SPECIALIZATION TASK: -20

BROKEN AUTHENTICATION

Intern Name: Karishma Sahni

Batch: 17CF

What is Authentication?

Authentication is the process of verifying the identity of a user, device, or system in order to grant access to resources or data, ensuring only authorized individuals or entities can proceed.

Core Concepts:

Authentication operates by proving that the claimant is truly who they claim to be, using credentials such as passwords, security tokens, or biometrics (like fingerprints or facial scans). It is a foundational element in cybersecurity, protecting sensitive information and systems from unauthorized access.

Steps in Authentication:

- Identification: The user provides a specific identifier, such as a username.
- Authentication: The system verifies the claim using something the user knows (password), something they have (physical token), or something they are (biometric).
- Authorization: If authentication is successful, the user is granted permission to access the requested resources.

Types of Authentications:

- Single-factor authentication (SFA): e.g., password-based logins.
- Two-factor authentication (2FA): combines two types of credentials, such as a password and a mobile-generated code.
- Multifactor authentication (MFA): involves more than two methods, including biometrics or smart cards for enhanced security.

Importance in Security:

Effective authentication prevents data breaches, malware installation, and unauthorized system access, keeping organizational and personal data secure.

Weak authentication mechanisms can expose systems to attacks, making robust practices essential.

What is broken authentication and its types?

Broken authentication refers to vulnerabilities in an application's authentication and session management processes that allow attackers to bypass controls and impersonate legitimate users. This security flaw occurs when authentication mechanisms are implemented improperly, enabling attackers to compromise credentials, session tokens, or account details and gain unauthorized access to user accounts or privileges. Poorly managed passwords, insecure session IDs, or insufficient protection against brute-force attacks are common causes.

Types of Broken Authentication:

- Credential Stuffing: Attackers use automated tools to attempt logins with stolen usernames and passwords from previous breaches.
- Brute Force Attacks: Rapidly testing many password combinations until the correct one is found, exploiting weak password policies.
- Session Fixation/Hijacking: Exploiting improperly managed session IDs, where attackers steal or predict valid session tokens to impersonate a user.
- Poor Password Management: Weak, default, or easily guessable passwords make accounts susceptible to takeover.
- Missing Account Lockout: Lack of lockout after several failed login attempts invites repeated brute-force attempts.
- Unencrypted Credentials: Storing or transmitting authentication data in plain text exposes it to theft during network interception.
- Implementation Logic Flaws: Bugs or errors in the authentication flow that allow keys or tokens to be predicted, manipulated, or bypassed entirely.

Broken authentication is ranked among the top web application and API security risks, as exploitation can result in complete compromise of user accounts and sometimes full system control. Effective session management and robust credential handling are critical defences.

Broken authentication case studies reveal how weak authentication or mismanagement of tokens and credentials led to real-world breaches, account takeovers, or API abuse.

Example 1: Weak Password Policies and Credential Stuffing

Revolut, a global financial app, suffered a breach that exposed details of around 50,000 users. Attackers used social engineering to obtain employee credentials, leading to data leaks that could be used for identity theft or phishing. The lack of strong authentication measures, such as multi-factor authentication (MFA), made it easier for attackers to compromise legitimate accounts.

Example 2: Session Timeouts and Improper Logout

Session timeout misconfiguration can let attackers reuse authentication sessions. For example, if a user logs into a web mail service at a public terminal and the session persists, a subsequent user could access the previous user's account, leading to unauthorized data exposure. This scenario highlights the importance of correctly implementing session expiration and logout processes.

Example 3: JWT Misconfiguration in APIs

An API demo application allowed authentication bypass by simply modifying the payload of a JSON Web Token (JWT) without verifying its digital signature. Attackers could impersonate other users by changing fields in the JWT and submitting it to the API, demonstrating the critical risk of improper validation and weak token integrity controls.

Example 4: OAuth Access Token Leakage

In some web applications, OAuth access tokens were exposed via insecure redirect URIs or URLs. Attackers intercepted these tokens and used them to access sensitive user data and resources without needing valid login credentials. This vulnerability occurred due to failing to validate redirect URIs and exposing tokens in insecure ways.

Example 5: Credential Stuffing Across Services

Attackers breach a CRM database and reuse stolen usernames and passwords to access accounts on unrelated systems like a bank database, leveraging widespread password reuse for privilege escalation and unauthorized financial access.

Key Lessons:

Robust authentication involves strong password management, secure token handling, MFA, and rigorous session management. Real-world failures often exploit gaps in these controls, emphasizing the need for comprehensive, multi-layered defences against authentication-related threats.

Impact of broken authentication

Broken authentication vulnerabilities can have severe impacts, ranging from individual harm to organizational crises, due to attackers gaining unauthorized access to user accounts or privileged systems.

Key Impacts:

- **Compromised User Data**

Attackers can steal personal and financial information, such as names, addresses, emails, payment details, and even health records. This often leads to identity theft, fraud, and privacy violation for the end user.

- **Account Takeover**

Unauthorized access enables attackers to hijack user accounts, change settings, make fraudulent transactions, and launch further attacks against other users or internal resources. High-value targets include accounts with administrative or elevated privileges, which can lead to broader system compromise.

- **System Compromise**

Gaining control over privileged user accounts can allow attackers to manipulate, delete, or steal sensitive data and introduce malicious code or backdoors. Such compromise can disrupt business services or even result in complete control over the organization's infrastructure.

- **Reputational and Financial Damage**

Customers lose trust in the compromised business, contributing to long-term damage to its reputation. Organizations often incur immediate financial losses, legal fees, regulatory penalties (e.g., GDPR fines), and costs for forensic investigation and remediation.

- **Regulatory and Legal Penalties**

Breaches resulting from broken authentication can trigger penalties due to non-compliance with data protection laws (GDPR, CCPA, PCI-DSS, HIPAA).

Real-World Examples

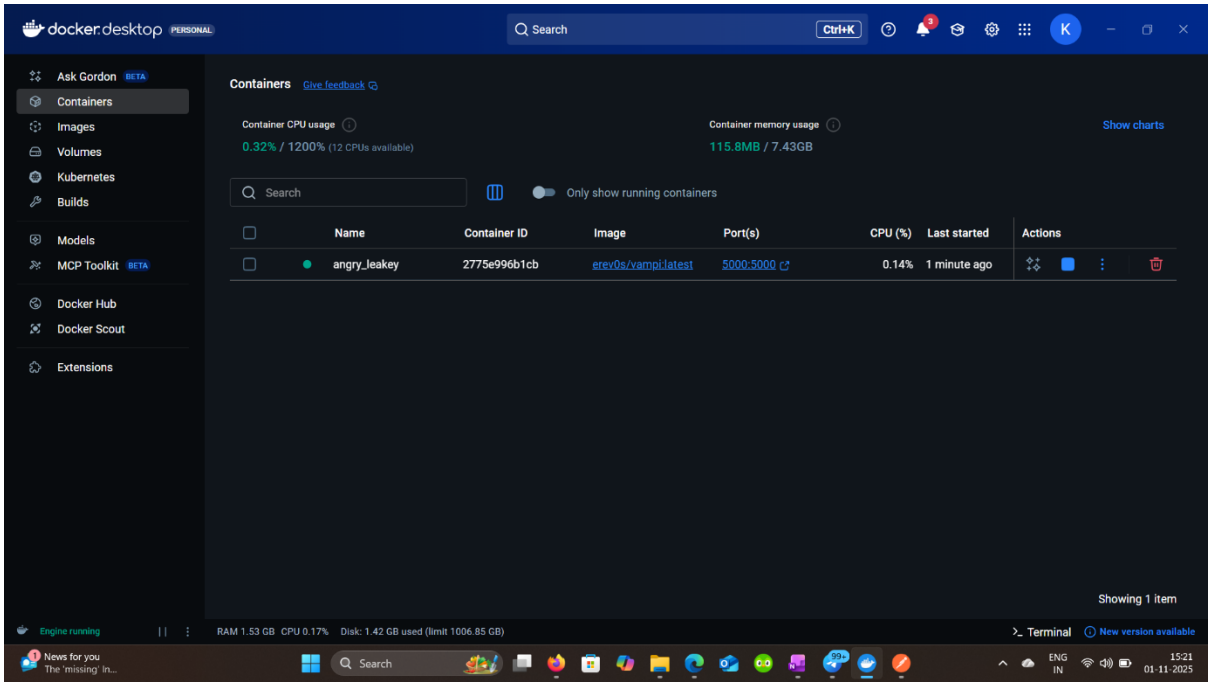
- In the Dunkin' Donuts credential stuffing incident, attackers exploited weak authentication to access loyalty accounts and redeem rewards, causing data and financial loss.
- WordPress session hijacking attacks abused session cookies and plugin flaws to escalate privileges and control websites.

- Major breaches often involve attackers leveraging broken authentication to pivot inside networks, launch ransomware, or steal intellectual property.

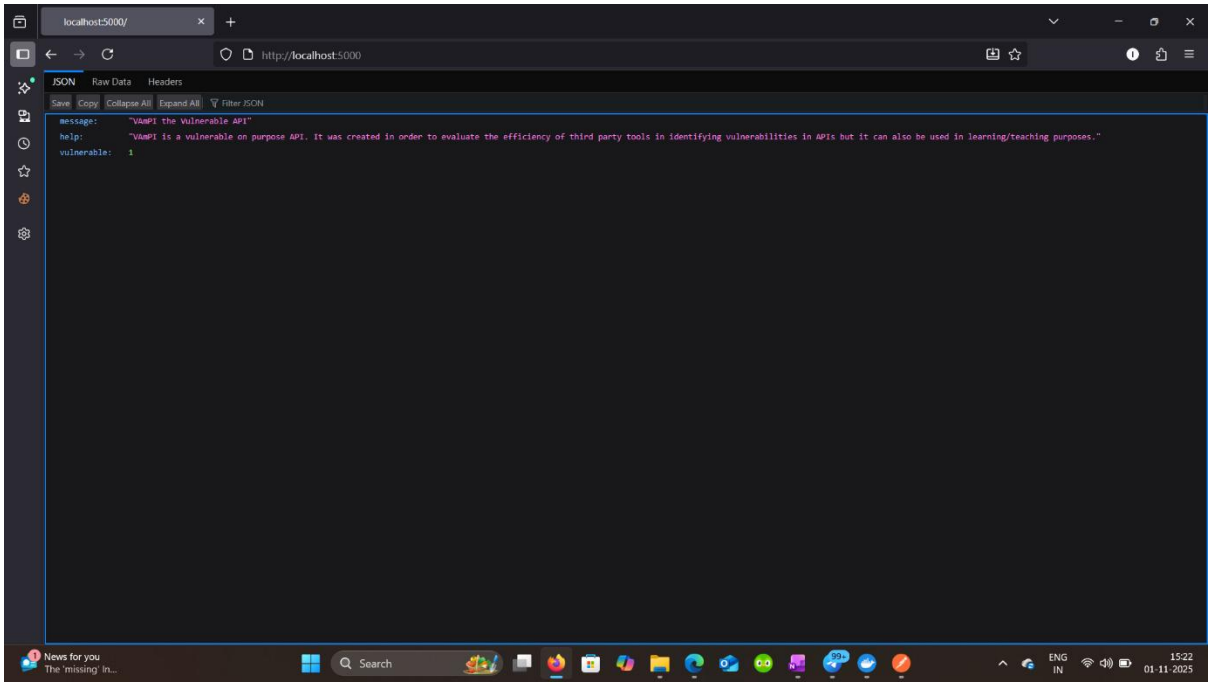
Broken authentication is one of the most critical threats facing web applications and APIs, affecting users, businesses, and infrastructure at multiple levels.

