SECURITY ANALYSIS REPORT

Comprehensive Code Security Assessment

Project: test_project

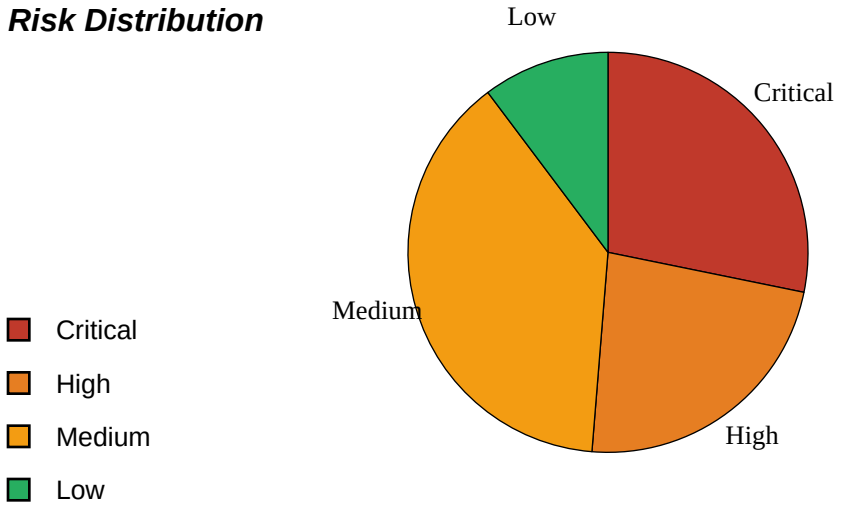
Generated: November 23, 2025 at 19:56:30

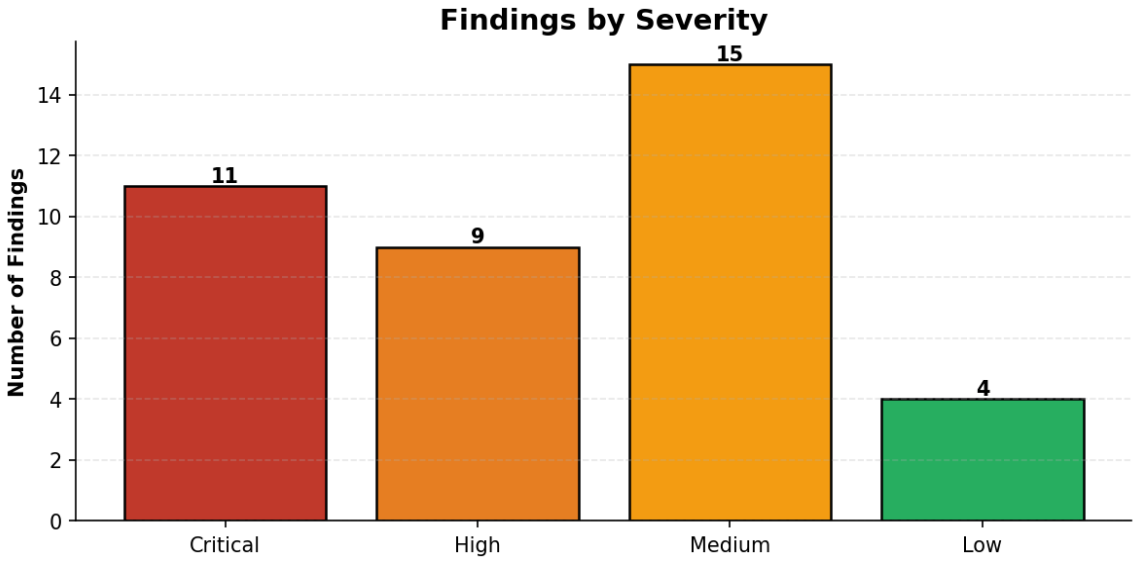
This report contains sensitive security information. Handle with care and distribute only to authorized personnel.

