# SQL Injection Vulnerability Report

## Affected Product

|  |  |
| --- | --- |
| **Attribute** | **Details** |
| **Product Name** | Online Shopping Portal Project |
| **Vendor** | PHPGurukul |
| **Version** | v2.1 |
| **Affected File** | Online Shopping Portal project-V2.0\shopping\check\_availability.php |
| **Affected Parameter** | email |
| **Method** | POST |
| **Vulnerability Type** | Time-Based Blind SQL Injection |

## Official Website

[PHPGurukul - Online Shopping Portal](https://phpgurukul.com/shopping-portal-free-download/)

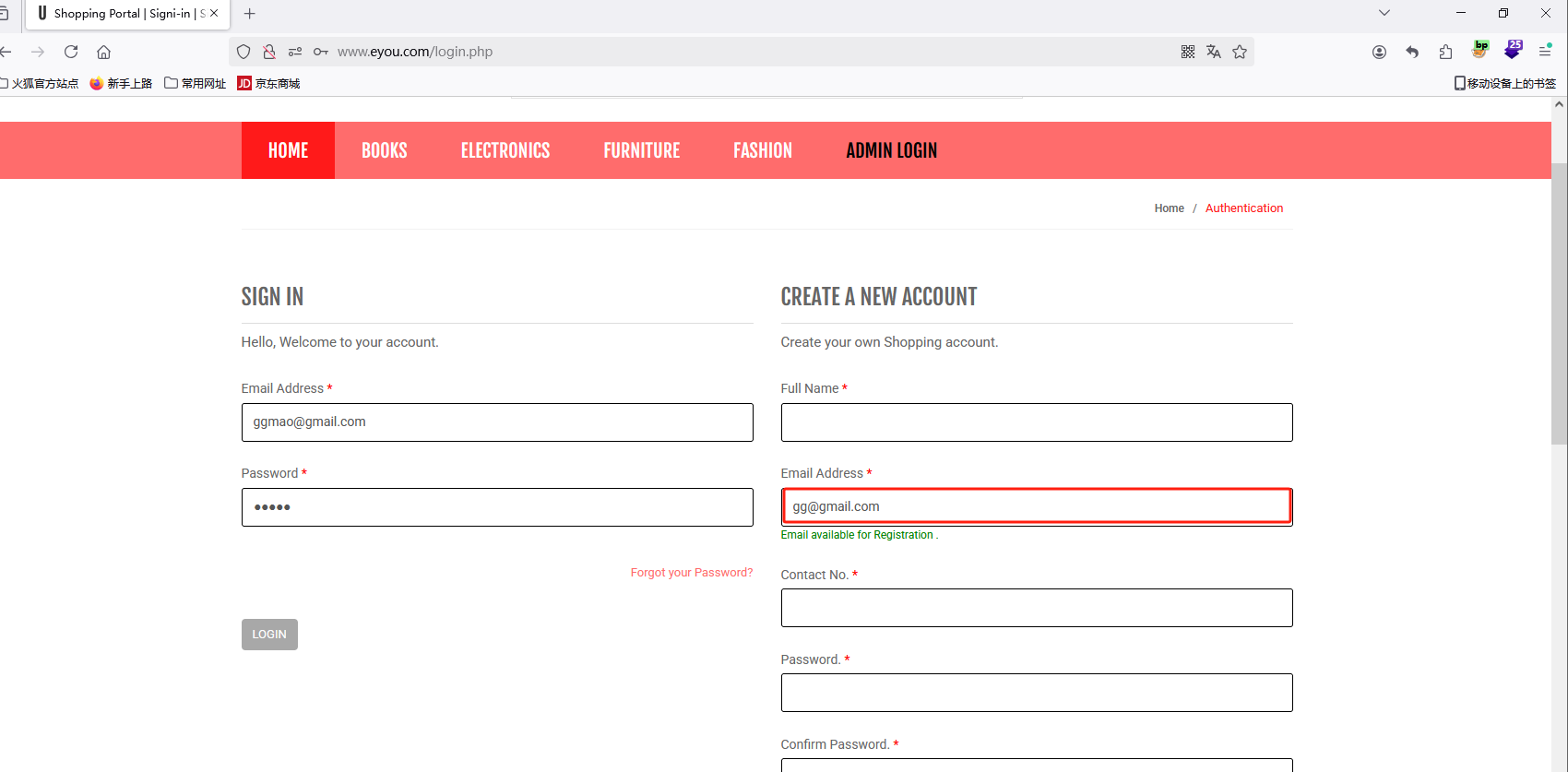
## Vulnerability Overview

A SQL Injection vulnerability exists in the email parameter of the **Online Shopping Portal Project v2.1**, allowing remote attackers 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.

