconfigurations. Typically, companies deploy security controls like web application firewalls, which are not configured to block undesired requests and attacks.

Likely Impact

Security misconfiguration can give intruders complete knowledge of API components. Firstly, it allows intruders to bypass security mechanisms. Stack trace or other detailed errors can provide the malicious actor access to confidential data and essential system details, further aiding the intruder in profiling the system and gaining entry.

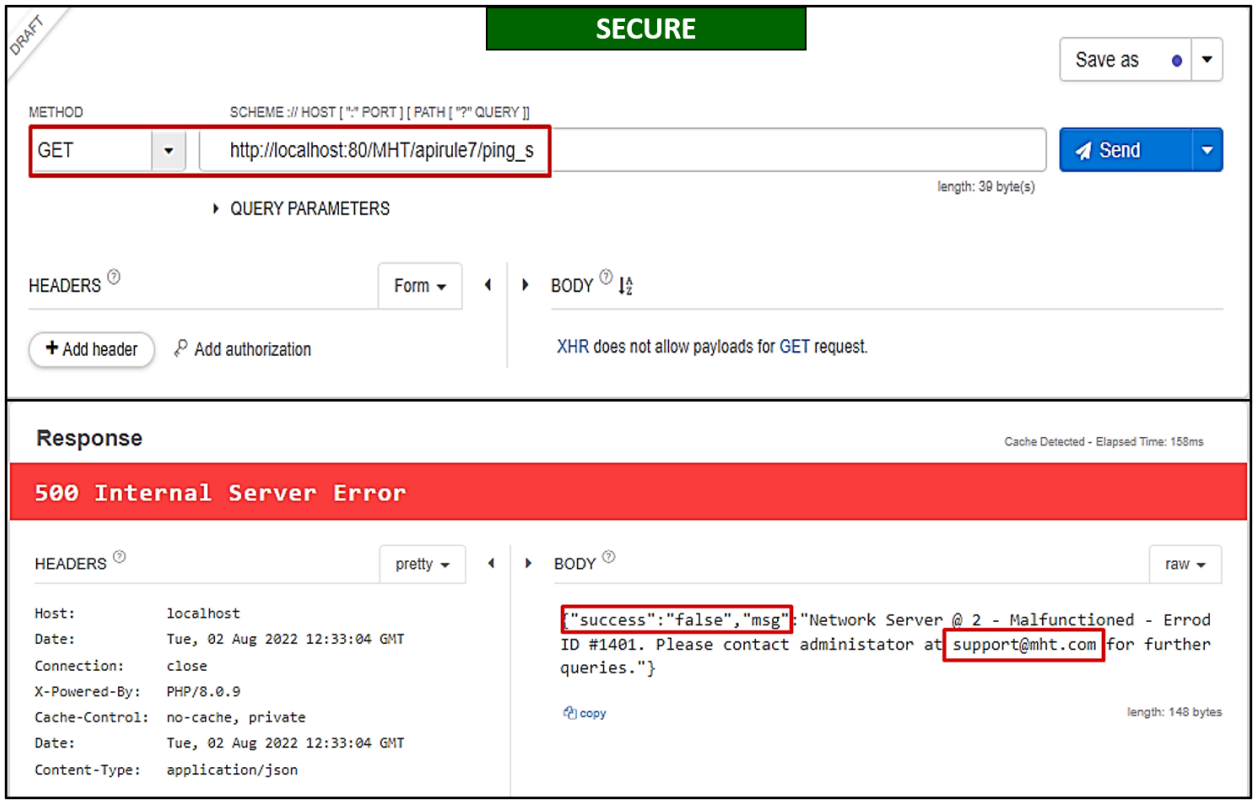
Practical Example

- Continue to use the Chrome browser and Talend API Tester for debugging in the VM.
- The company MHT is facing serious server availability issues. Therefore, they assigned Bob to develop an API endpoint /apirule7/ping_v (GET) that will share details regarding server health and status.
- Bob successfully designed the endpoint; however, he forgot to implement any error handling to avoid any information leakage.



- What is the issue here? In case of an unsuccessful call, the server sends a complete stack trace in response, containing function names, controller and route information, file path etc. An attacker can use the information for profiling and preparing specific attacks on the environment.

- The solution to the issue is pretty simple. Bob will create an API endpoint /apirule7/ping_s that will carry out error handling and only share desired information with the user (as shown below).



Mitigation Measures

- Limit access to the administrative interfaces for authorized users and disable them for other users.
- Disable default usernames and passwords for public-facing devices (routers, Web Application Firewall etc.).
- Disable directory listing and set proper permissions for every file and folder.
- Remove unnecessary pieces of code snippets, error logs etc. and turn off debugging while the code is in production.

