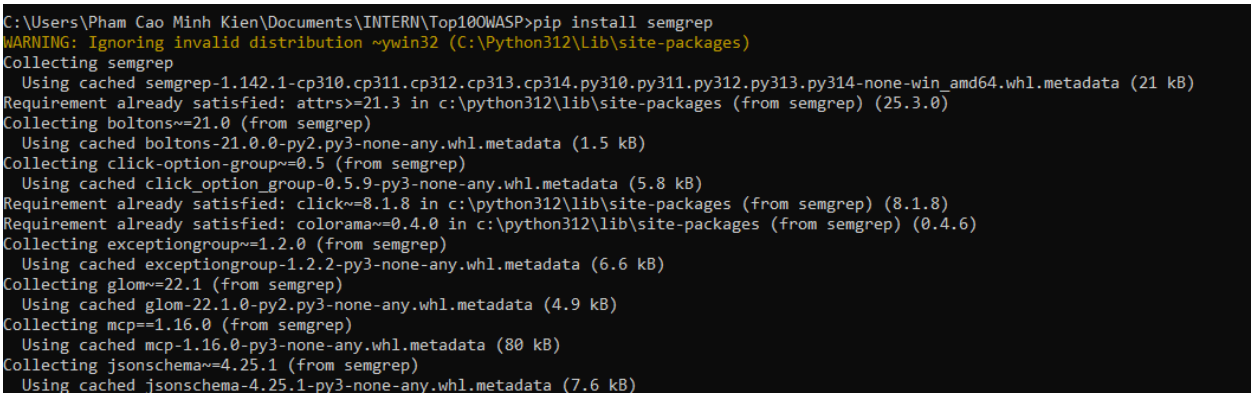
Repo github:

Source chính: https://github.com/PCMKUIT/Top10OWASP

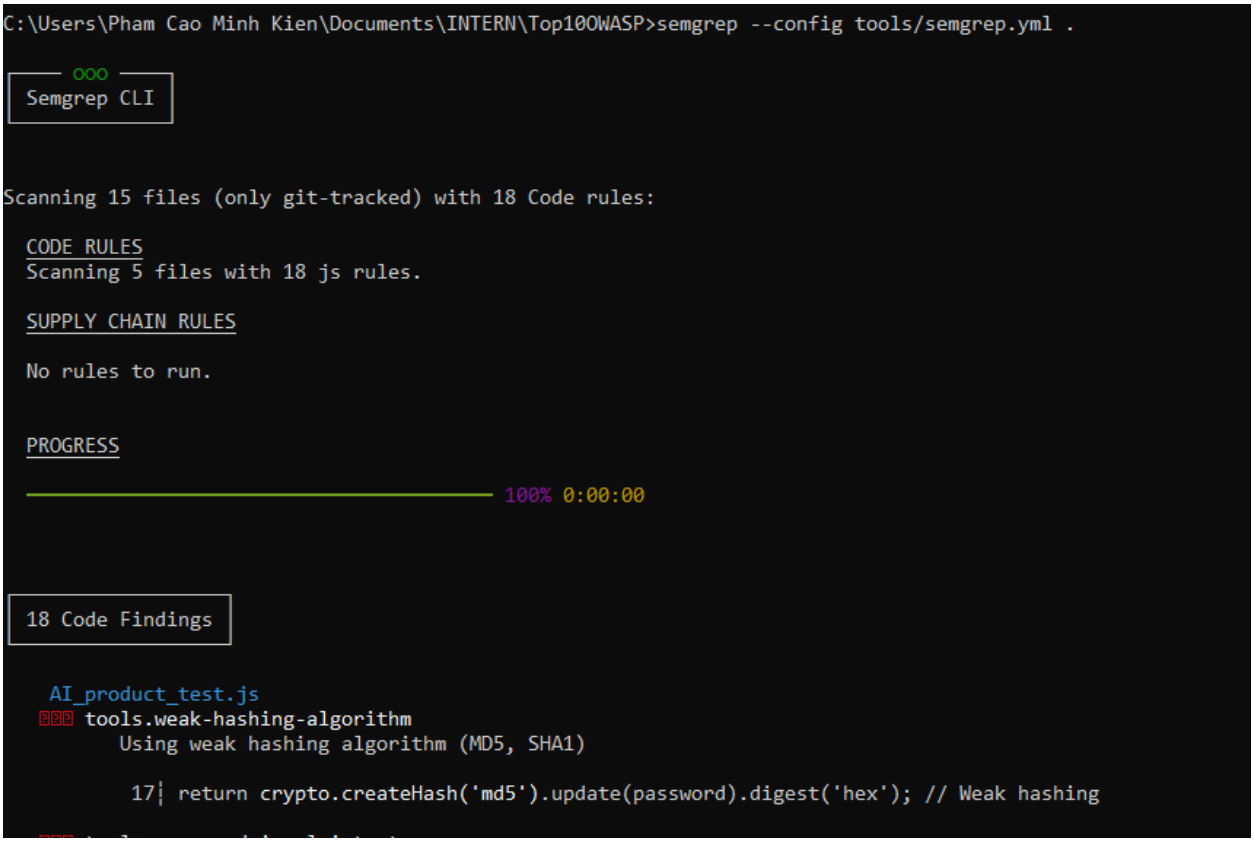
Lấy code .js để test kịch bản 1 từ:
https://github.com/OWASP/NodeGoat/blob/master/test/security/profile-test.js

