Security Audit Report

Generated on 26 Feb 2026 at 23:02
Target: C:\Users\royal\OneDrive\Desktop\Sentinel_AI\sample_app

OVERALL RISK SCORE

272

CRITICAL RISK

44 findings | 20.9s scan | 4 file(s)

12	20	10	
Must Fix Now	**Fix Before Launch**	**Fix Soon**	**Whe**

Files Scanned

- C:\Users\royal\OneDrive\Desktop\Sentinel_AI\sample_app\app (1).py
- C:\Users\royal\OneDrive\Desktop\Sentinel_AI\sample_app\requirements.txt
- C:\Users\royal\OneDrive\Desktop\Sentinel_AI\requirements.txt
- C:\Users\royal\OneDrive\Desktop\Sentinel_AI\.git

Severity Breakdown

Severity	Count	Meaning	Action
CRITICAL	12	Can lead to hacking or data theft	Fix immediately
HIGH	20	Serious vulnerability	Fix before going live
MEDIUM	10	Moderate security concern	Fix in next update
LOW	2	Minor issue	Fix when convenient

Security Findings

The following 44 security issues were detected, ordered by severity.

1. Hardcoded Secret

CRITICAL **Must Fix Now**

File: app (1).py | Line: 12 | Ref: CWE-798 | Detected by: Agent A

Potential hardcoded credential or secret detected on line 12.

Problematic code:

```
DB_PASSWORD = "admin123"
```

How to fix it:

Use environment variables or a secrets manager (e.g., HashiCorp Vault, AWS Secrets Manager) instead of hardcoding secrets.

2. Hardcoded Secret

CRITICAL **Must Fix Now**

File: app (1).py | Line: 13 | Ref: CWE-798 | Detected by: Agent A

Potential hardcoded credential or secret detected on line 13.

Problematic code:

```
API_KEY = "sk-abc123xyz789hardcoded"
```

How to fix it:

Use environment variables or a secrets manager (e.g., HashiCorp Vault, AWS Secrets Manager) instead of hardcoding secrets.

3. Exposed API Key

CRITICAL **Must Fix Now**

File: app (1).py | Line: 13 | Ref: CWE-798 | Detected by: Agent A

Potential hardcoded credential or secret detected on line 13.

Problematic code:

```
API_KEY = "sk-abc123xyz789hardcoded"
```

How to fix it:

Use environment variables or a secrets manager (e.g., HashiCorp Vault, AWS Secrets Manager) instead of hardcoding secrets.

4. Use of eval() Detected

CRITICAL **Must Fix Now**

File: app (1).py | Line: 79 | Ref: CWE-95 | Detected by: Agent A

eval() on line 79 can execute arbitrary code if user input is passed.

Problematic code:

```
result = eval(data)
```

How to fix it:

Avoid eval() entirely. Use safe alternatives like ast.literal_eval() for data parsing, or redesign the logic.

5. Login Bypass - Always-True Condition

CRITICAL **Must Fix Now**

File: app (1).py | Line: 52 | Ref: CWE-287 | Detected by: Agent B

Function 'admin_panel' contains an 'or True' expression which makes the condition always evaluate to True, bypassing authentication checks.

Problematic code:

```
if role == "admin" or True:
```

How to fix it:

Review and remove the 'or True' condition. Ensure all authentication conditions are properly evaluated.

6. Privilege Escalation - Role Check Bypass

CRITICAL **Must Fix Now**

File: app (1).py | Line: 52 | Ref: CWE-269 | Detected by: Agent B

Function 'admin_panel' has a role-based check that appears to be bypassed by a short-circuit boolean expression.

Problematic code:

```
def admin_panel(): role = request.args.get("role", "user") # Vulnerability: Login bypass via role
parameter if role == "admin" or True: return jsonify({"data": "sensitive_admin_da
```

How to fix it:

Audit all role-based conditionals. Ensure privilege checks cannot be bypassed.

7. Vulnerable Dependency: django

CRITICAL **Must Fix Now**

File: requirements.txt | Line: 2 | Ref: CWE-1035 | Detected by: Agent E

django==2.2.0 may be affected by CVE-2022-28346: Django SQL injection vulnerability in QuerySet.annotate().

Problematic code:

```
django==2.2.0
```

How to fix it:

Upgrade django to the latest patched version. Run: pip install --upgrade django

8. Vulnerable Dependency: pillow

CRITICAL **Must Fix Now**

File: requirements.txt | Line: 4 | Ref: CWE-1035 | Detected by: Agent E

pillow==8.0.0 may be affected by CVE-2022-22817: Pillow PIL.ImageMath.eval allows arbitrary code execution.