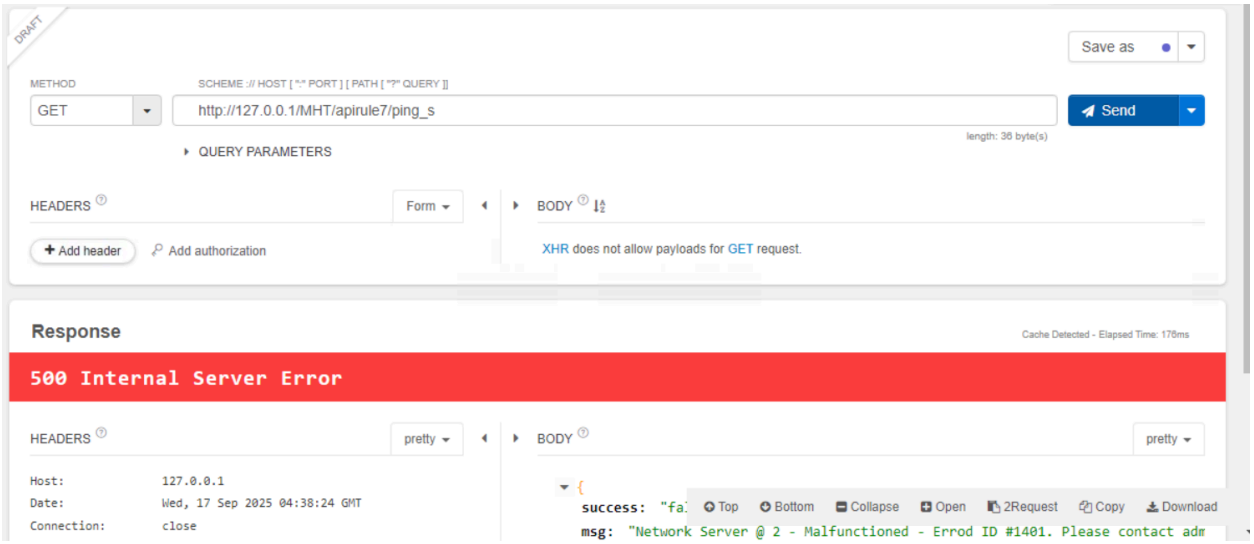
****

Answer the questions below:

Is it an excellent approach to show error logs from the stack trace to general visitors (yea/nay)?

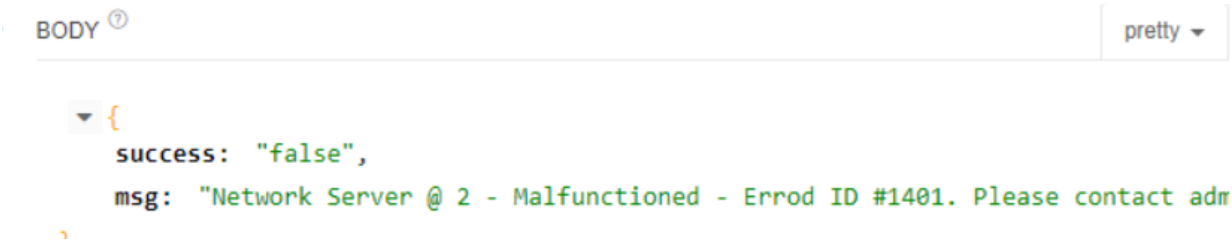
Answer: nay

Try to use the API call /apirule7/ping_s in the attached VM.



What is the HTTP response code?
Answer: 500

What is the Error ID number in the HTTP response message?



