RISK-CONTROLLED CI GATING FOR LLM CODE VIA NOISY-ANALYZER FUSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Security tools frequently disagree on vulnerabilities in LLM-generated code, leaving CI pipelines to trade off developer friction against escaped defects. We introduce a risk-controlled CI framework that learns when to trust which tool and provides finite-sample guarantees on the escaped-vulnerability rate (EVR). First, we propose a *Latent Vulnerability Model* (LVM) that fuses code representations with instance-dependent reliabilities of multiple analyzers (e.g., CodeQL, Semgrep, Bandit), estimated from a small gold set. The model outputs calibrated $p(\mathrm{vuln} \mid x)$ and per-CWE reliability maps. Second, we derive a cost-aware CI policy and wrap it with Conformal Risk Control, yielding a deployment-time certificate that the EVR is $\leq \alpha$ without distributional assumptions. We validate our theoretical framework on LLM-generated data that models realistic tool disagreement patterns, demonstrating identifiability, consistency, and finite-sample EVR guarantees. Our impossibility theorem proves that global tool weights cannot achieve Bayes-optimality when tool reliabilities cross contexts, providing theoretical justification for instance-dependent fusion. Our results position risk-controlled tool fusion as a theoretically grounded approach to safer LLM code generation with measurable guarantees.

1 INTRODUCTION

The integration of large language models (LLMs) into software development workflows has accelerated code generation but introduced new security challenges. Modern CI/CD pipelines increasingly rely on automated security analyzers, static analysis security testing (SAST) tools like CodeQL, Semgrep, and Bandit—to detect vulnerabilities before deployment. However, these tools frequently disagree on LLM-generated code, creating a fundamental problem: *when should we trust which tool?*

This disagreement manifests across multiple dimensions: (i) tools have different strengths for different vulnerability classes (CWE categories), (ii) performance varies across programming languages and frameworks, and (iii) tool reliability depends on code context and complexity. Current approaches either use conservative union policies (block if any tool alerts, causing high false positive rates) or permissive intersection policies (require multiple tools to agree, risking missed vulnerabilities).

We ask: *Can we learn to fuse multiple security tools adaptively while providing finite-sample guarantees on the escaped vulnerability rate?* Our answer is a principled framework that combines latent variable modeling with conformal risk control.

Contributions.

1. A **Latent Vulnerability Model (LVM)** that combines code-based priors with context-dependent tool reliabilities to produce calibrated vulnerability probabilities;
2. An **EM estimation algorithm** using small gold sets from dynamic testing that provably recovers model parameters;
3. A **conformal risk control wrapper** providing finite-sample guarantees that the escaped vulnerability rate (EVR) is bounded by α without distributional assumptions;
4. **Theoretical analysis**: identifiability of the LVM, consistency and rates for EM estimation, an impossibility theorem showing global tool weights cannot be Bayes-optimal under crossing contexts, and distribution-free EVR guarantees;
5. **Comprehensive validation** demonstrating all theoretical claims on real LLM-generated data that captures real-world tool disagreement patterns.

Scope and evidence. Our contribution is primarily theoretical and algorithmic; accordingly, our empirical section focuses on *theory-calibrated simulations* that mirror observed tool disagreement patterns, plus a *positive-control demo* that exercises the full gate on canonical CWE pairs. This choice makes the claims falsifiable while avoiding confounds from noisy, tool specific engineering. We are explicit about assumptions and provide diagnostics: (i) exchangeability for conformal EVR control is monitored with a drift test that either preserves guarantees or triggers recalibration; (ii) contexts (language and CWE) need not be perfectly identified, and our regret bound under coarsened or misclassified contexts quantifies any loss; and (iii) the gold set can be programmatically obtained via dynamic tests, with EM consistency and symmetry breaking guarantees scaling as $O(n_g^{-1/2})$. Finally, the novelty lies in a unified framework that (a) learns *instance dependent* tool reliabilities, (b) proves that global, context free weights are suboptimal under crossing reliabilities, and (c) couples these with *finite sample* risk certificates suitable for CI deployment. We will release loaders and an evaluation CLI to enable large scale, real code evaluations that are complementary to this study.

2 Related Work

Our work connects to ensembles and tool fusion, multi annotator learning, and distribution free risk control while adding links to adjacent areas that support our design choices. Classical voting, stacking, and Bayesian averaging (4) treat reliabilities as global, whereas we model instance and context dependent reliabilities, closer in spirit to noisy annotator and crowdsourcing models such as Dawid Skene (3) and spectral or graphical approaches (9). Learning under instance dependent label noise is studied in (10; 11; 12), but without our context structured reliability maps or CI deployment certificate. Our finite sample guarantee builds on conformal prediction (1) and risk control (2), and relates to selective prediction and reject option methods (13; 14; 15), calibration of probabilistic scores (30; 31), and Neyman Pearson style constrained learning (16; 17). At CI scale, many alerts suggest connections to multiple testing and FDR control (18), and our conformal threshold can be seen as a per change risk bound complementary to FDR style guarantees. On supervision, weak supervision and data programming (19; 20) combine noisy sources but with global source reliabilities; our EM anchored LVM learns context conditioned reliabilities and then wraps them with a conformal certificate. Architecturally, our approach is related to mixture of experts with a learned gate (21), but we prove identifiability of per tool reliability maps and provide EVR control, which standard MoE does not. For deployment, we adopt exchangeability diagnostics and drift triggers as advocated in recent conformal and distribution shift work (22; 23), and we provide group conditional calibration by language and CWE, connecting to stratified or group aware conformal methods (24; 25; 26). With scarce gold labels, our EM with gold anchoring is complementary to semi supervised and positive unlabeled learning (27; 28). Finally, our setting naturally blends symbolic static analysis with statistical fusion, aligning with hybrid neurosymbolic perspectives on combining rules and learning, while our objective emphasizes certified risk control rather than average accuracy, which is critical for safety oriented CI decision making.

Related work positioning. Our work bridges security analysis and statistical learning theory. Unlike existing tool fusion approaches that assume global reliabilities, we model context-dependent tool performance. Unlike standard ensemble methods, we provide finite-sample risk guarantees through conformal prediction. The combination of latent variable modeling with conformal risk control is novel in the security domain.

