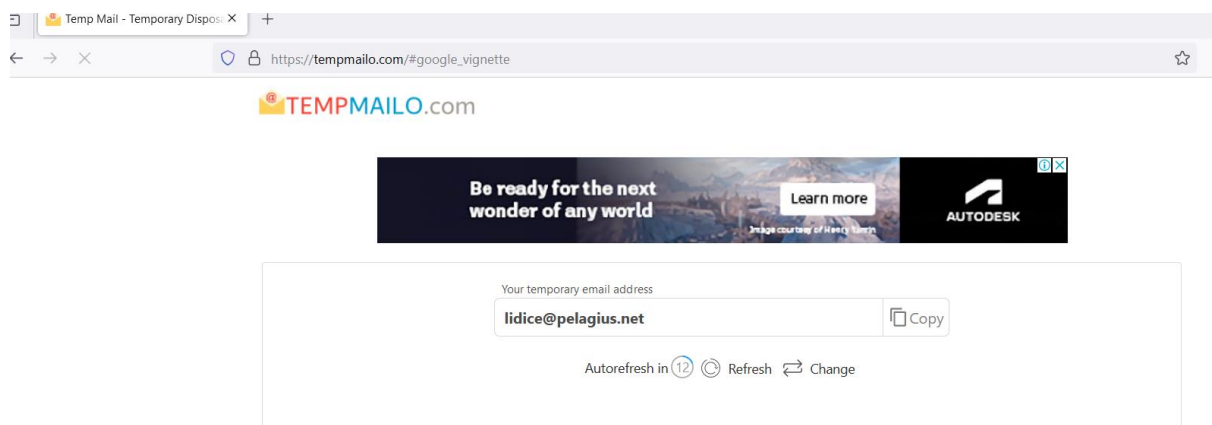


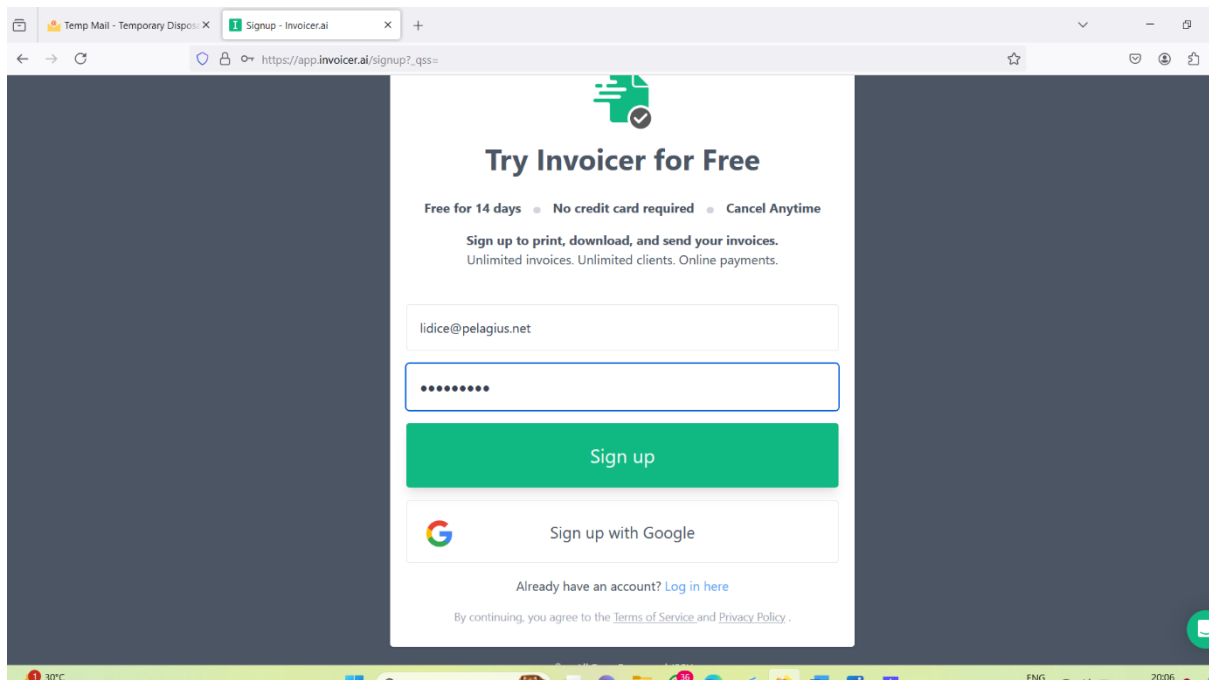
NAME: VAISHNAVI PARULEKAR	ROLL ID: 22203A0048
COURSE:CO (COMPUTER ENGINEERING)	Task Assignment: email rate-limiting feature on the website invoicer.ai
COMPANY: SECURER CYBER FUTURE	COLLEGE: VIDYALANKAR POLYTECHNIC