Answer: 1401
****

Vulnerability VIII - Injection

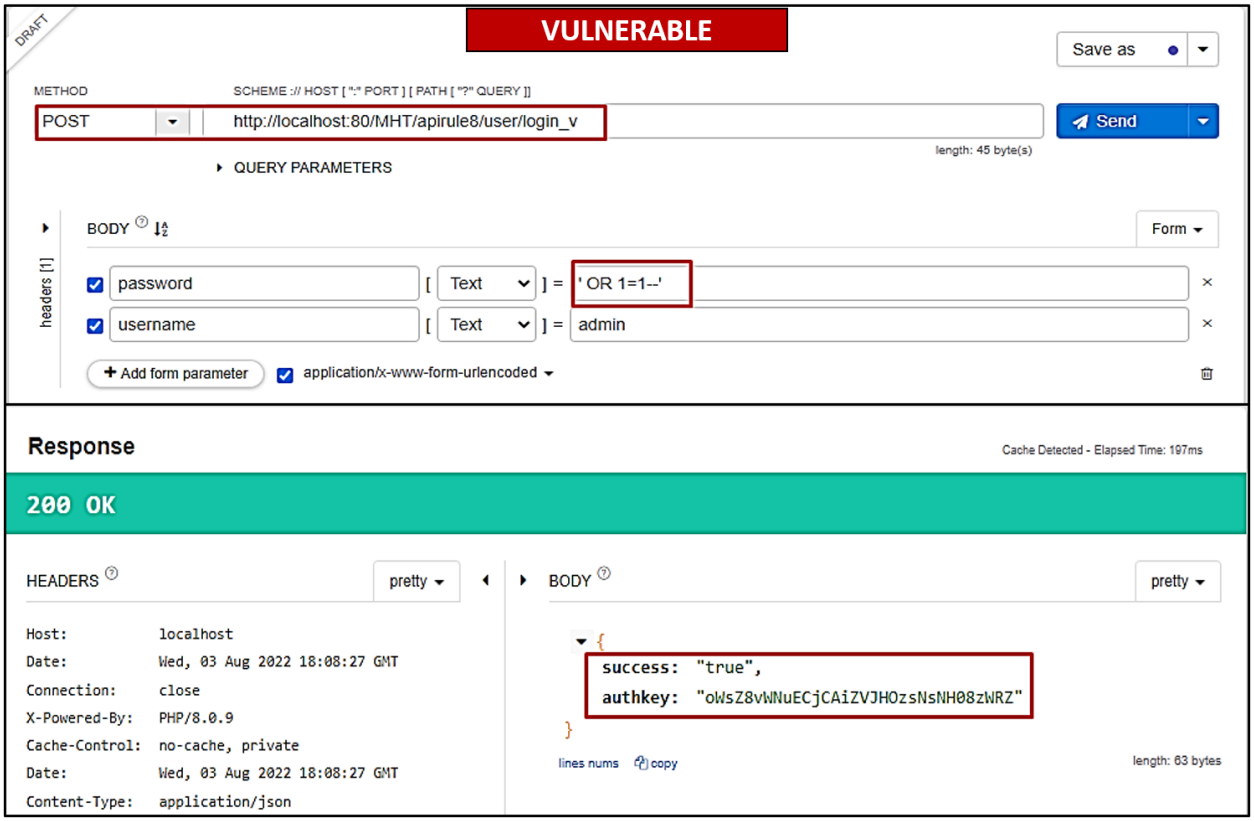
How Does it Happen?
Injection attacks are probably among the oldest API/web-based attacks and are still being carried out by hackers on real-world applications. Injection flaws occur when user input is not filtered and is directly processed by an API; thus enabling the attacker to perform unintended API actions without authorization. An injection may come from Structure Query Language (SQL), operating system (OS) commands, Extensible Markup Language (XML) etc. Nowadays, frameworks offer functionality to protect against this attack through automatic sanitization of data; however, applications built in custom frameworks like core PHP are still susceptible to such attacks.

Likely Impact

Injection flaws may lead to information disclosure, data loss, DoS, and complete account takeover. The successful injection attacks may also cause the intruders to access the sensitive data or even create new functionality and perform remote code execution.

