Security Vulnerability Report

Generated: 2025-03-05 00:00:37.381206

Summary

- Total Vulnerabilities: 2
- Critical: 2
- High: 0
- Medium: 0
- Low: 0
- Info: 0
- Risk Score: 22.00

Detailed Vulnerabilities

SQL_INJECTION (CRITICAL)

- **Description:** The application is vulnerable to SQL injection because it directly concatenates user-supplied input into an SQL query. An attacker can inject arbitrary SQL code by manipulating the 'id' parameter in the request.
- **Impact:** An attacker can read, modify, or delete sensitive data in the database, potentially leading to complete compromise of the application and underlying system.
- **Location:** main.py:9
- **CWE ID:** CWE-89
- **OWASP Category:** A03:2021 - Injection
- **CVSS Score:** 5.0
- **Remediation:** Use parameterized queries or prepared statements to prevent SQL injection. This ensures that user input is treated as data, not as part of the SQL query.

References:

- https://owasp.org/Top10/A03_2021-Injection/
- https://cwe.mitre.org/data/definitions/89.html

Proof of Concept:

```
Send a request like: /user?id=1 OR 1=1;--
```

Secure Code Example:

```
def get_user(): user_id = request.args.get('id', '') conn = sqlite3.connect('test.db')
cursor = conn.cursor() query = "SELECT * FROM users WHERE id = ?" try:
cursor.execute(query, (user_id,)) result = cursor.fetchone() except Exception as e:
result = str(e) conn.close() return jsonify({'result': result})
```

OS_COMMAND_INJECTION (CRITICAL)

- **Description:** The application is vulnerable to command injection because it executes user-supplied input as a system command without any sanitization. An attacker can inject arbitrary commands by manipulating the 'cmd' parameter in the request.
- **Impact:** An attacker can execute arbitrary commands on the server, potentially leading to complete compromise of the application and underlying system.
- **Location:** main.py:19
- **CWE ID:** CWE-78
- **OWASP Category:** A03:2021 - Injection
- **CVSS Score:** 5.0

- **Remediation:** Avoid using os.popen() or similar functions to execute commands based on user input. If command execution is necessary, use a safe API that properly sanitizes and validates input, or use a whitelist of allowed commands.

References:

- https://owasp.org/Top10/A03_2021-Injection/
- https://cwe.mitre.org/data/definitions/78.html

Proof of Concept:

```
Send a request like: /exec?cmd=ls -la
```

Secure Code Example:

```
import subprocess def exec_command(): command = request.args.get('cmd', '') # Example of whitelisting allowed commands if command not in ['safe_command1', 'safe_command2']: return jsonify({'error': 'Command not allowed'}) # Or handle the error appropriately try: result = subprocess.run(command, shell=False, capture_output=True, text=True, check=True) output = result.stdout except subprocess.CalledProcessError as e: output = str(e) return jsonify({'output': output})
```

Chained Vulnerabilities

Vulnerability Chain (Combined Severity: CRITICAL)

- **Attack Path:** Attack Chain Severity: CRITICAL Prerequisites: Command execution context, Database access, Input injection point, User interaction Attack Path Analysis: Step 1: SQL_INJECTION (CRITICAL) Location: main.py:9-17 Attack Vector: The application is vulnerable to SQL injection because it directly concatenates user-supplied input into an SQL query Potential Impact: An attacker can read, modify, or delete sensitive data in the database, potentially leading to complete compromise of the application and underlying system ↓ Chain Effect: This vulnerability could facilitate or amplify the next attack step Step 2: OS_COMMAND_INJECTION (CRITICAL) Location: main.py:19-22 Attack Vector: The application is vulnerable to command injection because it executes user-supplied input as a system command without any sanitization Potential Impact: An attacker can execute arbitrary commands on the server, potentially leading to complete compromise of the application and underlying system
- **Likelihood:** 0.57
- **Mitigation Priority:** 1

Prerequisites:

- Command execution context
- Database access
- Input injection point
- User interaction