Email rate-limiting feature on the website invoicer.ai

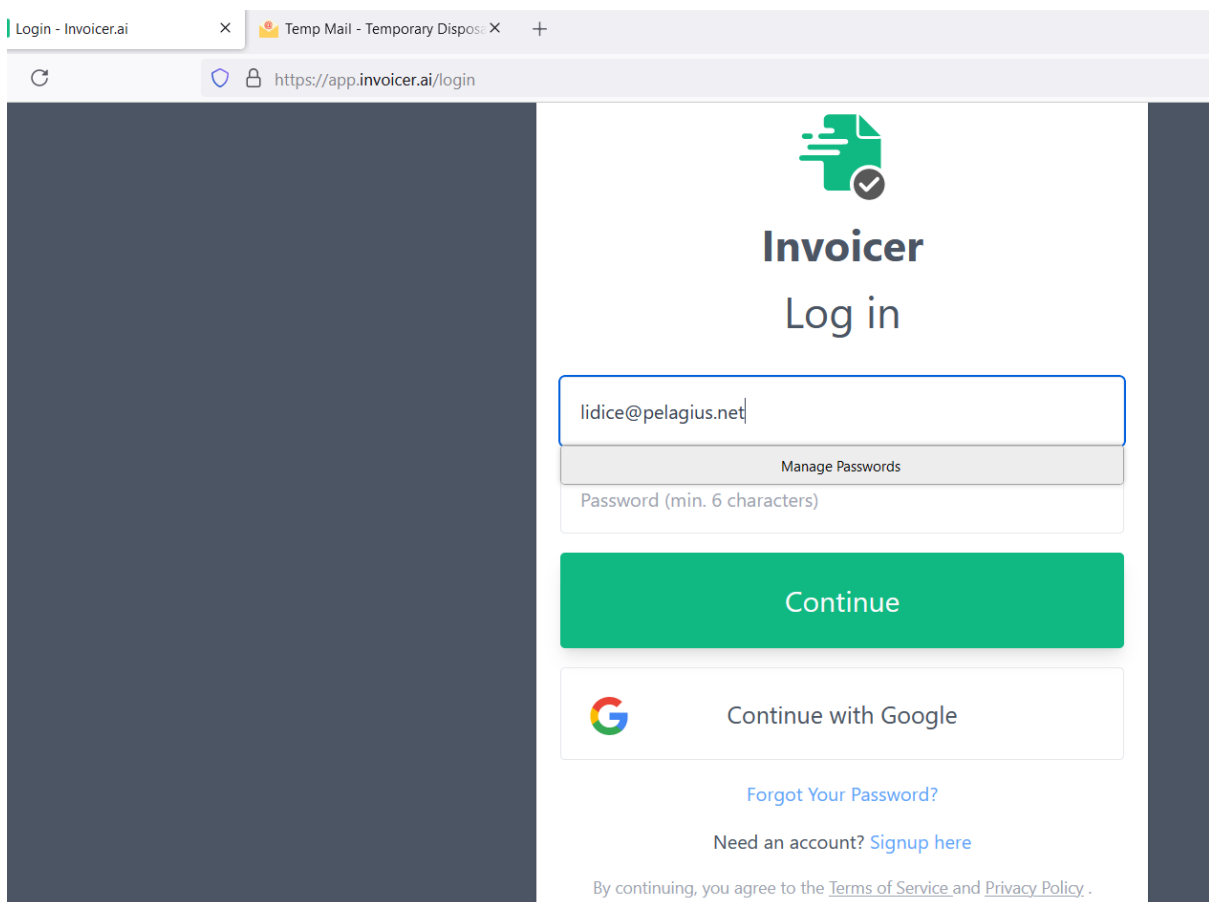
Detailed descriptions of the steps taken to perform the test. The results of testing, including any identified vulnerabilities.