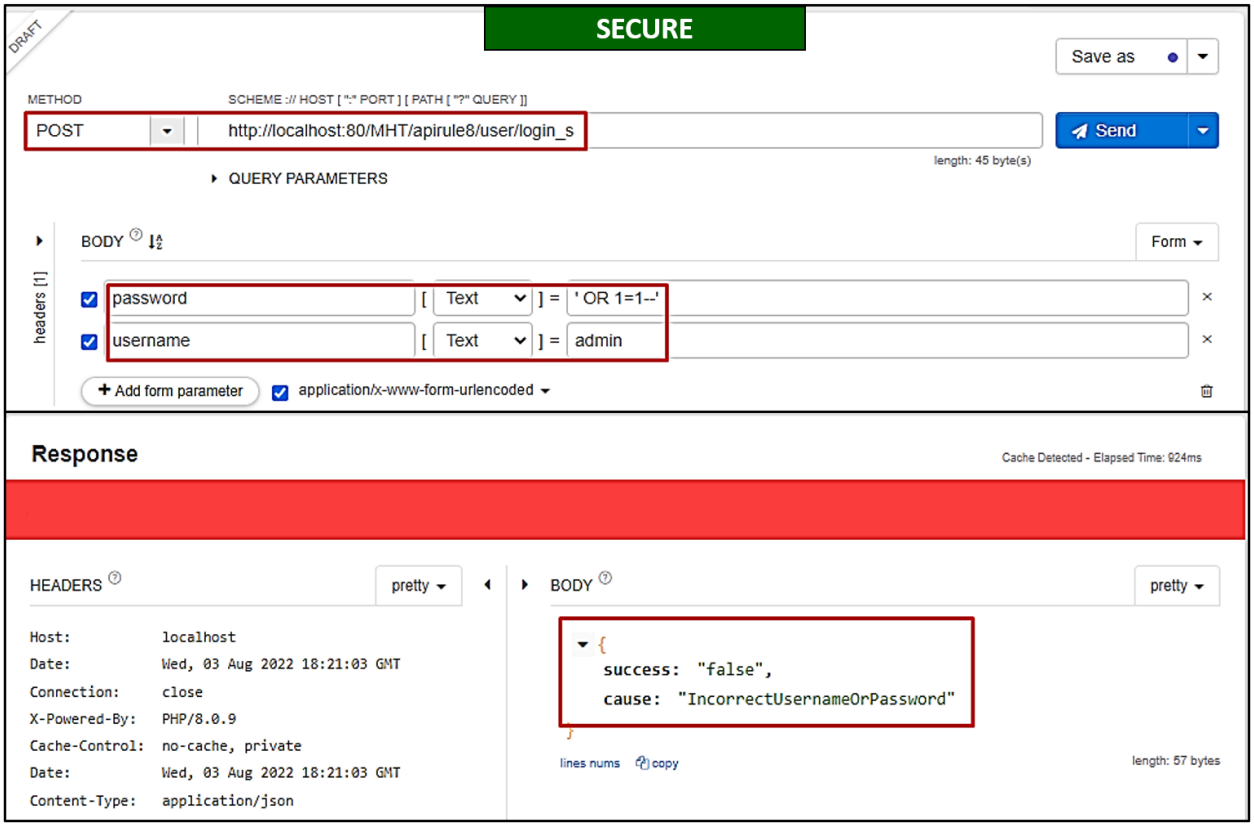
Practical Example

- Continue to use the Chrome browser and Talend API Tester for debugging in the VM.
- A few users of company MHT reported that their account password had changed, and they could not further log in to their original account. Consequently, the dev team found that Bob had developed a vulnerable login API endpoint /apirule8/user/login_v that is not filtering user input.
- A malicious attacker requires the username of the target, and for the password, they can use the payload ' OR 1=1--' and get an authorization key for any account (as shown below).



- Bob immediately realized his mistake; he updated the API endpoint to /apirule8/user/login_s and used parameterized queries and built-in filters of Laravel to sanitize user input.

- As a result, all malicious payloads on username and password parameters were effectively mitigated (as shown below)



Mitigation Measures

- Ensure to use a well-known library for client-side input validation.
- If a framework is not used, all client-provided data must be validated first and then filtered and sanitized.
- Add necessary security rules to the Web Application Firewall (WAF). Most of the time, injection flaws can be mitigated at the network level.
- Make use of built-in filters in frameworks like Laravel, Code Ignitor etc., to validate and filter data.

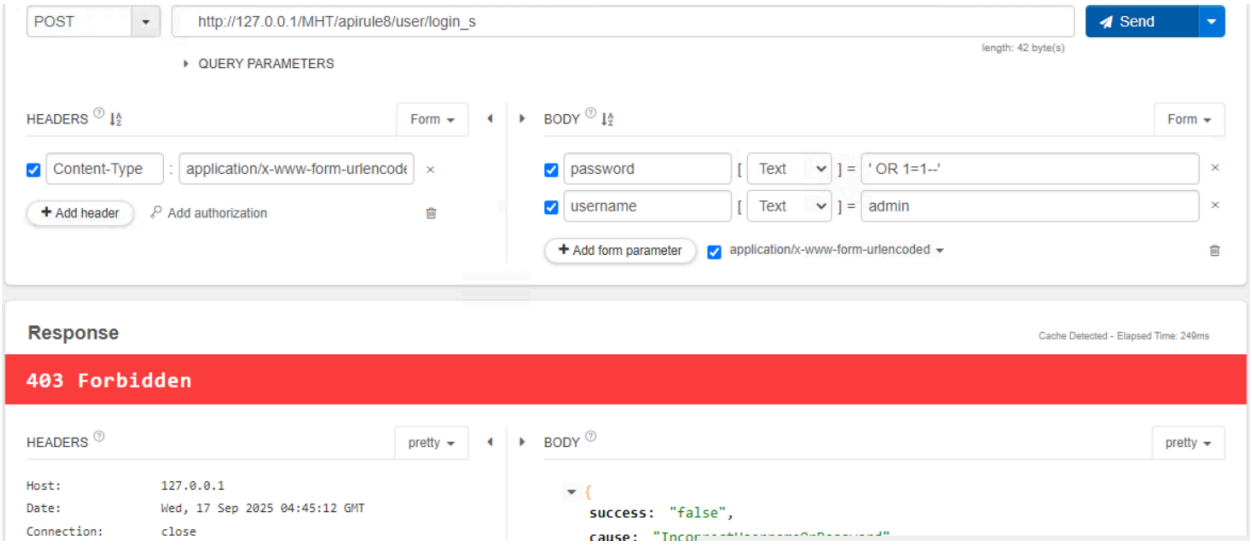
****

Answer the questions below:

