

amanyadav78 / CVE-2026-29861 Public[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Security and quality](#) [Insights](#)[..](#) [1 Branch](#) [0 Tags](#)  [Go to file](#) [Code](#) [...](#)[amanyadav78](#) Create README.md a85a4d3 · last week[README.md](#) [Create README.md](#) last week[README](#)

# CVE-2026-29861

## CVE Disclosure: CVE-2026-29861 — SQL Injection in PHP-MYSQL-User-Login-System

**Disclosure Date:** 10 April 2026

**CVE ID:** CVE-2026-29861 **Severity:** CRITICAL (CVSS 9.8)

### Summary

A critical SQL Injection vulnerability exists in `PHP-MYSQL-User-Login-System v1.0`, specifically within the `login.php` endpoint. The application directly incorporates unsanitized user inputs into SQL queries, allowing unauthenticated attackers to bypass authentication and gain full administrative access.

This issue has been assigned the identifier **CVE-2025-26198**. At the time of public disclosure, **no official patch** was available.

### Affected Product

- **Vendor:** Independent (keerti1924)
- **Project:** [PHP-MYSQL-User-Login-System](#)
- **Version:** v1.0
- **File:** `login.php`

- **Vulnerable Endpoint:**

```
http://localhost/PHP-MYSQL-User-Login-System/login.php
```

## Vulnerability Details

The admin login mechanism uses unsanitized input directly in SQL queries without any input validation or prepared statements:

```
$query = "SELECT * FROM admin WHERE username='$username' AND password='$password'";
```

This allows for injection payloads such as:

```
Username: ' OR '1'='1  
Password: [any value]
```

This bypasses authentication logic by evaluating to a true condition, thereby granting access to the admin dashboard.

## CWE Classification





CWE ID	Title
<a href="#">CWE-89</a>	Improper Neutralization of Special Elements used in an SQL Command

## CVSS v3.1 Score

Score	Severity	Vector String
9.8	CRITICAL	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

## Impact

A successful exploitation could result in:

-  Full **authentication bypass**
-  **Unauthorized access** to privileged admin features
-  Potential **data leakage or manipulation** using `UNION`-based SQL injection
-  Full **compromise of the backend database**

## Proof of Concept (PoC)

### 1. Clone the Repository

```
git clone https://github.com/keerti1924/PHP-MYSQL-User-Login-System.git
```



### 2. Host Locally

Use XAMPP/LAMP to deploy the project and navigate to:

```
http://localhost/PHP-MYSQL-User-Login-System/login.php
```







### 3. Payload Injection

Enter the following credentials in the login form:


- **Username:** ' OR '1'='1
- **Password:** [any value]

You will be logged in as the first admin user, verifying successful SQL injection.

## Recommendations

-  Replace dynamic SQL queries with **prepared statements** (`mysqli_prepare()` or **PDO**).
-  Perform **input validation and sanitization** for all user inputs.
-  Deploy a **Web Application Firewall (WAF)** to block known SQL injection patterns.
-  Conduct **regular code audits** and **penetration testing** for early detection.

## Timeline

Event	Date
Vulnerability Discovered	22 January 2026
Public Disclosure	10 April 2026
Patch Available	 Not available as of disclosure



## Credits

---

This vulnerability was discovered and responsibly disclosed by:

**Aman Yadav**

---



## References

---

- [OWASP - SQL Injection](#)
- [PortSwigger - SQL Injection](#)
- [CVE-2025-26198 on CVE.org](#)

---

## Releases

No releases published

---

## Packages

No packages published

---

## Contributors 1



**amanyadav78** Aman Yadav