SQL Injection in was found in “Beauty-Parlour-Management-System/bpms/index.php” in Project in PHP v1.1 allows remote attackers to execute arbitrary code via “name” POST request parameter.

* **Official Website URL**

<https://repo.freeprojectscodes.com/?wpdmpro=download-beauty-parlor-management-system-source-code>

* **Affected Product Name: -** Beauty Parlour Management System

|  |  |
| --- | --- |
| **Affected Vendor** | Source code and project |
| **Affected Code File** | Beauty-Parlour-Management-System/bpms/index.php |
| **Affected Parameter** | name |
| **Method** | POST |
| **Type** | Time-based blind |
| **Version** | V1.1 |

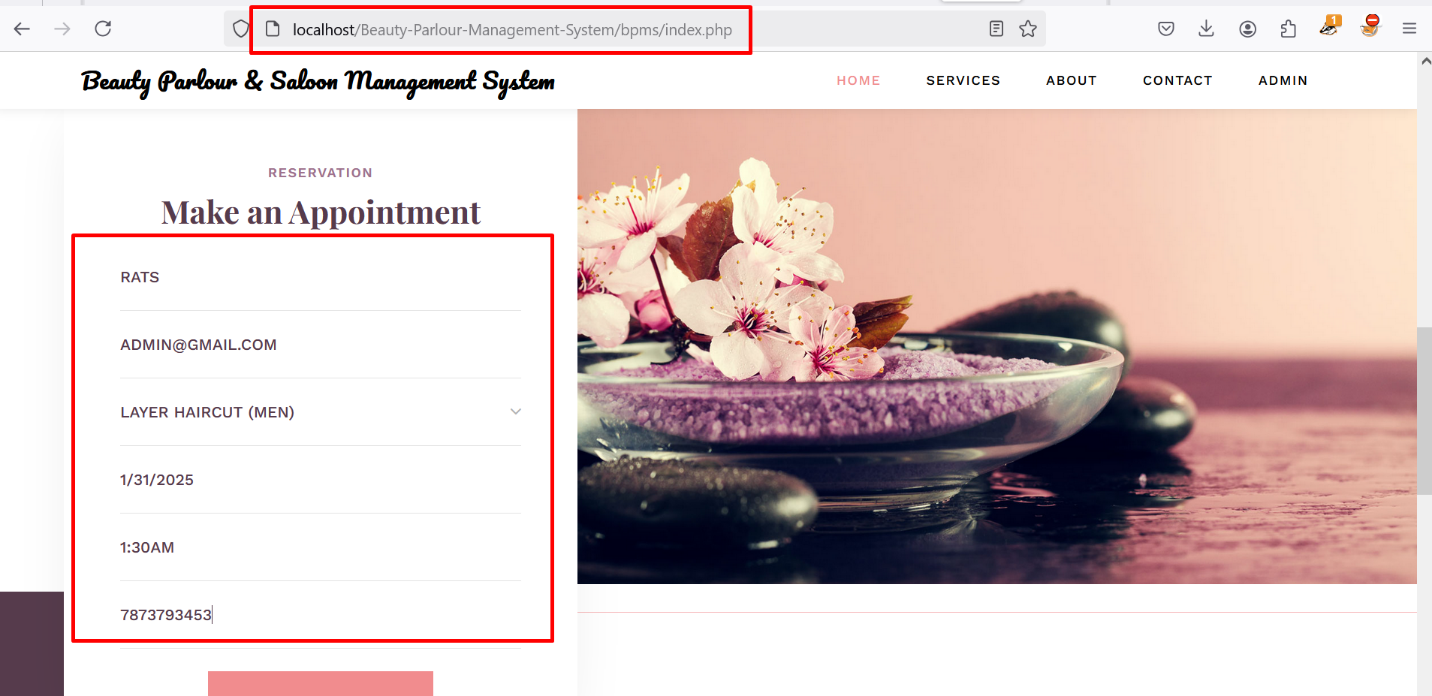
**Vulnerability Overview**

The vulnerability allows remote attackers to exploit the “name” parameter in Beauty Parlour Management System Project v1.1 to execute arbitrary SQL commands. By injecting time-delay payloads, attackers can determine the presence of a SQL Injection flaw by observing server response delays, confirming successful execution of SQL commands.