Problematic code:

```
pillow==8.0.0
```

How to fix it:

Upgrade pillow to the latest patched version. Run: pip install --upgrade pillow

9. Vulnerable Dependency: pyyaml

CRITICAL **Must Fix Now**

File: requirements.txt | Line: 5 | Ref: CWE-1035 | Detected by: Agent E

pyyaml==5.3 may be affected by CVE-2020-14343: PyYAML arbitrary code execution via yaml.load() without Loader.

Problematic code:

```
pyyaml==5.3
```

How to fix it:

Upgrade pyyaml to the latest patched version. Run: pip install --upgrade pyyaml

10. Vulnerable Dependency: django

CRITICAL **Must Fix Now**

File: requirements.txt | Line: 2 | Ref: CWE-1035 | Detected by: Agent E

django==2.2.0 may be affected by CVE-2022-28346: Django SQL injection vulnerability in QuerySet.annotate().

Problematic code:

```
django==2.2.0
```

How to fix it:

Upgrade django to the latest patched version. Run: pip install --upgrade django

11. Vulnerable Dependency: pillow

CRITICAL **Must Fix Now**

File: requirements.txt | Line: 4 | Ref: CWE-1035 | Detected by: Agent E

pillow==8.0.0 may be affected by CVE-2022-22817: Pillow PIL.ImageMath.eval allows arbitrary code execution.

Problematic code:

```
pillow==8.0.0
```

How to fix it:

Upgrade pillow to the latest patched version. Run: pip install --upgrade pillow

12. Vulnerable Dependency: pyyaml

CRITICAL **Must Fix Now**

File: requirements.txt | Line: 5 | Ref: CWE-1035 | Detected by: Agent E

pyyaml==5.3 may be affected by CVE-2020-14343: PyYAML arbitrary code execution via yaml.load() without Loader.

Problematic code:

```
pyyaml==5.3
```

How to fix it:

Upgrade pyyaml to the latest patched version. Run: pip install --upgrade pyyaml

13. SQL Injection Risk - String Concatenation in Query

HIGH **Fix Before Launch**

File: app (1).py | Line: 42 | Ref: CWE-89 | Detected by: Agent A

Line 42 appears to build a SQL query via string concatenation or f-string, which is vulnerable to SQL injection.

Problematic code:

```
query = "SELECT * FROM users WHERE username = '" + username + "' AND password = '" + password + "'"
```

How to fix it:

Use parameterized queries or an ORM (e.g., SQLAlchemy). Never concatenate user input directly into SQL.

14. SQL Injection Risk - String Concatenation in Query

HIGH Fix Before Launch

File: app (1).py | Line: 69 | Ref: CWE-89 | Detected by: Agent A

Line 69 appears to build a SQL query via string concatenation or f-string, which is vulnerable to SQL injection.

Problematic code:

```
query = f"SELECT * FROM products WHERE name LIKE '%{term}%'"
```

How to fix it:

Use parameterized queries or an ORM (e.g., SQLAlchemy). Never concatenate user input directly into SQL.

15. SQL Injection Risk - String Concatenation in Query

HIGH Fix Before Launch

File: app (1).py | Line: 91 | Ref: CWE-89 | Detected by: Agent A

Line 91 appears to build a SQL query via string concatenation or f-string, which is vulnerable to SQL injection.

Problematic code:

```
cursor.execute("UPDATE users SET password = '" + new_pass + "' WHERE id = " + user_id)
```

How to fix it:

Use parameterized queries or an ORM (e.g., SQLAlchemy). Never concatenate user input directly into SQL.

16. Plaintext Password Comparison

HIGH Fix Before Launch

File: app (1).py | Line: 35 | Ref: CWE-256 | Detected by: Agent A

Line 35 compares a password in plaintext. Passwords should be hashed before storage and comparison.

Problematic code:

```
if username == ADMIN_USER and password == ADMIN_PASS:
```

How to fix it:

Use bcrypt, argon2, or PBKDF2 for password hashing. Never store or compare plaintext passwords.

17. Unsanitized User Input Reaches eval()

HIGH Fix Before Launch

File: app (1).py | Line: 79 | Ref: CWE-20 | Detected by: Agent C

In function 'eval_endpoint', user-controlled variable(s) ['data'] flow directly into 'eval()' without apparent sanitization.

