Bayesian Evaluation of Developer Productivity Gains from OSS Vulnerability Scanning and Veracode Pipeline Scans

DbSaicle Research

January 1, 2026

Abstract

This paper quantifies developer productivity benefits from two automated security tools: an OSS vulnerability scan and a Veracode pipeline scan. We propose a hierarchical Bayesian framework to estimate reductions in time-to-remediate (TTR), vulnerability backlog, and rework rates while accounting for team and project heterogeneity. The model integrates tool usage telemetry, repository analytics, and vulnerability management data. Results are reported as posterior probabilities of improvement, credible intervals for effect sizes, and estimated economic return. This approach provides decision-grade evidence under uncertainty and supports adoption decisions across varied codebases.

1 Introduction

Modern software development teams face mounting security obligations while maintaining delivery velocity. Security findings, especially from open-source dependencies and static analysis, often inflate lead times and divert engineering effort. Automated scanning tools can reduce the time and effort required to identify and remediate vulnerabilities. However, organizations often lack statistically grounded evidence quantifying these benefits.

This paper introduces a Bayesian evaluation framework for two tools:

- `oss_vulnerability_scan`: detects dependency vulnerabilities and produces actionable remediation guidance.
- `veracode_pipeline_scan`: performs static analysis on built artifacts with rapid turnaround in CI.

We aim to measure productivity gains in terms of remediation speed, backlog reduction, and reduced rework. A Bayesian approach provides interpretable probabilities of improvement and robust uncertainty quantification, which are essential for decision making.

2 Tools Under Study

2.1 OSS Vulnerability Scan

The OSS scan tool ingests a project directory and performs dependency analysis for Maven, Gradle, or npm projects. It produces a Markdown report listing vulnerabilities by severity, affected versions, and recommended fixes.

2.2 Veracode Pipeline Scan

The Veracode tool uploads an artifact (JAR, WAR, EAR, ZIP, APK) to the pipeline scan API, polls scan status, and retrieves findings. It returns a Markdown report with findings by severity and location.

3 Research Questions

1. Does the OSS tool reduce vulnerability TTR compared to baseline workflows?
2. Does the Veracode tool reduce TTR or rework compared to baseline workflows?
3. Do the tools reduce the vulnerability backlog per sprint?
4. Is there a synergistic effect when both tools are used together?
5. What is the expected economic ROI from tool usage?

4 Data and Measurements

4.1 Outcome Variables

- **TTR (Time-to-Remediate)**: time from detection to verified fix (hours or days).
- **Backlog Count**: unresolved vulnerabilities per sprint.
- **Rework Rate**: reopened fixes or follow-up PRs per vulnerability.
- **Cycle Time** (optional): time from PR open to merge.

4.2 Treatment Variables

- `oss_usage_it`: count of OSS scans per project per sprint.
- `vera_usage_it`: count of Veracode pipeline scans per project per sprint.
- `oss_adopt_it`, `vera_adopt_it`: binary indicators for tool usage.
- Interaction: `oss_usage_it * vera_usage_it`.

4.3 Control Variables

- Project size (LOC, module count, dependency count).
- Team size, sprint length, release cadence.
- Language and build system.
- Time effects (calendar week or sprint).

4.4 Data Sources

- Tool telemetry logs (scan timestamps, scan IDs, findings count).
- Issue tracker data (vulnerability tickets and resolution timestamps).
- CI/CD logs (build time and artifact metadata).
- Repository analytics (PR cycle time, commit history).

5 Methodology

5.1 Hierarchical Bayesian Model

Let TTR_{it} be time-to-remediate for project i at time t. A log-normal model:

$$\log(TTR_{it}) \sim \mathcal{N}(\mu_{it}, \sigma)$$

$$\mu_{it} = \alpha_{team[i]} + \alpha_{proj[i]} + \delta_t + \beta_{oss} \cdot oss\_usage_{it} + \beta_{vera} \cdot vera\_usage_{it} + \beta_{int} \cdot (oss\_usage_{it} \cdot vera\_usage_{it}) + \gamma \cdot X_{it}$$

Backlog count can be modeled with a Negative Binomial:

$$backlog_{it} \sim \text{NegBin}(\lambda_{it}, \phi)$$

$$\log(\lambda_{it}) = \alpha_{team} + \alpha_{proj} + \delta_t + \beta_{oss} \cdot oss\_usage_{it} + \beta_{vera} \cdot vera\_usage_{it} + \gamma \cdot X_{it}$$

5.2 Priors

After standardizing predictors (mean 0, std 1):

$$\beta_* \sim \mathcal{N}(0, 0.5)$$

$$\alpha_{team} \sim \mathcal{N}(0, \sigma_{team}), \quad \sigma_{team} \sim \text{HalfNormal}(0.5)$$

$$\alpha_{proj} \sim \mathcal{N}(0, \sigma_{proj}), \quad \sigma_{proj} \sim \text{HalfNormal}(0.5)$$

$$\sigma \sim \text{HalfNormal}(0.5), \quad \phi \sim \text{HalfNormal}(1.0)$$

5.3 Causal Considerations

To reduce selection bias:

- Include time fixed effects (δ_t) for calendar shocks.
- Use difference-in-differences when adoption is staged.
- Control for project and team random effects.

6 Inference and Computation

Inference can be performed using Stan or PyMC with 4 chains and 2000-4000 iterations. Convergence is validated using $\hat{R} < 1.01$ and sufficient effective sample sizes.

Posterior summaries to report:

- $P(\beta_{oss} < 0)$ and $P(\beta_{vera} < 0)$ (improvement probability for TTR).
- Median and 95% credible intervals for effect sizes.
- Expected hours saved per sprint.