Testing VAmPI for Broken Authentication (Steps):

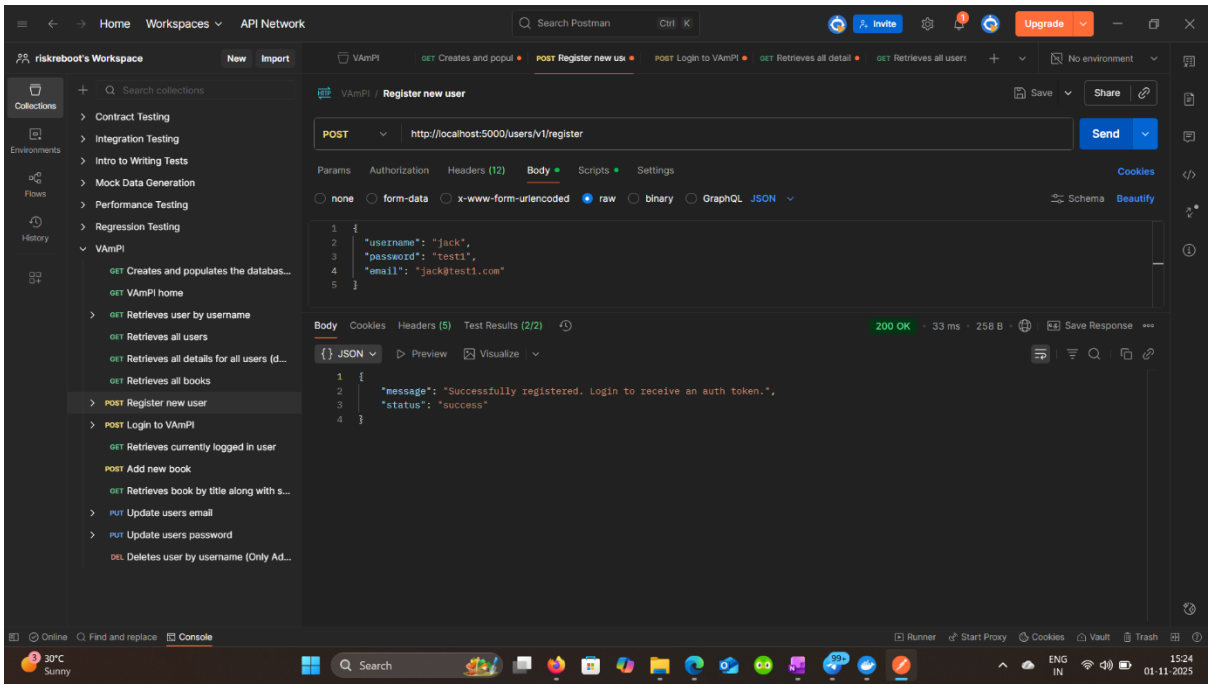
Step 1: Run VAmPI in docker desktop



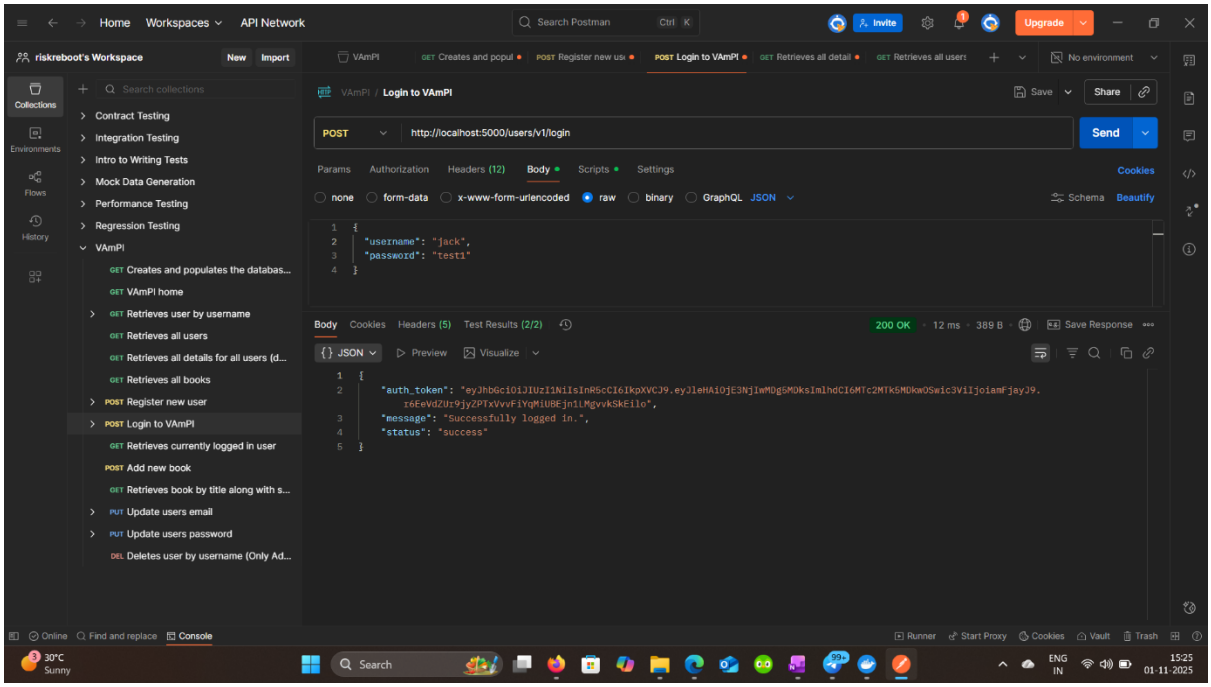
Step 2: Check VAmPI running in your browser at http://localhost:5000



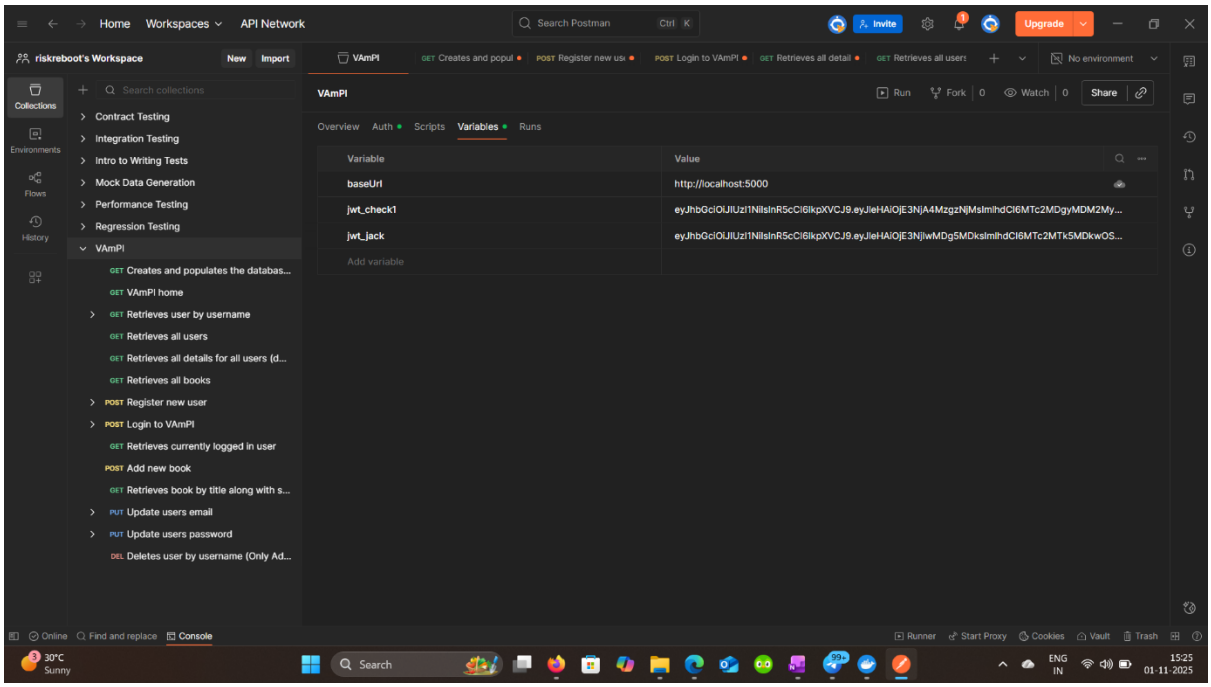
Step 3: In Postman, in your VAmPI collection use the POST Register new user request to create new user “Jack” with password “test1”



