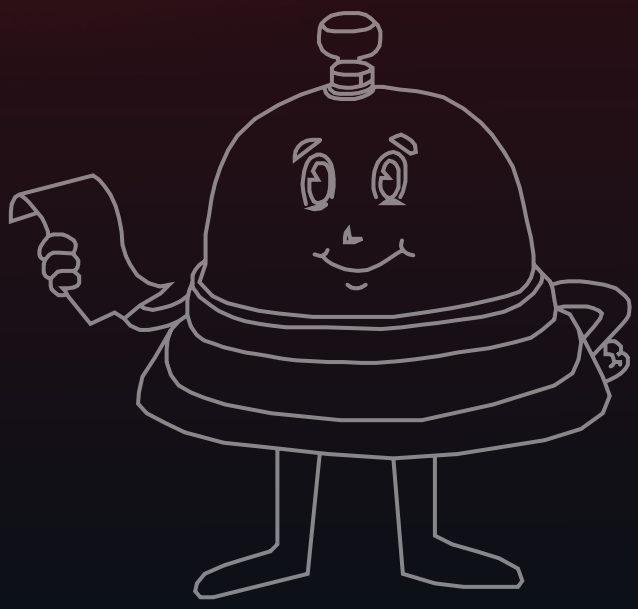
KLEAP

CYBERSECURITY

# API Security Report



# Table of Contents

# Statement of Confidentiality

This pentest report contains confidential and proprietary information belonging to KLEAP Technologies Pvt. Ltd. and Client. It is intended solely for the use of the Client and KLEAP Technologies Pvt. Ltd. The information provided within this report should not be disclosed, distributed, or shared with any third parties without the explicit written consent of both KLEAP Technologies Pvt. Ltd. and Client. Any unauthorized use or disclosure of this information is strictly prohibited and may result in legal action.

# Executive Summary

Client engaged KLEAP Technologies Pvt. Ltd. to perform penetration testing of the APIs. The primary goal of this API penetration testing project was to identify any potential areas of concern associated with the API in its current state and determine the extent to which the system may be breached by an attacker possessing a particular skill and motivation. The assessment was performed in accordance with the “best-in-class” practices as defined by ISECOM's Open Source Security Testing Methodology Manual (OSSTMM) and Open Web Application Security Project (OWASP).

KLEAP Technologies Pvt. Ltd. conducted the penetration testing during the period of March 20th, 2024 to April 11th, 2024. All testing activities were performed on the staging environment provided by the customer and completely isolated from the production data. While performing the testing activities, KLEAP Technologies Pvt. Ltd. emulated an external attacker without prior knowledge of the environment. To test the user-authenticated area and privilege escalation vulnerabilities, the customer supplied KLEAP Technologies Pvt. Ltd. credentials for several registered user and admin accounts.

# Scoping and Time Limitations

Scoping during the engagement did not permit denial of service or social engineering across all testing components.