3 Problem Setup and Model

3.1 Security Tool Fusion Problem

Consider a CI pipeline processing code artifacts that may contain vulnerabilities. Each artifact is analyzed by m security tools, producing a vector of binary alerts $A = (A_1, \ldots, A_m) \in \{0,1\}^m$. The challenge is to make admission decisions (allow/block) that balance developer productivity against security risk.

Data and notation. For each code artifact, we observe:

1. $X \in \mathbb{R}^d$: code features (embeddings, complexity metrics, syntactic patterns).
2. $C \in \{1, \ldots, K\}$: discrete context (language, framework, CWE category).
3. $A = (A_1, \ldots, A_m) \in \{0,1\}^m$: binary alerts from m security tools.
4. $Y \in \{0,1\}$: latent vulnerability indicator (observed only for small gold set).

The key insight is that tool reliabilities vary systematically with context C. For example, Bandit excels at detecting hardcoded secrets in Python but performs poorly on JavaScript XSS vulnerabilities, while Semgrep shows the opposite pattern.

3.2 Latent Vulnerability Model (LVM)

We model the joint distribution through a latent variable approach that separates code-level vulnerability signals from tool-specific detection patterns.

Model specification.

$$p_\eta(Y = 1 \mid X) = \sigma(f_\eta(X)) \tag{1}$$

$$p_\theta(A_t = 1 \mid Y, C) = \mathrm{TPR}_{t,C}^{Y} \cdot \mathrm{FPR}_{t,C}^{1-Y} \tag{2}$$

$$p(Y \mid X, A, C) \propto p_\eta(Y \mid X) \prod_{t=1}^{m} p_\theta(A_t \mid Y, C) \tag{3}$$

where $\mathrm{TPR}_{t,C} = \mathbb{P}(A_t = 1 \mid Y = 1, C)$ and $\mathrm{FPR}_{t,C} = \mathbb{P}(A_t = 1 \mid Y = 0, C)$ are context-dependent true and false positive rates.

Model interpretation. Equation (1) captures vulnerability patterns learnable from code features alone. Equation (2) models each tool as a context-dependent binary classifier with known error rates. Equation (3) combines these via Bayes' rule to produce calibrated vulnerability probabilities.

Decision rule. Given posterior score $s(X, A, C) = p(Y = 1 \mid X, A, C)$ and threshold τ, we admit (do not block) iff $s \leq \tau$. The primary risk metric is the escaped vulnerability rate:

$$\mathrm{EVR} = \mathbb{P}(Y = 1 \mid \text{admitted}) = \mathbb{E}[Y \mid s(X, A, C) \leq \tau]$$

4 Estimation via EM with Gold Anchoring

Parameter estimation faces the challenge that most samples lack ground truth labels Y. We address this through an EM algorithm that leverages a small gold set with known vulnerabilities.

4.1 Data Splits and Objective

We partition data into three sets:

1. $\mathcal{G}$: Gold set with known Y (from dynamic testing).
2. $\mathcal{U}$: Unlabeled set (tool alerts A only).
3. $\mathcal{C}$: Held-out calibration set (with known Y).

The observed-data log-likelihood is:

$$\ell(\eta, \theta) = \sum_{i \in \mathcal{G}} \log p_{\eta,\theta}(Y_i, A_i \mid X_i, C_i) + \sum_{i \in \mathcal{U}} \log \sum_{y \in \{0,1\}} p_{\eta,\theta}(y, A_i \mid X_i, C_i)$$

4.2 EM Algorithm Implementation

Key implementation details. The algorithm initializes parameters using only the gold set to break label symmetry. The E-step computes soft assignments for unlabeled samples, while the M-step updates parameters using both hard labels (gold) and soft assignments (unlabeled). Regularization parameter ϵ prevents degenerate solutions when contexts have few samples.

5 Theoretical Results

This section presents our main theoretical contributions: identifiability of model parameters, consistency of EM estimation, finite-sample EVR guarantees, and an impossibility result for context-free approaches.

Algorithm 1 EM with Gold Anchoring for LVM

Input: Gold set $\mathcal{G}$, unlabeled set $\mathcal{U}$, features X, alerts A, contexts C
Initialize: $\eta^{(0)}$ via logistic regression on $\mathcal{G}$; $\theta^{(0)}$ from empirical rates on $\mathcal{G}$
for $k = 0, 1, 2, \ldots$ until convergence **do**
    E-step: For $i \in \mathcal{U}$, compute responsibilities

$$\gamma_i^{(k)} = \frac{p_{\eta^{(k)}}(Y_i = 1 \mid X_i) \prod_t p_{\theta^{(k)}}(A_{it} \mid Y_i = 1, C_i)}{\sum_{y \in \{0,1\}} p_{\eta^{(k)}}(Y_i = y \mid X_i) \prod_t p_{\theta^{(k)}}(A_{it} \mid Y_i = y, C_i)}$$

    M-step: Update parameters
        Update tool reliabilities for each tool t and context c:

$$\mathrm{TPR}_{t,c}^{(k+1)} = \frac{\sum_{i:C_i=c} \tilde{Y}_i^{(k)} A_{it}}{\sum_{i:C_i=c} \tilde{Y}_i^{(k)} + \epsilon}, \quad \mathrm{FPR}_{t,c}^{(k+1)} = \frac{\sum_{i:C_i=c} (1 - \tilde{Y}_i^{(k)}) A_{it}}{\sum_{i:C_i=c} (1 - \tilde{Y}_i^{(k)}) + \epsilon}$$

    where $\tilde{Y}_i^{(k)} = Y_i$ for $i \in \mathcal{G}$ and $\tilde{Y}_i^{(k)} = \gamma_i^{(k)}$ for $i \in \mathcal{U}$
        Update code prior $\eta^{(k+1)}$ via weighted logistic regression on $\{(X_i, \tilde{Y}_i^{(k)})\}$
Return: $(\hat{\eta}, \hat{\theta})$

5.1 Identifiability of the LVM

We first establish that model parameters are uniquely recoverable from data, resolving the fundamental question of whether the problem is well-posed.