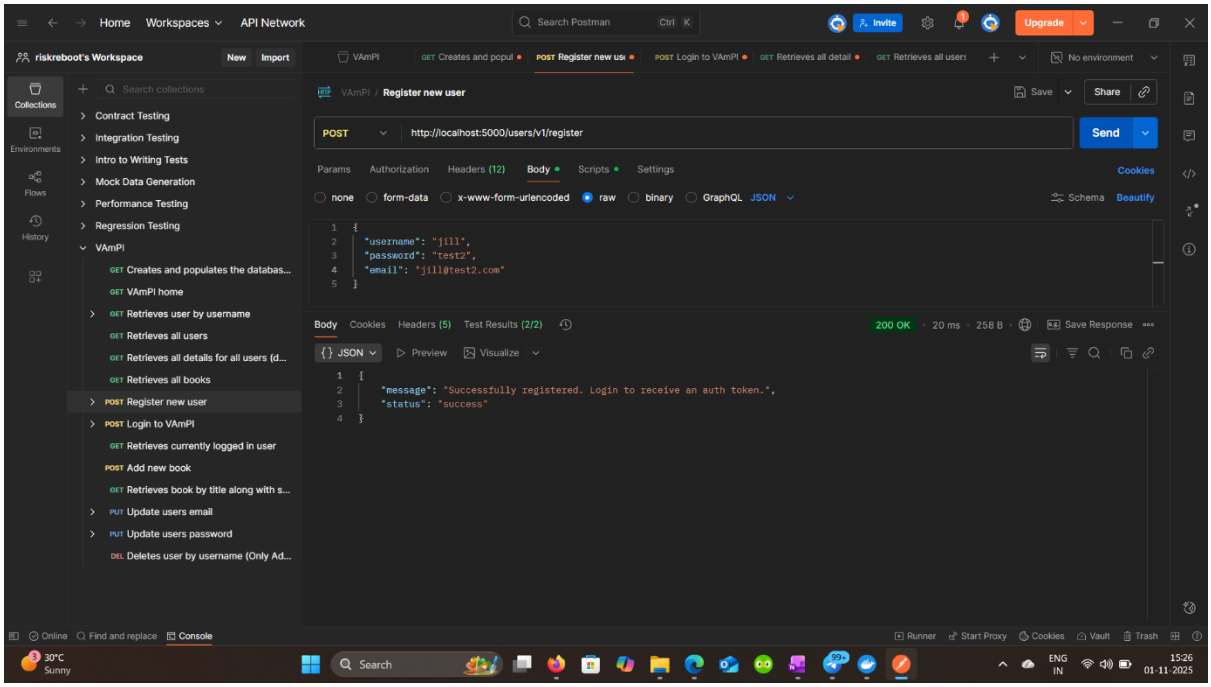
Step 4: Using the POST Login to VAmPI request, login as Jack, you will see a jwt token, copy it



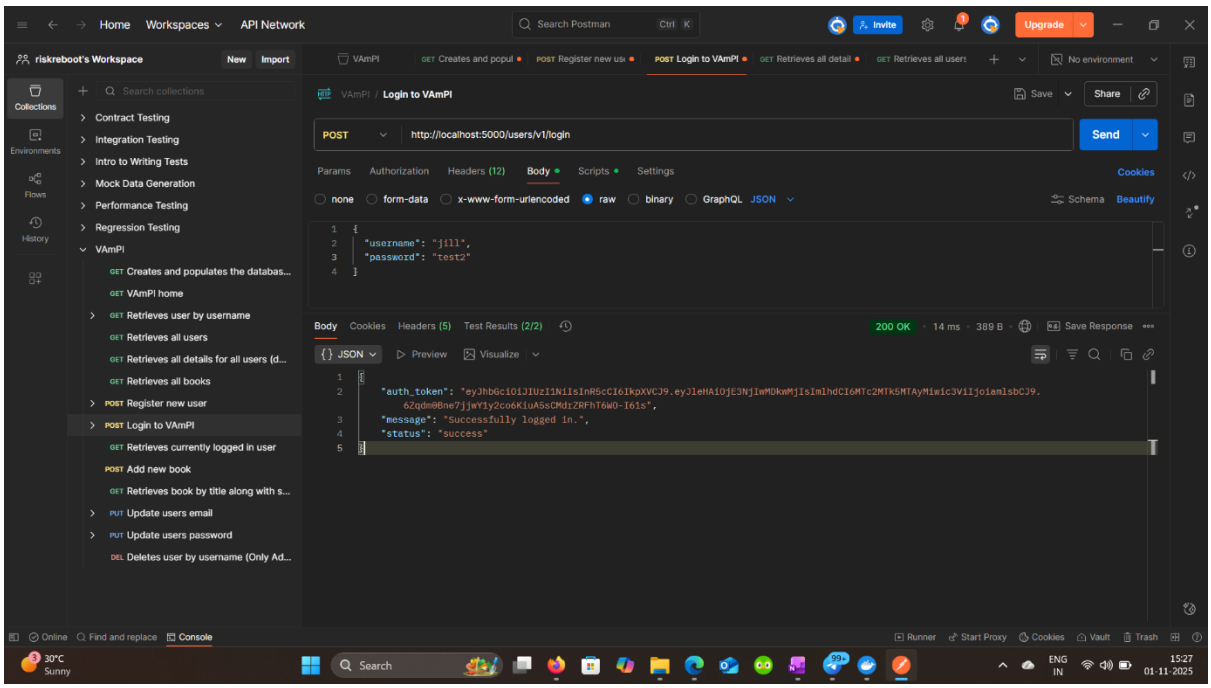
Step 5: In VAmPI > Variables > add a new variable jwt_jack and paste token



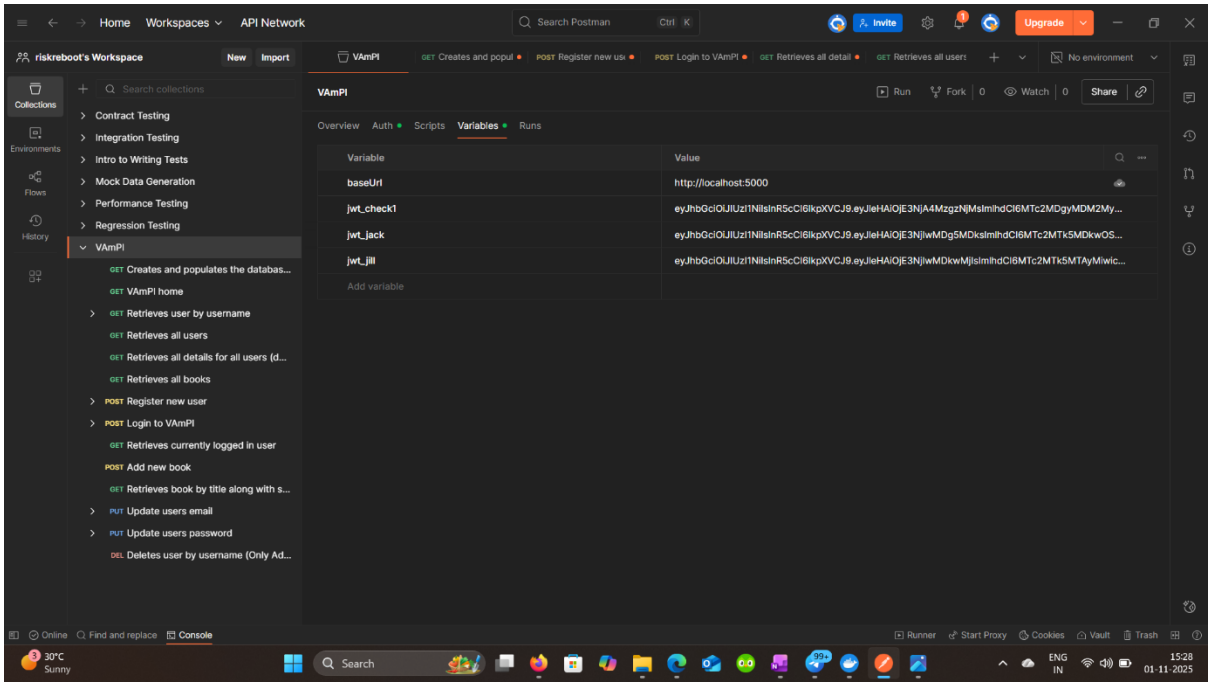
Step 6: Using POST Register new user request, create another user “Jill” with password “test2”



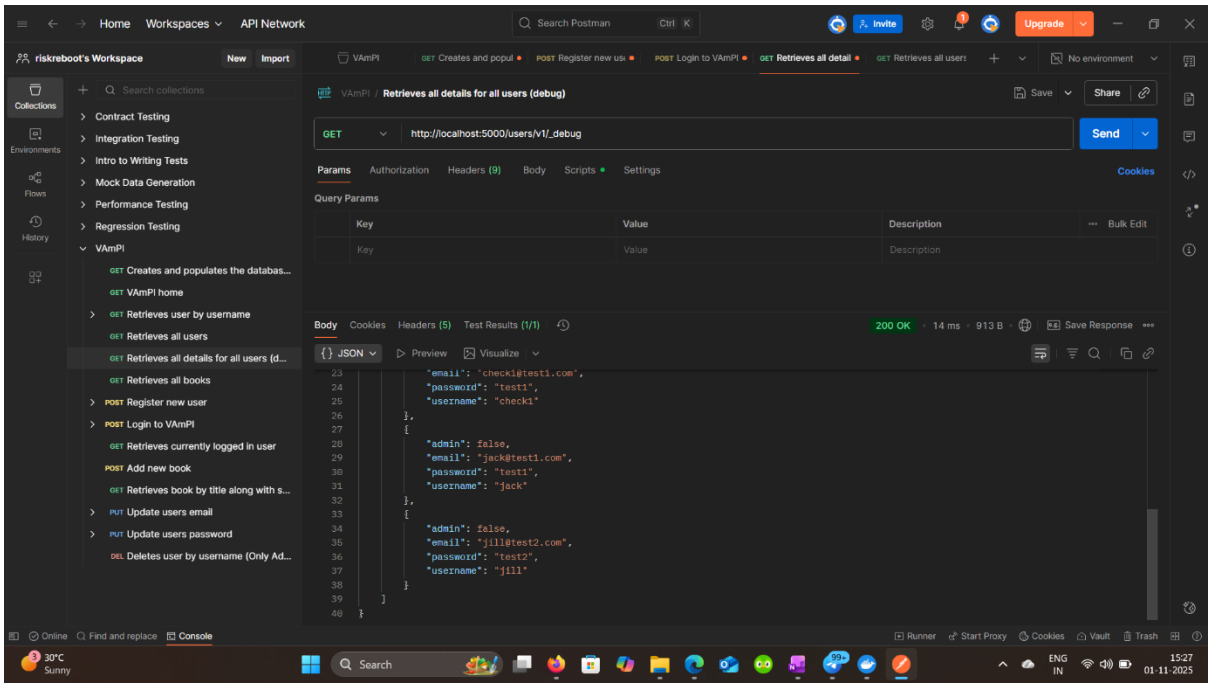
Step 7: Using the POST Login to VAmPI request, login as Jill, you will see a jwt token, copy it



