SQL injection was found in “http://localhost/Dairy-Farm-Shop-Management-System/dfsms/add-product.php” in PHPGurukul Dairy Farm Shop Management System Project in PHP v1.3 allows remote attackers to execute arbitrary code via “category, company, productname and productprice” POST request parameter.

* Official Website URL

https://phpgurukul.com/dairy-farm-shop-management-system-using-php-and-mysql/

* Affected Product Name:Dairy Farm Shop Management System

|  |  |
| --- | --- |
| Affected Vendor | Phpgurukul |
| Affected Code File | add-product.php |
| Affected Parameter | category, company, productname and productprice |
| Method | POST |
| Type | Time-based blind |
| Version | V 1.3 |

**Steps to Reproduce:**

1. **Log in to the Admin Panel:**

* Open the admin login page.
* Enter valid credentials and sign in.

A screenshot of a computer

AI-generated content may be incorrect.

1. **Navigate to the product section:**

* Click the “Add” button for add the product.
* Add anything in the input field.

A screenshot of a computer

AI-generated content may be incorrect.

1. **Intercept the Request:**

* Launch Burp Suite and configure your browser to route traffic through it.
* Enable Burp Suite Interceptor to capture requests.

A screenshot of a computer

AI-generated content may be incorrect.

1. **Modify the Request:**

* Capture the request when updating user details.
* Send it to the Burp Suite Repeater.
* Modify the **category, company, productname and productprice** parameter by injecting this payload:

'%2b(select\*from(select(sleep(20)))a)%2b'

A screenshot of a computer

AI-generated content may be incorrect.

1. **Send** **the Modified Request:**

• Forward the modified request in the Burp Suite Repeater.

• Observe the delay in the response time.

• The server will delay its response by 40 seconds, calculate all the parameter payload confirming the successful execution of the SLEEP () function, indicating a time-based SQL injection vulnerability.

A screenshot of a computer

AI-generated content may be incorrect.

**Recommended Mitigations:**

**Mitigation:** Implement proper input validation, output encoding, and Content Security Policy (CSP) to prevent malicious HTML injection.

**Impact**

* Data Theft: Unauthorized access to sensitive user or system data in the database.
* Data Manipulation: Modification or erasure of data, which destroys the integrity of data.
* Reconnaissance: Enumeration of the database structure, such as tables, columns, and schemas, for further exploitation.
* Financial Loss: Service denial, and possibly monetary losses to the production environment
* Loss of Reputation: Potential for loss of trust among users to either data breach or disruption in services.

**Recommended Mitigations:**

[SQL Injection Prevention - OWASP Cheat Sheet Series](https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html)