AppSecAI

AppSecAI Expert Triage Automation

Assessment of SAST Findings

Company: This is a test

Customer: Paul Smith

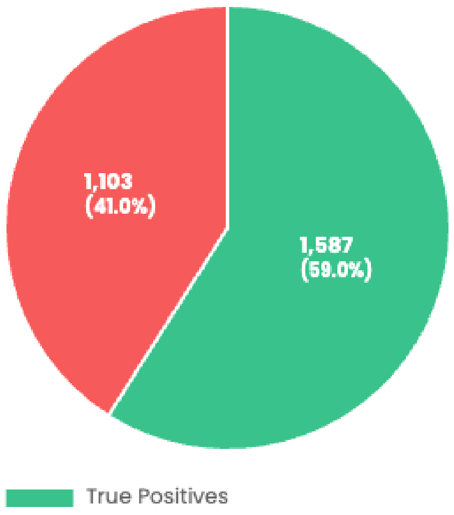
Report Date: January 23, 2025

Executive Summary

Static Application Security Testing (SAST) tools are integral to identifying vulnerabilities. But SAST "run" results must be manually triaged to remove false positives, a slow and expensive process that impedes application security, development and delivery. AppSecAI Expert Triage Automation (ETA) automates triage to lower manual triage time, tedium, and costs.

This report details the results from a single SARIF file called OWASP_Benchmark.SARIF.

2,690	1,587	1,103	41.0%
Total Vulnerabilities	True Positives	False Positives	False Positive Rate



Total Cost Savings:	**$50,973**
Manual Process:	$66,238
ETA Process:	$15,265

Total Hours Saved:	**340**
Manual Process:	442 hrs
ETA Process:	102 hrs

Total Company Savings (15 Repos)	
$ Savings:	$764,588
Hours Saved:	5,097 hrs

Key Findings Analysis: This security assessment revealed 2,690 potential vulnerabilities, with 59.0% confirmed as true positives requiring remediation. The implementation of AppSecAI ETA demonstrated significant efficiency gains, reducing manual triage and documentation time by 77% while maintaining high accuracy. When scaled across 15 repositories, this automation yields potential annual savings of $764,588 and 5,097 labor hours, enabling faster security assessments and more efficient resource allocation.

