

# Sample Evidence Report

Local Code Review - Structured Evidence - Professional PDF

## SENATTOREULTRACON DIAMOND

Sample project: fintech-backend-demo

**Sensitive client code in.**  
**100% local Windows review.**  
**Professional evidence + PDF out.**  
**No cloud upload.**

**SAMPLE GENERATED BY SENATTOREULTRACON DIAMOND**

This is a commercial sample generated from an intentionally vulnerable demo project.  
It demonstrates evidence structure, reporting output and workflow.

# Executive Summary

This sample report demonstrates how SENATTOREULTRACON DIAMOND can turn a local demo project into structured evidence, exportable technical artifacts and a professional PDF deliverable.

The analyzed project is intentionally vulnerable and exists only to demonstrate reporting output. It must not be deployed or reused in production.

<b>8</b> Total findings	<b>4</b> Critical
<b>3</b> High	<b>1</b> Medium

<b>Sample</b>	fintech-backend-demo
<b>Risk level</b>	CRITICAL
<b>Generated</b>	2026-06-18 10:43:55
<b>Workflow</b>	Local review -> structured evidence -> SARIF/JSON/TXT -> professional PDF

## Commercial Workflow Demonstrated

<b>1</b>	Local demo project / ZIP prepared
<b>2</b>	Code review evidence generated locally
<b>3</b>	Findings organized by severity and reference
<b>4</b>	SARIF / JSON / TXT outputs generated
<b>5</b>	Professional PDF evidence report produced

## Findings Overview

ID	Severity	Finding	Location
SUCD-SAMPLE-001	<b>CRITICAL</b>	Hardcoded secrets in backend source code	app.js:26
SUCD-SAMPLE-002	<b>CRITICAL</b>	SQL injection through concatenated request input	app.js:56
SUCD-SAMPLE-003	<b>HIGH</b>	Server-Side Request Forgery through user-controlled URL	app.js:109
SUCD-SAMPLE-004	<b>CRITICAL</b>	Command injection in report export workflow	app.js:135
SUCD-SAMPLE-005	<b>HIGH</b>	Path traversal in report file access	app.js:157
SUCD-SAMPLE-006	<b>HIGH</b>	Authorization bypass through request-controlled admin flag	app.js:175
SUCD-SAMPLE-007	<b>CRITICAL</b>	Sensitive debug endpoint exposes secrets and environment	app.js:197
SUCD-SAMPLE-008	<b>MEDIUM</b>	Overly permissive CORS policy	app.js:21

The overview above is designed for fast triage. The next section provides evidence, impact and remediation guidance for each finding.

## Detailed Evidence

Each finding below is formatted as a client-ready evidence block with location, reference, impact and remediation guidance.

SUCD-SAMPLE-001		CRITICAL
<b>Hardcoded secrets in backend source code</b>		
app.js:26   CWE-798   A07:2021 Identification and Authentication Failures		
<b>Evidence</b>	JWT_SECRET and ADMIN_API_KEY are hardcoded in the backend source.	
<b>Impact</b>	Secrets embedded in source code can be leaked, reused, extracted from repositories or exposed through logs/debug endpoints.	
<b>Remediation</b>	Move secrets to protected environment variables or a secrets manager, rotate exposed keys and remove debug exposure.	

SUCD-SAMPLE-002		CRITICAL
<b>SQL injection through concatenated request input</b>		
app.js:56   CWE-89   A03:2021 Injection		
<b>Evidence</b>	req.query and req.body values are concatenated directly into SQL queries.	
<b>Impact</b>	An attacker may manipulate SQL statements to bypass authentication, extract account data or modify transfer records.	
<b>Remediation</b>	Use parameterized queries/prepared statements and validate all request input before database access.	

SUCD-SAMPLE-003		HIGH
<b>Server-Side Request Forgery through user-controlled URL</b>		
app.js:109   CWE-918   A10:2021 Server-Side Request Forgery		
<b>Evidence</b>	The /api/statement/fetch endpoint fetches a URL fully controlled by the request.	
<b>Impact</b>	An attacker may force the backend to reach internal services, metadata endpoints or restricted network resources.	
<b>Remediation</b>	Use an allowlist of trusted hosts, block internal/private IP ranges and avoid fetching arbitrary user-controlled URLs.	

SUCD-SAMPLE-004		CRITICAL
<b>Command injection in report export workflow</b>		
app.js:135   CWE-78   A03:2021 Injection		
<b>Evidence</b>	format and period values from req.body are inserted into a shell command executed with child_process.exec.	
<b>Impact</b>	An attacker may inject shell metacharacters and execute arbitrary commands on the host system.	
<b>Remediation</b>	Avoid shell execution with user input. Use spawn/execFile with fixed arguments, strict validation and allowlisted values.	

SUCD-SAMPLE-005		HIGH
<b>Path traversal in report file access</b>		
app.js:157   CWE-22   A01:2021 Broken Access Control		
<b>Evidence</b>	Client-controlled file names are joined into filesystem paths without normalization and allowlist validation.	
<b>Impact</b>	An attacker may attempt to read or write files outside the intended reports directory.	
<b>Remediation</b>	Normalize paths, reject traversal sequences, restrict file access to an allowlisted directory and enforce safe filenames.	

SUCD-SAMPLE-006		HIGH
<b>Authorization bypass through request-controlled admin flag</b>		
app.js:175   CWE-862   A01:2021 Broken Access Control		
<b>Evidence</b>	Admin access can be granted when req.query.admin is set to true.	
<b>Impact</b>	A non-admin user may access administrative user data by manipulating request parameters.	
<b>Remediation</b>	Use server-side authorization based on verified identity, roles and permissions. Never trust client-controlled admin flags.	

SUCD-SAMPLE-007		CRITICAL
<b>Sensitive debug endpoint exposes secrets and environment</b>		
app.js:197   CWE-200   A05:2021 Security Misconfiguration		
<b>Evidence</b>	The debug endpoint returns secrets, working directory and process environment information.	
<b>Impact</b>	Attackers may obtain credentials, tokens, environment values and internal configuration details.	
<b>Remediation</b>	Remove debug endpoints from production, never return secrets and enforce authenticated diagnostic access only in controlled environments.	

SUCD-SAMPLE-008		MEDIUM
<b>Overly permissive CORS policy</b>		
app.js:21   CWE-942   A05:2021 Security Misconfiguration		
<b>Evidence</b>	CORS is configured with origin '*'.	
<b>Impact</b>	Overly broad cross-origin access may increase exposure of API responses and weaken browser-side trust boundaries.	
<b>Remediation</b>	Restrict CORS origins to trusted frontend domains and review credential handling.	

## Generated Output Files

This sample pack includes multiple output formats to demonstrate how the same review can be used for human-readable reporting and technical integration workflows.

Output	Purpose
appsec-evidence-report.pdf	Professional client-ready evidence report
appsec-evidence.sarif	SARIF export for security tooling and review workflows
appsec-evidence.json	Structured evidence data
appsec-evidence.txt	Plain-text evidence summary
raw-analyzer-output.json	Raw analyzer output captured for transparency

### Important Scope Note

This PDF is a commercial sample. The demo project is intentionally vulnerable. The report demonstrates evidence structure, export formats and professional delivery. It is not a guarantee that every vulnerability will be found in every real-world codebase.

SENATTOREULTRACON DIAMOND is positioned as a complementary offline evidence and reporting layer for technical code reviews. It is not a replacement for enterprise SAST platforms, DAST scanners, pentesting expertise or human security judgment.