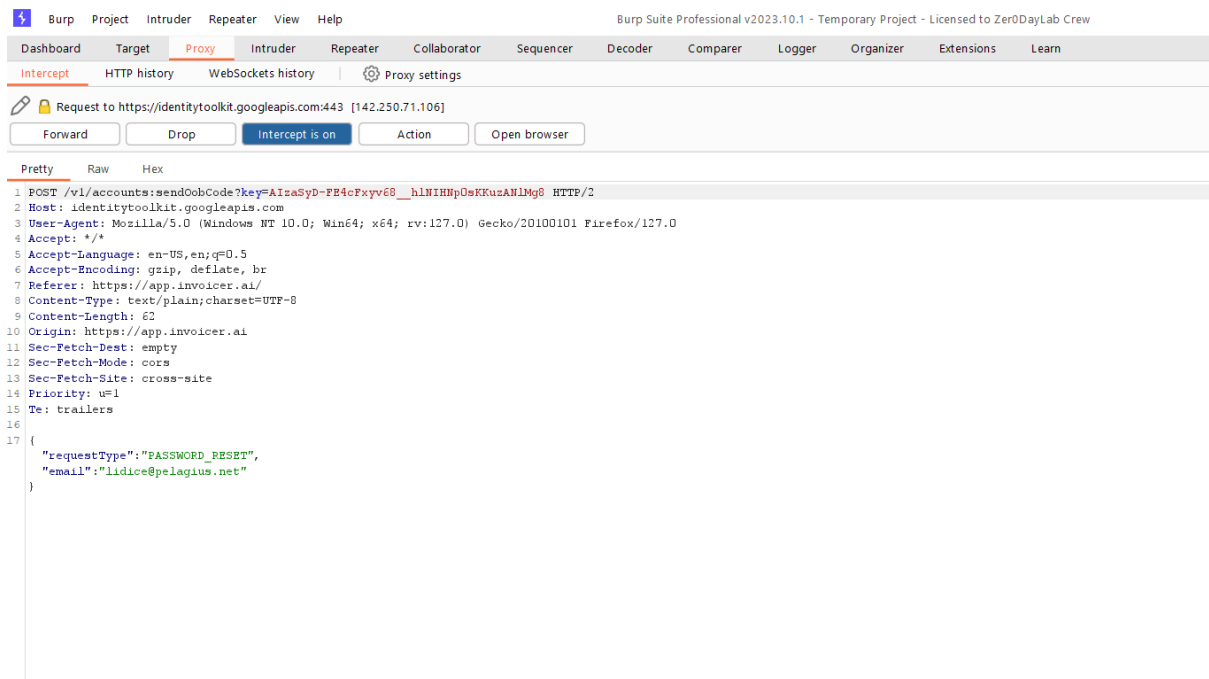
Took the email from the TEMPMAIL.O.com i.e lidice@pelagius.net