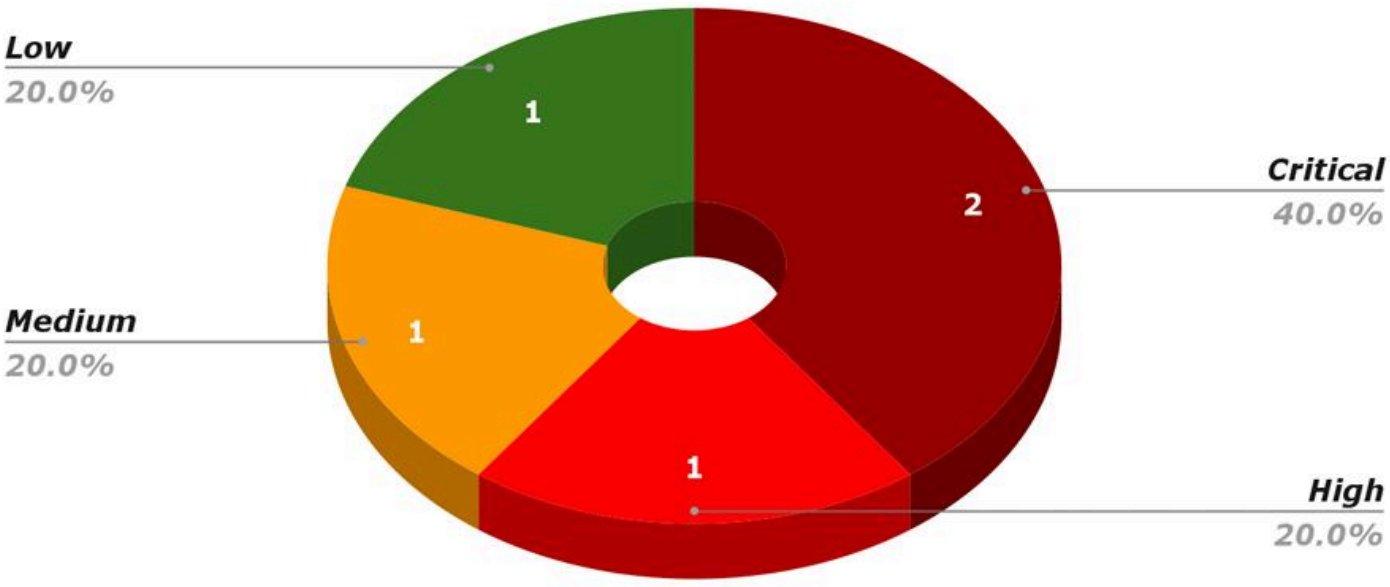
Time limitations were in place for testing. API penetration testing was permitted for seven (7) business days.

# Testing Summary

(Overall Summary of the findings)

| Scope | Critical | High | Medium | Low | Info | Total |
|---|---|---|---|---|---|---|
| Client AD | 2 | 1 | 1 | 1 | 0 | 5 |

Table 1: Findings per asset




**Vulnerability Index**

# Vulnerability Status

| Sr. No. | Vulnerability | Severity | Status |
|---|---|---|---|
| 1 | Excessive Information Disclosure | CRITICAL | OPEN |
| 2 | Mass Assignment leads to Privilege Escalation | CRITICAL | OPEN |
| 3 | SQL Injection | HIGH | OPEN |
| 4 | User Enumeration | MEDIUM | OPEN |
| 5 | Server Version Disclosure | LOW | OPEN |

# Recommendation

Based on the results of this assessment, KLEAP Technologies Pvt. Ltd. has the following high-level key recommendations.

## Key Recommendation (Network)

| | |
|---|---|
| **Key Issue** | During the security assessment of the APIs, it was observed that the root cause of the vulnerabilities found were mainly due to improper access control, security misconfigurations and weak cryptographic implementations. |
| **Recommendation** | We recommend implementing strong cryptographic control mechanisms, avoiding sensitive data exposure and also validating all user inputs. |

# Scope Summary

## In-Scope Assets

The following assets were considered explicitly in-scope for testing:

| Assets In-Scope | Hostname / CIDR / IP |
|---|---|
| APIs | VAmPI (Test API) |

## Out-of-Scope Assets

(If any)

| Assets Out-of-Scope | Hostname / CIDR / IP |
|---|---|
| N/A | N/A |

# Methodology

The pentest methodology employed by KLEAP Technologies Pvt. Ltd. follows a systematic approach to assess the security posture of client systems.

Our Penetration Testing Methodology is based on following guidelines and standards:

- Penetration Testing Execution Standard (PTES)
- NIST SP 800-115
- Open Source Security Testing Methodology Manual (OSSTMM)
- SANS: Conducting a Penetration Test on an Organization
- OWASP Testing Guide OWASP Top 10 Application Security Risks

## Engagement Phases

### 1. Reconnaissance

In this phase, the pentester gathers information about the target systems through passive reconnaissance and OSINT techniques. This includes identifying domain names, IP addresses, employee details, and any publicly available information. The goal is to gain a better understanding of the target's infrastructure, potential vulnerabilities, and attack surface.

