TRACE: EVIDENCE-GROUNDED EPISTEMIC CONTROL FOR RELIABLE AUTONOMOUS PENETRATION TESTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Autonomous penetration-testing agents increasingly rely on large language models to reason over security observations and orchestrate heterogeneous tools. However, these environments are partially observable and dynamic, and tool outputs may be incomplete, stale, conflicting, or misleading. Such epistemic errors can propagate through long-horizon trajectories, leading to redundant probing, premature decisions, and task failure. We propose TRACE, an evidence-grounded epistemic-control framework that maintains claim-centric beliefs using evidence provenance, reliability, temporal freshness, and explicit support–contradiction relations. TRACE actively acquires evidence using prospective information gain under cost, redundancy, and operational-risk constraints, and repairs affected beliefs when prior evidence becomes invalid or contradictory. Under a controlled same-backbone evaluation, TRACE increases CVE-Bench Success@1 from 25.0% to 32.5%. In separate epistemic-control analyses, TRACE reduces the repeated-action rate from 0.273 to 0.158 relative to heuristic evidence acquisition and improves recovery over dependency-agnostic belief repair. These results demonstrate the value of explicit evidence and belief management for robust autonomous penetration testing under unreliable observations.

1 INTRODUCTION

Large language model (LLM) agents have recently shown promise for autonomous penetration testing, where an agent must interpret security observations, invoke heterogeneous tools, maintain long-horizon context, and adapt its strategy as new information becomes available (Deng et al., 2024; Mayoral-Vilches et al., 2025; Zhu et al., 2026). Compared with traditional scripted automation, these agents can support flexible reconnaissance, vulnerability analysis, and multi-step decision-making. However, reliable autonomy in cybersecurity requires more than improved planning or broader tool access: agents rarely observe the true environment state directly and must instead infer it from tool outputs, network responses, and external interfaces that may be incomplete, stale, noisy, or mutually inconsistent.

Recent autonomous security agents have advanced tool orchestration, retrieval augmentation, memory, and long-horizon reasoning (Shen et al., 2025; Zhang et al., 2025b; Zhu et al., 2025). Meanwhile, general LLM-agent research has explored reasoning–action interleaving, uncertainty-aware information acquisition, and self-correction (Yao et al., 2023; Shinn et al., 2023; Hu et al., 2024). Nevertheless, many existing systems implicitly treat observations as reliable facts or leave evidence reliability to informal model reasoning. This assumption is problematic in penetration testing, where observations are tool-dependent and dynamically changing. An incorrect observation can be stored as context, influence future planning, and eventually cause repeated exploration, incorrect conclusions, or premature termination.

This motivates three fundamental challenges for reliable autonomous penetration testing: (1) how to maintain trustworthy beliefs from incomplete and conflicting evidence; (2) how to actively acquire additional evidence when current information is insufficient; and (3) how to recover when newly observed evidence invalidates previous beliefs. These challenges are tightly coupled because each decision changes the information state on which future actions depend.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

To address these challenges, we propose **TRACE**, an evidence-grounded epistemic-control framework for reliable autonomous penetration testing. Instead of directly mapping observations to actions, TRACE calibrates evidence–claim relation probabilities and aggregates evidence using explicit provenance, deterministic reliability, and temporal freshness, enabling the agent to distinguish uncertain observations from well-supported claims. TRACE further introduces a risk-aware epistemic controller that determines whether the agent should execute a task action or acquire additional evidence based on uncertainty, conflict, information gain, cost, and operational risk. When new evidence contradicts existing beliefs, TRACE performs dependency-aware belief repair by tracing affected claims, discounting unreliable dependencies, acquiring discriminative evidence, and replanning from the repaired state.

We evaluate TRACE in controlled cyber-range environments under clean, partial-observation, dynamic-change, adversarial-evidence, and tool-failure conditions using multiple LLM backbones. Beyond final task success, we evaluate belief calibration, unsupported-belief rate, evidence efficiency, conflict resolution, recovery performance, and repair-affected dependency depth. The results demonstrate that explicit epistemic control improves both penetration-testing capability and reliability under uncertain observations.

Our main contributions are summarized as follows:

- We formulate reliable autonomous penetration testing as an epistemic control problem under unreliable partial observations, introducing a claim-centric belief representation that explicitly models evidence provenance, reliability, freshness, uncertainty, and conflict.
- We propose **TRACE**, which integrates trust-calibrated belief estimation with risk-aware active evidence acquisition, enabling agents to identify unresolved critical claims and rank feasible evidence-acquisition actions by predicted utility.
- We introduce dependency-aware belief repair, which traces downstream dependencies potentially affected by invalidated assumptions, updates affected beliefs, and enables recovery beyond retry-based reflection.
- We establish a comprehensive evaluation protocol for autonomous penetration testing under incomplete, dynamic, adversarial, and failed observations, demonstrating improvements in both task performance and epistemic reliability.

