■■ SECURITY ANALYSIS REPORT

Comprehensive Code Security Assessment

Project: project

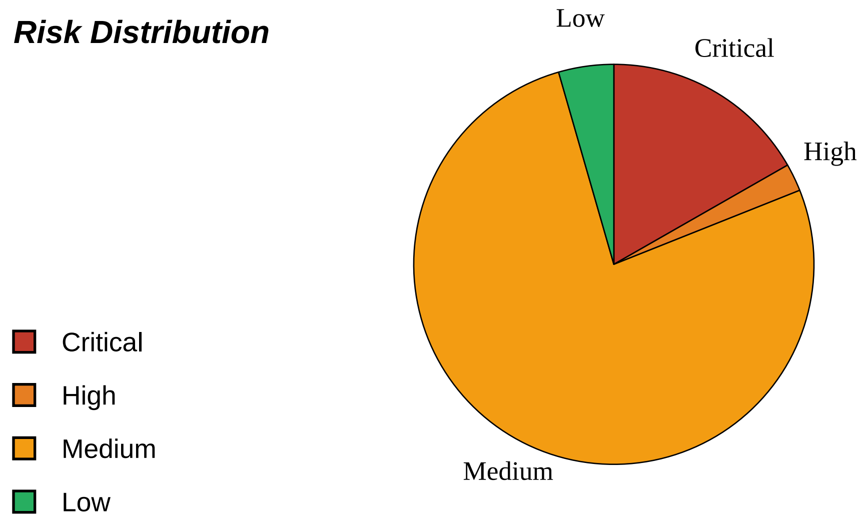
Generated: November 24, 2025 at 01:11:30

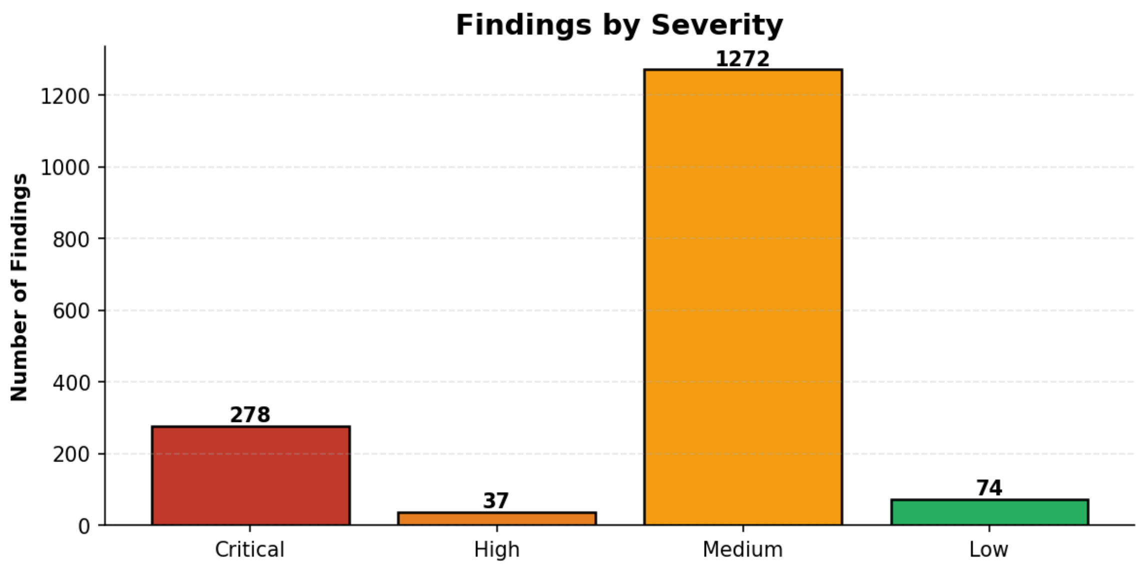
This report contains sensitive security information. Handle with care and distribute only to authorized personnel.

EXECUTIVE SUMMARY

Overall Risk Level	**CRITICAL**
Total Findings	1661
Critical Issues	278
High Issues	37
Medium Issues	1272
Low Issues	74
Languages Analyzed	python, php, json, javascript, env

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/	
■ ■■■ ANTIPATTERN_DETECTOR_README.md	
■ ■■■ COMPLETE_INTEGRATION_GUIDE.md	
■ ■■■ CRYPTOGRAPHY_INTEGRATION.md	
■ ■■■ DOCUMENTATION_SUMMARY.md	
■ ■■■ FEATURE_SUMMARY.txt	
■ ■■■ FINAL_INTEGRATION_SUMMARY.md	
■ ■■■ FINAL_SOLUTION.md	
■ ■■■ Feature_Implementation_Checklist.pdf	
■ ■■■ HOW_TO_USE.md	
■ ■■■ IMPLEMENTATION_SUMMARY.md	
■ ■■■ INTEGRATION_GUIDE.md	
■ ■■■ PROJECT_STRUCTURE.md	
■ ■■■ Project_Milestones_And_Progress.pdf	
■ ■■■ QUALITY_ANALYZER_README.md	
■ ■■■ QUICKSTART.md	
■ ■■■ QUICK_REFERENCE.txt	
■ ■■■ QUICK_START_GUIDE.md	
■ ■■■ README_QUALITY_ANALYZER.md	
■ ■■■ README_SECURITY_ANALYZER.md	
■ ■■■ **Reverser.py**	CRITICAL
■ ■■■ SECURITY_CHECKS_INTEGRATION.md	
■ ■■■ SOLUTION_SUMMARY.md	
■ ■■■ VALIDATION_FEATURES.md	
■ ■■■ **antipattern_detector.py**	CRITICAL
■ ■■■ complete_code_analysis_report.pdf	
■ ■■■ **concept_map_python.py**	CRITICAL
■ ■■■ **demo_analyzer.py**	CRITICAL