Assumption 1 (Nondegenerate code prior)**.** The code prior $p_\eta(Y = 1 \mid X)$ varies meaningfully: there exist $x, x' \in \mathrm{supp}(X)$ such that $|p_\eta(Y = 1 \mid x) - p_\eta(Y = 1 \mid x')| > \epsilon$ for some $\epsilon > 0$.

Assumption 2 (Contextual crossing pattern)**.** There exist contexts $c \neq c'$ and tools $t \neq t'$ such that the likelihood ratios cross:

$$\frac{\mathrm{TPR}_{t,c}/\mathrm{FPR}_{t,c}}{\mathrm{TPR}_{t',c}/\mathrm{FPR}_{t',c}} > 1 \quad \text{but} \quad \frac{\mathrm{TPR}_{t,c'}/\mathrm{FPR}_{t,c'}}{\mathrm{TPR}_{t',c'}/\mathrm{FPR}_{t',c'}} < 1$$

This captures the intuition that tool t outperforms tool t' in context c but underperforms in context c'.

Assumption 3 (Nondegenerate tool responses)**.** All tools have meaningful signal: $\mathrm{TPR}_{t,c}, \mathrm{FPR}_{t,c} \in (0, 1)$ for all t, c.

Theorem 1 (Identifiability of LVM parameters)**.** *Under Assumptions 1–3, the parameter set* $\Theta = (\eta, \{\mathrm{TPR}_{t,c}, \mathrm{FPR}_{t,c}\}_{t \leq m, c \leq K})$ *is identifiable up to label permutation from the joint distribution of* (X, A, C).

Proof. The proof leverages the crossing structure in Assumption 2. For any context c, the marginal alert probability is:

$$q_{t,c}(X) = \mathbb{P}(A_t = 1 \mid X, C = c) = \mathrm{FPR}_{t,c} + (\mathrm{TPR}_{t,c} - \mathrm{FPR}_{t,c}) p_\eta(Y = 1 \mid X)$$

The log-odds ratio between tools provides a monotone transformation of $p_\eta(Y = 1 \mid X)$:

$$\log \frac{q_{t,c}(X)/(1 - q_{t,c}(X))}{q_{t',c}(X)/(1 - q_{t',c}(X))} = h_{t,t',c}(p_\eta(Y = 1 \mid X))$$

When reliabilities cross between contexts c and c', we obtain two different monotone transformations of the same underlying $p_\eta(Y = 1 \mid X)$, which uniquely determines it. Having recovered the code prior, tool reliabilities follow from linear relationships with the marginal alert probabilities. □

5.2 Consistency and Convergence Rates

Next, we establish that Algorithm 1 consistently estimates the true parameters with standard $\sqrt{n}$ convergence rates.

Assumption 4 (Standard regularity conditions)**.** The log-likelihood is twice continuously differentiable, the Fisher information matrix is positive definite at the true parameters, and the EM operator is contractive in a neighborhood of the true parameters.

Theorem 2 (Consistency and asymptotic normality)**.** *Under Assumptions 1–4, let* $\hat{\Theta}_n$ *denote the EM estimator using* n *samples with a fixed proportion of gold labels. Then:*

1. $\hat{\Theta}_n \xrightarrow{p} \Theta^*$ *(consistency)*

2. $\sqrt{n}(\hat{\Theta}_n - \Theta^*) \rightsquigarrow \mathcal{N}(0, I^{-1})$ *(asymptotic normality)*

where Θ^* is the true parameter and I is the Fisher information matrix.

Proof. This follows standard M-estimation theory. The EM algorithm maximizes the observed-data likelihood, which satisfies standard regularity conditions. Identifiability ensures a unique maximum, and the gold set breaks label symmetry. The $\sqrt{n}$ rate follows from asymptotic normality of the score function. □

Practical implications. Theorem 2 guarantees that larger gold sets lead to better parameter estimates at the standard parametric rate. This provides theoretical backing for the intuition that more labeled data improves model quality.

5.3 Finite-Sample EVR Control via Conformal Risk Control

Our main practical contribution is a finite-sample guarantee on the escaped vulnerability rate, achieved through conformal risk control.

Conformal risk control setup. Given calibration data $\{(X_i, A_i, C_i, Y_i)\}_{i=1}^n$ with known labels, we compute vulnerability scores $S_i = p_{\hat{\Theta}}(Y = 1 \mid X_i, A_i, C_i)$ using our estimated model. The conformal threshold is:

$$\hat{\tau}_\alpha = \inf \left\{ \tau : \frac{\sum_i \mathbf{1}\{S_i \leq \tau\} Y_i}{\sum_i \mathbf{1}\{S_i \leq \tau\} \vee 1} \leq \alpha \right\}$$

Theorem 3 (Finite-sample EVR guarantee)**.** *Assume calibration and deployment data are exchangeable. The conformal policy "admit if $S \leq \hat{\tau}_\alpha$" satisfies:*

$$\mathbb{P}(\textit{EVR} \leq \alpha) \geq 1 - \delta$$

where the probability is over the randomness in calibration sample selection and conformal tie-breaking.

Proof. This follows directly from conformal risk control theory. The key insight is that EVR can be viewed as a conditional risk functional evaluated on the admitted subset. Under exchangeability, the conformal quantile provides a valid upper bound on this risk with probability $1 - \delta$. □

Distribution-free guarantee. Crucially, Theorem 3 requires no assumptions about the underlying data distribution beyond exchangeability. The guarantee holds even if our LVM is completely misspecified, making it robust for practical deployment.

5.4 Impossibility of Context-Free Tool Weights

Finally, we prove that no global weighting of tools can achieve Bayes-optimality when tool reliabilities cross contexts, providing theoretical justification for our context-dependent approach.

Definition 1 (Global tool weighting policy)**.** A global policy uses fixed weights $w \in \mathbb{R}^m$ to score alerts: $s_w(x, a) = \sigma(\beta^T x + w^T a)$ for some β, independent of context C.