Problematic code:

```
eval(data)
```

How to fix it:

Sanitize and validate all user-supplied data before passing to database queries, OS commands, or eval-like functions. Use parameterized queries for SQL.

18. Unsanitized User Input Reaches cursor.execute()

HIGH **Fix Before Launch**

File: app (1).py | Line: 91 | Ref: CWE-20 | Detected by: Agent C

In function 'reset_password', user-controlled variable(s) ['new_pass', 'user_id'] flow directly into 'cursor.execute()' without apparent sanitization.

Problematic code:

```
cursor.execute("UPDATE users SET password = '" + new_pass + "' WHERE id = " + user_id)
```

How to fix it:

Sanitize and validate all user-supplied data before passing to database queries, OS commands, or eval-like functions. Use parameterized queries for SQL.

19. Vulnerable Dependency: flask

HIGH **Fix Before Launch**

File: requirements.txt | Line: 1 | Ref: CWE-1035 | Detected by: Agent E

flask==0.12.3 may be affected by CVE-2018-1000656: Flask before 0.12.5 is vulnerable to Denial of Service via malicious JSON data.

Problematic code:

```
flask==0.12.3
```

How to fix it:

Upgrade flask to the latest patched version. Run: pip install --upgrade flask

20. Vulnerable Dependency: sqlalchemy

HIGH **Fix Before Launch**

File: requirements.txt | Line: 6 | Ref: CWE-1035 | Detected by: Agent E

sqlalchemy==1.3.0 may be affected by CVE-2019-7164: SQLAlchemy SQL injection via order_by() parameter.

Problematic code:

```
sqlalchemy==1.3.0
```

How to fix it:

Upgrade sqlalchemy to the latest patched version. Run: pip install --upgrade sqlalchemy

21. Vulnerable Dependency: werkzeug

HIGH **Fix Before Launch**

File: requirements.txt | Line: 8 | Ref: CWE-1035 | Detected by: Agent E

werkzeug==1.0.0 may be affected by CVE-2023-25577: Werkzeug multipart data parsing DoS vulnerability.

Problematic code:

```
werkzeug==1.0.0
```

How to fix it:

Upgrade werkzeug to the latest patched version. Run: pip install --upgrade werkzeug

22. Vulnerable Dependency: numpy

HIGH **Fix Before Launch**

File: requirements.txt | Line: 9 | Ref: CWE-1035 | Detected by: Agent E

numpy may be affected by CVE-2019-6446: NumPy pickle deserialization vulnerability via np.load().

Problematic code:

```
numpy
```

How to fix it:

Upgrade numpy to the latest patched version. Run: pip install --upgrade numpy

23. Vulnerable Dependency: urllib3

HIGH **Fix Before Launch**

File: requirements.txt | Line: 10 | Ref: CWE-1035 | Detected by: Agent E

urllib3==1.25.0 may be affected by CVE-2021-33503: urllib3 ReDoS vulnerability in URL parsing.

Problematic code:

```
urllib3==1.25.0
```

How to fix it:

Upgrade urllib3 to the latest patched version. Run: pip install --upgrade urllib3

24. Vulnerable Dependency: flask

HIGH **Fix Before Launch**

File: requirements.txt | Line: 1 | Ref: CWE-1035 | Detected by: Agent E

flask==0.12.3 may be affected by CVE-2018-1000656: Flask before 0.12.5 is vulnerable to Denial of Service via malicious JSON data.

Problematic code:

```
flask==0.12.3
```

How to fix it:

Upgrade flask to the latest patched version. Run: pip install --upgrade flask

25. Vulnerable Dependency: sqlalchemy

HIGH **Fix Before Launch**

File: requirements.txt | Line: 6 | Ref: CWE-1035 | Detected by: Agent E

sqlalchemy==1.3.0 may be affected by CVE-2019-7164: SQLAlchemy SQL injection via order_by() parameter.

Problematic code:

```
sqlalchemy==1.3.0
```

How to fix it:

Upgrade sqlalchemy to the latest patched version. Run: pip install --upgrade sqlalchemy

26. Vulnerable Dependency: werkzeug

HIGH Fix Before Launch

File: requirements.txt | Line: 8 | Ref: CWE-1035 | Detected by: Agent E

werkzeug==1.0.0 may be affected by CVE-2023-25577: Werkzeug multipart data parsing DoS vulnerability.