### 2. Scanning and Enumeration

In this phase, the pentester conducts active scanning to identify live hosts, open ports, and services running on the target systems. Tools like Nmap, Nessus, or OpenVAS are used to perform network scans and identify potential entry points. The identified services are then enumerated to gather more information, such as software versions, configurations, and potential vulnerabilities. This phase helps in identifying potential weaknesses and areas of focus for further assessment.

## 3. Vulnerability Assessment

In this phase, the pentester performs a comprehensive vulnerability assessment using a combination of automated tools and manual techniques. Commercial or open-source vulnerability scanners are utilized to identify common vulnerabilities and misconfigurations. The scan results are manually reviewed to validate and prioritize the identified vulnerabilities based on their severity and potential impact. This phase helps in identifying specific.

## 4. Exploitation

In this phase, the pentester attempts to exploit the identified vulnerabilities to gain unauthorized access or escalate privileges. Ethical hacking techniques are utilized to simulate real-world attack scenarios while ensuring no harm is caused to the target systems. The pentester may use various tools, scripts, or custom exploits to exploit the identified vulnerabilities. The goal is to demonstrate the potential impact of the vulnerabilities and assess the effectiveness of the target's security controls.

## 5. Reporting

In this final phase, the pentester compiles all findings, categorizes them based on severity levels, and provides detailed explanations, proof-of-concept demonstrations, and prioritized recommendations for remediation. The report includes a summary of the pentest engagement, an overview of the methodology used, and a comprehensive analysis of the vulnerabilities discovered. It also includes actionable recommendations to mitigate the identified vulnerabilities and improve the overall security posture of the target systems. The report serves as a valuable resource for the client to understand the security risks and take appropriate measures to address them.

# Vulnerability Classification & Severity

To categorize vulnerabilities according to a commonly understood vulnerability taxonomy, KLEAP Technologies Pvt. Ltd. uses the industry-standard Common Weakness Enumeration (CWE). CWE is a community-developed taxonomy of common software security weaknesses. It serves as a common language, a measuring stick for software security tools, and as a baseline for weakness identification, mitigation, and prevention efforts.

To rate the severity of vulnerabilities, KLEAP Technologies Pvt. Ltd. uses the industry standard Common Vulnerability Scoring System (CVSS) to calculate severity for each identified security vulnerability. CVSS provides a way to capture the principal characteristics of a vulnerability, and produce a numerical score reflecting its severity, as well as a textual representation of that score.

To help prioritize vulnerabilities and assist vulnerability management processes, KLEAP Technologies Pvt. Ltd. translates the numerical CVSS rating to a qualitative representation (such as low, medium, high and critical):

| CVSS Score v3.1 | |
|---|---|
| **Severity** | **Score** |
| **Critical** | 9.0 - 10.0 |
| **High** | 7.0 - 8.9 |
| **Medium** | 4.0 - 6.9 |
| **Low** | 0.1 - 3.9 |
| **Informational** | 0.0 |

# Findings Summary

Findings are sorted by their severity and grouped by the asset and CWE classification. Each asset section will contain a summary. Table 1 in the executive summary contains the total number of identified security vulnerabilities per asset per risk indication.