■ ■■■ **demo_antipattern_report.py**	CRITICAL
■ ■■■ **demo_quality_report.py**	CRITICAL
■ ■■■ **enhanced_analysis.py**	CRITICAL
■ ■■■ good_points.txt	
■ ■■■ **input processing.py**	CRITICAL
■ ■■■ **pdf_report_generator.py**	CRITICAL
■ ■■■ **project_documentation_generator.py**	CRITICAL
■ ■■■ project_tasks_list.txt	
■ ■■■ quality_analyzer.py	
■ ■■■ requirements.txt	
■ ■■■ **run_complete_analysis.py**	CRITICAL
■ ■■■ security_analysis.json	
■ ■■■ security_analysis_report.pdf	
■ ■■■ **test_antipattern_samples.py**	CRITICAL
■ ■■■ **test_quality_samples.py**	CRITICAL
■ ■■■ **test_vulnerable_sample.js**	CRITICAL
■ ■■■ **test_vulnerable_sample.py**	CRITICAL
■ ■■■ **validation_checker.py**	CRITICAL
■ ■■■ security_checks/	
■ ■ ■■■ README.md	
■ ■ ■■■ **__init__.py**	CRITICAL
■ ■ ■■■ **authentication_checker.py**	CRITICAL

INTELLIGENT FINDINGS TABLE (DEDUPLICATED)

Category	Unique Findings	Status
Dangerous Functions	147	■■
Hardcoded Secrets	26	■■
Taint Sources	27	■■
File/Network Ops	53	■■

■ *DANGEROUS FUNCTIONS*

File	Function	Category	Lines	Count	Risk
input processing.py	file	file_operations	33, 38, 46,	205	MEDIUM
pdf_report_generator.py	file	file_operations	26-27, 29, 3	141	MEDIUM
antipattern_detector.py	file	file_operations	21, 28, 32-3	110	MEDIUM
enhanced_analysis.py	file	file_operations	3, 22, 24, 2	98	MEDIUM
dotnet_frameworks.py	file	file_operations	23, 28, 31,	59	MEDIUM
project_documentation_gen	file	file_operations	32, 76, 80,	47	MEDIUM
python_frameworks.py	file	file_operations	24, 29, 32,	45	MEDIUM
authentication_checker.py	file	file_operations	22, 42, 56,	43	MEDIUM
javascript_frameworks.py	file	file_operations	22, 27, 30,	41	MEDIUM
validation_checker.py	file	file_operations	14, 36, 58,	38	MEDIUM
java_frameworks.py	file	file_operations	23, 28, 31,	36	MEDIUM
Reverser.py	file	file_operations	22-23, 28-34	35	MEDIUM
cryptography_checker.py	file	file_operations	31, 79, 92,	32	MEDIUM
test_antipattern_samples.	file	file_operations	2-3, 81, 84-	26	MEDIUM
input processing.py	exec	code_execution	33, 43-44, 5	24	CRITICAL
input processing.py	open	file_operations	34-35, 37-38	20	MEDIUM
base_checker.py	file	file_operations	32, 38, 43,	19	MEDIUM
validation_checker.py	exec	code_execution	119, 121-122	16	CRITICAL
demo_antipattern_report.p	file	file_operations	58-59, 61, 7	15	MEDIUM
test_vulnerable_sample.py	file	file_operations	36-37, 54-57	13	MEDIUM

■ *HARDCODED SECRETS*

File	Secret Type	Lines	Count	Severity

demo_analyzer.py	Aws Key	58, 72, 83	3	CRITICAL
concept_map_python.py	Password	291	1	HIGH
concept_map_python.py	Connection String	140	1	HIGH
concept_map_python.py	Connection String	140	1	HIGH
concept_map_python.py	Connection String	140	1	HIGH
demo_analyzer.py	Api Key	58	1	HIGH
demo_analyzer.py	Stripe Key	82	1	CRITICAL
test_antipattern_samples.py	Api Key	17	1	HIGH
test_antipattern_samples.py	Password	16	1	HIGH
test_antipattern_samples.py	Password	21	1	HIGH
test_vulnerable_sample.js	Api Key	16	1	HIGH
test_vulnerable_sample.js	Aws Key	12	1	CRITICAL
test_vulnerable_sample.js	Github Token	13	1	CRITICAL
test_vulnerable_sample.js	Slack Token	16	1	HIGH
test_vulnerable_sample.js	Jwt	15	1	HIGH