Can injection attacks be carried out to extract data from the database (yea/nay)?
Answer: yea

Can injection attacks result in remote code execution (yea/nay)?
Answer: yea

What is the HTTP response code if a user enters an invalid username or password?



Answer: 403

****

Vulnerability IX - Improper Asset Management

How Does it Happen?

Inappropriate Asset Management refers to a scenario where we have two versions of an API available in our system; let's name them APIv1 and APIv2. Everything is wholly switched to APIv2, but the previous version, APIv1, has not been deleted yet. Considering this, one might easily guess that the older version of the API, i.e., APIv1, doesn't have the updated or the latest security features. Plenty of other obsolete features of APIv1 make it possible to find vulnerable scenarios, which may lead to data leakage and server takeover via a shared database amongst API versions.

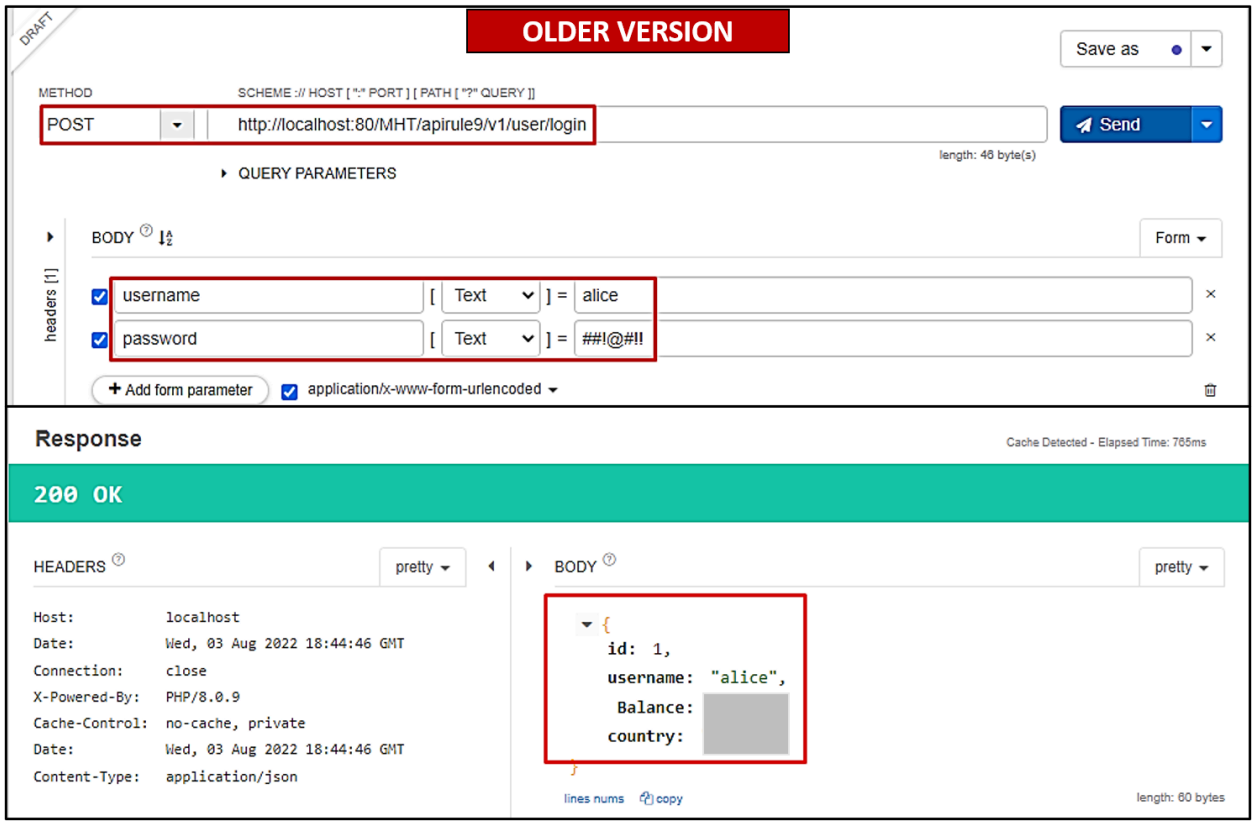
It is essentially about not properly tracking API endpoints. The potential reasons could be incomplete API documentation or absence of compliance with the Software Development Life Cycle. A properly maintained, up-to-date API inventory and proper documentation are more critical than hardware-based security control for an organization.