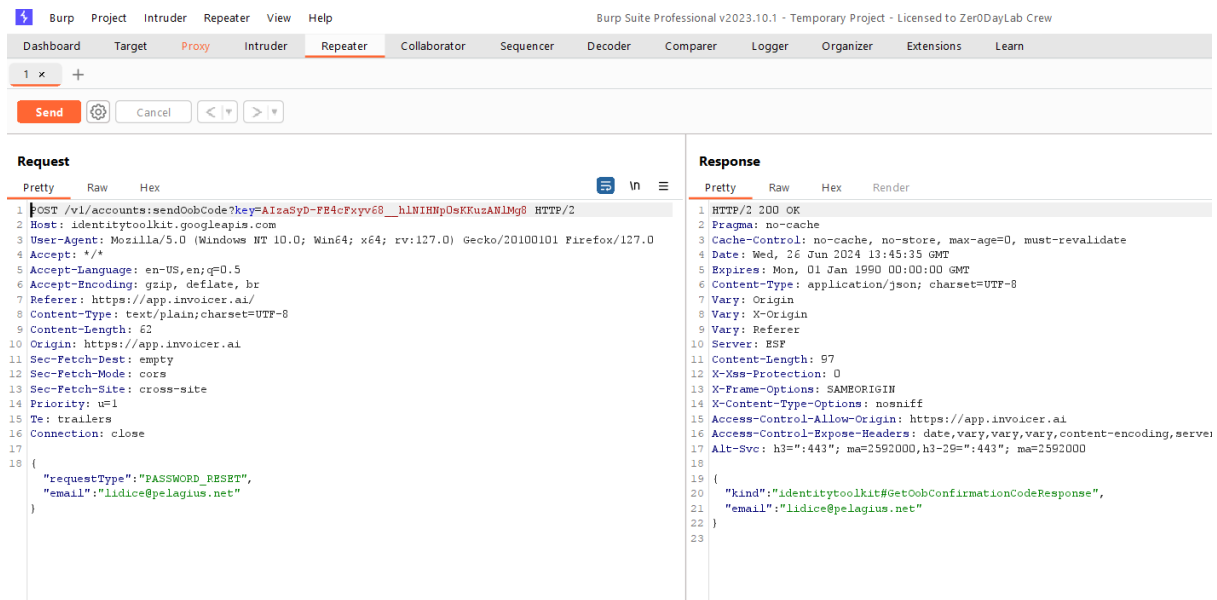
Did the sign up using the email and the password



Type the email id and did forget password while the intercept is on.



The traffic got catch in the intercept its showing “requestType”: PASSWORD_RESET”, and “email”: lidice@pelagius.net



Repeater screenshot

The image shows two side-by-side windows. The left window is Burp Suite Professional v2023.10.1, with the 'Repeater' tab selected. It displays a request to 'https://identitytoolkit.googleapis.com' with a POST method. The request body is a JSON object: {"requestType": "PASSWORD_RESET", "email": "lidice@pelagius.net"}. The right window is a web browser showing the 'Login - Invoicer.ai' page. The page has a green logo, the text 'Invoicer Log in', and a login form with fields for email (lidice@pelagius.net) and password (Password (min. 6 characters)). There is a green 'Continue' button and a 'Continue with Google' button.

The repeater is showing the same mail id as in the Log in page

The image shows the Burp Suite Professional v2023.10.1 'Intruder' tab. The 'Attack type' is set to 'Sniper'. The 'Payload positions' section is active, showing a list of positions for the request. The target is 'https://identitytoolkit.googleapis.com'. The request body is the same JSON object as in the previous image. The 'Update Host header to match target' checkbox is checked. On the right side, there are buttons for 'Add', 'Clear', 'Auto', and 'Refresh'.

Sent to intruder to add null payloads

1 x 2 x +

Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn

Positions Payloads Resource pool Settings

1 ? Payload sets

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.

Payload set: 1 Payload count: 20

Payload type: Null payloads Request count: 0

2 ? Payload settings [Null payloads]

This payload type generates payloads whose value is an empty string. With no payload markers configured, this can be used to repeatedly issue the base request unmodified.

