FEATURE IMPLEMENTATION CHECKLIST

Smart Comparison: Required vs Implemented

Generated: November 24, 2025

FEATURE IMPLEMENTATION MATRIX

Feature	Category	Required	Implemented	Status	Files
Recursively parse folders	Core Infrastruc	YES	YES	100%	7
Detect project language	Core Infrastruc	YES	YES	100%	7
Build AST	Core Infrastruc	YES	YES	100%	7
Extract functions/classes	Core Infrastruc	YES	YES	100%	7
eval/exec/system detection	Security Detect	YES	YES	100%	6
OWASP/CWE mapping	Security Detect	YES	YES	100%	6
Multi-language support	Security Detect	YES	YES	100%	6
Track user input	Data Flow Analy	YES	YES	100%	9
Propagation mapping	Data Flow Analy	YES	YES	100%	9
Taint analysis	Data Flow Analy	YES	YES	100%	9
Sanitization check	Data Flow Analy	YES	YES	100%	9
API keys	Security Detect	YES	YES	100%	4
JWT secrets	Security Detect	YES	YES	100%	4
Database credentials	Security Detect	YES	YES	100%	4
Base64 payloads	Security Detect	YES	YES	100%	4
Weak hashing	Security Checks	YES	YES	100%	6
Weak encryption	Security Checks	YES	YES	100%	6
Predictable random	Security Checks	YES	YES	100%	6
JWT issues	Security Checks	YES	YES	100%	6
Missing validation	Input Security	YES	YES	100%	6
Type checking	Input Security	YES	YES	100%	6
Boundary checks	Input Security	YES	YES	100%	6
Client-side only	Input Security	YES	YES	100%	6
Weak tokens	Security Checks	YES	YES	100%	6
Session management	Security Checks	YES	YES	100%	6
Cookie security	Security Checks	YES	YES	100%	6
Authorization checks	Security Checks	YES	NO	0%	-
Privilege escalation	Security Checks	YES	NO	0%	-
IDOR	Security Checks	YES	NO	0%	-
Sensitive data logging	Security Checks	YES	PARTIAL	30%	-

IMPLEMENTATION EVIDENCE

Authentication & Session

File	Path
comprehensive_analyzer.py	d:\project\comprehensive_analyzer.py
project_documentation_generator.py	d:\project\project_documentation_generator.py
test_all_functions.py	d:\project\test_all_functions.py
input_processor.py	d:\project\core\input_processor.py
authentication_checker.py	d:\project\security_checks\authentication_checker.py

Code Quality Enhancer

File	Path
comprehensive_analyzer.py	d:\project\comprehensive_analyzer.py
project_documentation_generator.py	d:\project\project_documentation_generator.py
project_documentation_generator.py	d:\project\project_documentation_generator.py
test_all_functions.py	d:\project\test_all_functions.py
quality_analyzer.py	d:\project\analyzers\quality_analyzer.py

Cryptography Misuse Detector

File	Path
comprehensive_analyzer.py	d:\project\comprehensive_analyzer.py
project_documentation_generator.py	d:\project\project_documentation_generator.py
test_all_functions.py	d:\project\test_all_functions.py
input_processor.py	d:\project\core\input_processor.py
cryptography_checker.py	d:\project\security_checks\cryptography_checker.py

Dangerous Functions Detector

File	Path
comprehensive_analyzer.py	d:\project\comprehensive_analyzer.py
project_documentation_generator.py	d:\project\project_documentation_generator.py

test_all_functions.py	d:\project\test_all_functions.py
input_processor.py	d:\project\core\input_processor.py
pdf_report_generator.py	d:\project\reports\pdf_report_generator.py

Data Flow Analysis

File	Path
comprehensive_analyzer.py	d:\project\comprehensive_analyzer.py
project_documentation_generator.py	d:\project\project_documentation_generator.py
test_all_functions.py	d:\project\test_all_functions.py
analyzer.py	d:\project\core\analyzer.py
input_processor.py	d:\project\core\input_processor.py

Dependency Vulnerability Scanner

File	Path
comprehensive_analyzer.py	d:\project\comprehensive_analyzer.py
project_documentation_generator.py	d:\project\project_documentation_generator.py
project_documentation_generator.py	d:\project\project_documentation_generator.py
project_documentation_generator.py	d:\project\project_documentation_generator.py
scan_vulnerabilities.py	d:\project\scan_vulnerabilities.py