Likely Impact

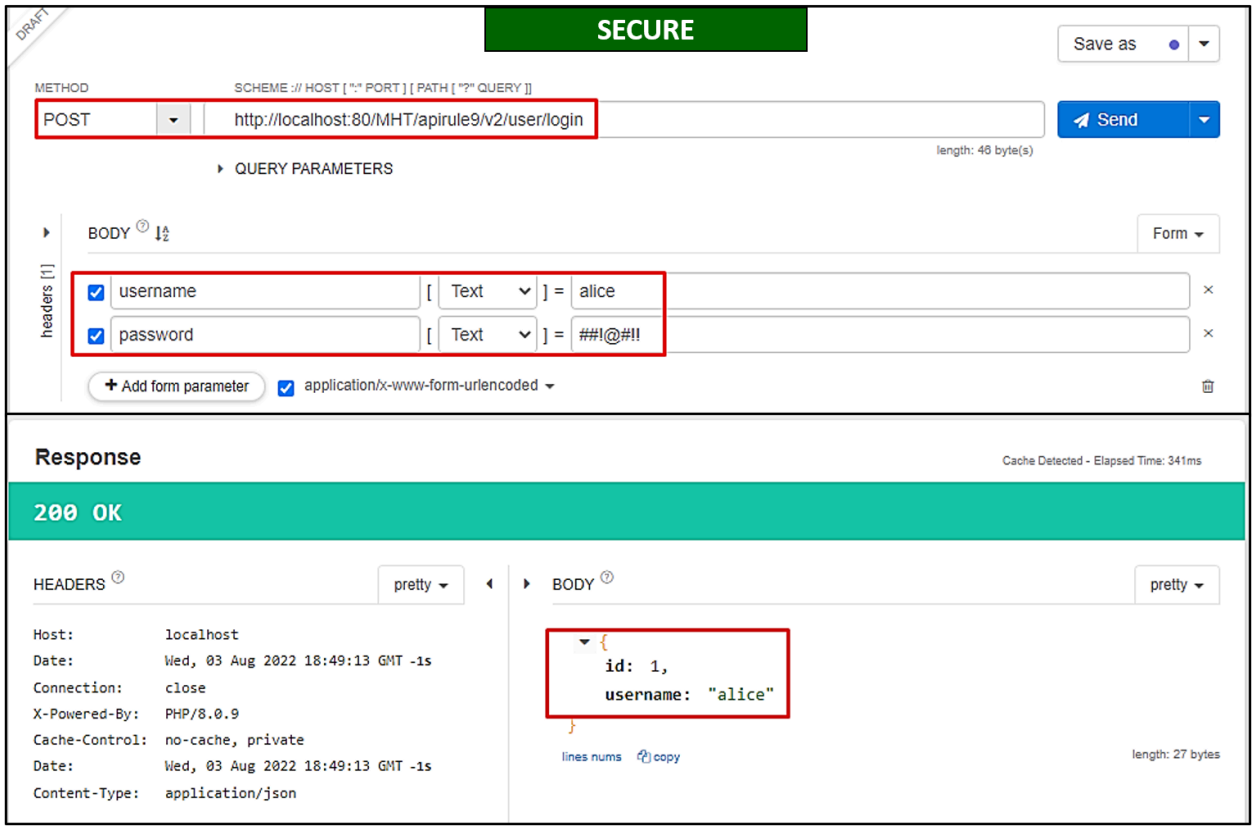
The older or the unpatched API versions can allow the intruders to get unauthorised access to confidential data or even complete control of the system.

Practical Example

- Continue to use the Chrome browser and Talend API Tester for debugging in the VM.
- During API development, the company MHT has developed different API versions like v1 and v2. The company ensured to use the latest versions and API calls but forgot to remove the old version from the server.
- Consequently, it was found that old API calls like apirule9/v1/user/login return more information like balance, address etc., against the user (as shown below).



