# AI-Assisted Secure Code Review

## Formal Evaluation Report

## 1. Experimental Setup

A controlled evaluation dataset was constructed to assess the effectiveness of the AI-Assisted Secure Code Review system. The dataset consisted of 10 intentionally vulnerable Python scripts and 5 secure (clean) Python scripts. The vulnerable samples included SQL injection, unsafe eval usage, command injection (shell=True), insecure deserialization, weak cryptographic hashing (MD5), weak randomness, insecure temporary file creation, HTTP requests without timeout configuration, improper input validation, and missing authentication checks. The clean samples implemented secure best practices including parameterized SQL queries, SHA-256 hashing, secure randomness using the secrets module, and safe subprocess invocation without shell=True. Each script was scanned via the GUI application and results were recorded in a structured evaluation sheet.

## 2. Evaluation Metrics

True Positives (TP): Vulnerable files correctly identified. False Negatives (FN): Vulnerable files missed. False Positives (FP): Clean files incorrectly flagged. True Negatives (TN): Clean files correctly ignored. Precision = TP / (TP + FP) Recall = TP / (TP + FN) F1 Score = Harmonic mean of Precision and Recall

## 3. Results

| Metric | Value |
|---|---|
| True Positives (TP) | 8 |
| False Negatives (FN) | 2 |
| False Positives (FP) | 0 |
| True Negatives (TN) | 5 |
| Precision | 1.00 |
| Recall | 0.80 |
| F1 Score | 0.89 |

## 4. Analysis

The system demonstrated high precision (100%), indicating no false positives in the evaluated clean dataset. This suggests conservative behavior and reliable filtering of secure code. Recall was measured at 80%, reflecting that two vulnerable scripts (improper input validation and missing authentication checks) were not detected. These missed cases represent logical and business-layer vulnerabilities rather than direct insecure API usage. The findings indicate strong performance in detecting explicit insecure patterns, with reduced effectiveness in identifying higher-level logical security flaws.

## 5. Limitations and Future Work

The dataset size remains limited and should be expanded for stronger statistical validity. Future improvements include expanding the dataset, automating evaluation workflows, benchmarking against standalone static analyzers, and integrating semantic analysis to improve detection of logical vulnerabilities.