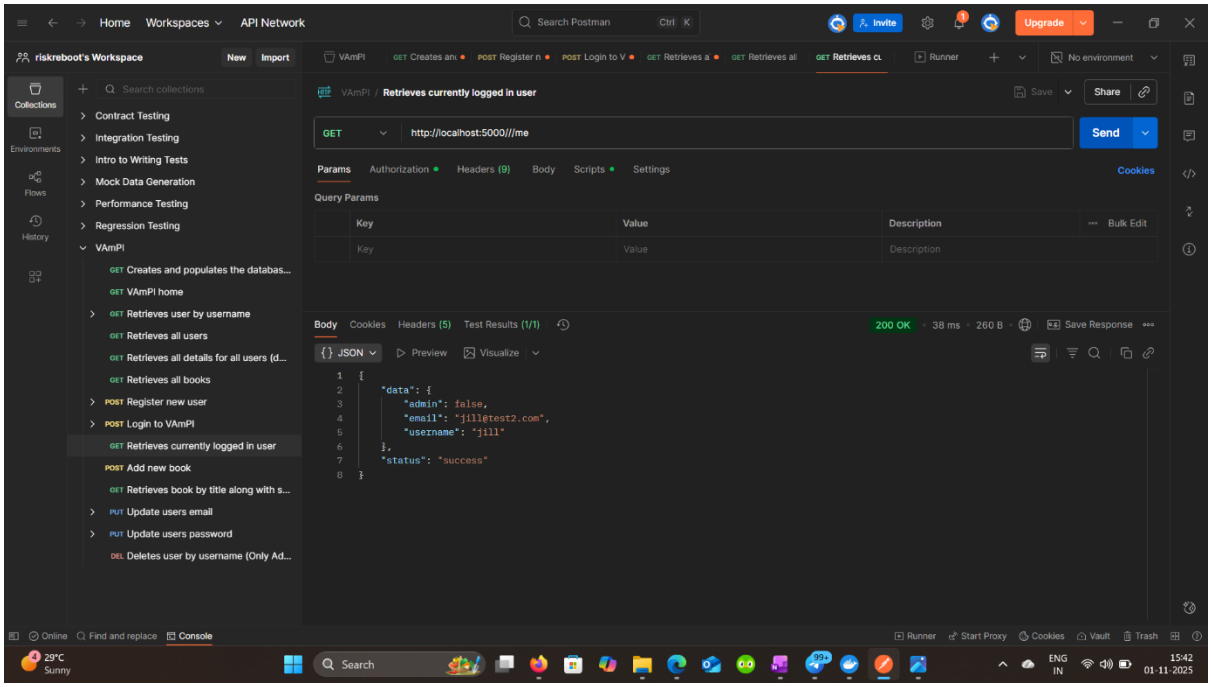
Step 8: In VAmPI > Variables > add a new variable jwt_jill and paste token



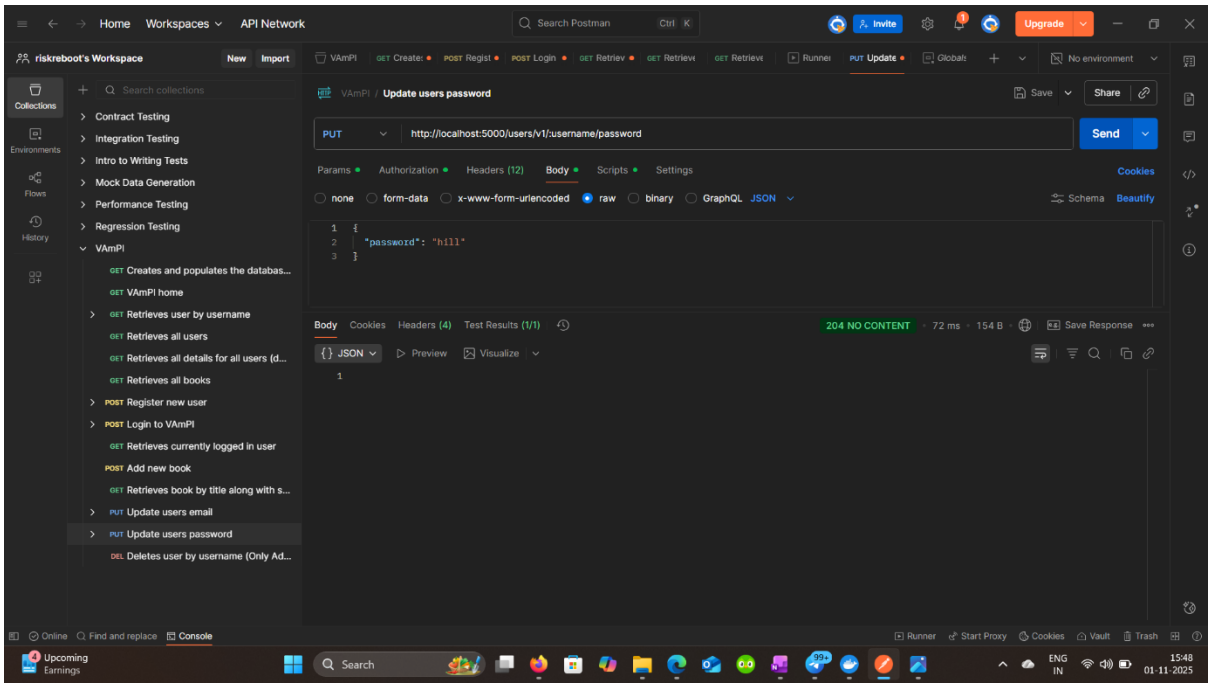
Step 9: Send GET Retrieve all details for all users request to verify credentials of both users



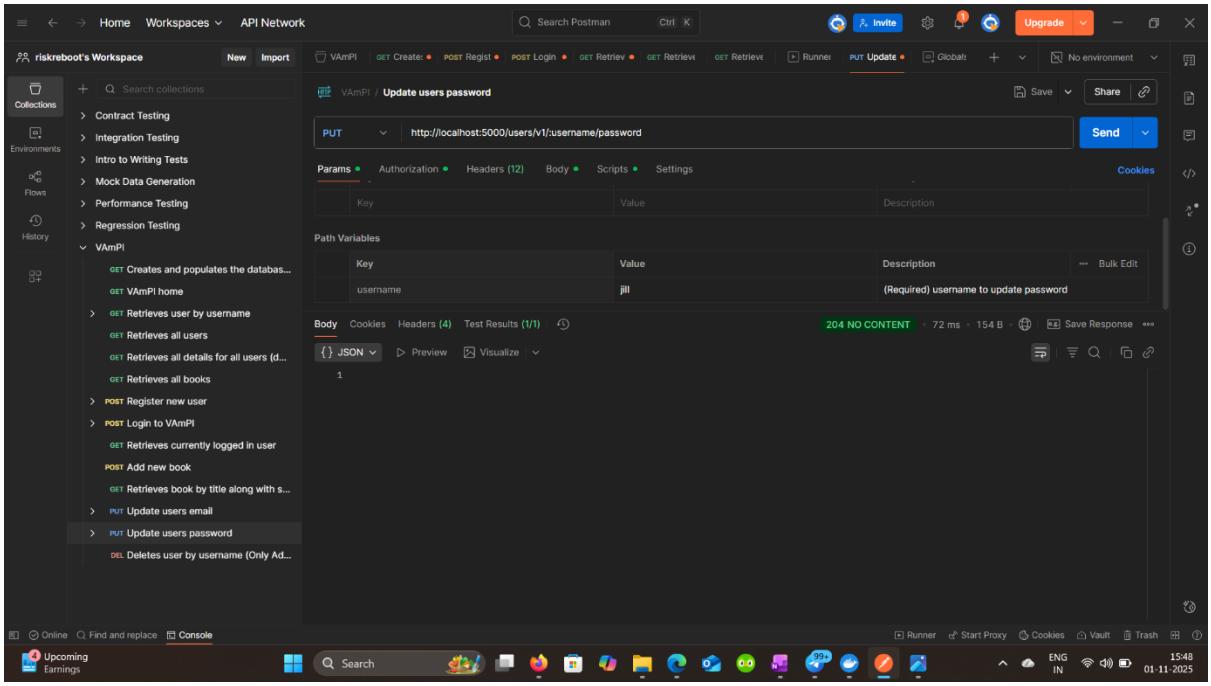
Step 10: If you send GET Retrieves currently logged in user, you will see, currently user Jill is logged in



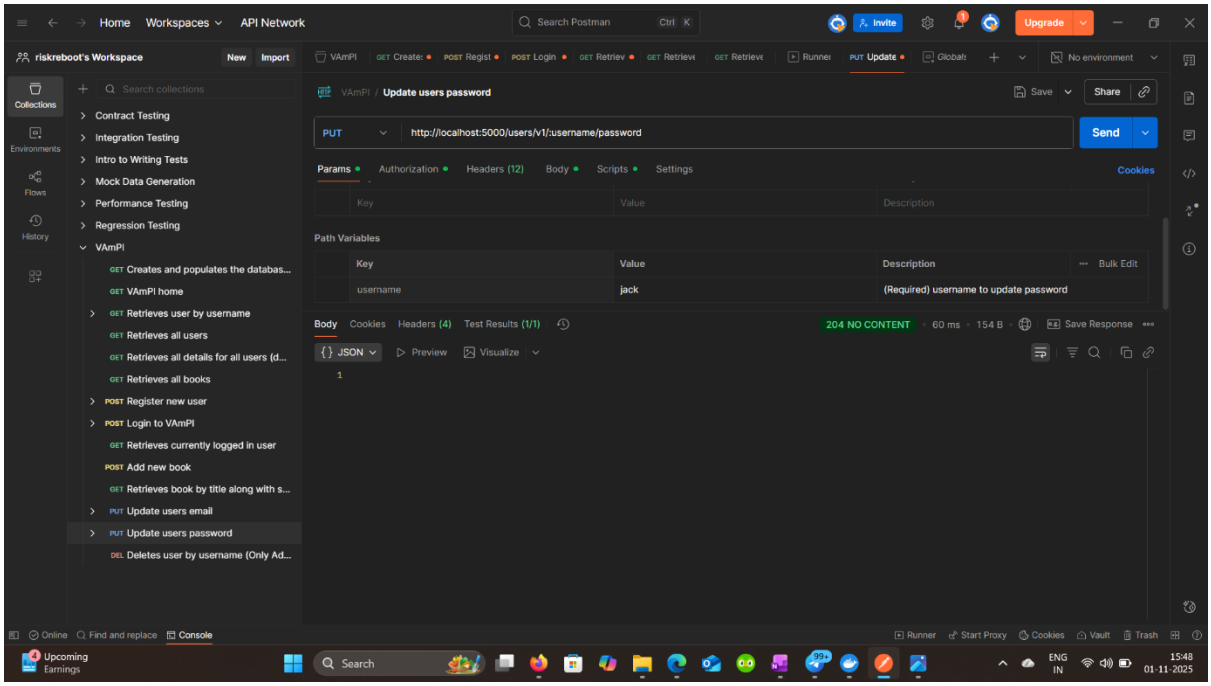
Step 11: In PUT Update users password request > Body > update new password