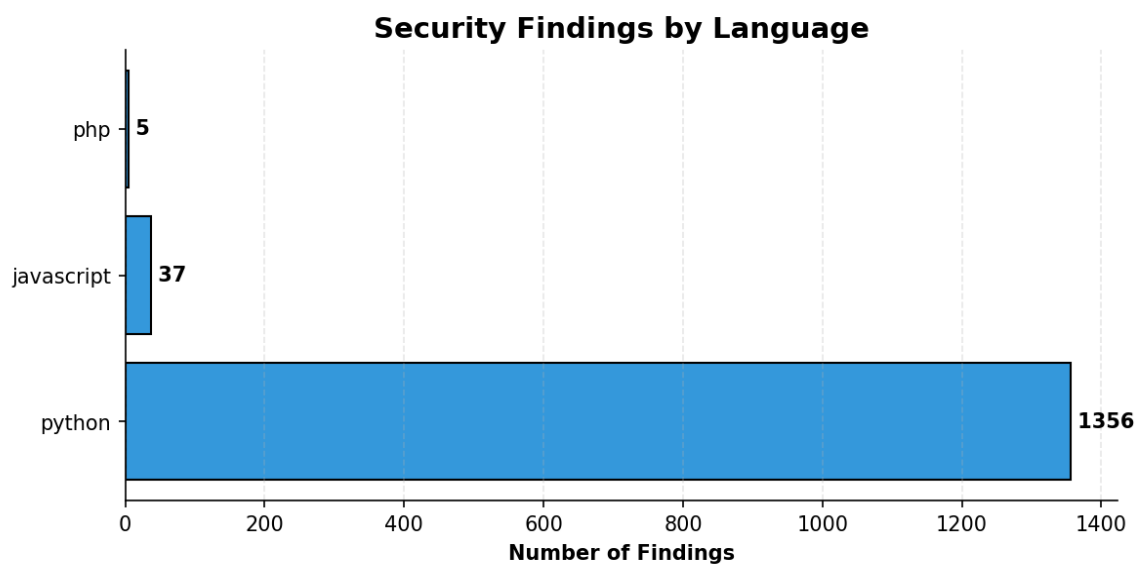
■ *USER INPUT SOURCES (TAINT ORIGINS)*

File	Input Source	Language	Lines	Count
test_vulnerable_sample.js	req.query	javascript	24, 31, 39,	8
input processing.py	sys.argv	python	72, 1397, 14	6
test_vulnerable_sample.py	request.args	python	28, 36, 43,	5
run_complete_analysis.py	sys.argv	python	193, 198	4
test_vulnerable_sample.py	sys.argv	python	79, 102	4
test_vulnerable_sample.js	req.body	javascript	61-62, 132	3
test_vulnerable_sample.js	process.env	javascript	116-117, 145	3
antipattern_detector.py	sys.argv	python	501	2
concept_map_python.py	input(	python	73, 294	2
demo_analyzer.py	sys.argv	python	61	2
demo_antipattern_report.py	sys.argv	python	177	2
demo_quality_report.py	sys.argv	python	174	2
Reverser.py	sys.argv	python	200, 203	2
python_frameworks.py	os.environ	python	73, 163	2
concept_map_python.py	sys.argv	python	73	1

■ *FILE & NETWORK OPERATIONS*

File	Operation Type	Lines	Count	Risk
input processing.py	Download	124, 258, 57	6	HIGH
input processing.py	File Read	308, 334, 35	5	MEDIUM
Reverser.py	File Read	47, 73, 97,	4	MEDIUM
test_antipattern_samples.py	Http Request	59, 73, 140,	4	MEDIUM
demo_analyzer.py	File Write	53, 71, 80	3	MEDIUM
input processing.py	File Read	54, 57, 251	3	MEDIUM
test_antipattern_samples.py	File Read	88, 153, 195	3	MEDIUM
test_vulnerable_sample.py	Socket	96	3	MEDIUM
antipattern_detector.py	File Read	81, 347	2	MEDIUM
input processing.py	File Write	46, 249	2	MEDIUM
input processing.py	File Write	54, 249	2	MEDIUM
input processing.py	File Read	46, 251	2	MEDIUM
input processing.py	Http Request	57, 256	2	MEDIUM
input processing.py	Http Request	47, 257	2	MEDIUM
input processing.py	Socket	57, 259	2	MEDIUM

DANGEROUS FUNCTIONS DETECTED



PYTHON - 1356 findings

Function	Category	File	Line
exec	code_execution	antipattern_detector.py	131
exec	code_execution	antipattern_detector.py	139
exec	code_execution	antipattern_detector.py	139
exec	code_execution	antipattern_detector.py	151
exec	code_execution	antipattern_detector.py	164
exec	code_execution	antipattern_detector.py	185
exec	code_execution	antipattern_detector.py	197
exec	code_execution	antipattern_detector.py	371
compile	code_execution	antipattern_detector.py	207
compile	code_execution	antipattern_detector.py	355
compile	code_execution	antipattern_detector.py	371
compile	code_execution	antipattern_detector.py	386
compile	code_execution	antipattern_detector.py	403
compile	code_execution	antipattern_detector.py	405
compile	code_execution	antipattern_detector.py	436