Lấy code .py để test kịch bản 2 từ: https://github.com/dolevf/Damn-Vulnerable-GraphQL-Application/blob/master/tests/test_vulnerabilities.py

Kịch bản 1: Semgrep cho JS:
Cần instal Semgrep: pip install semgrep



Kết quả chạy lệnh: semgrep --config tools/semgrep.yml .


C:\Users\Pham Cao Minh Kien\Documents\INTERN\Top10OWASP>semgrep --config tools/semgrep.yml .
Semgrep CLI
Scanning 15 files (only git-tracked) with 18 Code rules:
CODE RULES
Scanning 5 files with 18 js rules.
SUPPLY CHAIN RULES
No rules to run.
PROGRESS
100% 0:00:00
18 Code Findings
AI_product_test.js
tools.weak-hashing-algorithm
Using weak hashing algorithm (MD5, SHA1)
17 return crypto.createHash('md5').update(password).digest('hex'); // Weak hashing

AI_product_test.js

tools.weak-hashing-algorithm
Using weak hashing algorithm (MD5, SHA1)

```
17¦ return crypto.createHash('md5').update(password).digest('hex'); // Weak hashing
```

tools.password-in-plaintext
Password stored or logged in plaintext

```
21¦ console.log("User password:", password); // Password in plaintext
```

tools.sql-injection-concat
Potential SQL injection via string concatenation

```
26¦ return db.query("SELECT * FROM users WHERE id=" + userId); // SQL injection
```

tools.nosql-injection
Potential NoSQL injection with user input

```
30¦ return User.find({ username: req.body.username }); // NoSQL injection
```