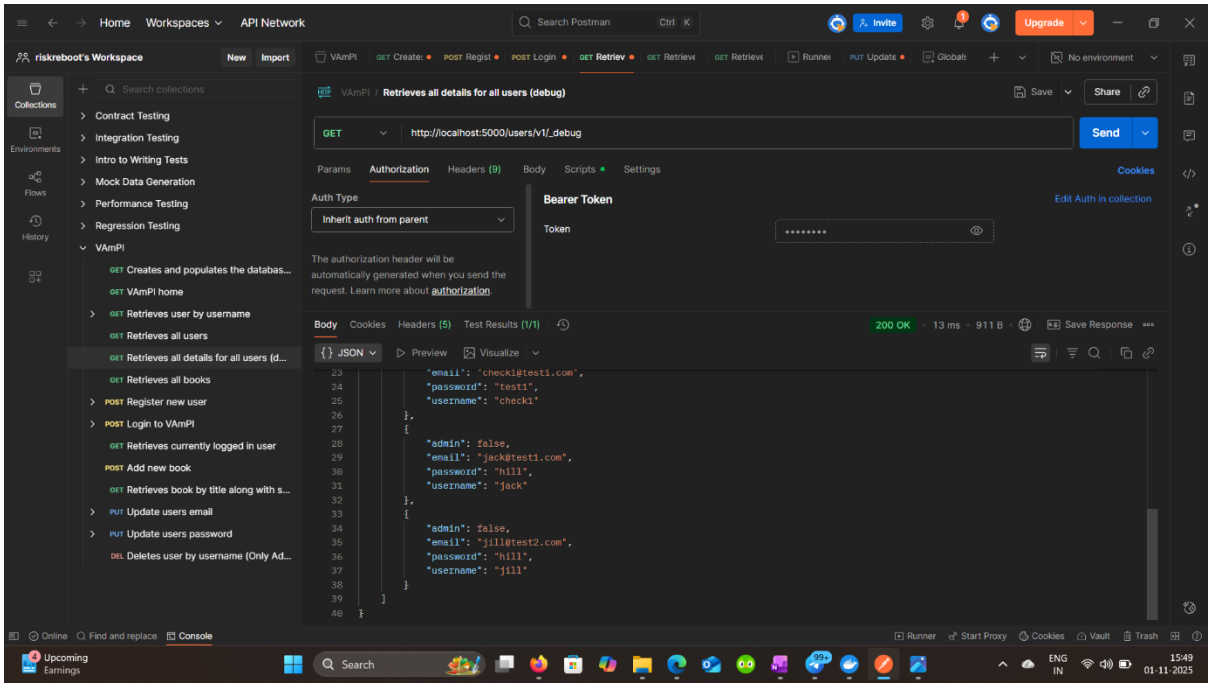
Step 12: In PUT Update users password request > Params > username value > enter Jill and send request you will get 204, means you have updated Jill's password while logged in as Jill



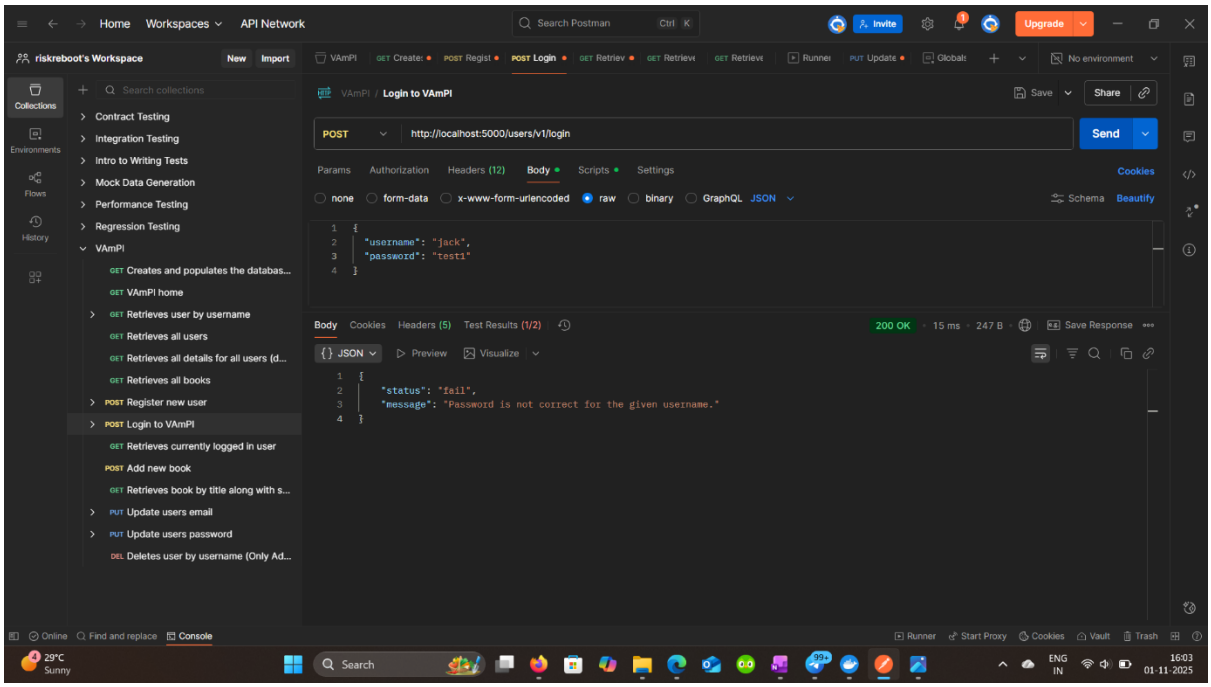
Step 13: In PUT Update users password request > Params > username value > enter Jack and send request you will get 204, means you have updated Jack's password while logged in as Jill, which should not have been possible, hence it is a vulnerability

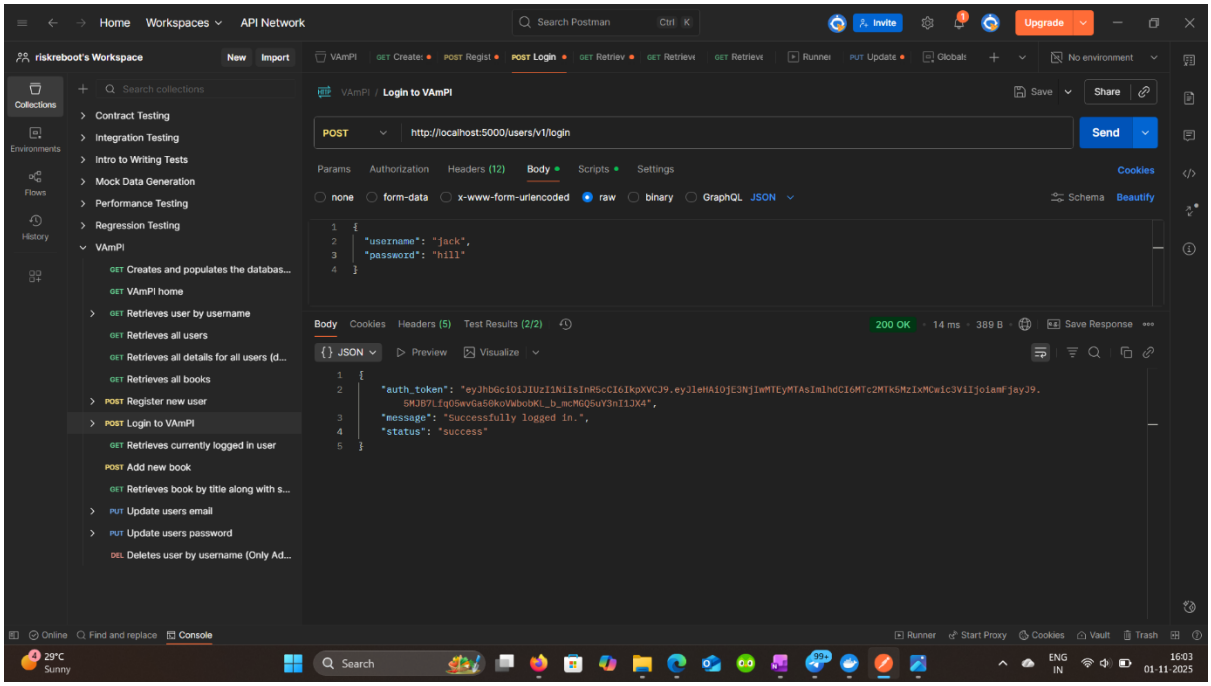


Step 14: Send GET Retrieves all details for all users request to verify password change



Step 15: If you send POST Login request as Jack with old password “test1” it will fail but with password “hill” it will succeed. User Jill was able to change User Jack’s password