7 Results (Template)

7.1 Remediation Time Reduction

Placeholder values below should be replaced with actual estimates:

- $P(\beta_{oss} < 0) = [0.92]$
- Median TTR reduction (OSS) = [18%], 95% CI [4%, 31%]
- $P(\beta_{vera} < 0) = [0.88]$
- Median TTR reduction (Veracode) = [12%], 95% CI [2%, 24%]

7.2 Backlog Reduction

Median backlog change per sprint: $[-1.8]$, 95% CI $[-3.1, -0.5]$.

7.3 Synergy

Interaction effect: $\beta_{int} = [-0.08]$, 95% CI $[-0.15, -0.01]$.

7.4 ROI

Expected hours saved per sprint multiplied by effective cost per hour yields net ROI. Report median and 95% credible interval:

$$\text{ROI} = [\text{median}], \quad 95\% \text{ CI} = [\text{low}, \text{high}]$$

8 Sample Figures

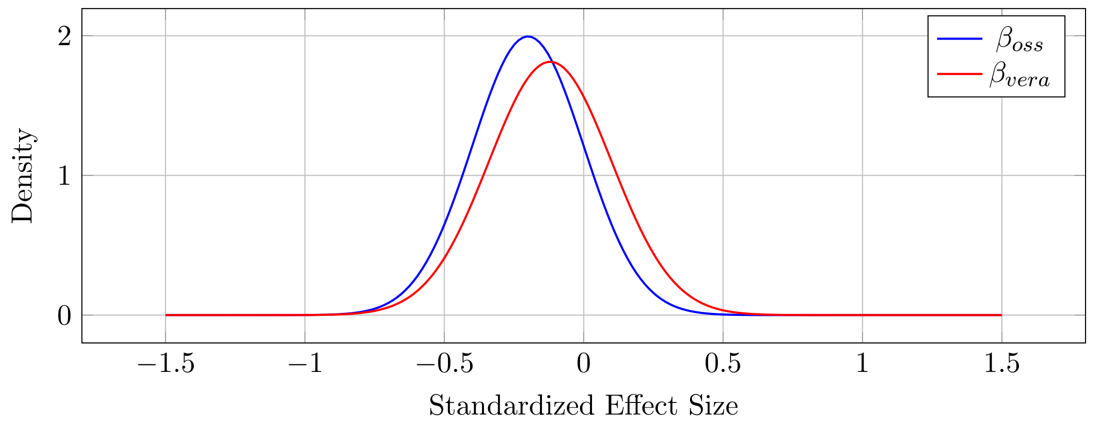

Figure 1: Sample posterior densities for OSS and Veracode effects (placeholder).

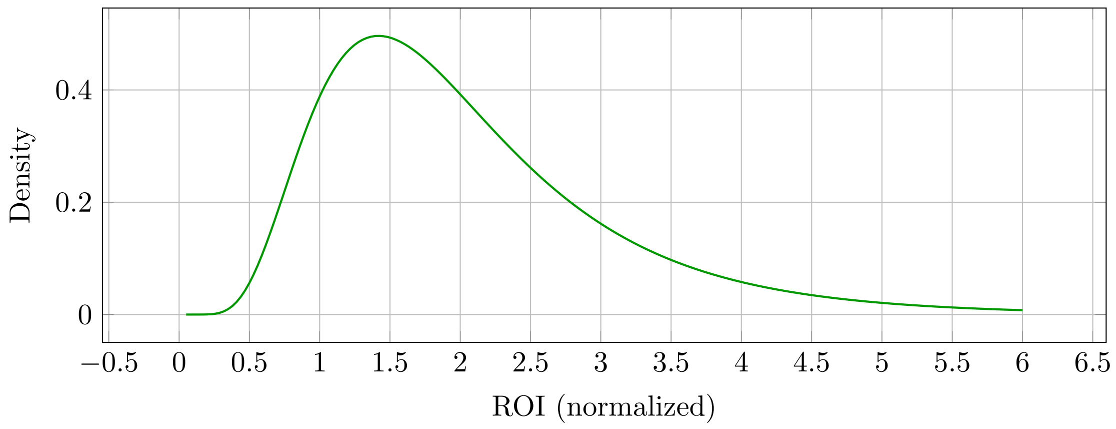

Figure 2: Sample ROI distribution (placeholder).

9 Discussion

The Bayesian framework provides interpretable probabilities of improvement and robust uncertainty estimates. Both tools are expected to reduce remediation time and vulnerability backlog, with possible synergy when used together. These results can guide adoption decisions across teams with heterogeneous workflows.

10 Threats to Validity

- Selection bias: early adopters may already be more efficient.
- Measurement error: missing or inconsistent timestamps.
- External shocks: release pressure or policy changes.

Mitigations include fixed effects, hierarchical modeling, and sensitivity checks.

11 Conclusion

This paper provides a statistical framework for quantifying productivity benefits of security automation tools. Bayesian inference enables decision-grade estimates of improvement probability and ROI. The framework is reusable for other developer tooling initiatives.

A Example Data Schema

- `project_id`, `team_id`, `sprint_id`
- `oss_scans_count`, `veracode_scans_count`
- `ttr_hours`, `vuln_backlog_count`, `rework_count`
- `loc`, `dependency_count`, `team_size`

B Analysis Pipeline (Sketch)

1. Extract telemetry and ticket data.
2. Normalize fields and standardize predictors.
3. Fit hierarchical model and validate convergence.
4. Report posterior summaries and ROI distribution.