Detailed Analysis

SAST Test File: OWASP_Benchmark.SARIF

Code Base: https://github.com/OWASP-Benchmark/BenchmarkJava.git

SAST Tool: SemGrep

Triage System: AppSecAI ETA 0.9

Analysis Time: 30 minutes

Total Vulnerabilities: 2,690

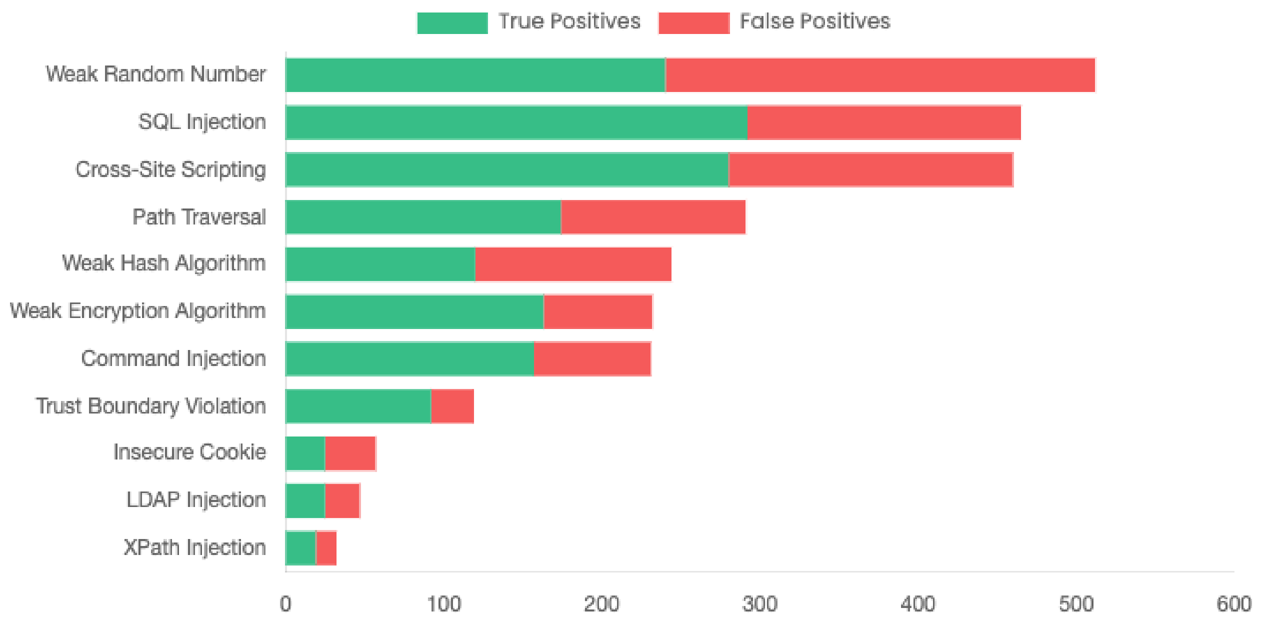
False Positives: 1,103

True Positives: 1,587

False Positive Rate: 41.0%

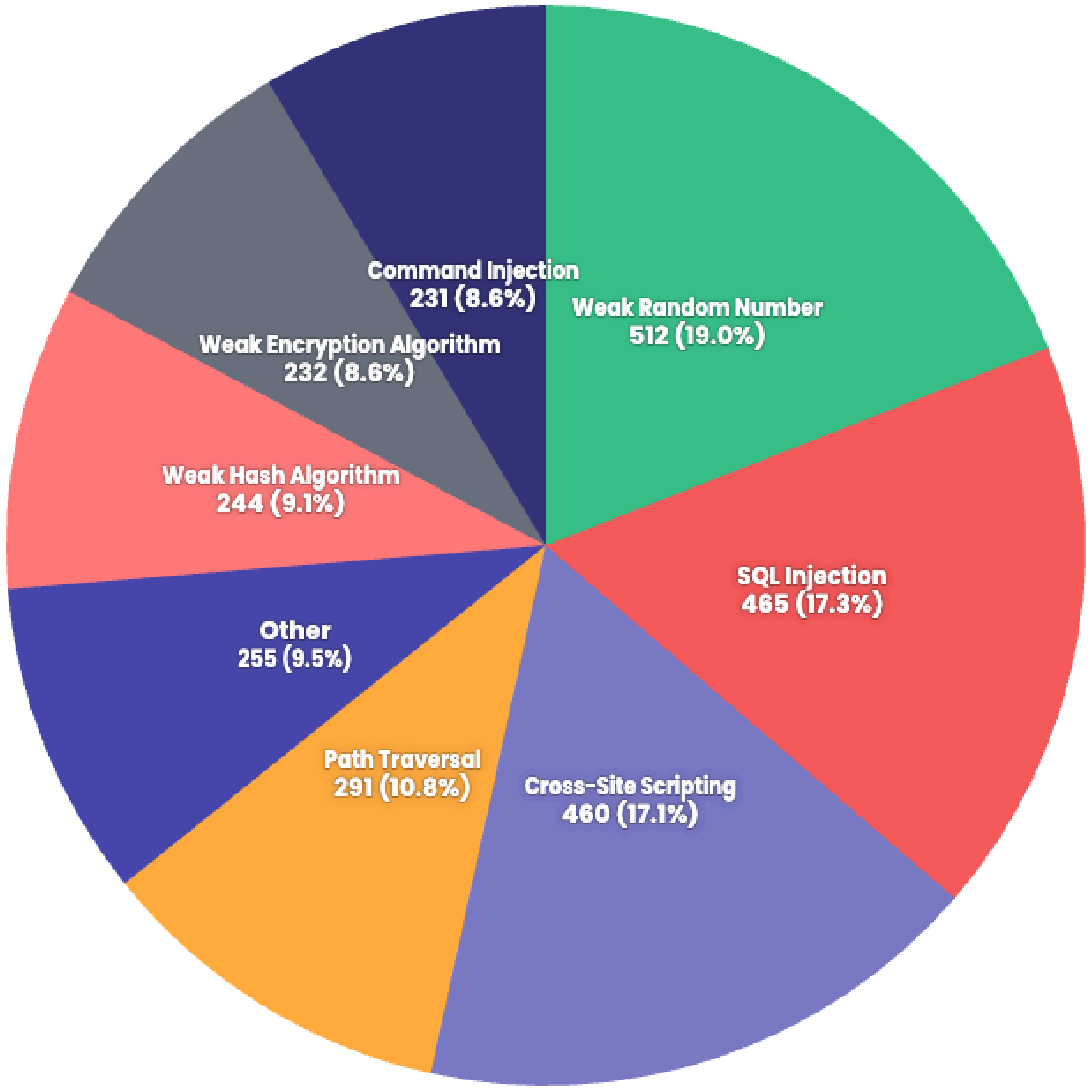
True Positive Rate: 59.0%

Vulnerability Type Distribution



Vulnerability Distribution Overview

Key Findings Analysis: The vulnerability assessment identified Weak Random Number and SQL Injection as the predominant security concerns, collectively representing a significant portion of all detected issues. Critical analysis of the distribution reveals that 53.4% of vulnerabilities are concentrated in just three categories (Weak Random Number, SQL Injection, Cross-Site Scripting), providing a clear prioritization framework for remediation efforts. This concentration pattern suggests that a targeted security hardening approach, focusing on these primary vulnerability types, would yield the highest security posture improvement with optimal resource utilization.



Financial Analysis and Cost Comparison

Cost Rates

Developer Cost: $200/hour

AppSec Analyst Cost: $150/hour

Time Metrics

Manual Triage: 1 min/vuln

Documentation: 15 min/vuln

ETA Triage: 0.5 min/vuln

Cost Metrics (Per Vulnerability)