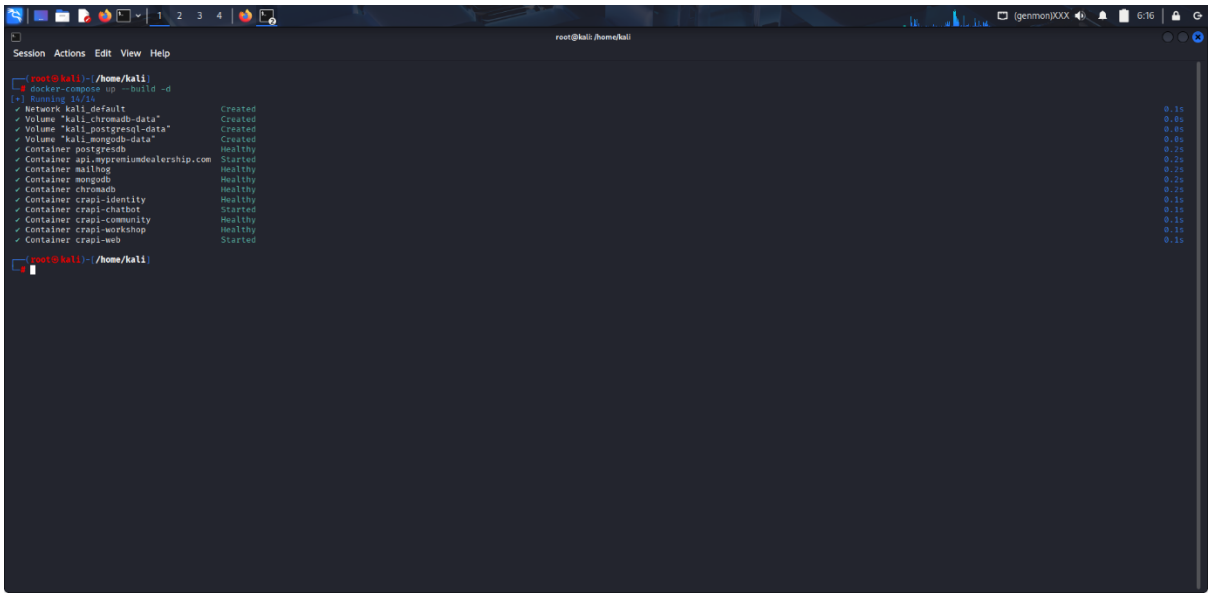




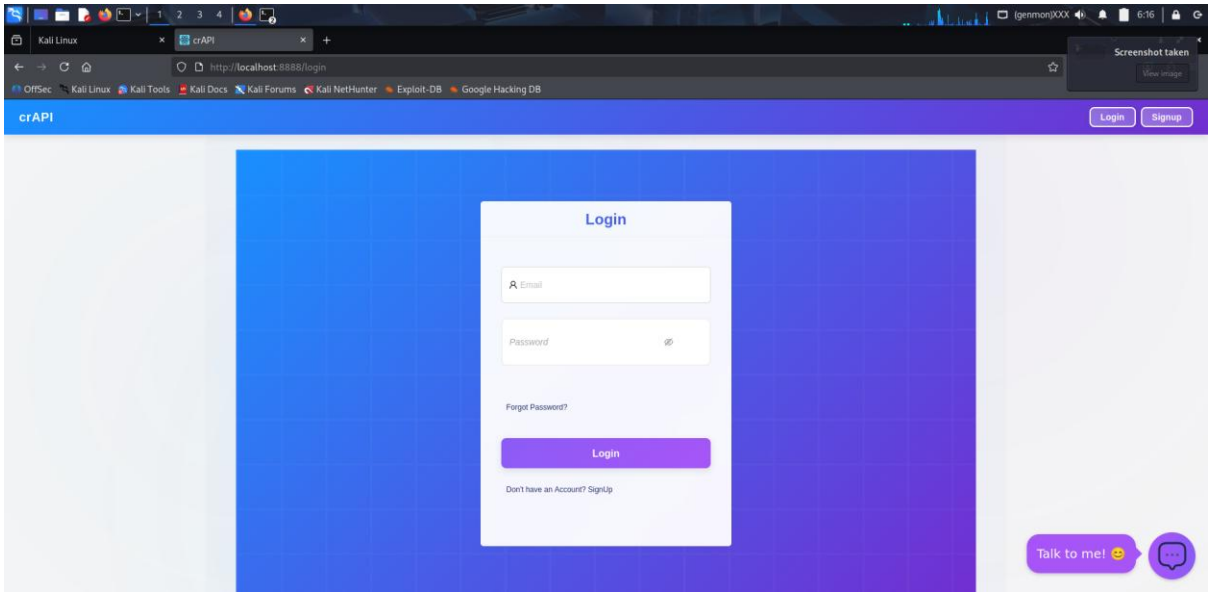
Being able to change other users email or password is Broken authentication vulnerability.

Now we will test crAPI for BOLA with a lab setup on Kali on virtual box and testing it on burpsuite on our windows

Step 1: Run crAPI through docker in Kali

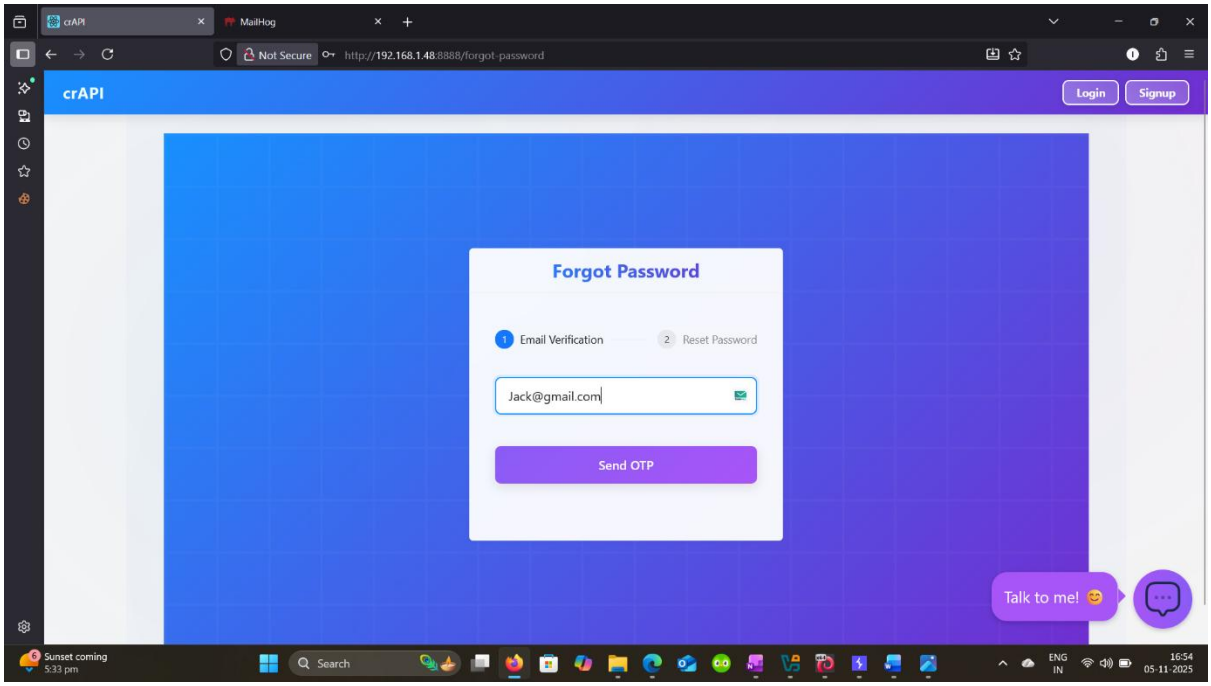


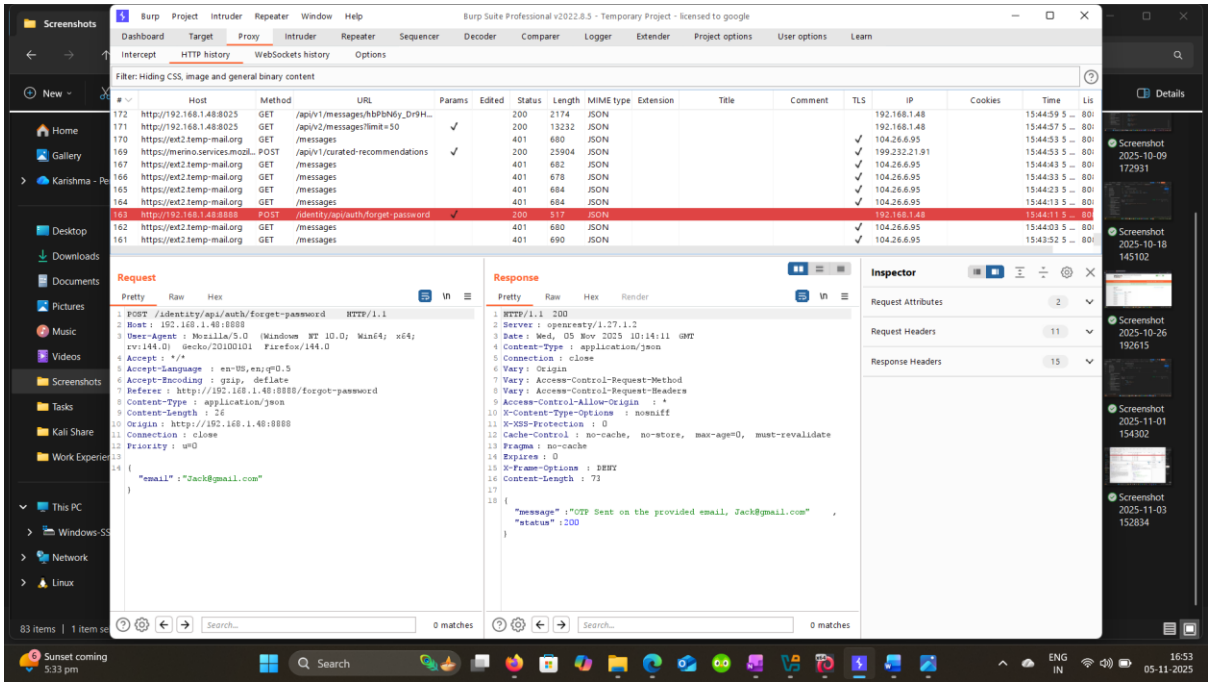
Step 2: Test if crAPI is working on our browser on kali at http://localhost:8888/login



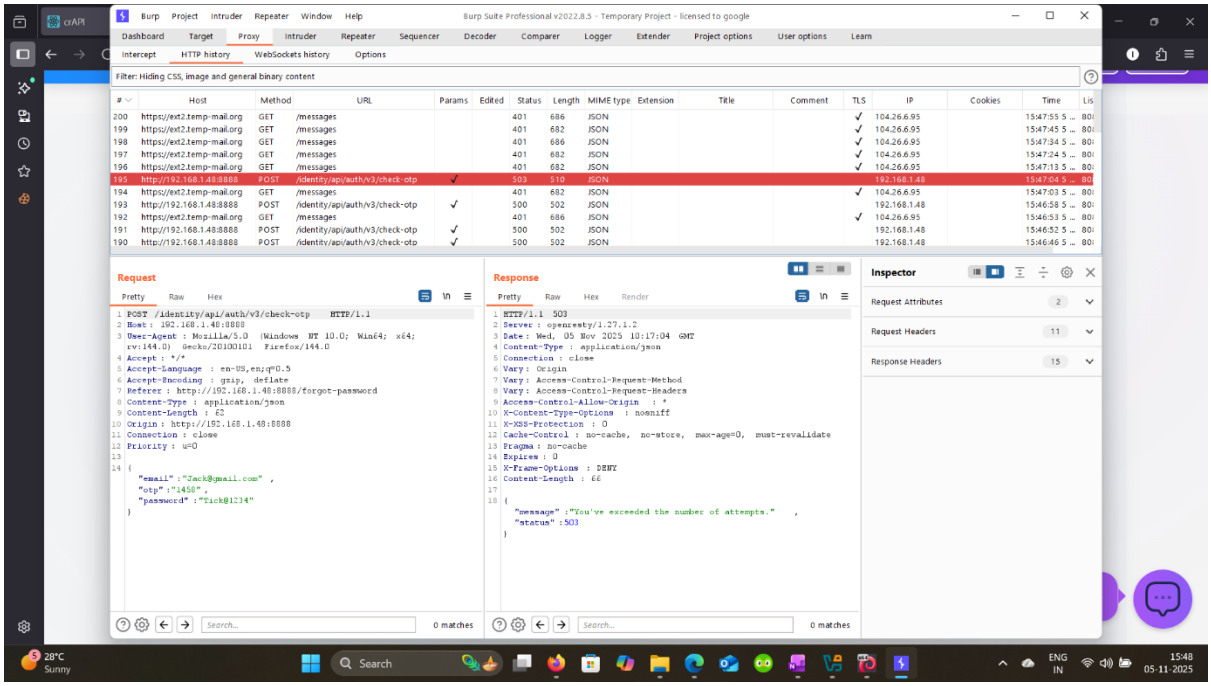
Step 3: In our Kali terminal, type ifconfig to find the IP on windows, in our case it is 192.168.1.48 and access crAPI on our browser in windows at http://192.168.1.48:8888 and mailhog at http://192.168.1.48:8025