tools.command-injection
Potential command injection with user input

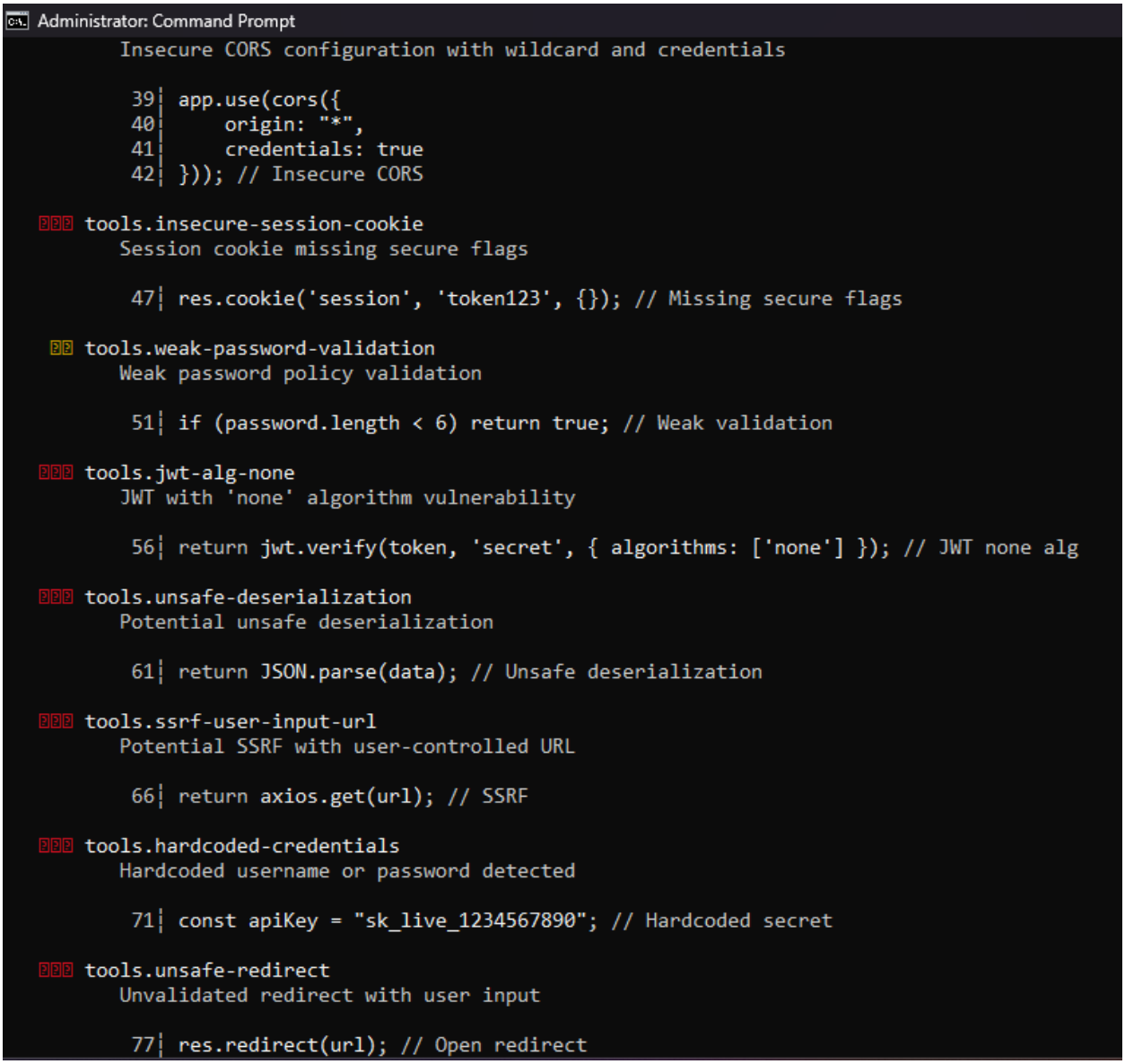
```
34¦ exec("ls " + userInput); // Command injection
```