JAVASCRIPT - 37 findings

Function	Category	File	Line

eval	code_execution	test_vulnerable_sample.js	22
eval	code_execution	test_vulnerable_sample.js	23
eval	code_execution	test_vulnerable_sample.js	25
eval	code_execution	test_vulnerable_sample.js	112
Function	code_execution	test_vulnerable_sample.js	44
Function	code_execution	test_vulnerable_sample.js	45
Function	code_execution	test_vulnerable_sample.js	47
Function	code_execution	test_vulnerable_sample.js	86
Function	code_execution	test_vulnerable_sample.js	95
Function	code_execution	test_vulnerable_sample.js	104
Function	code_execution	test_vulnerable_sample.js	111
Function	code_execution	test_vulnerable_sample.js	121
Function	code_execution	test_vulnerable_sample.js	126
setTimeout	code_execution	test_vulnerable_sample.js	52
setTimeout	code_execution	test_vulnerable_sample.js	55

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

Found 2353 potential taint flows

Flow #	1
Risk Level	HIGH
File	antipattern_detector.py
Source	sys.argv (line 501)
Sink	file (line 519)
Description	Tainted data from sys.argv may flow to file

Flow #	2
Risk Level	HIGH
File	antipattern_detector.py
Source	sys.argv (line 501)
Sink	file (line 526)
Description	Tainted data from sys.argv may flow to file

Flow #	3
Risk Level	HIGH
File	antipattern_detector.py
Source	sys.argv (line 501)
Sink	file (line 533)
Description	Tainted data from sys.argv may flow to file

Flow #	4
Risk Level	HIGH
File	antipattern_detector.py
Source	sys.argv (line 501)
Sink	file (line 537)
Description	Tainted data from sys.argv may flow to file

Flow #	5
Risk Level	HIGH
File	antipattern_detector.py
Source	sys.argv (line 501)
Sink	file (line 538)
Description	Tainted data from sys.argv may flow to file

Flow #	6

Risk Level	HIGH
File	antipattern_detector.py
Source	sys.argv (line 501)
Sink	file (line 538)
Description	Tainted data from sys.argv may flow to file

Flow #	7
Risk Level	HIGH
File	antipattern_detector.py
Source	sys.argv (line 501)
Sink	file (line 539)
Description	Tainted data from sys.argv may flow to file

Flow #	8
Risk Level	HIGH
File	antipattern_detector.py
Source	sys.argv (line 501)
Sink	file (line 540)
Description	Tainted data from sys.argv may flow to file

Flow #	9
Risk Level	HIGH
File	antipattern_detector.py
Source	sys.argv (line 501)
Sink	file (line 541)
Description	Tainted data from sys.argv may flow to file

Flow #	10
Risk Level	HIGH
File	antipattern_detector.py
Source	sys.argv (line 501)
Sink	file (line 542)
Description	Tainted data from sys.argv may flow to file

Flow #	11
Risk Level	HIGH
File	antipattern_detector.py
Source	sys.argv (line 501)
Sink	file (line 519)
Description	Tainted data from sys.argv may flow to file

Flow #	12
Risk Level	HIGH
File	antipattern_detector.py
Source	sys.argv (line 501)
Sink	file (line 526)
Description	Tainted data from sys.argv may flow to file

Flow #	13
Risk Level	HIGH
File	antipattern_detector.py
Source	sys.argv (line 501)
Sink	file (line 533)
Description	Tainted data from sys.argv may flow to file

Flow #	14
Risk Level	HIGH
File	antipattern_detector.py
Source	sys.argv (line 501)
Sink	file (line 537)
Description	Tainted data from sys.argv may flow to file

Flow #	15
Risk Level	HIGH
File	antipattern_detector.py
Source	sys.argv (line 501)
Sink	file (line 538)
Description	Tainted data from sys.argv may flow to file

Flow #	16
Risk Level	HIGH
File	antipattern_detector.py
Source	sys.argv (line 501)
Sink	file (line 538)
Description	Tainted data from sys.argv may flow to file

Flow #	17
Risk Level	HIGH
File	antipattern_detector.py
Source	sys.argv (line 501)