Step 4: Turn our proxy on to Burp and load our burpsuite, signup as a new user and in login page, click on forget password, input your email > send OTP, catch this request in burp and highlight it for further use

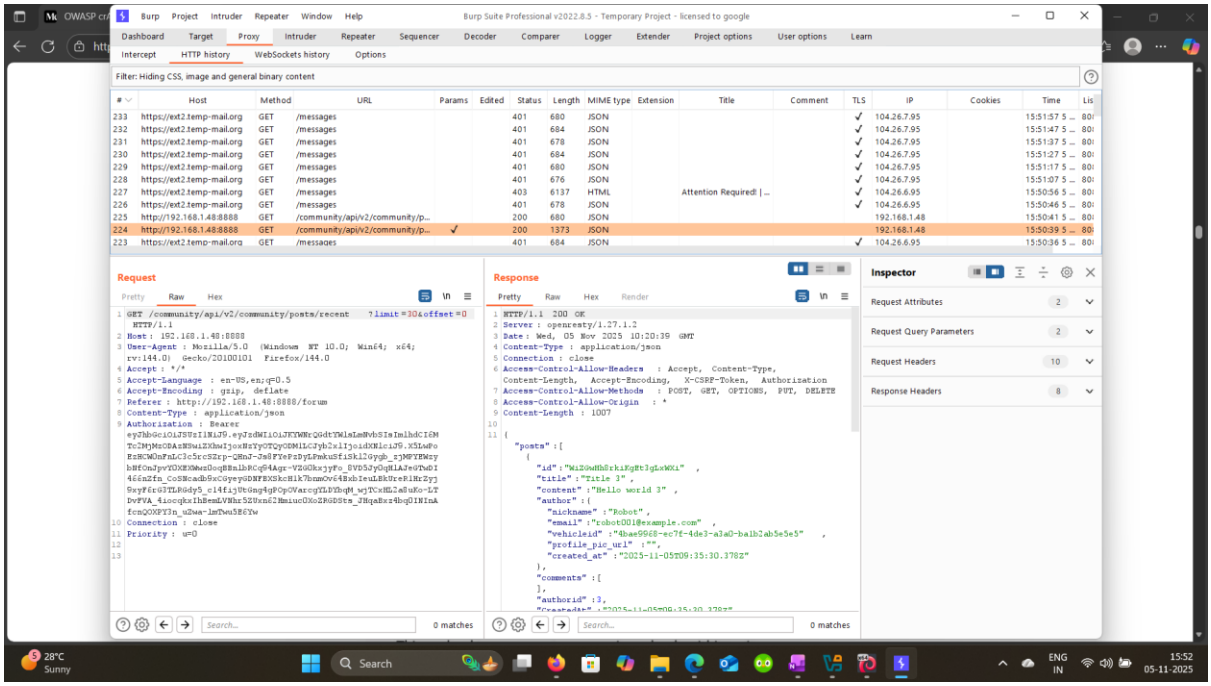




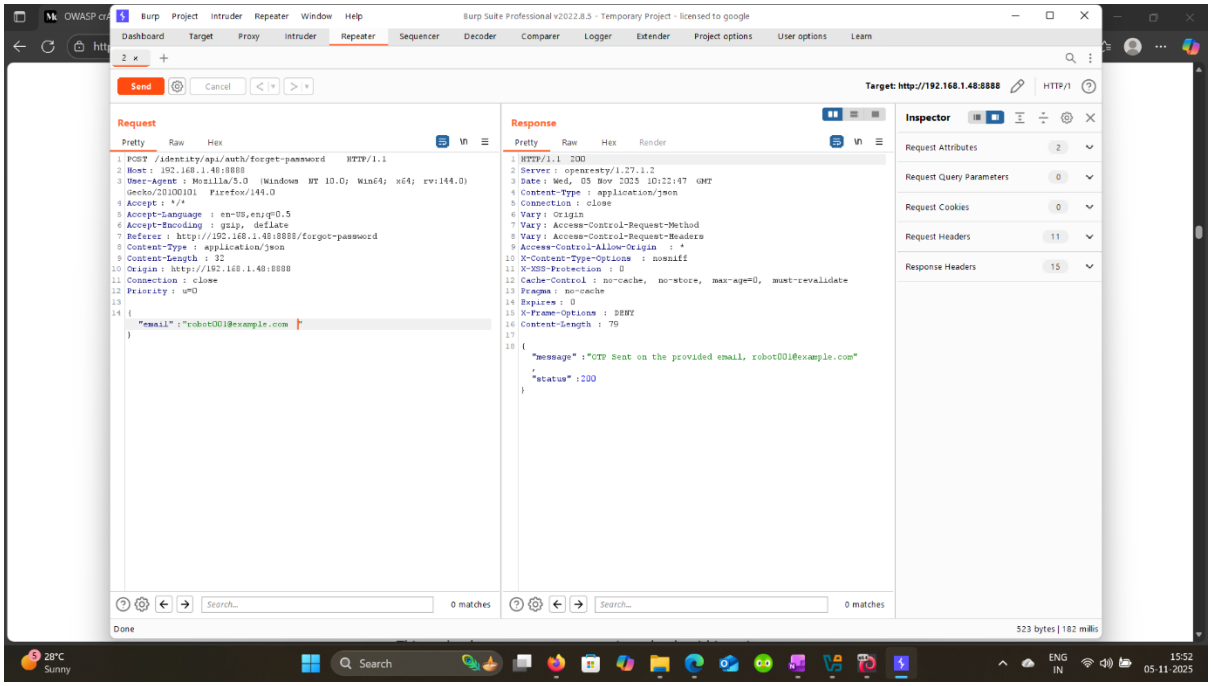
Step 5: Even though we have access to email and hence the OTP, we will input wrong OTP multiple times to check rate limiting, we notice after a few attempts we reached attempt limit and this is version 3, highlight this request too



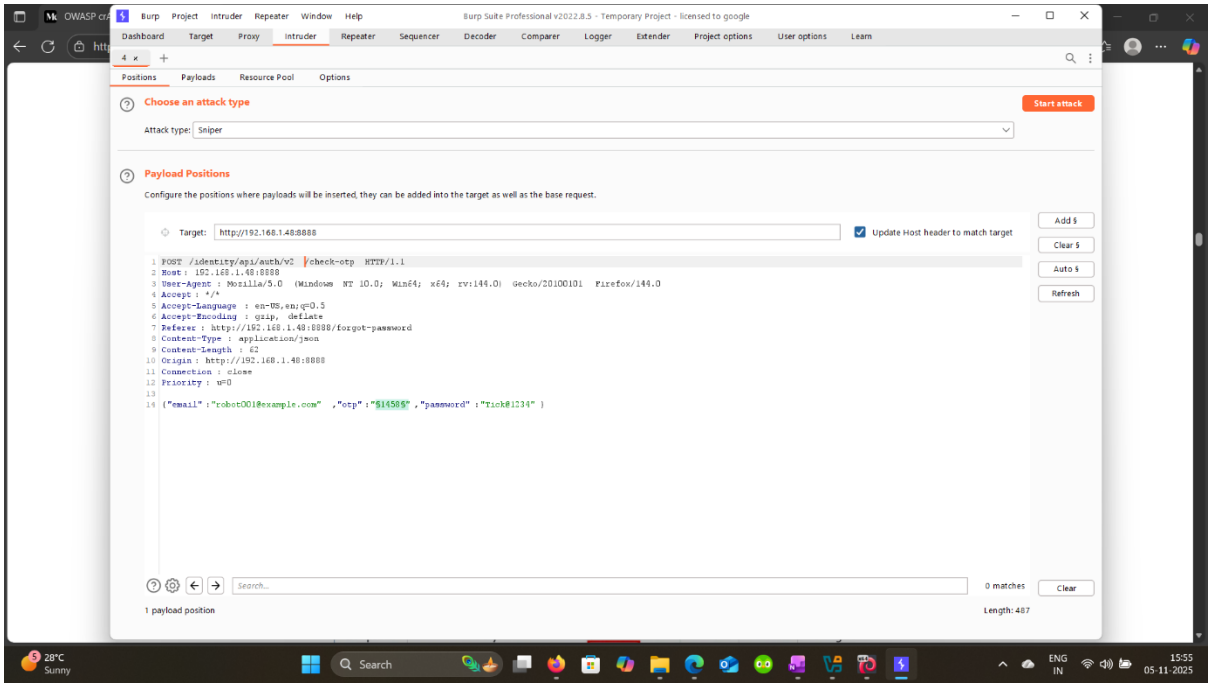
Step 6: We login as ourselves and go to community tab in crAPI and notice that the GET request of fetching comments reveals emails of other users, copy this email



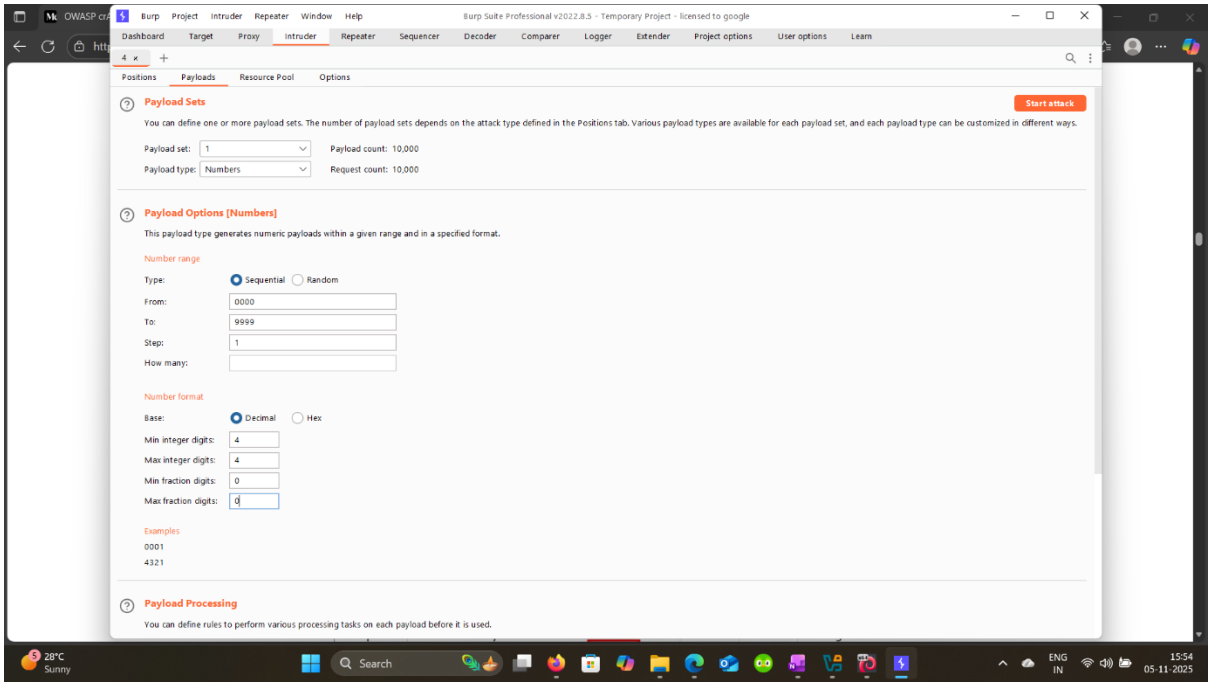
Step 7: Send the POST request for forgot password highlighted in Step 4 and send it to repeater, replace original user email with the one we copied in step 6 and send request, we get 200OK which means we have generated OTP for this random user