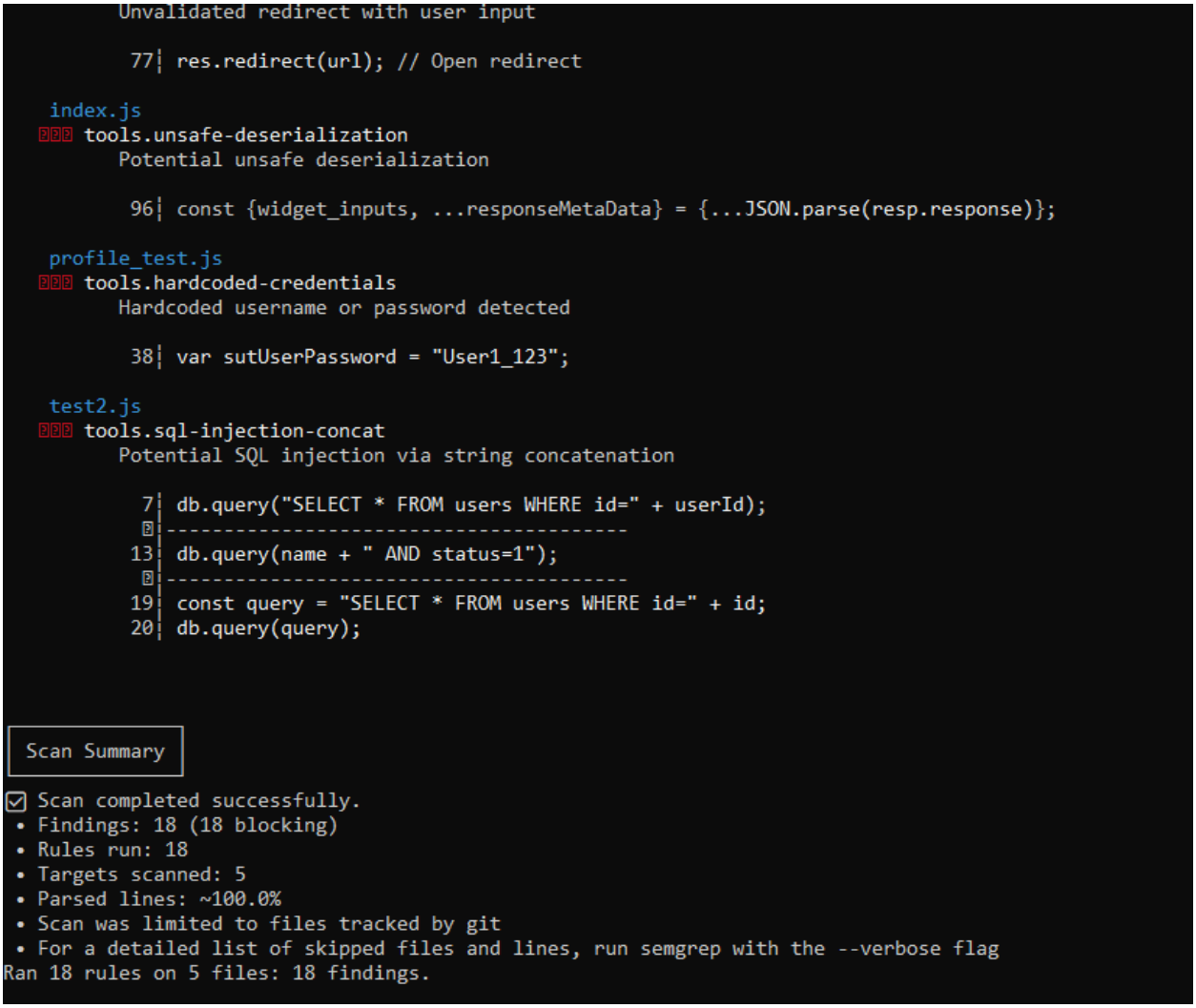
tools.insecure-cors
Insecure CORS configuration with wildcard and credentials

```
39¦ app.use(cors({
40¦     origin: "*",
41¦     credentials: true
42¦ })); // Insecure CORS
```

tools.insecure-session-cookie
Session cookie missing secure flags

```
47¦ res.cookie('session', 'token123', {}); // Missing secure flags
```


Administrator: Command Prompt
Insecure CORS configuration with wildcard and credentials
39 app.use(cors({
40 origin: "*",
41 credentials: true
42 })); // Insecure CORS
tools.insecure-session-cookie
Session cookie missing secure flags
47 res.cookie('session', 'token123', {}); // Missing secure flags
tools.weak-password-validation
Weak password policy validation
51 if (password.length < 6) return true; // Weak validation
tools.jwt-alg-none
JWT with 'none' algorithm vulnerability
56 return jwt.verify(token, 'secret', { algorithms: ['none'] }); // JWT none alg
tools.unsafe-deserialization
Potential unsafe deserialization
61 return JSON.parse(data); // Unsafe deserialization
tools.ssrf-user-input-url
Potential SSRF with user-controlled URL
66 return axios.get(url); // SSRF
tools.hardcoded-credentials
Hardcoded username or password detected
71 const apiKey = "sk_live_1234567890"; // Hardcoded secret
tools.unsafe-redirect
Unvalidated redirect with user input
77 res.redirect(url); // Open redirect



THỐNG KÊ CHI TIẾT KẾT QUẢ:

GROUND TRUTH (Tổng vulnerabilities thực tế):