- Bob being the developer of the endpoint, realized that he must immediately deactivate old and unused assets so that users can only access limited and desired information from the new endpoint /apirul9/v2/user/login (as shown below)



Mitigation Measures

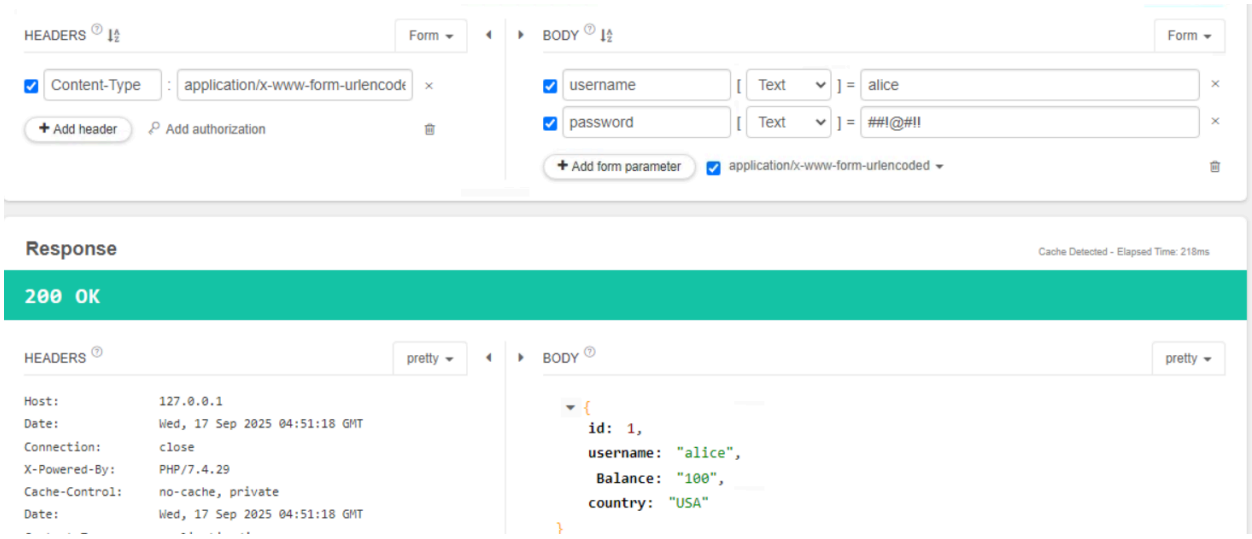
- Access to previously developed sensitive and deprecated API calls must be blocked at the network level.
- APIs developed for R&D, QA, production etc., must be segregated and hosted on separate servers.
- Ensure documentation of all API aspects, including authentication, redirects, errors, CORS policy, and rate limiting.
- Adopt open standards to generate documentation automatically.

****

Answer the questions below:

Is it good practice to host all APIs on the same server (yea/nay)?
Answer: nay

Make an API call to /apirule9/v1/user/login using the username "Alice" and password "##!@#!!".



What is the amount of balance associated with user Alice?
Answer: 100

What is the country of the user Alice?
Answer: USA
****

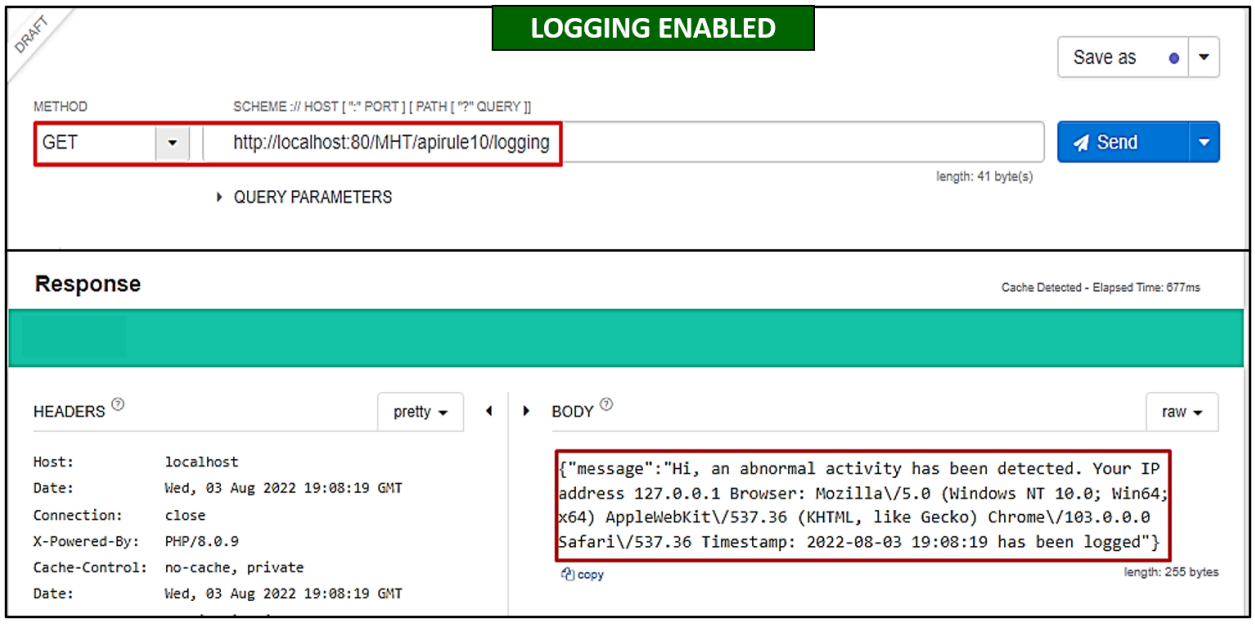
Vulnerability X - Insufficient Logging and Monitoring

