In-depth Weaknesses Documentation

1. Weakness Overview

Title	Example SQL Injection in Login Form
Identifier	WEAK-2024-001
Date Detected	2024-06-01
Detected By	John Doe
Severity	High
Status	Open

2. Description

The login form on the application does not sanitize user input before passing it to the SQL query, allowing attackers to inject arbitrary SQL statements. This could lead to unauthorized access, data leakage, or modification of database records.

3. Technical Details

- **Vulnerable Endpoint:** `/login`
- **Request Method:** POST
- **Payload Example:** `username=admin' -- &password=irrelevant`
- **Affected Parameter(s):** username
- **Affected Component:** Authentication

4. Proof of Concept (PoC)

```
POST /login HTTP/1.1
Host: example.com
Content-Type: application/x-www-form-urlencoded
username=admin' -- &password=irrelevant
```

The above payload logs into the application as "admin" without providing a valid password, demonstrating successful exploitation.

5. Impact

- Unauthorized data disclosure and access
- Potential data modification or deletion
- Privilege escalation
- Complete compromise of affected database

6. Mitigation & Recommendations

- Implement parameterized SQL queries for all database access
- Input validation and sanitization on server-side
- Conduct code review of authentication modules

- Apply database permissions to limit access

7. References

- CWE-89: Improper Neutralization of Special Elements used in an SQL Command

Important Notes

- Thorough documentation helps prioritize and remediate risks effectively.
- Always keep information accurate, actionable, and updated as the issue evolves.
- Include enough context for technical and non-technical stakeholders.
- Store documents in a secure, access-controlled location.