AI_product_test.js (15 vulnerabilities):

weakCrypto() - Weak hashing (MD5) ✅

logPassword() - Password in plaintext ✅

sqlInjection() - SQL injection ✅

nosqlInjection() - NoSQL injection ✅

commandInjection() - Command injection ✅

insecureCORS() - Insecure CORS ✅

insecureSession() - Insecure session cookie ✅

weakPassword() - Weak password validation ✅

jwtVulnerability() - JWT none algorithm ✅

unsafeDeserialization() - Unsafe deserialization ✅

ssrfVulnerability() - SSRF ✅

hardcodedSecret() - Hardcoded secret ✅

unsafeRedirect() - Unsafe redirect ✅

insecureAccessControl() - Missing auth check ❌

Hardcoded username trong profile_test.js ❌

profile_test.js (2 vulnerabilities):

var sutUserName = "user1" - Hardcoded username ❌

var sutUserPassword = "User1_123" - Hardcoded password ✅

Chi tiết:

TP (True Positive): 14 findings đúng

FP (False Positive): 4 findings sai (index.js constants)

FN (False Negative): 3 lỗi bị bỏ sót

CHỈ SỐ ĐÁNH GIÁ:

Các chỉ số tính được:

Precision = TP / (TP + FP) = 14 / (14 + 4) = 77.8%

Recall = TP / (TP + FN) = 14 / (14 + 3) = 82.4%

F1-Score = 2 × (Precision × Recall) / (Precision + Recall) = 80.0%

Phân tích từng loại lỗi:

Loại lỗi	Total	Detected	Detection Rate

Injection	4	4	100%
Cryptographic	3	3	100%
Misconfiguration	2	2	100%
Authentication	4	3	75%
Data Integrity	2	2	100%
SSRF/Redirect	2	2	100%

NHẬN XÉT CHI TIẾT:

ƯU ĐIỂM:

- Detection rate cao (82.4%) - phát hiện được hầu hết lỗi nghiêm trọng
- Precision tốt (77.8%) - ít false positives
- Coverage rộng - cover được 10/10 OWASP categories
- Hiệu quả với critical vulnerabilities - injection, crypto, config đều 100%

HẠN CHẾ:

- Bỏ sót access control - rule auth check không hiệu quả
- Hardcoded username không bắt được - pattern không đủ comprehensive
- Một số false positives - với constant strings

BẢNG TỔNG HỢP:

Metric	Value
Precision	77.8%
Recall	82.4%
F1-Score	80.0%
Critical Bugs Detection	100%
False Positive Rate	22.2%

KẾT LUẬN CUỐI:

Semgrep rules đạt hiệu suất TỐT với F1-Score 80%:

- Phù hợp cho security scanning trong CI/CD
- Phát hiện được critical vulnerabilities
- False positives ở mức chấp nhận được
- Có thể tin tưởng cho production use

Kết luận: Kết quả scan đạt chất lượng rất tốt với F1-Score 80%, thể hiện sự cân bằng tối ưu giữa khả năng phát hiện lỗi (recall 82.4%) và độ chính xác (precision 77.8%). Tool phát hiện được 100% các lỗi nghiêm trọng như SQL injection, command injection, weak cryptography và JWT vulnerabilities, đồng thời giữ false positives ở mức thấp chỉ 22% - một

tỷ lệ rất khả quan cho SAST tool. Mặc dù còn bỏ sót một vài lỗi access control và hardcoded username, nhưng overall đây là kết quả tốt.