How Does it Happen?
Insufficient logging & monitoring reflects a scenario when an attacker conducts malicious activity on your server; however, when you try to track the hacker, there is not enough evidence available due to the absence of logging and monitoring mechanisms. Several organizations only focus on infrastructure logging like network events or server logging but lack API logging and monitoring. Information like the visitor's IP address, endpoints accessed, input data etc., along with a timestamp, enables the identification of threat attack patterns. If logging mechanisms are not in place, it would be challenging to identify the attacker and their details. Nowadays, the latest web frameworks can automatically log requests at different levels like error, debug, info etc. These errors can be logged in a database or file or even passed to a SIEM solution for detailed analysis.

Likely Impact
Inability to identify attacker or hacker behind the attack.

Practical Example

- Continue to use the Chrome browser and Talend API Tester for debugging in the VM.
- In the past, the company MHT has been susceptible to multiple attacks, and the exact culprit behind the attacks could not be identified. Therefore, Bob was assigned to make an API endpoint /apirule10/logging (GET) that will log users' metadata (IP address, browser version etc.) and save it in the database as well (as shown below).



- Later, it was also decided that the same would be forwarded to a SIEM solution for correlation and analysis.

Mitigation Measures

- Ensure use of the Security Information and Event Management (SIEM) system for log management.
- Keep track of all denied accesses, failed authentication attempts, and input validation errors, using a format imported by SIEM and enough detail to identify the intruder.
- Handle logs as sensitive data and ensure their integrity at rest and transit. Moreover, implement custom alerts to detect suspicious activities as well.

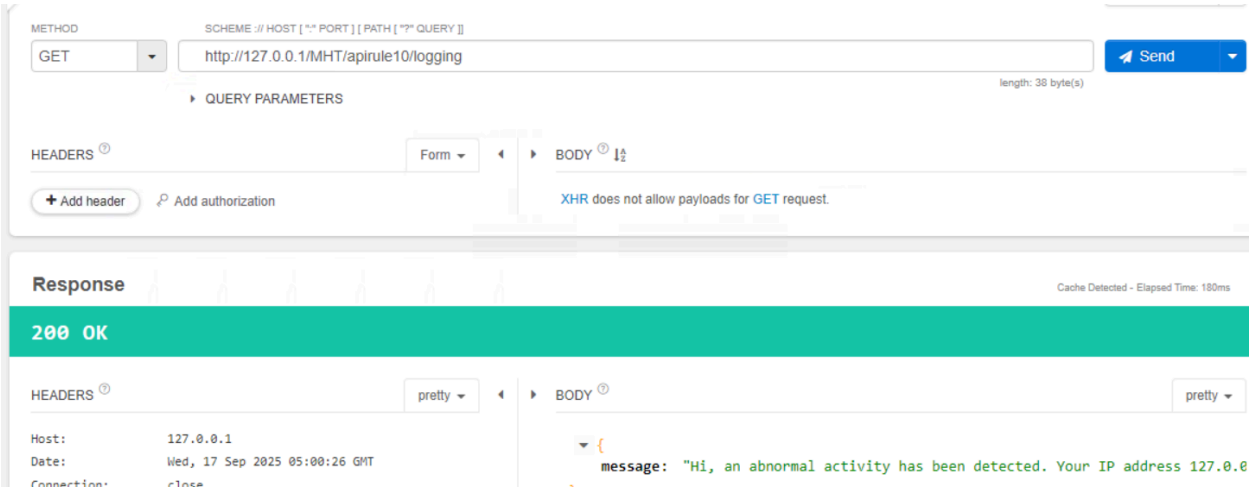
****

Answer the questions below:

Should the API logs be publically accessible so that the attacker must know they are being logged (yea/nay)?

Answer: nay

What is the HTTP response code in case of successful logging of user information?



Answer: 200