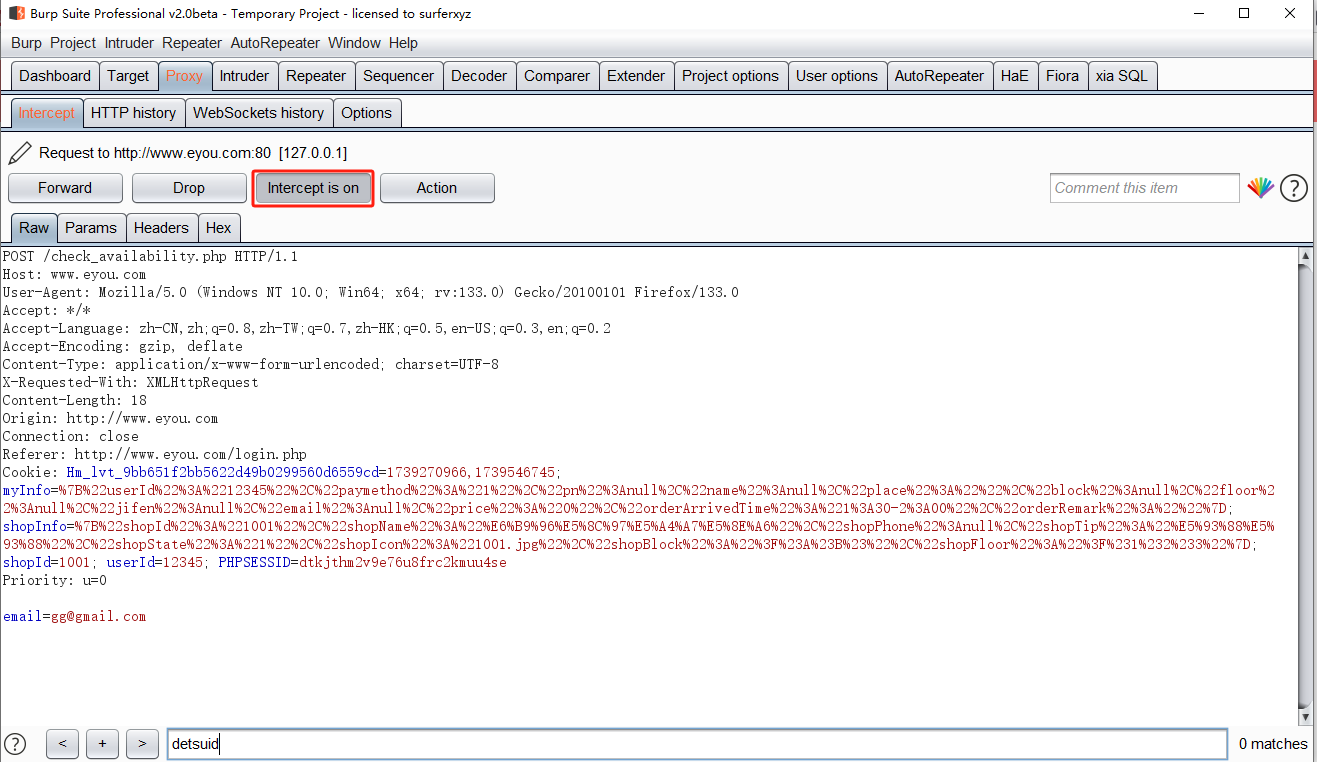
## Steps to Reproduce

1. **Access the Vulnerable URL:**

|  |
| --- |
| http://www.eyou.com/login.php |



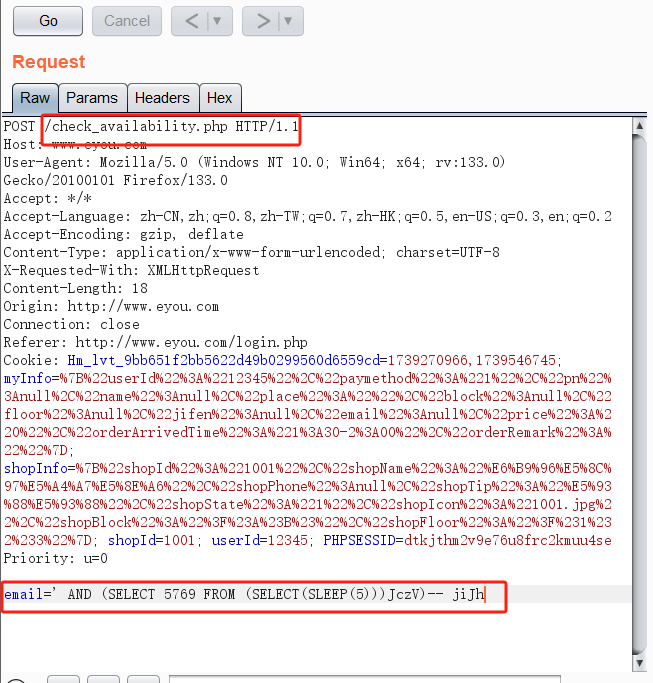
1. **Intercept the Request:**

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

1. **Modify the Parameter:**

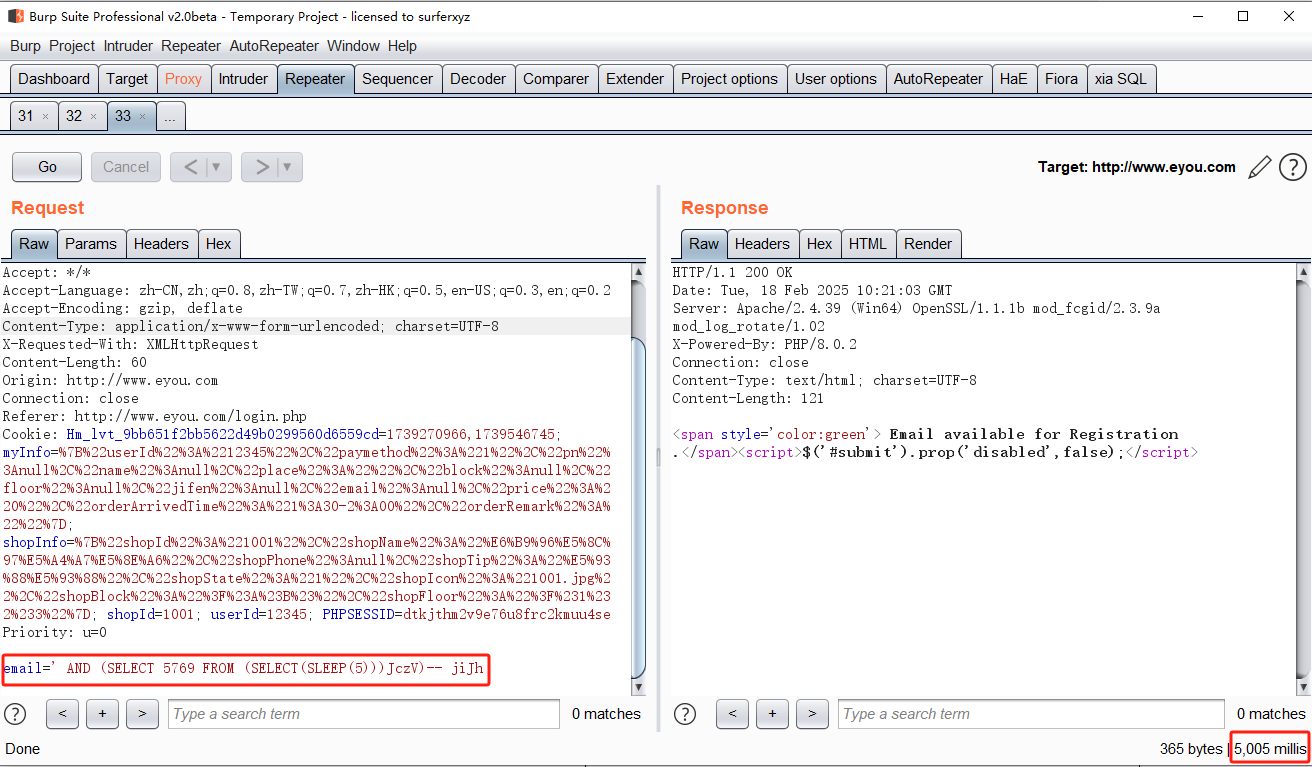
**Send the request to Burp Suite Repeater and modify the email parameter with the following payload:**

|  |
| --- |
| **' AND (SELECT 5769 FROM (SELECT(SLEEP(5)))JczV)-- jiJh** |



4.**Send the Modified Request:**

* Forward the modified request in Burp Suite Repeater.
* Observe the delay in the response time.
* The server will delay its response by 5 seconds, confirming successful execution of the SLEEP() function, indicating a **time-based SQL injection vulnerability**.



## Impact

* **Data Theft:** Unauthorized access to sensitive user or system data.
* **Data Manipulation:** Modification or deletion of database records.
* **Credential Exposure:** Extraction of usernames, passwords, or authentication details.
* **Server Compromise:** Potential exploitation of underlying server systems.
* **Reconnaissance:** Enumeration of database structures (tables, columns, schemas).
* **Financial Loss:** Downtime and potential monetary losses.
* **Loss of Reputation:** User trust degradation due to service disruption or data breaches.

## Recommended Mitigations

* **Use Prepared Statements (Parameterized Queries).**
* **Sanitize User Inputs:** Validate and filter all incoming data.
* **Implement Web Application Firewall (WAF).**
* **Use the Principle of Least Privilege (PoLP) for database users.**
* **Regularly Update and Patch the Application.**
* **Monitor Logs for Suspicious Activities.**

For detailed guidelines, refer to: [OWASP SQL Injection Prevention Cheat Sheet](https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html).