SAST Security Scan Report

Scan ID: sast_d87551a1
Filename: vuln.py
Scan Date: 2025-04-27 10:33:17

Vulnerability Summary

Risk Level	Count
High	1
Medium	1
Low	0
Total	2

Detailed Findings

SQLI-001: SQL Injection (CVSS: 9.0)

Severity: HIGH
Description: Do■rudan SQL sorgusu olu■turma güvenlik aç■■■na neden olabilir
File: vuln.py (Line: 1056)
Vulnerable Code:

```
// SQL sorgusu örne■i Injection def login(user_input): query = f"SELECT * FROM users WHERE name = '{user_input}'"
```

Recommendation:
Parametreli sorgular veya ORM kullan■n
AI-Generated Solution:
AI Solution: To fix SQL injection vulnerabilities: • Always use parameterized queries or prepared statements • Never concatenate user input directly into SQL strings • Use an ORM (Object-Relational Mapping) library • Implement proper input validation and sanitization • Apply the principle of least privilege for database accounts Example code fix: # Instead of: query = f"SELECT * FROM users WHERE username = '{username}';" # Use parameterized queries: query = "SELECT * FROM users WHERE username = %s;" cursor.execute(query, (username,)) # Or with SQLAlchemy: user = Users.query.filter_by(username=username).first()

HCP-001: Hardcoded Password (CVSS: 5.0)

Severity: MEDIUM
Description: Kod içinde sabit ■ifre kullan■m■ tespit edildi
File: vuln.py (Line: 508)
Vulnerable Code:

```
// ■ifre tan■m■ örne■i ad def weak_crypto(): import hashlib password = "admin123" hashed = hashlib.md5(pa
```

Recommendation:
■ifreleri environment variable veya güvenli bir yap■land■rma dosyas■nda saklay■n
AI-Generated Solution:
AI Solution: No specific solution available for this vulnerability. Please consult security best practices or contact a security expert.