## Findings Overview

During the engagement, 5 unique vulnerabilities were found across mainly 3 different vulnerability categories. The most common vulnerability types identified were:

- Broken Object Level Authorization
- Injection
- Security Misconfiguration

Exploring the findings further by their actual vulnerability type as defined by CWE, Table 3 shows the number of individual findings and its distribution of severity.

| Vulnerabilities | Critical | High | Medium | Low | Info | Total |
|---|---|---|---|---|---|---|
| Excessive Information Disclosure | 1 | 0 | 0 | 0 | 0 | 0 |
| Mass Assignment leads to Privilege Escalation | 1 | 0 | 0 | 0 | 0 | 0 |
| SQL Injection | 0 | 1 | 0 | 0 | 0 | 0 |
| User Enumeration | 0 | 0 | 1 | 0 | 0 | 0 |
| Server Version Disclosure | 0 | 0 | 0 | 1 | 0 | 0 |
| Server Version Disclosure | 2 | 1 | 1 | 1 | 0 | 0 |

Table 3: severity distribution across vulnerability types

## Findings Overview as per OWASP Standards

During the engagement, 5 unique vulnerabilities were found across mainly 3 different vulnerability categories. The most common vulnerability types identified were:

- Broken Object Level Authorization
- Injection
- Security Misconfiguration

Exploring the findings further by their actual vulnerability type as defined by CWE, Table 3 shows the number of individual findings and its distribution of severity.

| Vulnerabilities | Results | Findings |
|---|---|---|
| API1:2023 Broken Object Level Authorization | **Fail** | 1 |
| API2:2023 Broken Authentication | **Pass** | 0 |
| API3:2023 Broken Object Property Level Authorization | **Fail** | 1 |
| API4:2023 Unrestricted Resource Consumption | **Pass** | 0 |
| API5:2023 Broken Function Level Authorization | **Pass** | 0 |
| API6:2023 Unrestricted Access to Sensitive Business Flows | **Pass** | 0 |
| API7:2023 Server Side Request Forgery | **Pass** | 0 |
| API8:2023 Security Misconfiguration | **Fail** | 1 |
| API9:2023 Improper Inventory Management | **Pass** | 0 |
| API10:2023 Unsafe Consumption of APIs | **Pass** | 0 |

# Technical Findings Details

## 01: Excessive Information Disclosure

| Vulnerability Severity | CWE ID |
|---|---|
| Critical | CEW-200 |
| **OWASP Category** | CVSS Score |
| API3:2023 - Broken Object Property Level Authorization | 9.1 |

### Vulnerability Description

Excessive Information Disclosure occurs when APIs reveal more fields, data, and information than the client requires through the API response.

### Vulnerable API

/users/v1/_debug

### Impact

The vulnerable API exposes excessive and highly sensitive data like user credentials including user email, usernames and passwords.

### Steps to Reporduce

1. Make an API call to the above mentioned API and observe credentials of all users are exposed.

## Remediation

- Strictly avoid all sensitive data like passwords, api keys, license keys, etc. from being disclosed.
- Ensure other users' email and usernames are not visible to any user while making API calls.

## References

https://owasp.org/API-Security/editions/2019/en/0xa3-excessive-data-exposure/

# 02: Mass Assignment leads to Privilege Escalation

| Vulnerability Severity | CWE ID |
|---|---|
| Critical | 915 |
| **CVE ID** | CVSS Score |
| API3:2023 - Broken Object Property Level Authorization | 9.0 |

## Vulnerability Description

During the analysis, it was found that anyone can successfully register as admin and get all the admin privileges.