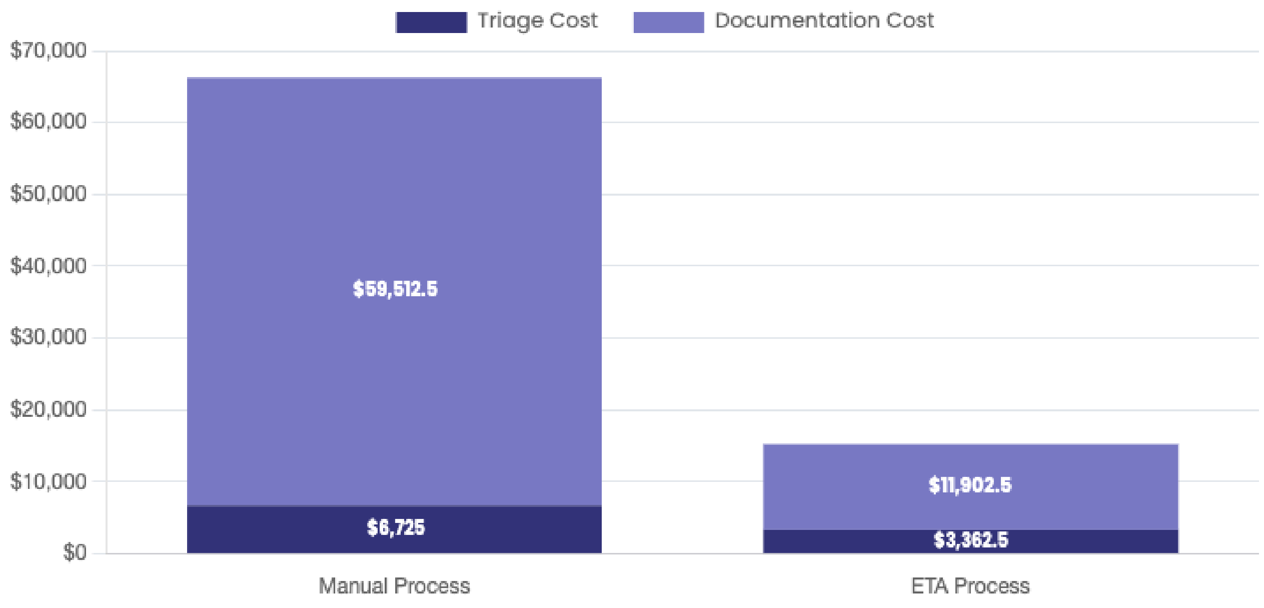
Manual Triage: $2.50

Documentation: $37.50

ETA Triage: $1.25

Fix Cost: $400.00

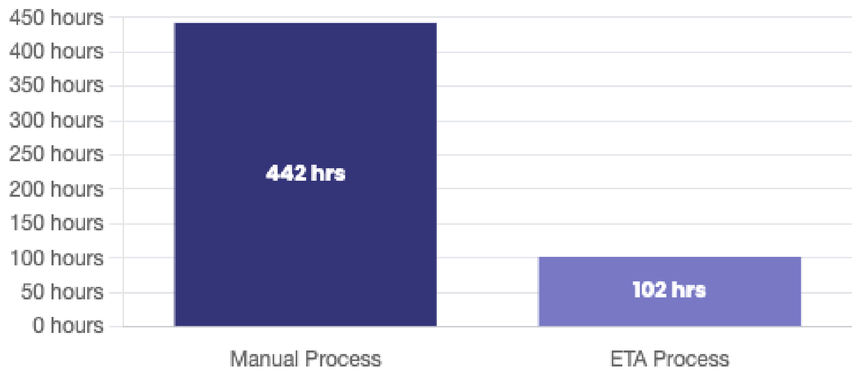
Process Cost Comparison - Total Savings: $50,972.5 (77.0%)



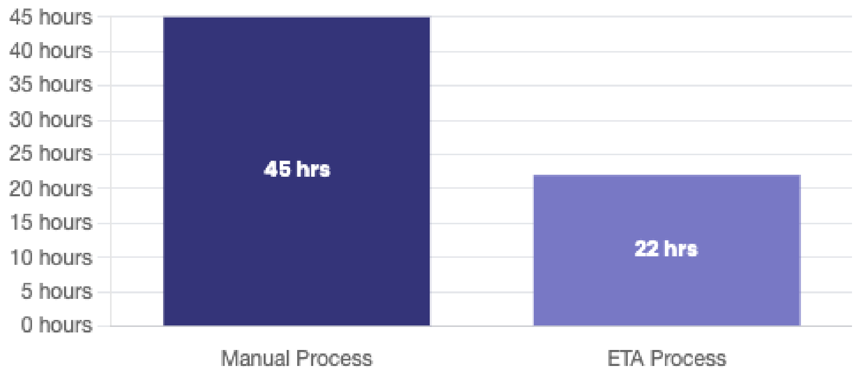
Labor Cost Analysis

Key Findings Analysis: Implementation of AppSecAI ETA demonstrates substantial labor efficiency gains across all security assessment processes. Total labor hours were reduced by 340 hours (77.0%), with particularly significant improvements in triage activities (22 hours saved, 50.0% reduction) and documentation tasks (317 hours saved, 80.0% reduction). This dramatic decrease in manual effort enables security teams to process vulnerability assessments 4x faster while maintaining high accuracy, significantly accelerating the security review cycle.