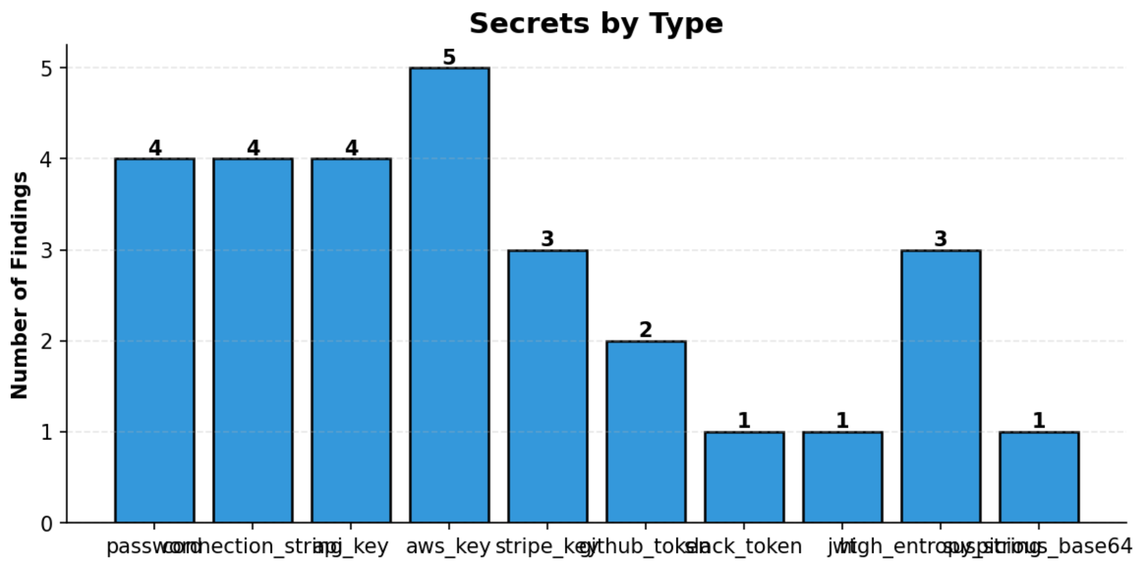
Sink	file (line 539)
Description	Tainted data from sys.argv may flow to file

Flow #	18
Risk Level	HIGH
File	antipattern_detector.py
Source	sys.argv (line 501)
Sink	file (line 540)
Description	Tainted data from sys.argv may flow to file

Flow #	19
Risk Level	HIGH
File	antipattern_detector.py
Source	sys.argv (line 501)
Sink	file (line 541)
Description	Tainted data from sys.argv may flow to file

Flow #	20
Risk Level	HIGH
File	antipattern_detector.py
Source	sys.argv (line 501)
Sink	file (line 542)
Description	Tainted data from sys.argv may flow to file

HARDCODED SECRETS & CREDENTIALS



Total secrets found: 28

Type	File	Line	Preview
password	concept_map_python.py	291	password = "admin123"
connection_stri	concept_map_python.py	140	mongodb://',
connection_stri	concept_map_python.py	140	mysql://',
connection_stri	concept_map_python.py	140	postgresql://',
api_key	demo_analyzer.py	58	API_KEY = "AKIAIOSFODNN7EXAMPLE"
aws_key	demo_analyzer.py	58	AKIAIOSFODNN7EXAMPLE
aws_key	demo_analyzer.py	72	AKIAIOSFODNN7EXAMPLE
aws_key	demo_analyzer.py	83	AKIAIOSFODNN7EXAMPLE
stripe_key	demo_analyzer.py	82	sk_live_abcdefghijklmnopqrstuvwx
api_key	test_antipattern_samples.	17	api_key = "sk_live_123456789abcdef"
password	test_antipattern_samples.	16	password = "SuperSecret123!"
password	test_antipattern_samples.	21	PASSWORD = "admin123"
api_key	test_vulnerable_sample.js	16	API_TOKEN = "xoxp-123456789012-123456789...
aws_key	test_vulnerable_sample.js	12	AKIAIOSFODNN7EXAMPLE
github_token	test_vulnerable_sample.js	13	ghp_abcdefghijklmnopqrstuvwxyz1234567890
slack_token	test_vulnerable_sample.js	16	xoxp-123456789012-123456789012-abcdefghi...
jwt	test_vulnerable_sample.js	15	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ...
stripe_key	test_vulnerable_sample.js	14	sk_live_abcdefghijklmnopqrstuvwx
high_entropy_st	test_vulnerable_sample.js	N/A	ghp_abcdefghijklmnopqrstuvwxyz1234567890...
high_entropy_st	test_vulnerable_sample.js	N/A	xoxp-123456789012-123456789012-abcdefghi...

FRAMEWORK-SPECIFIC SECURITY FINDINGS

Total Framework Findings: 33

● Critical: 4
● High: 13
● Medium: 16
● Low: 0

■ *CRITICAL - 4 findings*

#	Issue	File	Type	Line
1	Hardcoded credentials detected in Node.js code	test_vulnerable_sample.js	exposure	16
2	Use of eval() detected - potential code injection	test_vulnerable_sample.js	code_execution	25
3	Use of eval() detected - potential code injection	test_vulnerable_sample.js	code_execution	112
4	Hardcoded credentials detected in Node.js code	script.js	exposure	4

Key Recommendations:
• Move credentials to environment variables using process.env
• Avoid eval(). Use safer alternatives like JSON.parse() or Function constructor