Framework Misconfiguration

File	Path
project_documentation_generator.py	d:\project\project_documentation_generator.py
base_checker.py	d:\project\security_checks\base_checker.py
__init__.py	d:\project\security_checks\__init__.py
dotnet_frameworks.py	d:\project\security_checks\framework_checks\dotnet_framework
javascript_frameworks.py	d:\project\security_checks\framework_checks\javascript_frame

Hardcoded Secrets Detector

File	Path
comprehensive_analyzer.py	d:\project\comprehensive_analyzer.py

project_documentation_generator.py	d:\project\project_documentation_generator.py
test_all_functions.py	d:\project\test_all_functions.py
input_processor.py	d:\project\core\input_processor.py

Input Processing Module

File	Path
comprehensive_analyzer.py	d:\project\comprehensive_analyzer.py
project_documentation_generator.py	d:\project\project_documentation_generator.py
scan_vulnerabilities.py	d:\project\scan_vulnerabilities.py
test_all_functions.py	d:\project\test_all_functions.py
input_processor.py	d:\project\core\input_processor.py

Input Validation & Sanitization

File	Path
comprehensive_analyzer.py	d:\project\comprehensive_analyzer.py
project_documentation_generator.py	d:\project\project_documentation_generator.py
test_all_functions.py	d:\project\test_all_functions.py
input_processor.py	d:\project\core\input_processor.py
validation_checker.py	d:\project\security_checks\validation_checker.py

Security Anti-Patterns

File	Path
comprehensive_analyzer.py	d:\project\comprehensive_analyzer.py
project_documentation_generator.py	d:\project\project_documentation_generator.py
test_all_functions.py	d:\project\test_all_functions.py
antipattern_detector.py	d:\project\analyzers\antipattern_detector.py

Security Report Generator

File	Path
comprehensive_analyzer.py	d:\project\comprehensive_analyzer.py
project_documentation_generator.py	d:\project\project_documentation_generator.py

test_all_functions.py	d:\project\test_all_functions.py
analyzer.py	d:\project\core\analyzer.py
demo_antipattern_report.py	d:\project\demos\demo_antipattern_report.py

■ GOLDEN WORKFLOW INSIGHT

Advanced Project Management Insight:

The most effective security tool development follows the "Detection-First, Reporting-Second" principle. Your current implementation demonstrates strong detection capabilities (dangerous functions, secrets, taint analysis, vulnerability scanning) which are the foundation.

■ Vulnerability Scanner Architecture:

The project now includes a comprehensive dependency vulnerability scanner with:

- **Multi-Format Parsing:** Automatic detection and parsing of requirements.txt, package.json, pom.xml, build.gradle
- **Multi-Source API Integration:** OSV (Open Source Vulnerabilities), NVD (National Vulnerability Database), GitHub Security Advisories
- **Version-Aware Matching:** Precise CVE matching based on installed versions
- **Smart Rate Limiting:** Respectful API usage with configurable delays
- **Comprehensive Reporting:** Integrated into JSON/PDF outputs with severity grouping and exploit detection

■ Module Structure:

- modules/vulnerability_scanner/models.py - Data models (Dependency, Vulnerability, ScanResult)
- modules/vulnerability_scanner/parsers.py - Multi-format dependency file parsers
- modules/vulnerability_scanner/api_clients.py - OSV/NVD/GitHub API clients
- modules/vulnerability_scanner/scanner.py - Main scanning orchestration
- scan_vulnerabilities.py - Standalone CLI tool for dedicated scanning

Next-Level Optimization:

1. Implement incremental scanning (only analyze changed files)
2. Add caching layer for AST parsing and vulnerability results (10x speed improvement)
3. Create plugin architecture for custom security rules
4. Build CI/CD integration templates (GitHub Actions, GitLab CI)
5. Add machine learning for false positive reduction
6. Implement SLA-based vulnerability prioritization (EPSS scores)

Hidden Gem: Consider implementing a "Security Score Card" that tracks improvement over time. This gamifies security for development teams and shows measurable progress in reducing vulnerabilities across sprints. Track metrics like: vulnerabilities fixed per sprint, mean time to remediation (MTTR), and security debt reduction.