EXECUTIVE SUMMARY

Overall Risk Level	CRITICAL
Total Findings	39
Critical Issues	11
High Issues	9
Medium Issues	15
Low Issues	4
Languages Analyzed	python, json, env, php, javascript

Risk Distribution





FILE TREE HIERARCHY WITH RISK INDICATORS

Risk Level	Indicator	Description
CRITICAL	🔴	Immediate attention required - Critical vulnerabilities
HIGH	🟠	Review and fix soon - High risk issues
MEDIUM	🟡	Security concern - Should be addressed
LOW	🟢	Minor issue - Low priority
CLEAN	⚪	No security issues detected

■ project/	
■ ■■■ COMPLETE_INTEGRATION_GUIDE.md	
■ ■■■ CRYPTOGRAPHY_INTEGRATION.md	
■ ■■■ DOCUMENTATION_SUMMARY.md	
■ ■■■ FINAL_INTEGRATION_SUMMARY.md	
■ ■■■ Feature_Implementation_Checklist.pdf	
■ ■■■ PROJECT_STRUCTURE.md	
■ ■■■ Project_Milestones_And_Progress.pdf	
■ ■■■ QUICKSTART.md	
■ ■■■ README_SECURITY_ANALYZER.md	
■ ■■■ Reverser.py	
■ ■■■ SECURITY_CHECKS_INTEGRATION.md	
■ ■■■ VALIDATION_FEATURES.md	
■ ■■■ concept_map_python.py	
■ ■■■ demo_analyzer.py	
■ ■■■ enhanced_analysis.py	
■ ■■■ good_points.txt	
■ ■■■ input processing.py	
■ ■■■ pdf_report_generator.py	
■ ■■■ project_documentation_generator.py	
■ ■■■ project_tasks_list.txt	
■ ■■■ requirements.txt	
■ ■■■ security_analysis.json	
■ ■■■ security_analysis_report.pdf	
■ ■■■ test_vulnerable_sample.js	
■ ■■■ test_vulnerable_sample.py	
■ ■■■ validation_checker.py	
■ ■■■ security_checks/	

■ ■ ■■■ README.md	
■ ■ ■■■ __init__.py	
■ ■ ■■■ authentication_checker.py	
■ ■ ■■■ base_checker.py	
■ ■ ■■■ cryptography_checker.py	
■ ■ ■■■ framework_checks/	
■ ■ ■■■ __init__.py	
■ ■ ■■■ dotnet_frameworks.py	
■ ■ ■■■ java_frameworks.py	
■ ■ ■■■ javascript_frameworks.py	
■ ■ ■■■ python_frameworks.py	
■ ■■■ test_project/	
■ ■■■ app.py	
■ ■■■ script.js	
■ ■■■ server.php	
■ ■■■ utils.py	
■ ■■■ config/	
■ ■ ■■■ creds.env	
■ ■ ■■■ settings.json	
■ ■■■ src/	
■ ■■■ helper.py	
■ ■■■ module.js	

INTELLIGENT FINDINGS TABLE (DEDUPLICATED)

Category	Unique Findings	Status
Dangerous Functions	6	■■
Hardcoded Secrets	1	■■
Taint Sources	0	✓
File/Network Ops	1	■■

■ *DANGEROUS FUNCTIONS*

File	Function	Category	Lines	Count	Risk
script.js	Function	code_execution	6, 11	2	CRITICAL
server.php	eval	code_execution	25	1	CRITICAL
server.php	system	command_injectio	12	1	CRITICAL
server.php	shell_exec	command_injectio	13	1	CRITICAL
server.php	exec	command_injectio	13	1	CRITICAL
server.php	include	file_operations	3	1	MEDIUM