**Steps to Reproduce:**

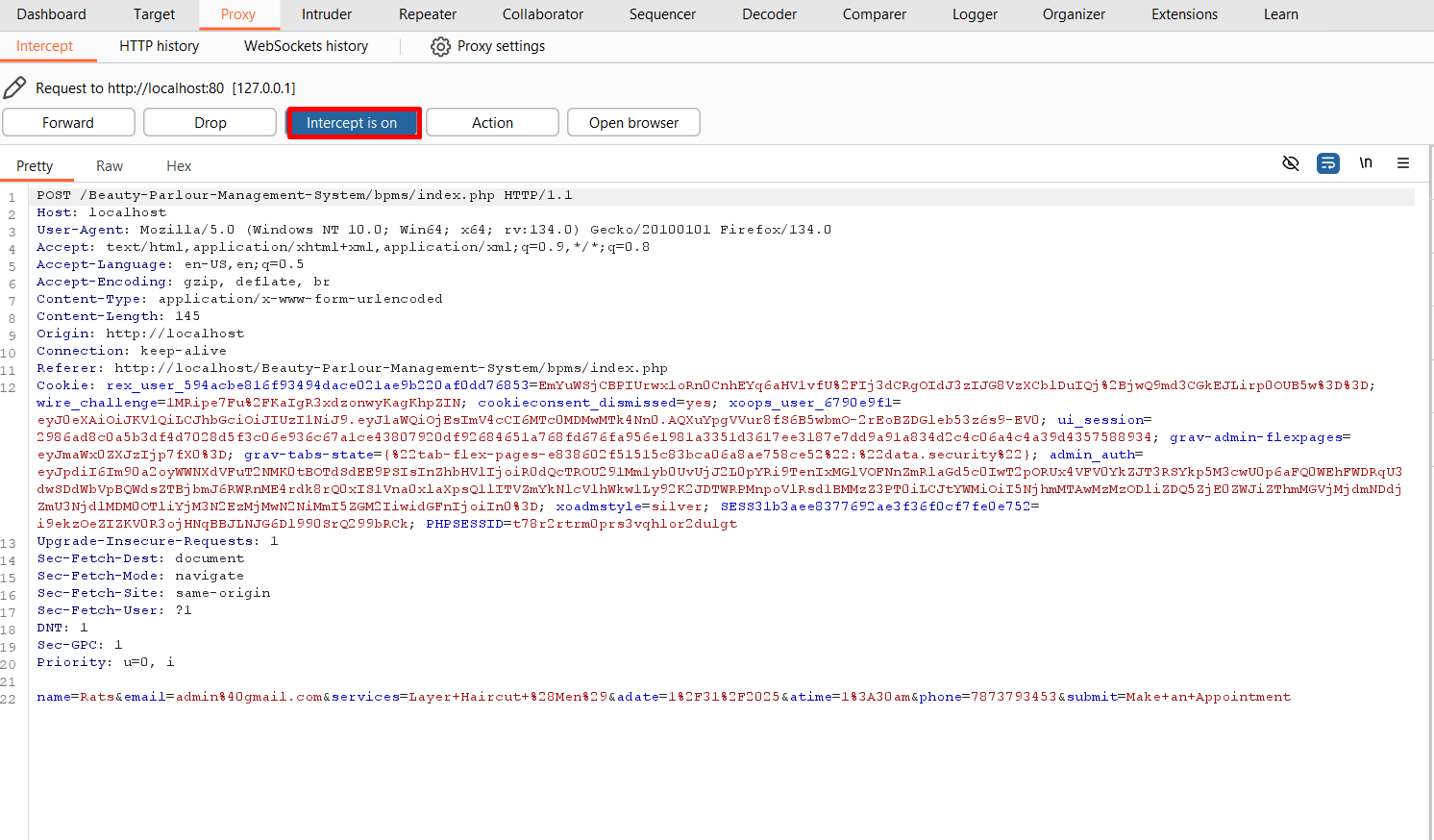
1. **Access the URL**

<http://localhost/Beauty-Parlour-Management-System/bpms/index.php> for Make an Appointment