## Vulnerable API

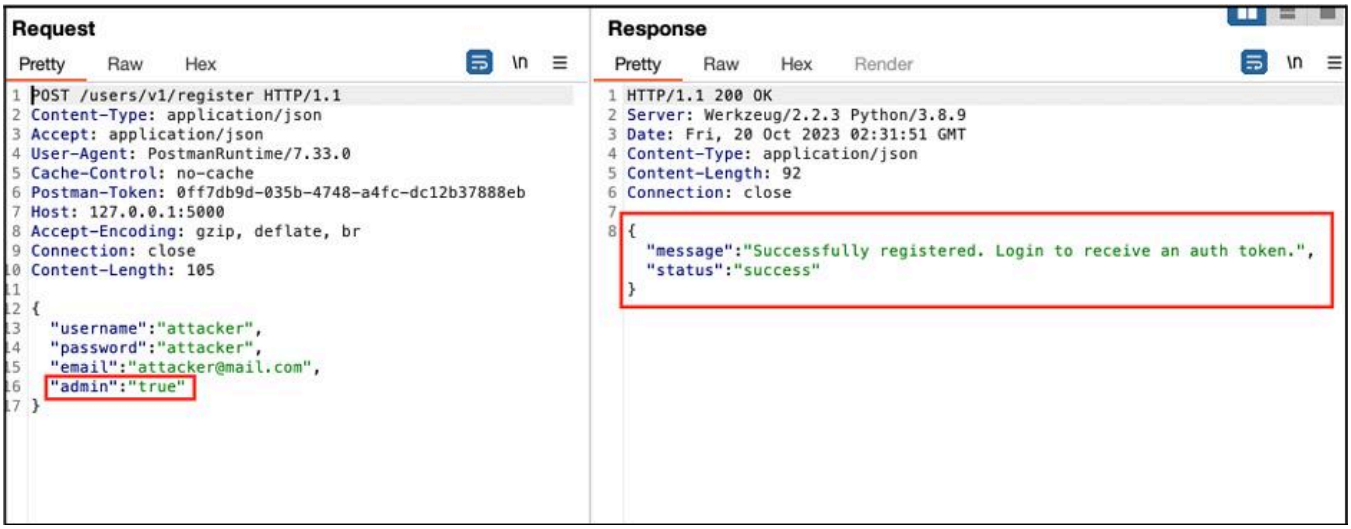
/users/v1/register

## Impact

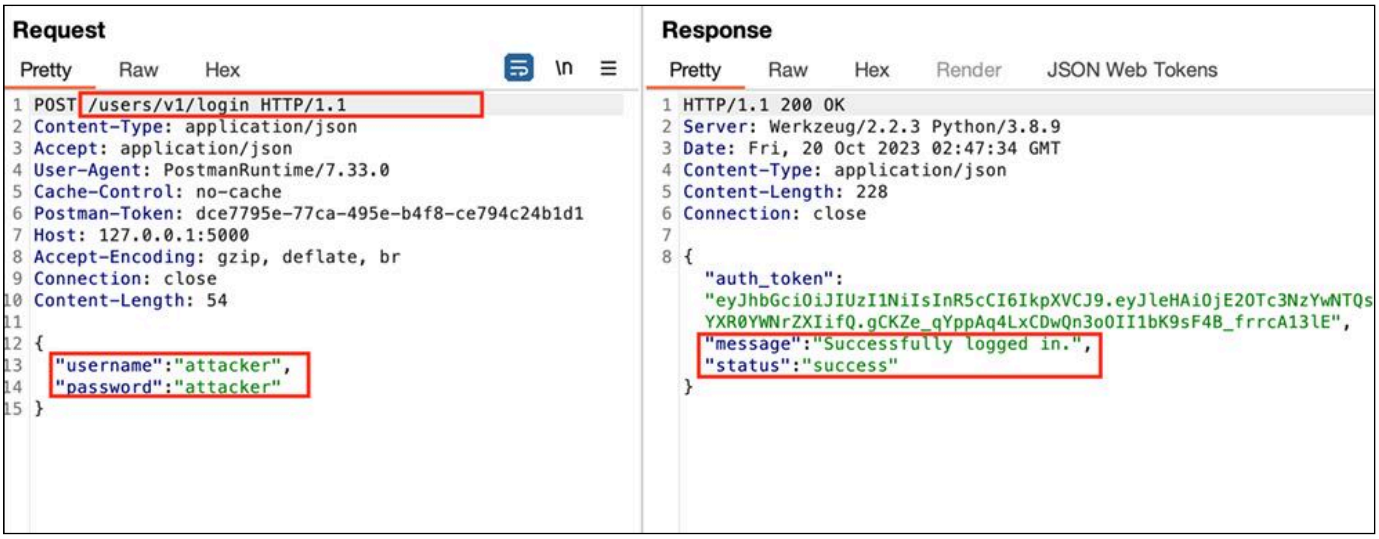
Using admin privileges, an attacker can access & modify all sensitive data and even delete other genuine users.