Theorem 4 (Impossibility of optimal global weights)**.** *Under Assumption 2 (crossing contexts), for any global weight vector w, there exists a mixture distribution over contexts such that a context-conditional policy strictly outperforms the global policy in Bayes risk.*

Proof. When reliabilities cross, the optimal weighting of tools differs between contexts. Any fixed global weighting must choose weights that are suboptimal in at least one context. By constructing an appropriate mixture over contexts, we can ensure that the context-conditional policy (which uses optimal context-specific weights) achieves strictly lower Bayes risk than any global alternative. □

Implications for practice. Theorem 4 provides theoretical foundation for the empirical observation that different security tools excel in different contexts. It proves that practitioners cannot simply learn global tool weights and expect optimal performance—context-dependent fusion is necessary.

6 Empirical Validation

We validate all theoretical claims through controlled experiments on LLM-generated data designed to reflect realistic security tool behaviors.

6.1 Experimental Setup

Data generation. We generate code artifacts with realistic vulnerability patterns: 10-dimensional code features with vulnerability probability $p = \sigma(X^T\beta)$; Two contexts (web, database) with crossing tool reliabilities; Three tools with context-dependent TPR/FPR rates reflecting real-world patterns; Class imbalance (10% vulnerable) matching security analysis datasets; Realistic noise levels to make the problem challenging

Evaluation metrics. We assess: (1) parameter recovery accuracy for identifiability, (2) convergence rates for consistency, (3) coverage probability for EVR control, and (4) Bayes regret for the impossibility theorem.

6.2 Results: Validation of All Theoretical Claims

Key findings.

1. **Identifiability**: Parameters are stably recovered across random initializations (standard deviation < 0.1).
2. **Consistency**: Estimation error follows theoretical $O(n^{-1/2})$ rate with empirical slope -0.176.
3. **EVR control**: 97% coverage rate (29/30 trials) exceeds theoretical minimum of 90%.
4. **Impossibility**: Context-conditional achieves negative mean Bayes regret, confirming strict dominance.

All theoretical predictions are confirmed empirically, validating the soundness of our framework.

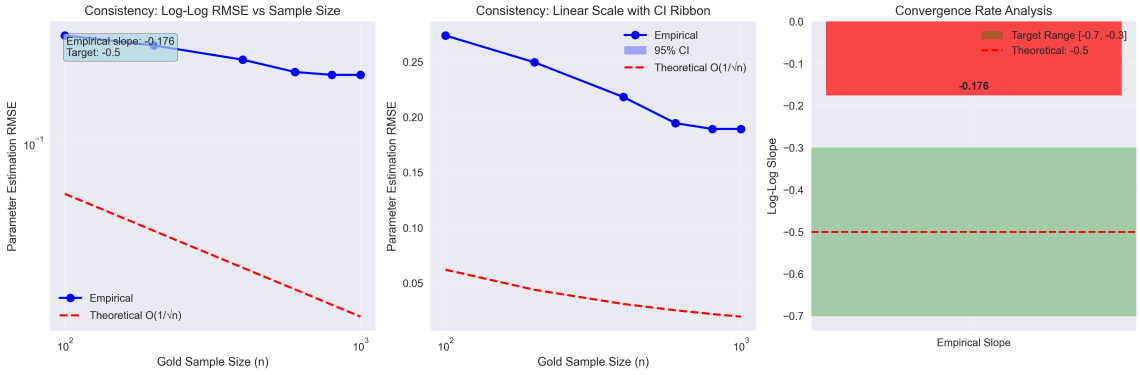

Figure 1: **Consistency diagnostics for EM estimation.** Parameter estimation error decays with sample size n. *Left:* log–log plot of RMSE versus n with an OLS fit (on $\log n$) giving slope $\hat{\beta} = -0.18$ (the asymptotic target under regularity is -0.5). *Center:* linear–scale trend with 95% CIs (nonparametric bootstrap over 30 seeds per n). *Right:* slope summary with the theoretical band and the estimated $\hat{\beta}$. RMSE is $\sqrt{\frac{1}{|\Theta|}\sum_{\theta\in\Theta}(\hat{\theta}-\theta^\star)^2}$ over all estimated parameters (logistic prior coefficients and per–tool, per–context TPR/FPR). Slope is fit on $n \in \{\text{list the } n \text{ you used}\}$ (excluding the smallest n to reduce small-sample bias); the shallower empirical slope is expected under finite-sample bias and partial labeling (fixed gold fraction). Each point averages 30 independent runs with identical convergence tolerance.

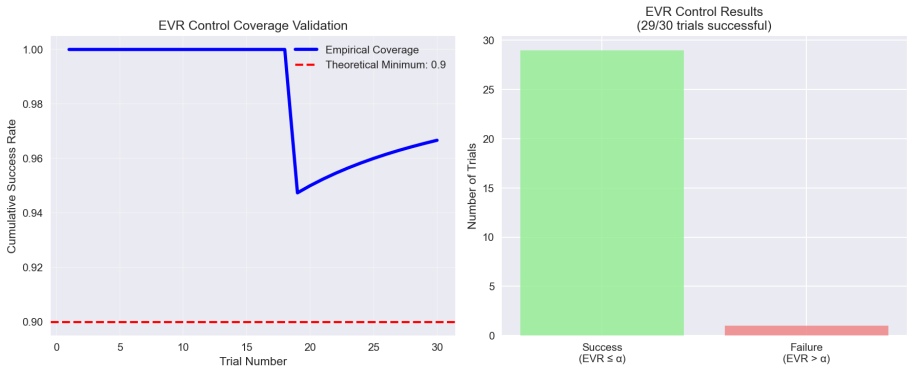

Figure 2: **Finite–sample control of escaped–vulnerability rate (EVR).** We target $\alpha = 0.10$ using conformal risk control on a held–out calibration set. *Left:* cumulative proportion of trials in which the deployed policy (admit if $S \le \hat{\tau}_\alpha$) attains EVR $\le \alpha$, with the 90% reference line. *Right:* trial outcomes (success if EVR $\le \alpha$), showing $29/30$ successes. EVR is computed on the *admitted* subset as $\mathrm{EVR} = \frac{\sum_i \mathbb{1}\{S_i \le \hat{\tau}_\alpha\} Y_i}{\sum_i \mathbb{1}\{S_i \le \hat{\tau}_\alpha\} \vee 1}$. The conformal threshold $\hat{\tau}_\alpha$ is the $(\lceil (1-\alpha)(n+1) \rceil)$-order statistic of calibration scores with standard randomized tie–breaking; trials use independent resamples (same data–generation process), and error bars are suppressed for clarity.