☒ Generate 20 payloads

☐ Continue indefinitely

3 ? Payload processing

You can define rules to perform various processing tasks on each payload before it is used.

Add Enabled Rule

Edit

Remove

Up

Down

4 ? Payload encoding

This setting can be used to URL-encode selected characters within the final payload, for safe transmission within HTTP requests.

☒ URL-encode these characters:

Addes Null payload to generate 20 payloads and start the attack

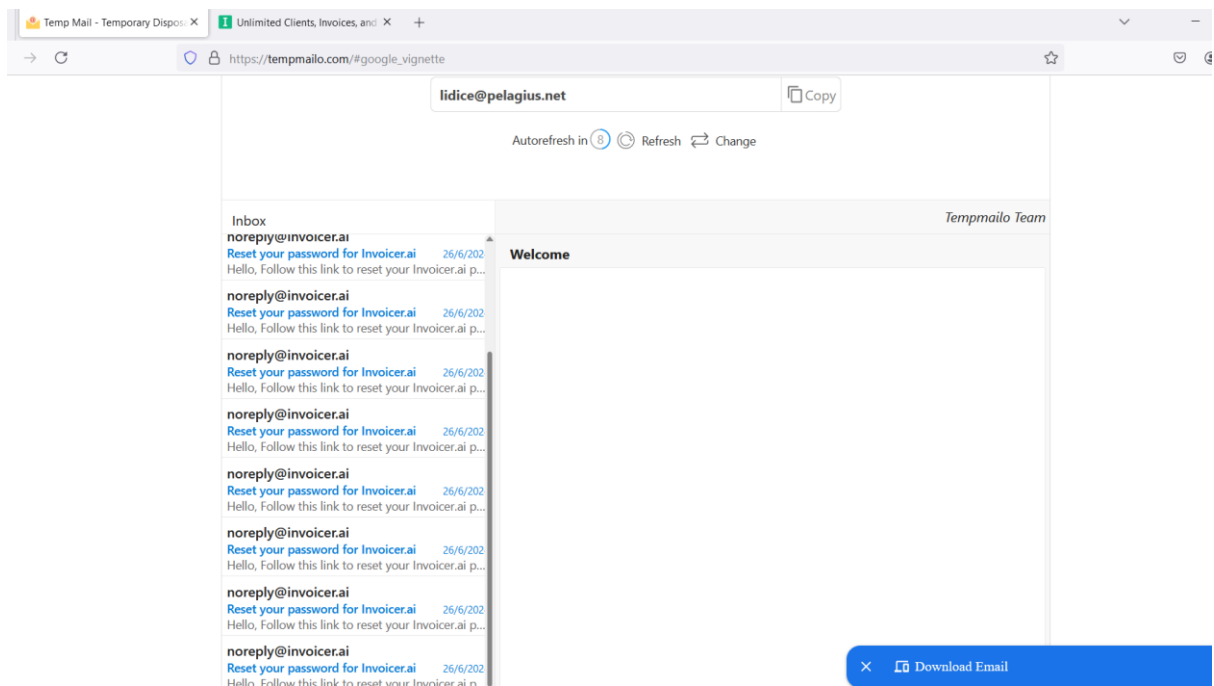
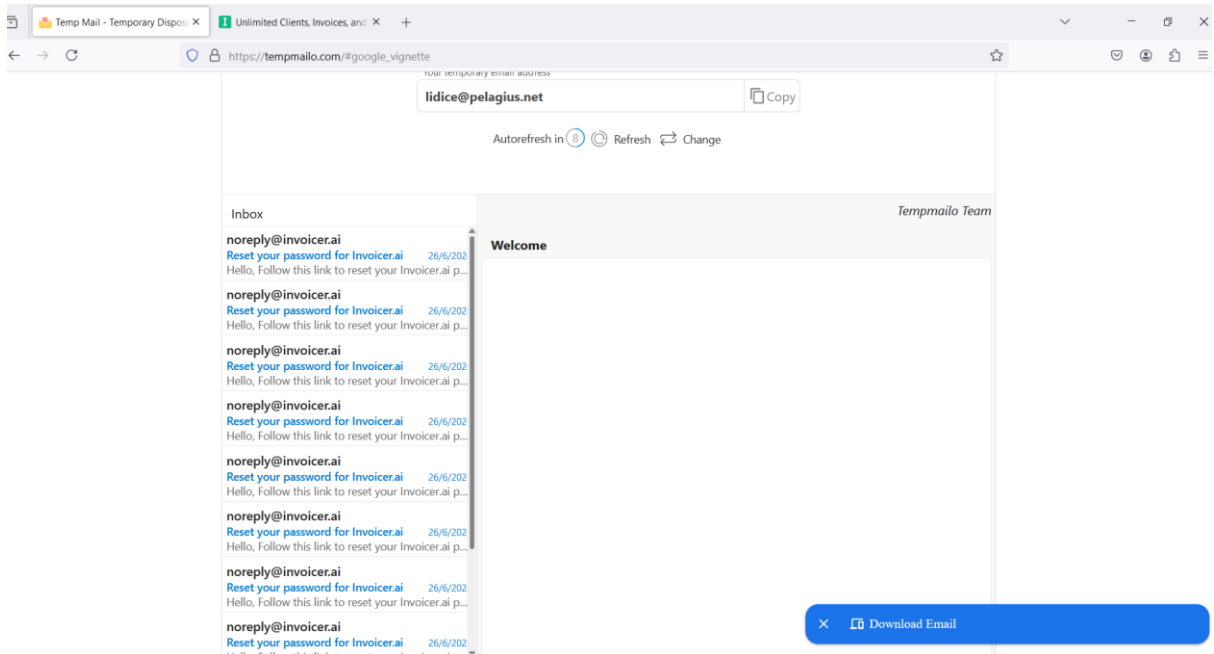
Attack Save Columns 3. Intruder attack of https://identitytoolkit.googleapis.com - Temporary attack - Not saved to project file