## Step to Reproduce (Evidences)

1. Set the "admin": "true" while registering on the above API and observe you are successfully registered as an Admin user.




2. Login to the above created admin account.




3. Using admin privileges, observe you can delete any user now.


## Remediation

- If possible, avoid using functions that automatically bind a client's input into code variables, internal objects, or object properties ("Mass Assignment").
- Allow changes only to the object's properties that should be updated by the client.
- Keep returned data structures to the bare minimum, according to the business/functional requirements for the endpoint.

## References

- https://traceable.ai/owasp-api/mass-assignment

# 03: SQL Injection

| Vulnerability Severity | CWE ID |
|---|---|
| High | 89 |
| **CVE ID** | CVSS Score |
| N/A | 7.5 |

## Vulnerability Description

SQL injection (SQLi) is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database. This can allow an attacker to view data that they are not normally able to retrieve.

## Vulnerable API

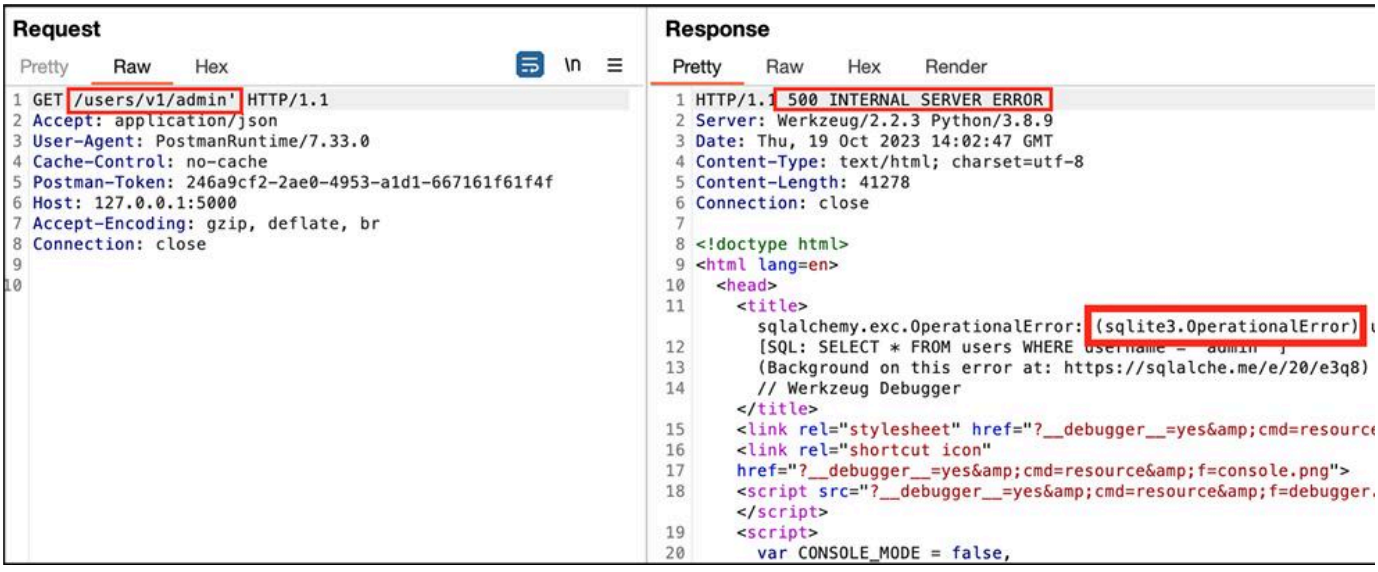
/users/v1/username

## Impact

SQL injection attack in the present scenario can result in unauthorized access to sensitive data, such as tables of the application database.