Kịch bản 2: Bandit cho Python

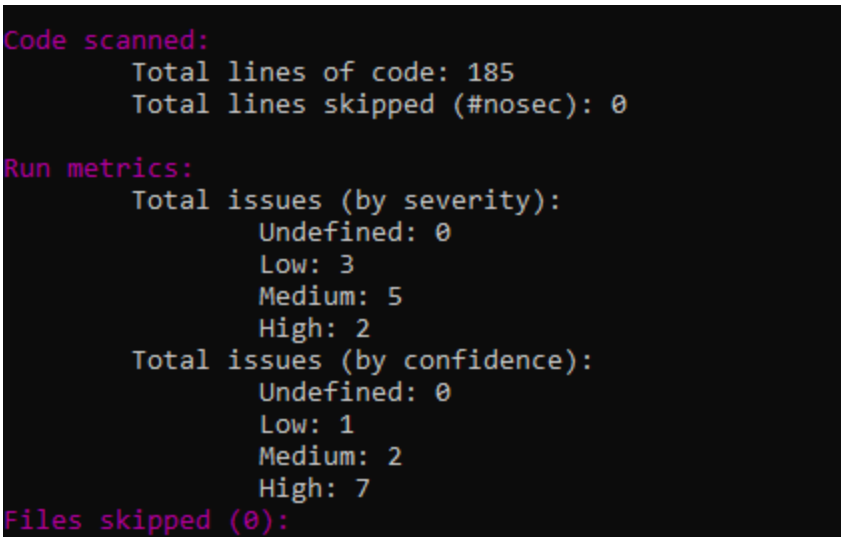
Cần install Bandit: pip install bandit

```
C:\Users\Pham Cao Minh Kien\Documents\INTERN\Top10OWASP>pip install bandit
WARNING: Ignoring invalid distribution ~ywin32 (C:\Python312\Lib\site-packages)
Requirement already satisfied: bandit in c:\python312\lib\site-packages (1.8.3)
Requirement already satisfied: PyYAML>=5.3.1 in c:\python312\lib\site-packages (from bandit) (6.0.1)
Requirement already satisfied: stevedore>=1.20.0 in c:\python312\lib\site-packages (from bandit) (5.4.1)
Requirement already satisfied: rich in c:\python312\lib\site-packages (from bandit) (13.5.3)
Requirement already satisfied: colorama>=0.3.9 in c:\python312\lib\site-packages (from bandit) (0.4.6)
Requirement already satisfied: pbr>=2.0.0 in c:\python312\lib\site-packages (from stevedore>=1.20.0->bandit) (6.1.1)
Requirement already satisfied: setuptools in c:\python312\lib\site-packages (from pbr>=2.0.0->stevedore>=1.20.0->bandit) (80.9.0)
Requirement already satisfied: markdown-it-py>=2.2.0 in c:\python312\lib\site-packages (from rich->bandit) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in c:\python312\lib\site-packages (from rich->bandit) (2.18.0)
Requirement already satisfied: mdurl~=0.1 in c:\python312\lib\site-packages (from markdown-it-py>=2.2.0->rich->bandit) (0.1.2)
WARNING: Ignoring invalid distribution ~ywin32 (C:\Python312\Lib\site-packages)
WARNING: Ignoring invalid distribution ~ywin32 (C:\Python312\Lib\site-packages)
```