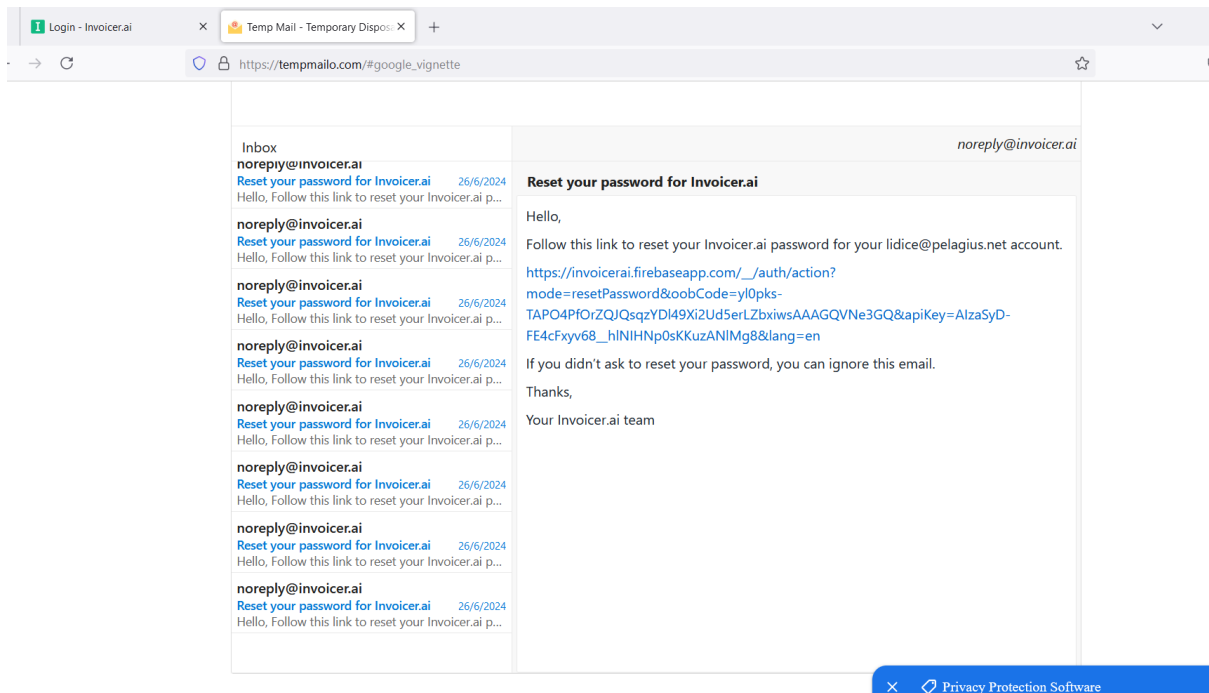
Results Positions Payloads Resource pool Settings