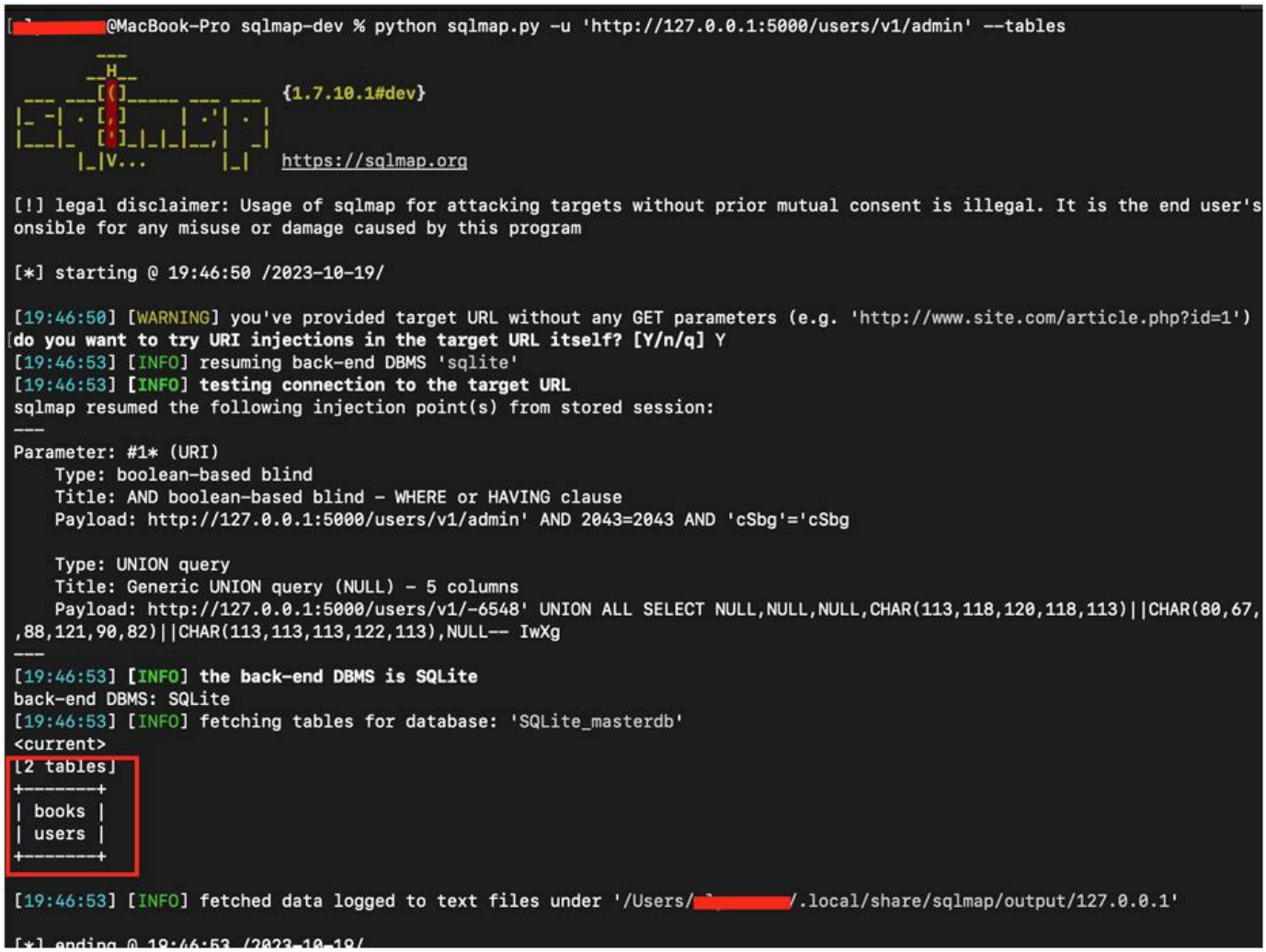
## Step to Reproduce (Evidences)