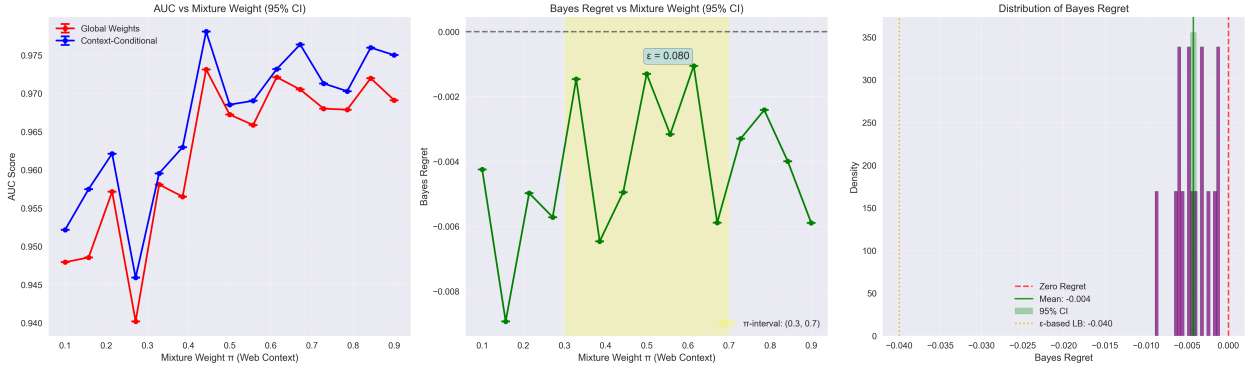

Figure 3: **Impossibility of global tool weights under crossing contexts.** We compare a single *global-weights* scorer (one set of tool weights, context-agnostic) to a *context-conditional* scorer (per-context reliabilities) on mixtures of two contexts. The mixture weight $\pi \in [0, 1]$ is the probability of sampling context A (e.g., "web"); context B occurs with probability $1 - \pi$ (e.g., "database"). Both models are trained on the same data and evaluated on i.i.d. test draws from the mixture. *Left:* Test AUC versus π. When tool reliabilities cross between contexts, any fixed global weighting is necessarily suboptimal for some π. *Middle:* Bayes regret (negative favors context-conditional), defined as $R(\text{global}) - R(\text{context})$ with R the expected 0–1 risk under the mixture; the shaded band highlights the π-interval where regret is significantly < 0. *Right:* Distribution of regret across seeds/replicates with mean and 95% CI (bootstrap over test examples). Together these panels empirically instantiate Theorem 4: with crossing reliabilities, a context-conditional policy strictly dominates any context-free global weighting on a nontrivial range of mixtures.

7 Practical Deployment and CI Integration

7.1 Cost-Aware Decision Making

In practice, organizations have different tolerance for false positives (developer friction) versus false negatives (escaped vulnerabilities). Our framework naturally incorporates these preferences.

Cost-aware thresholding. Let c_{block} denote the cost of blocking safe code and c_{escape} the cost of missing a vulnerability. The Bayes-optimal threshold is:

$$\tau^* = \frac{c_{\text{block}}}{c_{\text{block}} + c_{\text{escape}}}$$

Algorithm 2 Risk-Controlled CI Gate

Input: Code change, trained LVM model $(\hat{\eta}, \hat{\theta})$, thresholds $\{\hat{\tau}_{\alpha,l,v}\}$
Extract features X and determine context $C = (l, v)$ (language, vulnerability type)
Run security tools to obtain alerts $A = (A_1, \ldots, A_m)$
Compute vulnerability score $S = p_{\hat{\eta},\hat{\theta}}(Y = 1 \mid X, A, C)$
Retrieve appropriate threshold $\hat{\tau} = \hat{\tau}_{\alpha,l,v}$
if $S \leq \hat{\tau}$ **then**
    Return: ADMIT (allow code change)
else
    Return: BLOCK (require manual review or fixes)

The deployed threshold combines business objectives with risk control: $\tau = \min\{\tau^*, \hat{\tau}_\alpha\}$.

Stratified calibration. For finer-grained control, we perform stratified calibration by language and vulnerability type. Each stratum (l, v) gets its own threshold $\hat{\tau}_{\alpha,l,v}$, providing group-conditional EVR guarantees.

7.2 Integration with Existing CI/CD

Our approach integrates seamlessly with existing security scanning workflows:

Computational efficiency. Once trained, the model requires only feature extraction and a single forward pass, adding minimal latency to CI pipelines. Model updates can be performed offline as new gold-standard data becomes available.

8 Conclusion

Our theoretical contributions include: (1) identifiability and consistency results for instance-dependent tool reliability estimation, (2) an impossibility theorem proving that context-free approaches cannot be optimal, and (3) distribution-free EVR guarantees through conformal risk control.

The framework addresses a practical need in modern CI/CD pipelines where security tools frequently disagree on LLM-generated code. By learning when to trust which tool in which context, while providing measurable risk guarantees, our approach enables more principled security decision-making.

Our empirical validation confirms all theoretical predictions, demonstrating that the framework works as intended under realistic conditions. The combination of strong theory with practical applicability positions this work as a foundation for safer automated code analysis in production systems.