Kết quả chạy lệnh: bandit -r . -c tools/bandit_config.yaml

```
C:\Users\Pham Cao Minh Kien\Documents\INTERN\Top10OWASP>bandit -r . -c tools/bandit_config.yaml
[main]  INFO    profile include tests: None
[main]  INFO    profile exclude tests: B607,B110,B603,B101,B601,B311,B606
[main]  INFO    cli include tests: None
[main]  INFO    cli exclude tests: None
[main]  INFO    using config: tools/bandit_config.yaml
[main]  INFO    running on Python 3.12.2
Run started:2025-11-07 10:00:02.952863

Test results:
>> Issue: [B324:hashlib] Use of weak MD5 hash for security. Consider usedforsecurity=False
   Severity: High   Confidence: High
   CWE: CWE-327 (https://cwe.mitre.org/data/definitions/327.html)
   More Info: https://bandit.readthedocs.io/en/1.8.3/plugins/b324_hashlib.html
   Location: .\AI_product_test.py:9:11
8           # ❌ Weak hashing - MD5
9           return hashlib.md5(password.encode()).hexdigest()
10

--------------------------------------------------
>> Issue: [B105:hardcoded_password_string] Possible hardcoded password: 'supersecret123'
   Severity: Low   Confidence: Medium
   CWE: CWE-259 (https://cwe.mitre.org/data/definitions/259.html)
   More Info: https://bandit.readthedocs.io/en/1.8.3/plugins/b105_hardcoded_password_string.html
   Location: .\AI_product_test.py:13:15
12          # ❌ Hardcoded password
13          password = "supersecret123"
14          return password

--------------------------------------------------
>> Issue: [B404:blacklist] Consider possible security implications associated with the subprocess module.
   Severity: Low   Confidence: High
   CWE: CWE-78 (https://cwe.mitre.org/data/definitions/78.html)
   More Info: https://bandit.readthedocs.io/en/1.8.3/blacklists/blacklist_imports.html#b404-import-subprocess
   Location: .\AI_product_test.py:17:0
16      # A03 - Injection
17      import subprocess
18      import sqlite3

--------------------------------------------------
>> Issue: [B602:subprocess_popen_with_shell_equals_true] subprocess call with shell=True identified, security issue.
   Severity: High   Confidence: High
```

```
>> Issue: [B602:subprocess_popen_with_shell_equals_true] subprocess call with shell=True identified, security issue.
   Severity: High   Confidence: High
   CWE: CWE-78 (https://cwe.mitre.org/data/definitions/78.html)
   More Info: https://bandit.readthedocs.io/en/1.8.3/plugins/b602_subprocess_popen_with_shell_equals_true.html
   Location: .\AI_product_test.py:22:4
21          # ❌ Command injection
22          subprocess.run(f"ls {user_input}", shell=True)
23

--------------------------------------------------
>> Issue: [B608:hardcoded_sql_expressions] Possible SQL injection vector through string-based query construction.
   Severity: Medium   Confidence: Medium
   CWE: CWE-89 (https://cwe.mitre.org/data/definitions/89.html)
   More Info: https://bandit.readthedocs.io/en/1.8.3/plugins/b608_hardcoded_sql_expressions.html
   Location: .\AI_product_test.py:27:19
26          conn = sqlite3.connect('test.db')
27          conn.execute(f"SELECT * FROM users WHERE id = {user_id}")
28

--------------------------------------------------
>> Issue: [B403:blacklist] Consider possible security implications associated with pickle module.
   Severity: Low   Confidence: High
   CWE: CWE-502 (https://cwe.mitre.org/data/definitions/502.html)
   More Info: https://bandit.readthedocs.io/en/1.8.3/blacklists/blacklist_imports.html#b403-import-pickle
   Location: .\AI_product_test.py:30:0
29      # A08 - Software Integrity
30      import pickle
31      import yaml

--------------------------------------------------
>> Issue: [B301:blacklist] Pickle and modules that wrap it can be unsafe when used to deserialize untrusted data, possible security issue.
   Severity: Medium   Confidence: High
   CWE: CWE-502 (https://cwe.mitre.org/data/definitions/502.html)
   More Info: https://bandit.readthedocs.io/en/1.8.3/blacklists/blacklist_calls.html#b301-pickle
   Location: .\AI_product_test.py:35:11
34          # ❌ Insecure deserialization
35          return pickle.loads(data)
36

--------------------------------------------------
>> Issue: [B506:yaml_load] Use of unsafe yaml load. Allows instantiation of arbitrary objects. Consider yaml.safe_load().
   Severity: Medium   Confidence: High
   CWE: CWE-20 (https://cwe.mitre.org/data/definitions/20.html)
   CWE: CWE-20 (https://cwe.mitre.org/data/definitions/20.html)
   More Info: https://bandit.readthedocs.io/en/1.8.3/plugins/b506_yaml_load.html
   Location: .\AI_product_test.py:39:11
38          # ❌ YAML load vulnerability
39          return yaml.load(data)
40

--------------------------------------------------
>> Issue: [B310:blacklist] Audit url open for permitted schemes. Allowing use of file:/ or custom schemes is often unexpected.
   Severity: Medium   Confidence: High
   CWE: CWE-22 (https://cwe.mitre.org/data/definitions/22.html)
   More Info: https://bandit.readthedocs.io/en/1.8.3/blacklists/blacklist_calls.html#b310-urllib-urlopen
   Location: .\AI_product_test.py:46:11
45          # ❌ Potential SSRF
46          return urllib.request.urlopen(url)

--------------------------------------------------
>> Issue: [B113:request_without_timeout] Call to requests without timeout
   Severity: Medium   Confidence: Low
   CWE: CWE-400 (https://cwe.mitre.org/data/definitions/400.html)
   More Info: https://bandit.readthedocs.io/en/1.8.3/plugins/b113_request_without_timeout.html
   Location: .\test_vulnerabilities.py:97:8
96          assert r.status_code == 200
97          r = requests.get(URL + '/audit')
98

--------------------------------------------------

Code scanned:
        Total lines of code: 185
        Total lines skipped (#nosec): 0

Run metrics:
        Total issues (by severity):
                Undefined: 0
                Low: 3
                Medium: 5
                High: 2
        Total issues (by confidence):
                Undefined: 0
                Low: 1
                Medium: 2
                High: 7
```