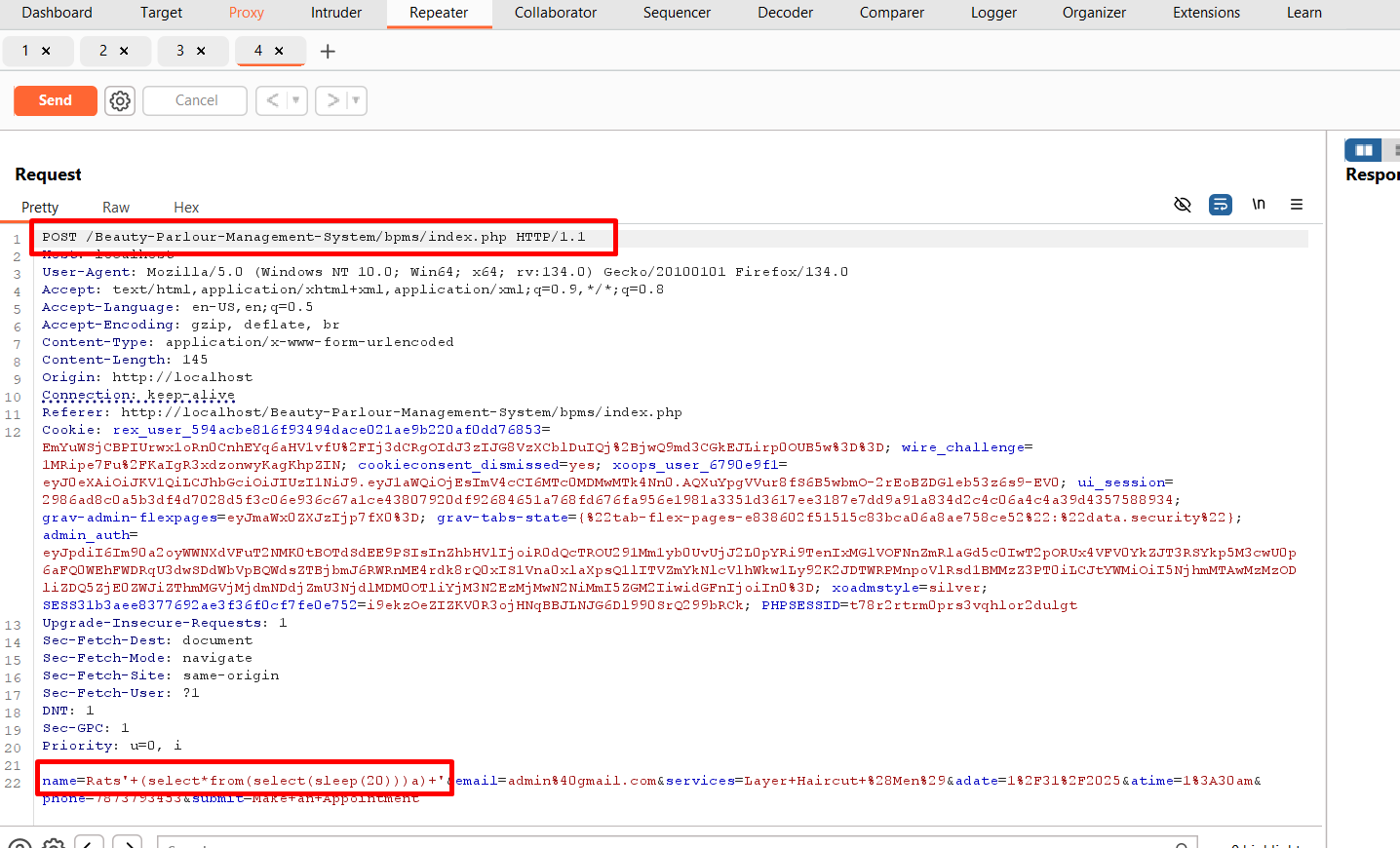
1. **Intercept the Request:**

**•** Enable Burp Suite and set up the browser to route traffic through it.



1. **Modify the Parameter:**

• Send the request to the Burp Suite Repeater and modify the “**name**” parameter with the following payload: (**'%2b(select\*from(select(sleep(20)))a)%2b'**)



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

Description automatically generated

**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.
* Credential Exposure: Exploitation to obtain usernames, passwords, or other authentication details.
* Server Compromise: Use of database queries for exploitation of underlying server systems or gaining shell access.
* 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)