■■ *HIGH - 13 findings*

#	Issue	File	Type	Line
1	Flask application missing CSRF protection	pdf_report_generator.py	config	N/A
2	Express.js missing Helmet security middleware	test_vulnerable_sample.js	config	N/A
3	Express.js missing CSRF protection	test_vulnerable_sample.js	config	N/A
4	child_process used without input sanitization	test_vulnerable_sample.js	command_injecti	N/A
5	Flask debug mode enabled	test_vulnerable_sample.py	config	116
6	Flask application missing CSRF protection	test_vulnerable_sample.py	config	N/A
7	Flask application missing CSRF protection	authentication_checker.py	config	N/A
8	Flask application missing CSRF protection	__init__.py	config	N/A
9	Django debug mode enabled in production	python_frameworks.py	config	46
10	Django ALLOWED_HOSTS is empty	python_frameworks.py	config	76
11	Django ALLOWED_HOSTS allows all hosts (*)	python_frameworks.py	config	76
12	FastAPI CORS allows credentials with wildcard orig	python_frameworks.py	config	N/A
13	Flask application missing CSRF protection	__init__.py	config	N/A

Key Recommendations:
• Install and configure flask-wtf for CSRF protection
• Install and use helmet: npm install helmet && app.use(helmet())
• Install and use csurf middleware for CSRF protection
• Validate and sanitize all inputs before executing commands. Use execFile or spawn with array argumen
• Disable debug mode in production: app.run(debug=False)

■ *MEDIUM - 16 findings*

#	Issue	File	Type	Line
1	Flask SESSION_COOKIE_SECURE not enabled	pdf_report_generator.py	config	N/A
2	Flask SESSION_COOKIE_HTTPONLY not enabled	pdf_report_generator.py	config	N/A
3	Express.js X-Powered-By header not disabled	test_vulnerable_sample.js	misconfiguratio	N/A
4	Express.js missing rate limiting middleware	test_vulnerable_sample.js	config	N/A
5	Deprecated or weak cryptographic function detected	test_vulnerable_sample.js	weak_crypto	122
6	Flask SESSION_COOKIE_SECURE not enabled	test_vulnerable_sample.py	config	N/A
7	Flask SESSION_COOKIE_HTTPONLY not enabled	test_vulnerable_sample.py	config	N/A
8	Flask SESSION_COOKIE_SECURE not enabled	authentication_checker.py	config	N/A
9	Flask SESSION_COOKIE_HTTPONLY not enabled	authentication_checker.py	config	N/A
10	Flask SESSION_COOKIE_SECURE not enabled	__init__.py	config	N/A
11	Flask SESSION_COOKIE_HTTPONLY not enabled	__init__.py	config	N/A
12	Django SECURE_SSL_REDIRECT is disabled	python_frameworks.py	config	117
13	FastAPI/Uvicorn reload enabled in production	python_frameworks.py	config	218
14	FastAPI CORS configured to allow all origins (*)	python_frameworks.py	config	235
15	Flask SESSION_COOKIE_SECURE not enabled	__init__.py	config	N/A
16	Flask SESSION_COOKIE_HTTPONLY not enabled	__init__.py	config	N/A

Frameworks Detected: Django, Express.js, FastAPI, Flask, Node.js

CRYPTOGRAPHY MISUSE ANALYSIS

Total Cryptography Issues: 23
● Weak Encryption: 5
● ECB Mode: 5
● Unsalted Passwords: 1
● JWT Issues: 5
● Weak Hashing: 7
● Predictable Random: 0

WEAK ENCRYPTION ALGORITHMS - 5 findings

#	Issue	File	Line	Pattern
1	Weak encryption algorithm detected: 'DES	cryptography_checker	99	DES.new
2	Weak encryption algorithm detected: 'DES	cryptography_checker	99	DES3.new
3	Weak encryption algorithm detected: 'ARC	cryptography_checker	99	ARC4.new
4	Weak encryption algorithm detected: 'Blo	cryptography_checker	99	Blowfish.new
5	Weak encryption algorithm detected: 'mod	cryptography_checker	99	mode=ECB

Recommendations:
• Use AES-256-GCM or ChaCha20-Poly1305 for encryption

ECB MODE USAGE - 5 findings

#	Issue	File	Line	Pattern
1	ECB mode encryption detected: 'ECB'	input processing.py	1009	ECB
2	ECB mode encryption detected: 'ECB'	pdf_report_generator	1784	ECB
3	ECB mode encryption detected: 'MODE_ECB'	cryptography_checker	286	MODE_ECB
4	ECB mode encryption detected: 'mode=AES.	cryptography_checker	286	mode=AES.MODE_ECB
5	ECB mode encryption detected: 'ECB'	cryptography_checker	7	ECB

Recommendations:
• Use CBC, GCM, or CTR mode instead of ECB. ECB mode leaks patterns in plaintext.