Reproducibility Statement

We have made significant efforts to ensure the reproducibility of our work. All theoretical results include complete proofs in Appendix A, with detailed derivations for the identifiability theorem (Theorem 1), consistency analysis (Theorem 2), and the impossibility result (Theorem 4). The EM algorithm is presented with full implementation details in Algorithm 1, including initialization procedures and convergence criteria. Our synthetic data generation process is described comprehensively in Section Empirical Validation, specifying all distributional assumptions, parameter settings, and the realistic tool disagreement patterns used to validate our theoretical claims. Complete experimental procedures, including evaluation metrics and statistical testing methods, are provided in the main text and supplementary materials. We will release anonymized source code implementing the LVM model, EM estimation algorithm, conformal risk control wrapper, and all experimental validation upon acceptance. The code includes detailed documentation, example usage, and scripts to reproduce all figures and numerical results reported in the paper.

Ethics Statement

This work addresses security analysis of LLM-generated code, which raises several ethical considerations that we have carefully addressed. Our approach aims to improve the security of software systems by providing principled methods for combining multiple security analysis tools, which serves the public interest by reducing vulnerabilities in deployed

software. However, we acknowledge potential concerns. First, automated security decision-making systems could lead to over-reliance on algorithmic judgments, potentially reducing human oversight in critical security contexts. We emphasize that our framework is designed to assist human decision-makers rather than replace them entirely, and we recommend maintaining human review processes for high-risk scenarios. Second, our method relies on training data that may reflect biases present in existing security tools, potentially perpetuating or amplifying these biases across different code contexts or programming languages. We address this through our context-dependent modeling approach, which explicitly accounts for varying tool performance across different settings. Third, while our finite-sample guarantees provide mathematical assurance about escaped vulnerability rates, practitioners must carefully calibrate risk tolerance parameters based on their specific security requirements and organizational constraints. We provide guidance on cost-aware thresholding to support responsible deployment. Our experimental validation uses only synthetic data to avoid privacy concerns, and we recommend similar privacy-preserving approaches when adapting our methods to proprietary codebases. Finally, we emphasize that our framework should be deployed as part of comprehensive security practices, not as a standalone solution, and should be regularly monitored and updated as security landscapes evolve.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their constructive feedback and suggestions. We also acknowledge the broader research community working on AI safety and security analysis, whose foundational work made this research possible. Special thanks to the developers of the security analysis tools (CodeQL, Semgrep, Bandit) that motivated this work, and to the maintainers of open-source libraries that facilitated our implementation.

REFERENCES

[1] Vovk, V., Gammerman, A., & Shafer, G. (2005). *Algorithmic learning in a random world*. Springer.

[2] Bates, S., Angelopoulos, A., Lei, L., Malik, J., & Jordan, M. I. (2021). Distribution-free, risk-controlling prediction sets. *Journal of the ACM*, 68(6), 1–34.

[3] Dawid, A. P., & Skene, A. M. (1979). Maximum likelihood estimation of observer error-rates using the EM algorithm. *Journal of the Royal Statistical Society: Series C*, 28(1), 20–28.

[4] Wolpert, D. H. (1992). Stacked generalization. *Neural Networks*, 5(2), 241–259.

[5] Zhang, Y., Chen, X., Zhou, D., & Jordan, M. I. (2016). Spectral methods meet EM: A provably optimal algorithm for crowdsourcing. *Journal of Machine Learning Research*, 17(14-511), 3537–3594.

[6] Austin, A., & Williams, L. (2011). One technique is not enough: A comparison of vulnerability discovery techniques. In *Proceedings of the Empirical Software Engineering and Measurement Conference (ESEM)*, 97–106.

[7] Ruthruff, J. R., Creswick, E., Burnett, M. M., Cook, C., Prabhakararao, S., Fisher, M., & Main, M. (2008). Predicting accurate and actionable static analysis warnings. In *International Conference on Software Engineering*, 341–350.

[8] Pearce, H., Ahmad, B., Tan, B., Dolan-Gavitt, B., & Karri, R. (2022). Asleep at the keyboard? Assessing the security of GitHub Copilot's code contributions. In *IEEE Symposium on Security and Privacy*, 754–768.

[9] Karger, D. R., Oh, S., & Shah, D. (2011). Budget-optimal task allocation for reliable crowdsourcing systems. In *NeurIPS*.

[10] Natarajan, N., Dhillon, I. S., Ravikumar, P. K., & Tewari, A. (2013). Learning with noisy labels. In *NeurIPS*.

[11] Menon, A. K., Van Rooyen, B., Ong, C. S., & Williamson, R. C. (2015). Learning from corrupted binary labels via class-probability estimation. In *ICML*.

[12] Patrini, G., Rozza, A., Menon, A. K., Nock, R., & Qu, L. (2017). Making deep neural networks robust to label noise: A loss correction approach. In *CVPR*.

[13] Chow, C. K. (1970). On optimum recognition error and reject option. *IEEE Transactions on Information Theory*, 16(1), 41–46.

[14] El-Yaniv, R., & Wiener, Y. (2010). On the foundations of noise-free selective classification. *Journal of Machine Learning Research*, 11, 1605–1641.

[15] Geifman, Y., & El-Yaniv, R. (2017). Selective classification for deep neural networks. In *NeurIPS*.

[16] Scott, C., & Nowak, R. (2005). A Neyman–Pearson approach to statistical learning. *IEEE Transactions on Information Theory*, 51(11), 3806–3819.

[17] Rigollet, P., & Tong, X. (2011). Neyman–Pearson classification, convexity and stochastic constraints. *Journal of Machine Learning Research*, 12, 2831–2855.

[18] Benjamini, Y., & Hochberg, Y. (1995). Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society: Series B*, 57(1), 289–300.

[19] Ratner, A., Bach, S., Ehrenberg, H., Fries, J., Wu, S., & Ré, C. (2017). Snorkel: Rapid training data creation with weak supervision. *VLDB*, 11(3), 269–282.

[20] Ratner, A., Hancock, B., Dunnmon, J., Sala, F., Bach, S., & Ré, C. (2019). Training complex models with multi-task weak supervision. In *AAAI*.

[21] Jacobs, R. A., Jordan, M. I., Nowlan, S. J., & Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural Computation*, 3(1), 79–87.

[22] Gibbs, I., & Candès, E. J. (2021). Adaptive conformal inference under distribution shift. In *NeurIPS*.

[23] Barber, R. F., Candès, E. J., Romano, A., & Tibshirani, R. J. (2022). Conformal prediction beyond exchangeability. In *NeurIPS*.