Problematic code:

```
werkzeug==1.0.0
```

How to fix it:

Upgrade werkzeug to the latest patched version. Run: pip install --upgrade werkzeug

27. Vulnerable Dependency: numpy

HIGH Fix Before Launch

File: requirements.txt | Line: 9 | Ref: CWE-1035 | Detected by: Agent E

numpy may be affected by CVE-2019-6446: NumPy pickle deserialization vulnerability via np.load().

Problematic code:

```
numpy
```

How to fix it:

Upgrade numpy to the latest patched version. Run: pip install --upgrade numpy

28. Vulnerable Dependency: urllib3

HIGH Fix Before Launch

File: requirements.txt | Line: 10 | Ref: CWE-1035 | Detected by: Agent E

urllib3==1.25.0 may be affected by CVE-2021-33503: urllib3 ReDoS vulnerability in URL parsing.

Problematic code:

```
urllib3==1.25.0
```

How to fix it:

Upgrade urllib3 to the latest patched version. Run: pip install --upgrade urllib3

29. PostgreSQL Connection String with Credentials in Git History

HIGH Fix Before Launch

File: .git | Ref: CWE-312 | Detected by: Agent F

A potential secret was found in commit b1cd13ae (2026-02-24) by BM840. Even if deleted from current code, this secret remains accessible in git history to anyone with repo access.

Problematic code:

```
Commit: b1cd13ae | initial commit (r'postgres://[^:]+:[^@]+@',
```

How to fix it:

1. Rotate/revoke the exposed secret immediately. 2. Use 'git filter-repo' or BFG Repo Cleaner to purge from history. 3. Force-push the cleaned history. 4. Use environment variables for all secrets going forward.

30. OpenAI API Key in Git History

HIGH Fix Before Launch

File: .git | Ref: CWE-312 | Detected by: Agent F

A potential secret was found in commit b1cd13ae (2026-02-24) by BM840. Even if deleted from current code, this secret remains accessible in git history to anyone with repo access.

Problematic code:

```
Commit: b1cd13ae | initial commit "code_snippet": "API_KEY = \"sk-abc123xyz789hardcoded\"",
```

How to fix it:

1. Rotate/revoke the exposed secret immediately. 2. Use 'git filter-repo' or BFG Repo Cleaner to purge from history. 3. Force-push the cleaned history. 4. Use environment variables for all secrets going forward.

31. Password in Git History

HIGH Fix Before Launch

File: .git | Ref: CWE-312 | Detected by: Agent F

A potential secret was found in commit b1cd13ae (2026-02-24) by BM840. Even if deleted from current code, this secret remains accessible in git history to anyone with repo access.

Problematic code:

```
Commit: b1cd13ae | initial commit DB_PASSWORD = "admin123"
```

How to fix it:

1. Rotate/revoke the exposed secret immediately. 2. Use 'git filter-repo' or BFG Repo Cleaner to purge from history. 3. Force-push the cleaned history. 4. Use environment variables for all secrets going forward.

32. API Key in Git History

HIGH Fix Before Launch

File: .git | Ref: CWE-312 | Detected by: Agent F

A potential secret was found in commit b1cd13ae (2026-02-24) by BM840. Even if deleted from current code, this secret remains accessible in git history to anyone with repo access.

Problematic code:

```
Commit: b1cd13ae | initial commit API_KEY = "sk-abc123xyz789hardcoded"
```

How to fix it:

1. Rotate/revoke the exposed secret immediately. 2. Use 'git filter-repo' or BFG Repo Cleaner to purge from history. 3. Force-push the cleaned history. 4. Use environment variables for all secrets going forward.

33. Debug Mode Enabled

MEDIUM Fix Soon

File: app (1).py | Line: 16 | Ref: CWE-215 | Detected by: Agent A

Debug mode is enabled on line 16. In production, this exposes stack traces, internal state, and may enable the interactive debugger.

Problematic code:

```
DEBUG = True
```

How to fix it:

Disable debug mode in production. Use environment-based configuration (e.g., DEBUG = os.getenv('DEBUG', 'False') == 'True').

34. Debug Mode Enabled

MEDIUM Fix Soon

File: app (1).py | Line: 17 | Ref: CWE-215 | Detected by: Agent A

Debug mode is enabled on line 17. In production, this exposes stack traces, internal state, and may enable the interactive debugger.

Problematic code:

```
app.config['DEBUG'] = True
```

How to fix it:

Disable debug mode in production. Use environment-based configuration (e.g., DEBUG = os.getenv('DEBUG', 'False') == 'True').

35. Unauthenticated Route Handler

MEDIUM Fix Soon

File: app (1).py | Line: 30 | Ref: CWE-306 | Detected by: Agent B

Route function 'login' does not appear to verify user authentication before processing the request.