UNSALTED PASSWORD HASHING - 1 findings

#	Issue	File	Line	Pattern
1	Password hashing without salt or proper	test_vulnerable_samp	N/A	N/A

Recommendations:
• Use bcrypt, argon2, or scrypt for password hashing with automatic salting

JWT SECURITY ISSUES - 5 findings

#	Issue	File	Line	Pattern
1	JWT 'none' algorithm or disabled verific	cryptography_checker	343	algorithm="none"
2	JWT 'none' algorithm or disabled verific	cryptography_checker	343	algorithm='none'
3	JWT 'none' algorithm or disabled verific	cryptography_checker	343	algorithms=["none"]
4	JWT signature verification disabled	cryptography_checker	344	verify_signature=False
5	JWT signature verification disabled	cryptography_checker	344	verify=False

Recommendations:
• Always use a strong signing algorithm (HS256, RS256, ES256) and verify JWT signatures

WEAK HASHING ALGORITHMS - 7 findings

#	Issue	File	Line	Pattern
1	Weak hashing algorithm detected: 'crypto	test_vulnerable_samp	122	crypto.createHash('md5')
2	Weak hashing algorithm detected: 'hashli	test_vulnerable_samp	76	hashlib.md5
3	Weak hashing algorithm detected: 'md5('	test_vulnerable_samp	76	md5(
4	Weak hashing algorithm detected: 'hashli	cryptography_checker	38	hashlib.md5
5	Weak hashing algorithm detected: 'hashli	cryptography_checker	38	hashlib.sha1
6	Weak hashing algorithm detected: 'md5('	cryptography_checker	38	md5(
7	Weak hashing algorithm detected: 'sha1('	cryptography_checker	38	sha1(

Recommendations:
• Use SHA-256, SHA-512, or SHA-3 instead of MD5/SHA1

AUTHENTICATION & SESSION SECURITY

Total Authentication & Session Issues: 40
● Auth Bypass: 10
● Insecure Cookies: 0
● Weak Session Timeout: 0
● Missing Session Rotation: 3
● Missing MFA: 3
● Weak Password Policy: 24

AUTHENTICATION BYPASS - 10 findings

#	Issue	File	Line	Type
1	Potential authentication bypass detected	input processing.py	1058	potential_auth_bypas
2	Potential authentication bypass detected	input processing.py	1069	potential_auth_bypas
3	Potential authentication bypass detected	pdf_report_generator	1887	potential_auth_bypas
4	Potential authentication bypass detected	pdf_report_generator	1903	potential_auth_bypas
5	Potential authentication bypass detected	pdf_report_generator	1916	potential_auth_bypas
6	Potential authentication bypass detected	authentication_check	327	potential_auth_bypas
7	Potential authentication bypass detected	authentication_check	331	potential_auth_bypas
8	Potential authentication bypass detected	authentication_check	379	potential_auth_bypas
9	Potential authentication bypass detected	authentication_check	332	potential_auth_bypas
10	Potential authentication bypass detected	server.php	18	potential_auth_bypas

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

MISSING SESSION ROTATION - 3 findings

#	Issue	File	Line	Type
1	Login functionality detected without session	cryptography_checker	N/A	missing_session_rota
2	Login functionality detected without session	app.py	N/A	missing_session_rota
3	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 - 3 findings

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

Recommendations:
• Implement multi-factor authentication for sensitive applications

WEAK PASSWORD POLICY - 24 findings

#	Issue	File	Line	Type
1	Password handling without length validation	antipattern_detector	N/A	missing_password_len
2	Password handling without length validation	concept_map_python.p	N/A	missing_password_len
3	Password handling without length validation	demo_analyzer.py	N/A	missing_password_len
4	Password handling without length validation	demo_antipattern_rep	N/A	missing_password_len
5	Password handling without length validation	enhanced_analysis.py	N/A	missing_password_len
6	Password handling without length validation	pdf_report_generator	N/A	missing_password_len
7	Password handling without length validation	project_documentatio	N/A	missing_password_len
8	Password handling without length validation	Reverser.py	N/A	missing_password_len
9	Password handling without complexity requirem	Reverser.py	N/A	missing_password_com
10	Password handling without length validation	run_complete_analysi	N/A	missing_password_len
11	Password handling without length validation	test_antipattern_sam	N/A	missing_password_len
12	Password handling without length validation	test_vulnerable_samp	N/A	missing_password_len
13	Password handling without length validation	test_vulnerable_samp	N/A	missing_password_len
14	Password handling without complexity requirem	test_vulnerable_samp	N/A	missing_password_com
15	Password handling without length validation	cryptography_checker	N/A	missing_password_len

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

CODE QUALITY & MAINTAINABILITY ANALYSIS

✓ Quality analysis not performed or module not loaded.

ANTI-PATTERN & SECURITY ISSUES DETECTION

Total Anti-Pattern Issues: 112
● Password Variables: 8
● SQL Concatenation: 12
● API Without Timeout: 8
● Unsafe File Paths: 24
● Dead Code: 59
● .env Issues: 1

■ *PASSWORD/SECRET VARIABLES - 8 findings*

Hardcoded passwords and secrets pose critical security risks.