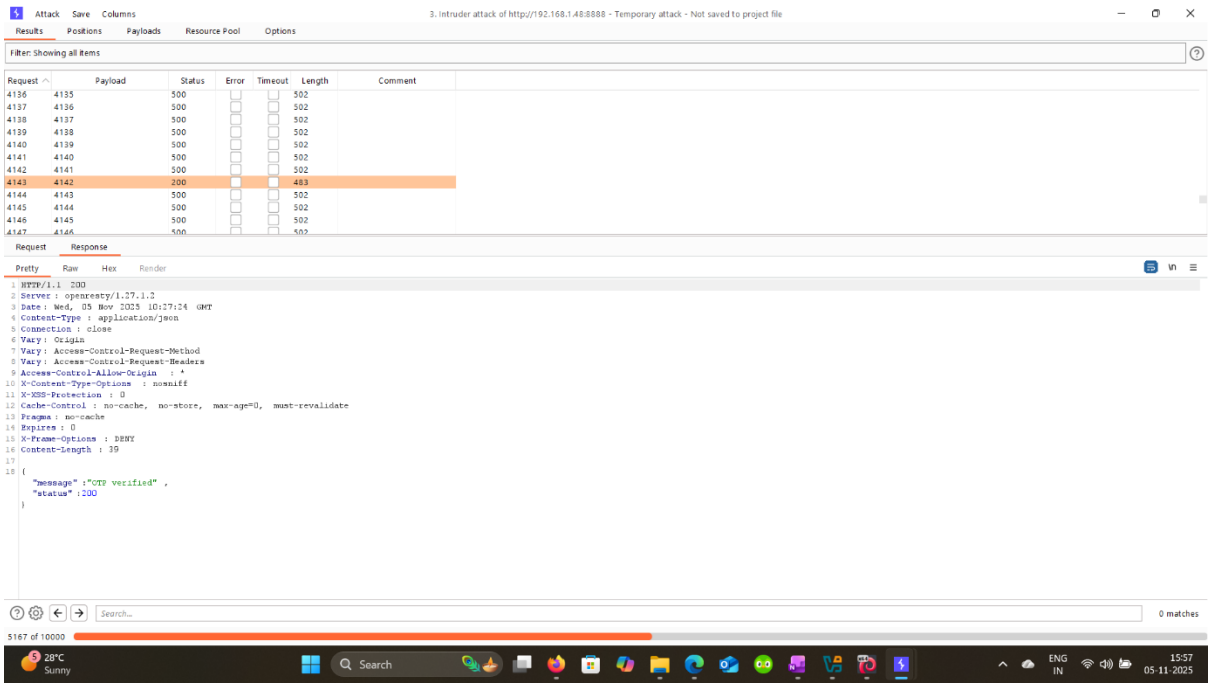
Step 8: Send POST check OTP request highlighted in Step 5 to intruder and clear all payloads except OTP, and change v3 to v2, set attack type to sniper



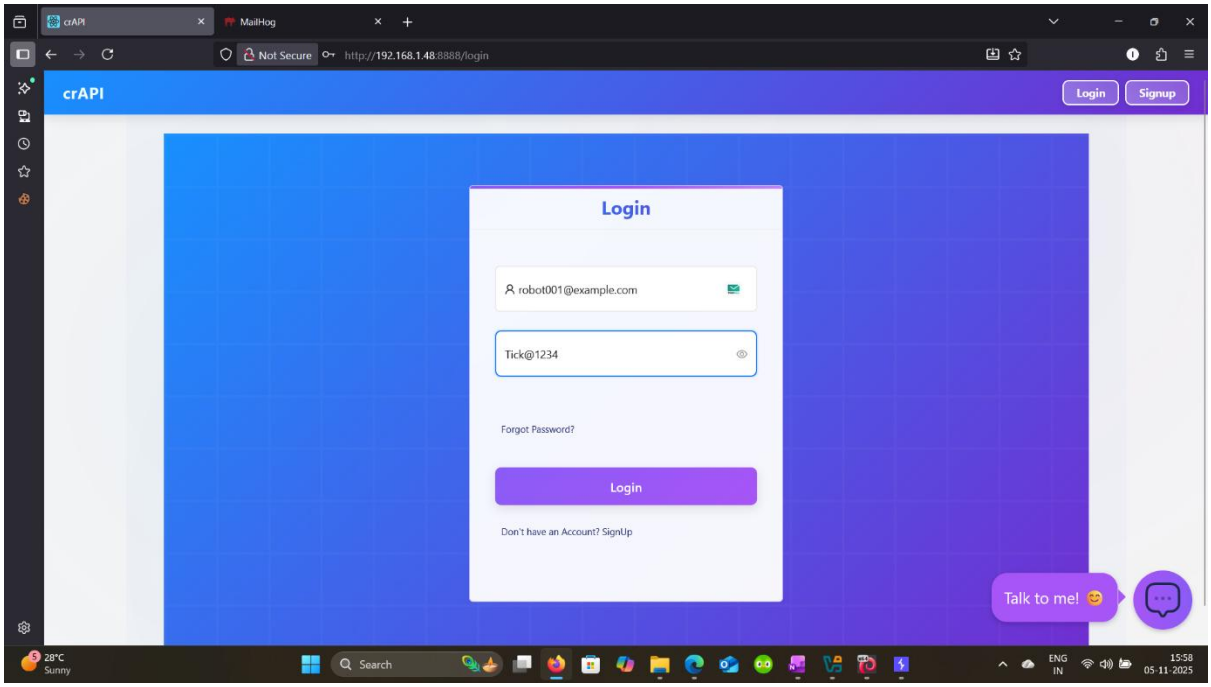
Step 9: In Payloads, set parameters as shown in screenshot and start attack.

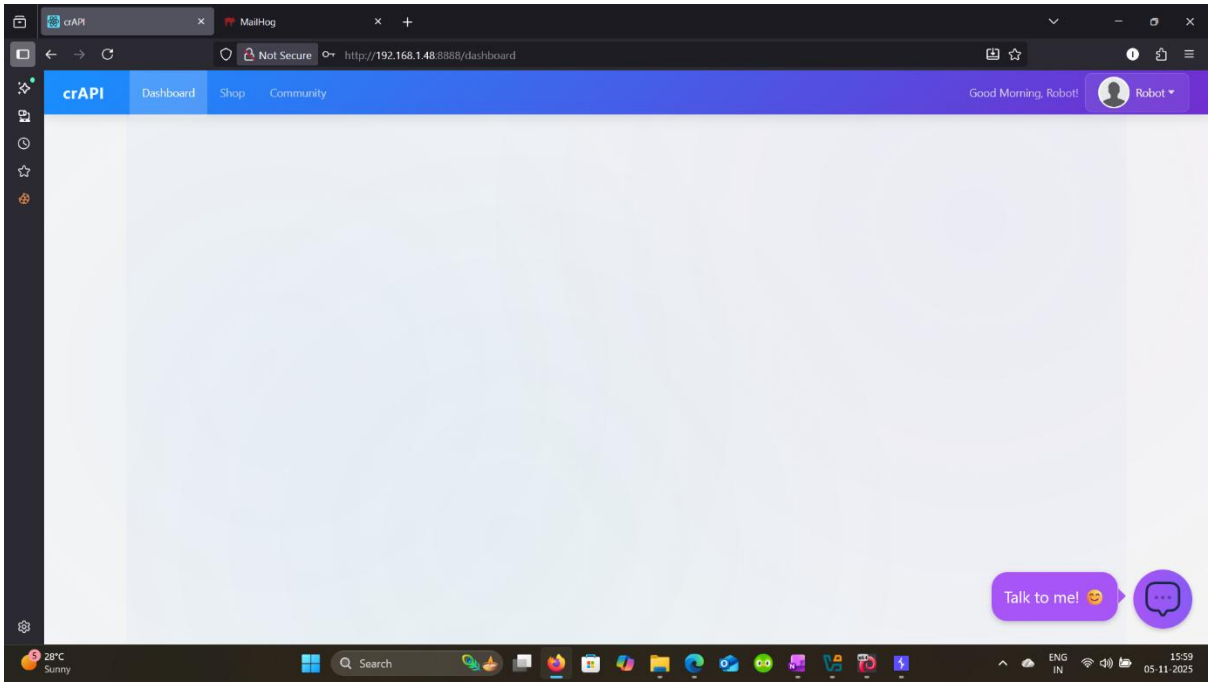


Step 10: One of the requests will show 200 status, OTP verified successfully



Step 11: Now use that random users email and the password you used in the attack to login to this user's account





Being able to change other users email or password is Broken authentication vulnerability.

References:

https://www.techtarget.com/searchsecurity/definition/authentication

https://www.microsoft.com/en-us/security/business/security-101/what-is-authentication

https://www.geeksforgeeks.org/computer-networks/authentication-in-computer-network/

https://portswigger.net/web-security/authentication

https://brightsec.com/blog/broken-authentication-impact-examples-and-how-to-fix-it/

https://www.descope.com/learn/post/broken-authentication

https://www.contrastsecurity.com/glossary/broken-authentication

https://qawerk.com/blog/what-is-broken-authentication/

https://www.loginradius.com/blog/identity/what-is-broken-authentication

https://www.pynt.io/learning-hub/owasp-top-10-guide/broken-authentication-in-apis-and-web-apps-risks-and-mitigations