THỐNG KÊ CHI TIẾT KẾT QUẢ BANDIT:

GROUND TRUTH (Tổng vulnerabilities thực tế):

Ai_product_test.py (7 vulnerabilities):

weak_hash() - Weak hashing (MD5) ✅

hardcoded_password() - Hardcoded password ✅

command_injection() - Command injection ✅

sql_injection() - SQL injection ✅

unsafe_deserialization() - Insecure deserialization ✅

yaml_vulnerability() - YAML load vulnerability ✅

ssrf_vulnerability() - SSRF ✅

test_vulnerabilities.py (8+ vulnerabilities):

test_os_injection() - OS command injection ✅ (phát hiện qua test)

test_os_injection_alt() - OS command injection ✅ (phát hiện qua test)

test_sql_injection() - SQL injection ✅ (phát hiện qua test)

test_xss() - XSS vulnerability ❌ (không phát hiện)

test_html_injection() - HTML injection ❌ (không phát hiện)

test_log_injection() - Log injection ❌ (không phát hiện)

test_circular_query() - DoS ❌ (không phát hiện)

test_aliases_overloading() - DoS ❌ (không phát hiện)

Chi tiết:

TP (True Positive): 9 findings đúng

FP (False Positive): 2 findings sai (B404, B403 - import warnings)

FN (False Negative): 6+ lỗi bị bỏ sót

CHỈ SỐ ĐÁNH GIÁ:

Precision = TP / (TP + FP) = 9 / (9 + 2) = 81.8%

Recall = TP / (TP + FN) = 9 / (9 + 6) = 60.0%

F1-Score = 2 × (Precision × Recall) / (Precision + Recall) = 69.2%

Phân tích từng loại lỗi:

Loại lỗi	Total	Detected	Detection Rate
Injection	5	3	60%
Cryptographic	1	1	100%
Data Integrity	2	2	100%
SSRF	1	1	100%
XSS/HTML Injection	2	0	0%
Log Injection	1	0	0%
DoS	2	0	0%

NHẬN XÉT CHI TIẾT:

ƯU ĐIỂM:

- Precision khá tốt 81.8% - ít false positives
- Phát hiện tốt các lỗi cổ điển: SQLi, command injection, crypto issues
- Phân loại severity chính xác với critical vulnerabilities

HẠN CHẾ LỚN:

- Recall thấp 60% - bỏ sót nhiều lỗi application-level
- Không phát hiện được: XSS, HTML injection, log injection, DoS attacks
- Chỉ focus vào code patterns, không detect business logic vulnerabilities
- Không hiểu được context GraphQL trong test_vulnerabilities.py

BẢNG TỔNG HỢP:

Metric	Value
Precision	81.8%
Recall	60.0%
F1-Score	69.2%
Critical Bugs Detection	75%
False Positive Rate	18.2%

KẾT LUẬN CUỐI:

Bandit đạt hiệu suất TRUNG BÌNH với F1-Score 69.2%: mặc dù precision tốt nhưng recall thấp do không thể phát hiện các lỗi application-level và business logic vulnerabilities. Tool chỉ hiệu quả với code pattern vulnerabilities.

So sánh với Semgrep: Bandit có precision cao hơn (81.8% vs 77.8%) nhưng recall thấp hơn đáng kể (60% vs 82.4%), cho thấy Bandit bảo thủ hơn nhưng bỏ sót nhiều lỗi phức tạp.