SQL Injection in was found in “Park-Ticketing-Management-System-Project/ptms/ normal-bwdates-reports-details.php” in PHPGurukul Park Ticketing Management System Project in PHP v2.0 allows remote attackers to execute arbitrary code via “todate” POST request parameter.

* Official Website URL

https://phpgurukul.com/park-ticketing-management-system-using-php-and-mysql/

* Affected Product Name:Park Ticketing Management System Using PHP and MySQL

|  |  |
| --- | --- |
| Affected Vendor | Phpgurukul |
| Affected Code File | normal-bwdates-reports-details.php |
| Affected Parameter | today |
| Method | POST |
| Type | Time-based-blind |
| Version | V 2.0 |

**Steps to Reproduce:**

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

* Open the admin login page.
* Enter your credentials and sign

A screenshot of a computer

AI-generated content may be incorrect.

1. **Go to PTMS Admin section:**

* Navigate to the “Report” section.
* Click the “Normal People Report” section.
* Choose any fromdate to todate 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, **todate** parameter by injecting this payload:

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

A screenshot of a computer program

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 20 seconds, 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)