1. Intercept the above API request. Add ‘ (single quote) at the end of the request and observe the response.




2. Using sqlmap, extract tables against the vulnerable API endpoint.

## Step to Reproduce (Evidences)




## Remediation

- Filter database inputs: Detect and filter out malicious code from user inputs.
- Restrict database code: Prevent unintended database queries and exploration by limiting database procedures and code.
- Restrict database access: Prevent unauthorized data access, exfiltration, or deletion through access control restrictions

## References

- https://owasp.org/www-community/attacks/SQL_Injection

# 04: User Enumeration

| Vulnerability Severity | CWE ID |
|---|---|
| Medium | 204 |
| **CVE ID** | CVSS Score |
| API8:2023 - Security Misconfiguration | 5.9 |

## Vulnerability Description

Whenever an attacker enters an incorrect password, the vulnerable API prompts the error as an incorrect password.

## Vulnerable API

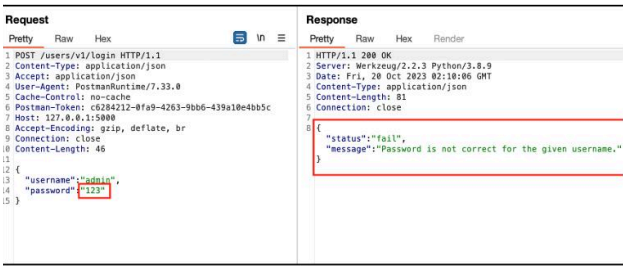
/users/v1/login

## Impact

User enumeration can help an attacker to bruteforce a password for the valid username.

## Step to Reproduce (Evidences)

1. Login using valid username but invalid password and observe the below error message.




## Remediation

- The login API should always prompt "Invalid username or password" whenever login fails.

## References

- https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/03-Identity_Management_Testing/04-Testing_for_Account_Enumeration_and_Guessable_User_Account

# 05: Server Version Disclosure

| Vulnerability Severity | CWE ID |
|---|---|
| Low | 200 |
| **CVE ID** | CVSS Score |
| N/A | 3.7 |

## Vulnerability Description

The Server header discloses the server version details that handled the request.
The server version details are as follows :
Server: Werkzeug/2.2.3 Python/3.8.9

## Vulnerable API

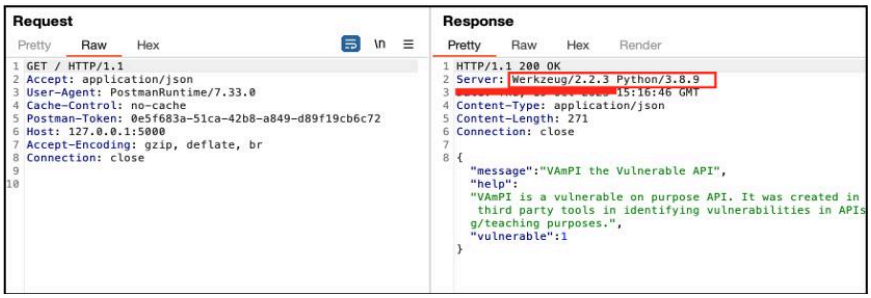
All API Endpoints (Base URL)

## Impact

Using the information in this header, attackers can find vulnerabilities easily. An attacker might use the disclosed information to harvest specific security vulnerabilities for the version identified.

## Step to Reproduce (Evidences)

1. Intercept the API request and observe the server version being disclosed in the response.




## Remediation

- Disable Server Header in the Response.

## References

- https://www.thesmartscanner.com/vulnerability-list/server-version-disclosure

KLEAP

CYBERSECURITY

https://kleapcybersecurity.com/

info@kleapcybersecurity.com

4111, Briargrove Circle, Raleigh,
North Carolina, 27607, USA