FEATURE IMPLEMENTATION CHECKLIST

Smart Comparison: Required vs Implemented

Generated: November 23, 2025

FEATURE IMPLEMENTATION MATRIX

Feature	Category	Required	Implemented	Status	Files
Recursively parse folders	Core Infrastruc	YES	YES	100%	6
Detect project language	Core Infrastruc	YES	YES	100%	6
Build AST	Core Infrastruc	YES	YES	100%	6
Extract functions/classes	Core Infrastruc	YES	YES	100%	6
eval/exec/system detection	Security Detect	YES	YES	100%	5
OWASP/CWE mapping	Security Detect	YES	YES	100%	5
Multi-language support	Security Detect	YES	YES	100%	5
Track user input	Data Flow Analy	YES	YES	100%	5
Propagation mapping	Data Flow Analy	YES	YES	100%	5
Taint analysis	Data Flow Analy	YES	YES	100%	5
Sanitization check	Data Flow Analy	YES	YES	100%	5
API keys	Security Detect	YES	YES	100%	2
JWT secrets	Security Detect	YES	YES	100%	2
Database credentials	Security Detect	YES	YES	100%	2
Base64 payloads	Security Detect	YES	YES	100%	2
Weak hashing	Security Checks	YES	YES	100%	4
Weak encryption	Security Checks	YES	YES	100%	4
Predictable random	Security Checks	YES	YES	100%	4
JWT issues	Security Checks	YES	YES	100%	4
Missing validation	Input Security	YES	YES	100%	5
Type checking	Input Security	YES	YES	100%	5
Boundary checks	Input Security	YES	YES	100%	5
Client-side only	Input Security	YES	YES	100%	5
Weak tokens	Security Checks	YES	NO	0%	-
Session management	Security Checks	YES	NO	0%	-
Cookie security	Security Checks	YES	PARTIAL	30%	-
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

Code Quality Enhancer

File	Path
enhanced_analysis.py	d:\project\enhanced_analysis.py
input processing.py	d:\project\input processing.py
pdf_report_generator.py	d:\project\pdf_report_generator.py
project_documentation_generator.py	d:\project\project_documentation_generator.py

Cryptography Misuse Detector

File	Path
input processing.py	d:\project\input processing.py
project_documentation_generator.py	d:\project\project_documentation_generator.py
test_vulnerable_sample.js	d:\project\test_vulnerable_sample.js
test_vulnerable_sample.py	d:\project\test_vulnerable_sample.py

Dangerous Functions Detector

File	Path
concept_map_python.py	d:\project\concept_map_python.py
enhanced_analysis.py	d:\project\enhanced_analysis.py
input processing.py	d:\project\input processing.py
pdf_report_generator.py	d:\project\pdf_report_generator.py
project_documentation_generator.py	d:\project\project_documentation_generator.py

Data Flow Analysis

File	Path
enhanced_analysis.py	d:\project\enhanced_analysis.py
input processing.py	d:\project\input processing.py
pdf_report_generator.py	d:\project\pdf_report_generator.py
project_documentation_generator.py	d:\project\project_documentation_generator.py

test_vulnerable_sample.py	d:\project\test_vulnerable_sample.py

Hardcoded Secrets Detector

File	Path
input processing.py	d:\project\input processing.py
project_documentation_generator.py	d:\project\project_documentation_generator.py

Input Processing Module

File	Path
concept_map_python.py	d:\project\concept_map_python.py
input processing.py	d:\project\input processing.py
pdf_report_generator.py	d:\project\pdf_report_generator.py
project_documentation_generator.py	d:\project\project_documentation_generator.py
test_vulnerable_sample.js	d:\project\test_vulnerable_sample.js

Input Validation & Sanitization

File	Path
concept_map_python.py	d:\project\concept_map_python.py
enhanced_analysis.py	d:\project\enhanced_analysis.py
input processing.py	d:\project\input processing.py
project_documentation_generator.py	d:\project\project_documentation_generator.py
validation_checker.py	d:\project\validation_checker.py

Security Report Generator

File	Path
input processing.py	d:\project\input processing.py
pdf_report_generator.py	d:\project\pdf_report_generator.py
project_documentation_generator.py	d:\project\project_documentation_generator.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) which are the foundation.

Next-Level Optimization:
1. Implement incremental scanning (only analyze changed files)
2. Add caching layer for AST parsing (10x speed improvement)
3. Create plugin architecture for custom rules
4. Build CI/CD integration templates
5. Add machine learning for false positive reduction

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.