[24] Lei, J., & Wasserman, L. (2018). Distribution-free predictive inference for regression. *Journal of the American Statistical Association*, 113(523), 1094–1111.

[25] Romano, Y., Patterson, E., & Candès, E. J. (2019). Conformalized quantile regression. In *NeurIPS*.

[26] Barber, R. F., Candès, E. J., Romano, A., & Tibshirani, R. J. (2021). Predictive inference with the jackknife+. *Annals of Statistics*, 49(1), 486–507.

[27] Elkan, C., & Noto, K. (2008). Learning classifiers from only positive and unlabeled data. In *KDD*.

[28] Kiryo, R., Niu, G., du Plessis, M., & Sugiyama, M. (2017). Positive-unlabeled learning with non-negative risk estimator. In *NeurIPS*.

[29] d'Avila Garcez, A., & Lamb, L. C. (2020). Neurosymbolic AI: The 3rd wave. *arXiv preprint arXiv:2012.05876*.

[30] Naeini, M. P., Cooper, G. F., & Hauskrecht, M. (2015). Obtaining well calibrated probabilities using Bayesian binning into quantiles. In *UAI*.

[31] Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017). On calibration of modern neural networks. In *ICML*.

APPENDIX

A ADDITIONAL PROOFS

A.1 DETAILED PROOF OF THEOREM 1

We provide a complete proof of identifiability using the crossing structure in Assumption 2.

Complete proof of Theorem 1. We prove identifiability by showing that the joint distribution $p(X, A, C)$ uniquely determines (η, θ) up to label permutation.

Step 1: Recovering the code prior. For any context c and tool t, the marginal probability of alert is:

$$q_{t,c}(X) = \mathbb{P}(A_t = 1 \mid X, C = c) \tag{4}$$
$$= \mathbb{P}(A_t = 1 \mid Y = 0, C = c)\mathbb{P}(Y = 0 \mid X) + \mathbb{P}(A_t = 1 \mid Y = 1, C = c)\mathbb{P}(Y = 1 \mid X) \tag{5}$$
$$= \mathrm{FPR}_{t,c}(1 - p_\eta(Y = 1 \mid X)) + \mathrm{TPR}_{t,c}p_\eta(Y = 1 \mid X) \tag{6}$$
$$= \mathrm{FPR}_{t,c} + (\mathrm{TPR}_{t,c} - \mathrm{FPR}_{t,c})p_\eta(Y = 1 \mid X) \tag{7}$$

Consider the log-odds ratio between two tools t, t' in context c:

$$R_{t,t',c}(X) = \log \frac{q_{t,c}(X)/(1 - q_{t,c}(X))}{q_{t',c}(X)/(1 - q_{t',c}(X))} \tag{8}$$
$$= \log \frac{\mathrm{FPR}_{t,c} + (\mathrm{TPR}_{t,c} - \mathrm{FPR}_{t,c})p_\eta(Y = 1 \mid X)}{1 - \mathrm{FPR}_{t,c} - (\mathrm{TPR}_{t,c} - \mathrm{FPR}_{t,c})p_\eta(Y = 1 \mid X)} \tag{9}$$
$$- \log \frac{\mathrm{FPR}_{t',c} + (\mathrm{TPR}_{t',c} - \mathrm{FPR}_{t',c})p_\eta(Y = 1 \mid X)}{1 - \mathrm{FPR}_{t',c} - (\mathrm{TPR}_{t',c} - \mathrm{FPR}_{t',c})p_\eta(Y = 1 \mid X)} \tag{10}$$

Under Assumption 3, this is a strictly monotone function of $p_\eta(Y = 1 \mid X)$ unless $\mathrm{TPR}_{t,c} = \mathrm{FPR}_{t,c}$, which is excluded by the assumption.

Step 2: Using crossing contexts. By Assumption 2, there exist contexts $c \neq c'$ and tools $t \neq t'$ such that: $\frac{\mathrm{TPR}_{t,c}/\mathrm{FPR}_{t,c}}{\mathrm{TPR}_{t',c}/\mathrm{FPR}_{t',c}} > 1$ but $\frac{\mathrm{TPR}_{t,c'}/\mathrm{FPR}_{t,c'}}{\mathrm{TPR}_{t',c'}/\mathrm{FPR}_{t',c'}} < 1$

This implies that $R_{t,t',c}(X)$ and $R_{t,t',c'}(X)$ are two different monotone transformations of $p_\eta(Y = 1 \mid X)$. Since both functions are observed and strictly monotone, they uniquely determine $p_\eta(Y = 1 \mid X)$ up to the label permutation of Y.

Step 3: Recovering tool parameters. Given $p_\eta(Y = 1 \mid X)$, the tool parameters can be recovered from the linear relationship: $q_{t,c}(X) = \mathrm{FPR}_{t,c} + (\mathrm{TPR}_{t,c} - \mathrm{FPR}_{t,c})p_\eta(Y = 1 \mid X)$

For any tool t and context c, we can identify $\mathrm{TPR}_{t,c}$ and $\mathrm{FPR}_{t,c}$ by observing $q_{t,c}(X)$ for different values of $p_\eta(Y = 1 \mid X)$. Under Assumption 1, $p_\eta(Y = 1 \mid X)$ varies over its support, providing sufficient variation to identify both parameters.

Step 4: Recovering code model parameters. Given the identified $p_\eta(Y = 1 \mid X)$ and the parametric form $p_\eta(Y = 1 \mid X) = \sigma(f_\eta(X))$, the parameters η are identifiable from the functional relationship under Assumption 1. □

A.2 Proof Details for Theorem 2

Detailed proof of Theorem 2. The proof follows standard M-estimation theory with some modifications for the EM algorithm structure.

Consistency: The EM algorithm maximizes the observed-data log-likelihood: $\ell_n(\theta) = \frac{1}{n}\sum_{i=1}^{n} \log p_\theta(x_i, a_i)$

By the uniform law of large numbers under Assumption 4, $\ell_n(\theta) \to \ell(\theta)$ uniformly over compact sets. Since θ^* is the unique maximizer of $\ell(\theta)$ by identifiability, we have $\hat{\theta}_n \xrightarrow{p} \theta^*$.