#	File	Line	Variable	Language	Severity
1	test_antipattern_s	16	password	python	CRITICAL
2	test_antipattern_s	17	api_key	python	CRITICAL
3	test_antipattern_s	18	secret	python	CRITICAL
4	test_antipattern_s	21	DATABASE_PASSWORD	python	CRITICAL
5	test_antipattern_s	131	api_key	python	CRITICAL
6	test_vulnerable_sa	18	API_KEY	python	CRITICAL
7	test_vulnerable_sa	22	SECRET_TOKEN	python	CRITICAL
8	app.py	4	API_KEY	python	CRITICAL

Recommendations:
• Use environment variables instead of hardcoded secrets
• Implement proper secret management (Vault, AWS Secrets Manager)
• Never commit secrets to version control

■ *SQL INJECTION RISKS - 12 findings*

SQL queries built with string concatenation are vulnerable to SQL injection attacks.

#	File	Line	Pattern	Language
1	test_antipattern_sampl	35	execute() with concatenat	python
2	test_antipattern_sampl	35	execute() with f-string v	python
3	test_antipattern_sampl	35	execute() with concatenat	python
4	test_antipattern_sampl	47	execute() with concatenat	python

5	test_antipattern_sampl	47	execute() with f-string v	python
6	test_antipattern_sampl	47	execute() with concatenat	python
7	test_antipattern_sampl	137	execute() with concatenat	python
8	test_antipattern_sampl	137	execute() with f-string v	python
9	test_antipattern_sampl	137	execute() with concatenat	python
10	test_antipattern_sampl	171	execute() with concatenat	python
11	test_antipattern_sampl	171	execute() with f-string v	python
12	test_antipattern_sampl	171	execute() with concatenat	python

Recommendations:
• Use parameterized queries with placeholders (?, %s)
• Implement ORM libraries (SQLAlchemy, Sequelize, Django ORM)
• Never concatenate user input into SQL queries

■■ *API CALLS WITHOUT TIMEOUT - 8 findings*

API calls without timeout can cause application hangs and resource exhaustion.

#	File	Line	Method	Language
1	test_antipattern_samples.	59	get	python
2	test_antipattern_samples.	66	post	python
3	test_antipattern_samples.	73	get	python
4	test_antipattern_samples.	74	post	python
5	test_antipattern_samples.	75	put	python
6	test_antipattern_samples.	140	get	python
7	test_vulnerable_sample.js	88	fetch/axios	javascript
8	test_vulnerable_sample.py	92	post	python

Recommendations:
• Python: Add timeout parameter to requests.get/post()
• JavaScript: Use AbortController for fetch() or timeout config for axios
• Set reasonable timeout values (e.g., 30 seconds for normal API calls)

■ *UNSAFE FILE PATH ACCESS - 24 findings*

File operations with unsanitized user input can lead to path traversal attacks.

#	File	Line	Operation	Language
1	antipattern_detector.p	80	open	python
2	antipattern_detector.p	346	open	python
3	antipattern_detector.p	431	open	python
4	input processing.py	307	open	python
5	input processing.py	333	open	python

6	input processing.py	357	open	python
7	input processing.py	395	open	python
8	input processing.py	420	open	python
9	input processing.py	441	open	python
10	project_documentation_	80	open	python
11	project_documentation_	164	open	python
12	Reverser.py	46	open	python
13	Reverser.py	72	open	python
14	Reverser.py	96	open	python
15	Reverser.py	134	open	python
16	Reverser.py	160	open	python
17	test_antipattern_sampl	94	remove	python
18	test_antipattern_sampl	156	remove	python
19	test_antipattern_sampl	87	open	python
20	test_antipattern_sampl	152	open	python

Recommendations:
• Validate and sanitize all file paths from user input
• Use os.path.join() or path.join() for safe path construction
• Check paths against allowed directories (whitelist approach)
• Reject paths containing '..' or absolute paths from users

■■ *.ENV FILE SECURITY - 1 findings*

.env files containing secrets must be properly secured.

#	File	Line	Issue
1	creds.env	1	Password in .env

Recommendations:
• Ensure .env files are in .gitignore
• Never commit .env files to version control
• Provide .env.example with placeholder values
• Use proper secret management in production

SECURITY & QUALITY RECOMMENDATIONS

■ *CRITICAL*

• Remove all hardcoded secrets and passwords immediately
• Fix SQL injection vulnerabilities - use parameterized queries
• Fix code execution vulnerabilities (eval, exec)
• Address command injection flaws
• Remediate insecure deserialization
• Fix infinite loops that can cause system hangs
• Secure .env files and never commit them to version control

■ *HIGH*

• Implement input validation for all user inputs
• Use parameterized queries for ALL database operations
• Replace weak cryptographic algorithms
• Sanitize all file paths from user input
• Add proper error handling in empty catch blocks
• Add timeout to all API/HTTP requests
• Validate file paths against path traversal attacks

■ *MEDIUM*

• Implement logging and monitoring
• Set up automated security scanning
• Conduct regular security code reviews
• Use secret management tools (Vault, AWS Secrets Manager)
• Remove dead/unreachable code to improve maintainability
• Add timeout parameters to prevent resource exhaustion

■ *BEST PRACTICES*

• Implement defense-in-depth strategy
• Follow principle of least privilege
• Keep dependencies up to date
• Document security assumptions
• Enforce consistent naming conventions across the codebase
• Use linters and formatters for code quality
• Use environment variables for configuration
• Implement ORM libraries instead of raw SQL