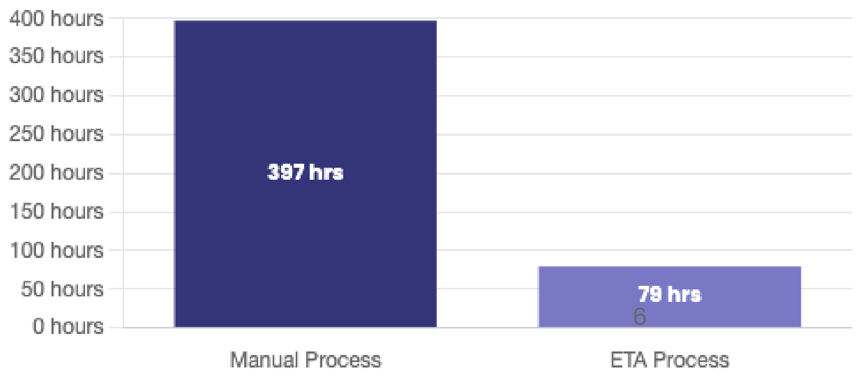
Total Labor Hours - Hours Saved: 340 (77.0%) - Cost Savings: $50,972.5 (77.0%)



Triage Labor Hours - Hours Saved: 22 (50.0%) - Cost Savings: $3,362.5 (50.0%)



Documentation Labor Hours - Hours Saved: 317 (80.0%) - Cost Savings: $47,610 (80.0%)



Appendix: Glossary of Terms

This glossary provides definitions for key terms used throughout the report to ensure clear understanding of security concepts, processes, and technologies discussed.

Application Security (AppSec)

The practice of protecting applications from security threats and vulnerabilities throughout the software development lifecycle, including development and test.

AppSecAI ETA (Expert Triage Automation)

An AI-powered solution that automates and accelerates vulnerability triage by removing false positives and documenting true positives.

False Positive

A reported security vulnerability that is determined to be incorrect or non-exploitable.

True Positive

A correctly identified security vulnerability that requires remediation.

Static Application Security Testing (SAST)

A well established security testing methodology that analyzes source code or compiled binaries to identify vulnerabilities without executing the application. Typical SAST tool results will contain false positives which need to be removed.

Triage

The process of reviewing, categorizing, and prioritizing security findings to determine their validity and impact.

SARIF (Static Analysis Results Interchange Format)

A standardized format for reporting the results of static analysis tools.

Remediation

The process of fixing identified security vulnerabilities to eliminate risks.

Triage and Fix Automation

The use of AI and machine learning to streamline security processes, reducing manual effort and cost.

Security Posture

The overall security status of an organization's applications and infrastructure based on risk assessments and mitigation strategies.

Technical Vulnerability Analysis

Data Analysis Summary:
Based on analysis of 2,690 vulnerability records, several significant patterns emerge: 1. **Language Distribution:** None is the predominant language (100.0% of vulnerabilities). 2. **Top Categories:** weakrand: 512 (19.0%); sqli: 465 (17.3%); xss: 460 (17.1%) 3. **CWE Types:** CWE-330: 512; CWE-89: 465; CWE-79: 460 4. **Data Flow:** Sources: RequestGetHeaders.code, RequestGetParameterValues.code, RequestGetQueryString.code; Sinks: SecureRandomNextDouble2.code, MessageDigestGetInstance-S-P2.code, RandomNextInt.code 5. **Risk:** 79.9% of vulnerabilities involve confirmed vulnerable sinks.

Language Distribution

Language	Count	%
None	2,690	100.0

Top Categories

Category	Count	%
weakrand	512	19.0
sqli	465	17.3
xss	460	17.1
pathtraver	291	10.8
hash	244	9.1

Top Sources

Source	Count	%
RequestGetHeaders.code	295	11.0
RequestGetParameterValues.code	282	10.5
RequestGetQueryString.code	279	10.4
RequestGetParameterMap.code	264	9.8
RequestGetParameterNames.code	252	9.4

Top Sinks

Sink	Count	%
SecureRandomNextDouble2.code	38	1.4
MessageDigestGetInstance-S-P2.code	35	1.3
RandomNextInt.code	34	1.3
SessionSetAttribute-S-O^.code	34	1.3
SecureRandomNextLong.code	34	1.3