Filter: Showing all items

Request ^	Payload	Status code	Error	Timeout	Length	Comment
0		200	<input type="checkbox"/>	<input type="checkbox"/>	682	
1	null	200	<input type="checkbox"/>	<input type="checkbox"/>	682	
2	null	200	<input type="checkbox"/>	<input type="checkbox"/>	682	
3	null	200	<input type="checkbox"/>	<input type="checkbox"/>	682	
4	null	200	<input type="checkbox"/>	<input type="checkbox"/>	682	
5	null	200	<input type="checkbox"/>	<input type="checkbox"/>	682	
6	null	200	<input type="checkbox"/>	<input type="checkbox"/>	682	
7	null	200	<input type="checkbox"/>	<input type="checkbox"/>	682	
8	null	200	<input type="checkbox"/>	<input type="checkbox"/>	682	
9	null	200	<input type="checkbox"/>	<input type="checkbox"/>	682	
10	null	200	<input type="checkbox"/>	<input type="checkbox"/>	682	
11	null	200	<input type="checkbox"/>	<input type="checkbox"/>	682	
12	null	200	<input type="checkbox"/>	<input type="checkbox"/>	682	
13	null	200	<input type="checkbox"/>	<input type="checkbox"/>	682	
14	null	200	<input type="checkbox"/>	<input type="checkbox"/>	682	
15	null	200	<input type="checkbox"/>	<input type="checkbox"/>	682	
16	null	200	<input type="checkbox"/>	<input type="checkbox"/>	682	
17	null	200	<input type="checkbox"/>	<input type="checkbox"/>	682	
18	null	200	<input type="checkbox"/>	<input type="checkbox"/>	682	
19	null	200	<input type="checkbox"/>	<input type="checkbox"/>	682	
20	null	200	<input type="checkbox"/>	<input type="checkbox"/>	682	

The 20 null payloads





The email from the inbox

The vulnerability is of receiving a huge number of reset password verification code in my email using the same Intruder attack with null payloads.

Potential impacts of the identified issues.

The inbox is showing huge number of emails of reset password verification code it requires Implementation of proper rate limiting on password reset requests is essential to mitigate these risks, ensuring the security, reliability, and user satisfaction of the service.

If not controlled it may lead to various issues: -

Security Risks

User Experience

Operational Issues as the way the user operates

Reputation Damage for both trust and damage of brand

Recommendations for mitigation if any vulnerabilities are found.

Limit the number of passwords reset requests a user can make within a specific timeframe.

Increase the time interval between subsequent reset requests after each attempt.

Implement CAPTCHA to verify that the requests are coming from humans and not automated bots, especially after a certain number of failed attempts.

Implement the I am not the robot for any automated tools are accessing or not

Educate users about phishing attacks and how to recognize legitimate communications from your service.

Encourage the use of strong, unique passwords and provide guidelines for creating them by using special symbols in it like @, #, \$, & etc.

Validate all inputs and outputs to prevent injection attacks and other common vulnerabilities.