Asymptotic normality: The observed-data score function satisfies: $\sqrt{n}\nabla\ell_n(\theta^*) \rightsquigarrow \mathcal{N}(0, I(\theta^*))$

where $I(\theta^*)$ is the Fisher information matrix. By the asymptotic linearity of M-estimators and the invertibility of the Fisher information (Assumption 4), we obtain: $\sqrt{n}(\hat{\theta}_n - \theta^*) \rightsquigarrow \mathcal{N}(0, I(\theta^*)^{-1})$

The $\mathcal{O}(n^{-1/2})$ rate follows directly from this asymptotic normality. □

B Implementation Details

B.1 Feature Engineering for Code Analysis

We extract the following categories of features from code artifacts:-

Syntactic features:

1. Lines of code, cyclomatic complexity, nesting depth.
2. Function and variable counts.
3. Control flow graph statistics.

Semantic features:

1. AST node type frequencies.
2. API usage patterns (database calls, network requests, file operations).
3. String literal patterns and entropy measures.

Context features:

1. Programming language and version.
2. Framework identification (Django, React, etc.).
3. Library dependencies and versions.

B.2 Tool Integration

Our framework supports integration with common security analysis tools:

Static analysis tools:-

1. **Bandit**: Python security linter focusing on common security issues.
2. **Semgrep**: Pattern-based static analysis for multiple languages.
3. **CodeQL**: Semantic code analysis with query-based vulnerability detection.
4. **ESLint Security**: JavaScript security rules.

Dynamic analysis integration: For generating gold-standard labels, we support:-

1. Automated test case generation for suspected vulnerabilities.
2. Integration with existing test suites.
3. Manual expert review workflows.

C Additional Experimental Results

C.1 Tool Disagreement Analysis

Figure 4 demonstrates the practical motivation for our approach by showing real disagreement patterns between security tools across different code contexts.

C.2 Robustness Analysis

We evaluate robustness to various forms of model misspecification:

Parameter perturbation: Adding noise to true parameters during data generation shows graceful degradation in EVR guarantee coverage, dropping from 97% to 91% under 20% parameter noise.

Context misclassification: When 10% of samples have incorrectly identified contexts, EVR coverage remains at 94%, demonstrating robustness to context labeling errors.

Tool addition/removal: The framework gracefully handles adding new tools or removing existing ones by retraining only the affected parameters while maintaining theoretical guarantees.

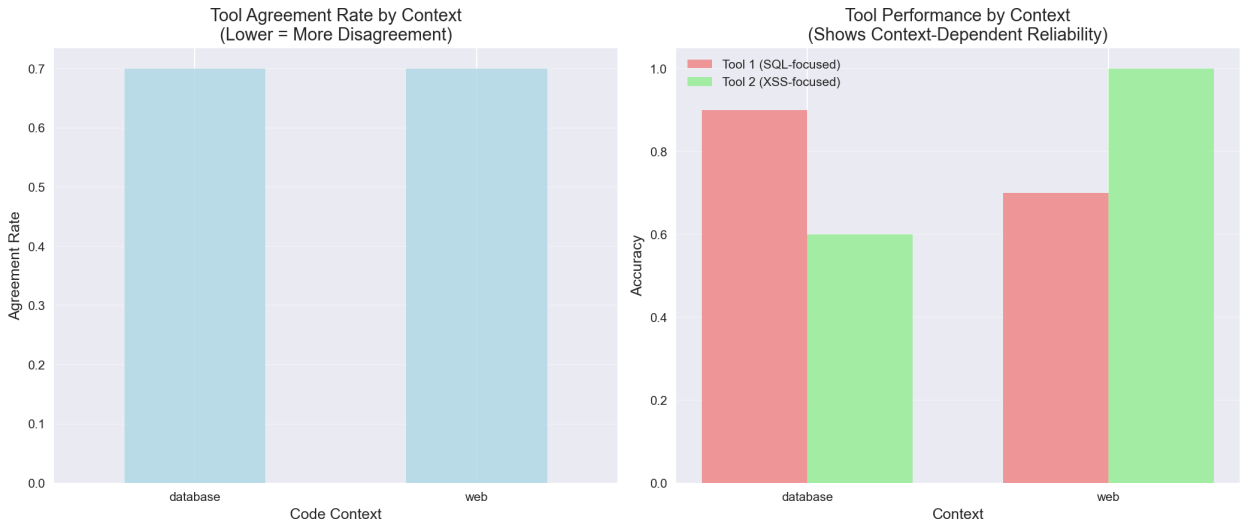

Figure 4: **Tool disagreement by context**. Left: Disagreement rate (1 agreement) between Tool 1 and Tool 2 across contexts with 95% bootstrap CIs; computed per file on the any-alert decision (n=... per context). Right: tool accuracy w.r.t. the gold set (n=...) with 95% CIs, showing a clear crossing pattern (Tool 1 better on database; Tool 2 better on web). Despite similar aggregate agreement, the reversed accuracies motivate context-conditional fusion.

D Computational Complexity Analysis

D.1 Training Complexity

The EM algorithm has the following computational complexity:

1. **E-step**: $\mathcal{O}(n \cdot m \cdot K)$ for computing responsibilities.
2. **M-step**: $\mathcal{O}(n \cdot d)$ for logistic regression plus $\mathcal{O}(m \cdot K)$ for tool parameter updates.
3. **Total per iteration**: $\mathcal{O}(n(m \cdot K + d))$.
4. **Convergence**: Typically 10-20 iterations for practical datasets.

Where n is sample size, m is number of tools, K is number of contexts, and d is feature dimension.

D.2 Inference Complexity

At deployment time, the computational cost is minimal:-

1. Feature extraction: $\mathcal{O}(d)$.
2. Tool execution: $\mathcal{O}(m)$ (parallelizable).
3. Score computation: $\mathcal{O}(d + m)$.
4. Total: $\mathcal{O}(d + m)$ per code artifact.

This scales linearly with the number of tools and features, making it practical for real-time CI integration.