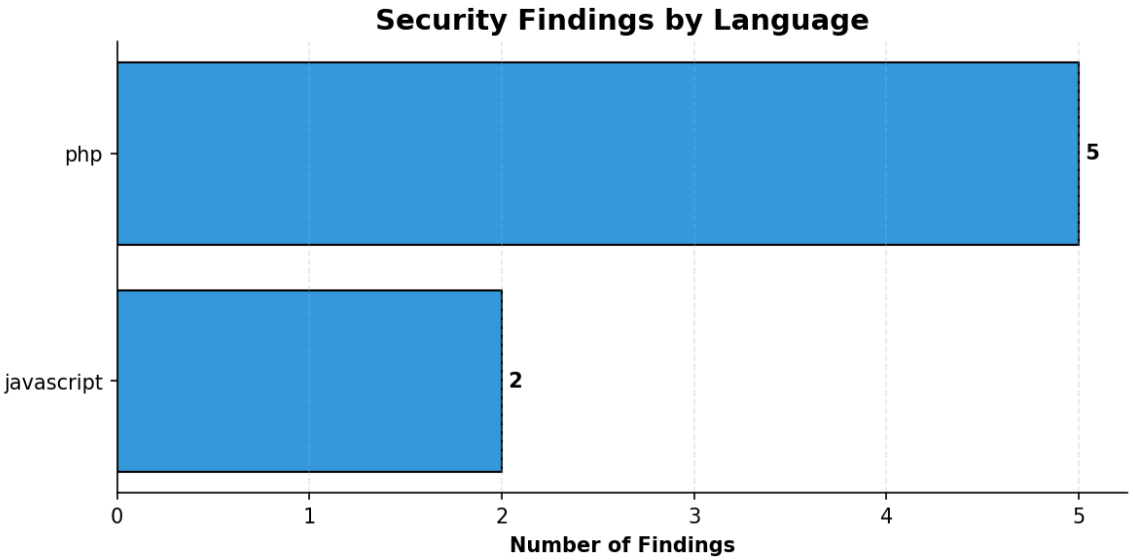
■ *HARDCODED SECRETS*

File	Secret Type	Lines	Count	Severity
server.php	Password	5	1	HIGH

■ *FILE & NETWORK OPERATIONS*

File	Operation Type	Lines	Count	Risk
script.js	File Write	14	1	MEDIUM

DANGEROUS FUNCTIONS DETECTED



JAVASCRIPT - 2 findings

Function	Category	File	Line
Function	code_execution	script.js	6
Function	code_execution	script.js	11

PHP - 5 findings

Function	Category	File	Line
eval	code_execution	server.php	25
system	command_injection	server.php	12
shell_exec	command_injection	server.php	13
exec	command_injection	server.php	13
include	file_operations	server.php	3

TAINT FLOW ANALYSIS

✓ No direct taint flows detected.

HARDCODED SECRETS & CREDENTIALS

Total secrets found: 1

Type	File	Line	Preview
password	server.php	5	PASSWORD = "php-password-9999"

FRAMEWORK-SPECIFIC SECURITY FINDINGS

Total Framework Findings: 1
- Critical: 1
- High: 0
- Medium: 0
- Low: 0

CRITICAL - 1 findings

#	Issue	File	Type	Line
1	Hardcoded credentials detected in Node.js code	script.js	exposure	4

Key Recommendations:
• Move credentials to environment variables using process.env

Frameworks Detected: Node.js

CRYPTOGRAPHY MISUSE ANALYSIS

✓ No cryptography misuse issues detected.

AUTHENTICATION & SESSION SECURITY

Total Authentication & Session Issues: 11
- Auth Bypass: 1
- Insecure Cookies: 0
- Weak Session Timeout: 0
- Missing Session Rotation: 2
- Missing MFA: 2
- Weak Password Policy: 6

AUTHENTICATION BYPASS - 1 findings

#	Issue	File	Line	Type
1	Potential authentication bypass detected	server.php	18	potential_auth_bypas

Recommendations:
• Review authentication logic for hardcoded bypasses or weak conditions

MISSING SESSION ROTATION - 2 findings

#	Issue	File	Line	Type
1	Login functionality detected without session	app.py	N/A	missing_session_rota
2	Login functionality detected without session	server.php	N/A	missing_session_rota

Recommendations:
• Regenerate session ID after login to prevent session fixation: session.regenerate() o

MISSING MULTI-FACTOR AUTHENTICATION - 2 findings

#	Issue	File	Line	Type
1	Authentication code without MFA/2FA implement	app.py	N/A	missing_mfa
2	Authentication code without MFA/2FA implement	server.php	N/A	missing_mfa

Recommendations:
• Implement multi-factor authentication for sensitive applications

WEAK PASSWORD POLICY - 6 findings

#	Issue	File	Line	Type

1	Password handling without length validation	app.py	N/A	missing_password_len
2	Password handling without complexity requirem	app.py	N/A	missing_password_com
3	Password handling without length validation	server.php	N/A	missing_password_len
4	Password handling without complexity requirem	server.php	N/A	missing_password_com
5	Password handling without length validation	settings.json	N/A	missing_password_len
6	Password handling without complexity requirem	settings.json	N/A	missing_password_com

Recommendations:

• Enforce minimum password length (at least 8-12 characters)

• Enforce password complexity (uppercase, lowercase, numbers, special characters)

SECURITY RECOMMENDATIONS

■ *CRITICAL*

- Remove all hardcoded secrets immediately
- Fix code execution vulnerabilities (eval, exec)
- Address command injection flaws
- Remediate insecure deserialization

■ *HIGH*

- Implement input validation for all user inputs
- Use parameterized queries for database operations
- Replace weak cryptographic algorithms
- Sanitize all file paths

■ *MEDIUM*

- Implement logging and monitoring
- Set up automated security scanning
- Conduct regular security code reviews
- Use secret management tools (Vault, AWS Secrets Manager)

■ *BEST PRACTICES*

- Implement defense-in-depth strategy
- Follow principle of least privilege
- Keep dependencies up to date
- Document security assumptions