Problematic code:

```
def login(): username = request.form.get("username") password = request.form.get("password") #
Vulnerability: Plaintext password comparis
```

How to fix it:

Add authentication decorators or token validation before processing sensitive route logic.

36. Unauthenticated Route Handler

MEDIUM Fix Soon

File: app (1).py | Line: 63 | Ref: CWE-306 | Detected by: Agent B

Route function 'search' does not appear to verify user authentication before processing the request.

Problematic code:

```
def search(): term = request.args.get("q", "") db = get_db() cursor = db.cursor() # Vulnerability:
Another raw SQL injection quer
```

How to fix it:

Add authentication decorators or token validation before processing sensitive route logic.

37. Unauthenticated Route Handler

MEDIUM Fix Soon

File: app (1).py | Line: 76 | Ref: CWE-306 | Detected by: Agent B

Route function 'eval_endpoint' does not appear to verify user authentication before processing the request.

Problematic code:

```
def eval_endpoint(): data = request.json.get("expression", "") # Vulnerability: eval() on user input
result = eval(data) return jsonif
```

How to fix it:

Add authentication decorators or token validation before processing sensitive route logic.

38. Unauthenticated Route Handler

MEDIUM Fix Soon

File: app (1).py | Line: 84 | Ref: CWE-306 | Detected by: Agent B

Route function 'reset_password' does not appear to verify user authentication before processing the request.

Problematic code:

```
def reset_password(): user_id = request.form.get("user_id") new_pass = request.form.get("password")
db = get_db() cursor = db.cursor()
```

How to fix it:

Add authentication decorators or token validation before processing sensitive route logic.

39. Vulnerable Dependency: requests

MEDIUM Fix Soon

File: requirements.txt | Line: 3 | Ref: CWE-1035 | Detected by: Agent E

requests==2.18.0 may be affected by CVE-2018-18074: Requests library sends HTTP Authorization header to redirected hosts.

Problematic code:

```
requests==2.18.0
```

How to fix it:

Upgrade requests to the latest patched version. Run: pip install --upgrade requests

40. Vulnerable Dependency: jinja2

MEDIUM Fix Soon

File: requirements.txt | Line: 7 | Ref: CWE-1035 | Detected by: Agent E

jinja2==2.10.0 may be affected by CVE-2020-28493: Jinja2 ReDoS vulnerability in urlize filter.

Problematic code:

```
jinja2==2.10.0
```

How to fix it:

Upgrade jinja2 to the latest patched version. Run: pip install --upgrade jinja2

41. Vulnerable Dependency: requests

MEDIUM Fix Soon

File: requirements.txt | Line: 3 | Ref: CWE-1035 | Detected by: Agent E

requests==2.18.0 may be affected by CVE-2018-18074: Requests library sends HTTP Authorization header to redirected hosts.

Problematic code:

```
requests==2.18.0
```

How to fix it:

Upgrade requests to the latest patched version. Run: pip install --upgrade requests

42. Vulnerable Dependency: jinja2

MEDIUM **Fix Soon**

File: requirements.txt | Line: 7 | Ref: CWE-1035 | Detected by: Agent E

jinja2==2.10.0 may be affected by CVE-2020-28493: Jinja2 ReDoS vulnerability in urlize filter.

Problematic code:

```
jinja2==2.10.0
```

How to fix it:

Upgrade jinja2 to the latest patched version. Run: pip install --upgrade jinja2

43. Unpinned Dependency: numpy

LOW **Fix When Possible**

File: requirements.txt | Line: 9 | Ref: CWE-1104 | Detected by: Agent E

numpy has no version pinned. Unpinned dependencies can introduce breaking changes or vulnerabilities automatically.

Problematic code:

```
numpy
```

How to fix it:

Pin to a specific version: numpy==<version>. Use 'pip freeze' to get current versions.

44. Unpinned Dependency: numpy

LOW **Fix When Possible**

File: requirements.txt | Line: 9 | Ref: CWE-1104 | Detected by: Agent E

numpy has no version pinned. Unpinned dependencies can introduce breaking changes or vulnerabilities automatically.

Problematic code:

```
numpy
```

How to fix it:

Pin to a specific version: numpy==<version>. Use 'pip freeze' to get current versions.

Report Complete

This report was generated by SentinelAI on 26 Feb 2026 at 23:02.

Total issues found: **44** | Risk Score: **272** | Risk Level: **CRITICAL RISK**