2 Related Work

2.1 Autonomous cybersecurity agents

LLM-based security agents have advanced from interactive penetration-testing assistants such as PentestGPT, CAI, and HPTSA to systems with multi-agent coordination, retrieval augmentation, adaptive workflows, and richer tool interaction (Deng et al., 2024; Mayoral-Vilches et al., 2025; Zhu et al., 2026; Shen et al., 2025; Stafeev et al., 2025; Huang et al., 2026; Lin et al., 2025; Ren et al., 2025). Meanwhile, benchmarks such as Cybench, CVE-Bench, BountyBench, PACEbench, CyberGym, and CTF-Dojo increasingly emphasize executable vulnerabilities, multi-stage interaction, and realistic environments (Zhang et al., 2025b; Zhu et al., 2025; Zhang et al., 2025a; Liu et al., 2026; Wang et al., 2026; Zhuo et al., 2025). These efforts primarily evaluate task completion and tool use, while explicit management of unreliable, stale, or conflicting observations remains less studied.

2.2 Epistemic state and active information acquisition

Partial observability is classically addressed through belief states over latent environment states (Kaelbling et al., 1998). LLM-agent methods such as ReAct interleave reasoning and environment interaction, while uncertainty-aware approaches investigate when additional information should be acquired (Yao et al., 2023; Hu et al., 2024). TRACE builds on this perspective but explicitly models evidence provenance, reliability, temporal freshness, and support–contradiction relations, which are particularly important when security observations originate from heterogeneous and potentially inconsistent tools.

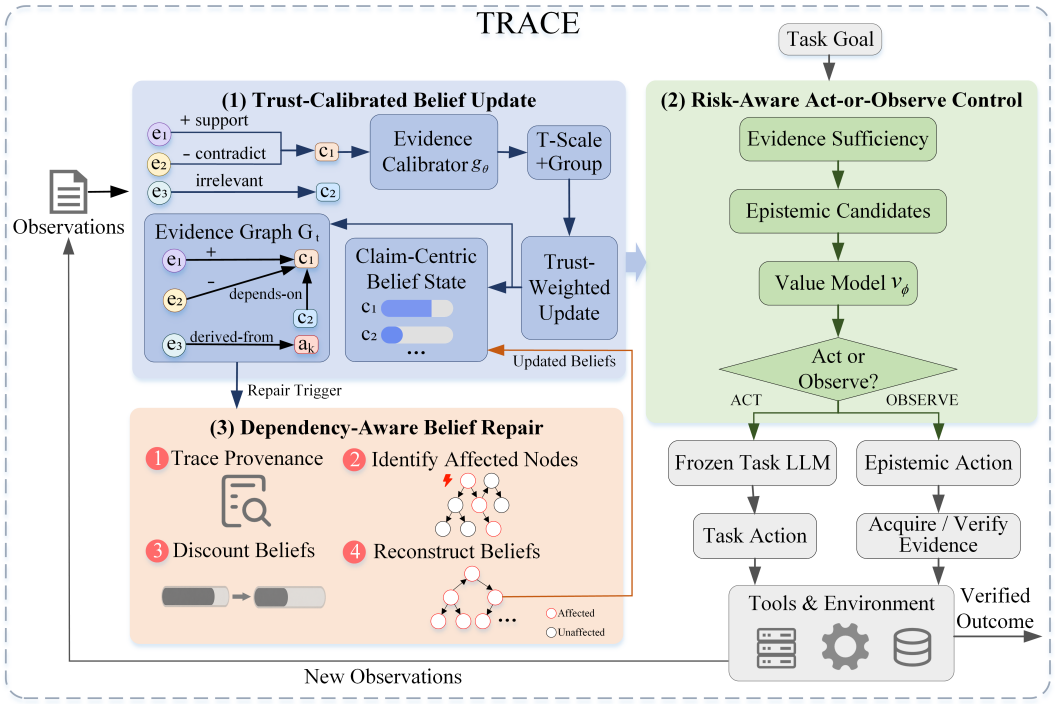


Figure 1: Overview of **TRACE**. (1) Trust-calibrated belief update converts observations into calibrated evidence and maintains a claim-centric evidence graph with confidence-valued claim beliefs. (2) Risk-aware act-or-observe control assesses evidence sufficiency, evaluates epistemic actions, and decides between task execution and evidence acquisition. (3) Dependency-aware belief repair traces affected claims, discounts unreliable beliefs, and reconstructs consistent belief states. Only g_θ and v_ϕ are trained, while the task LLM remains frozen.

2.3 Verification and Belief Recovery

Reflection and tool-assisted verification can improve agent reliability through feedback and iterative correction (Shinn et al., 2023; Gou et al., 2024). However, intrinsic self-correction without external feedback remains unreliable (Huang et al., 2024). Security agents also employ execution-based verification and exploitation confirmation (Shen et al., 2025; Zhu et al., 2025). TRACE differs by maintaining a persistent claim-centric epistemic state: when evidence invalidates an earlier belief, it traces affected dependencies, reconstructs the corresponding beliefs, and replans from the repaired state rather than only retrying the latest action.

3 Method

We introduce **TRACE**, an evidence-grounded epistemic control framework for reliable autonomous penetration testing under unreliable partial observations. Unlike agents that map observations directly to actions, TRACE maintains a claim-centric belief state grounded in structured evidence. As illustrated in Figure 1, observations are converted into calibrated evidence for belief updating. The epistemic controller then determines whether to execute a task action or acquire additional evidence. When new observations invalidate previous assumptions, TRACE performs dependency-aware belief repair before continuing interaction.

3.1 Problem Setup

We consider an agent interacting with a partially observable environment with latent state s_t, observation o_t, task actions $\mathcal{A}_T$, and epistemic actions $\mathcal{A}_E$ for acquiring, verifying, or disambiguating evidence. Rather than estimating the full latent state, TRACE maintains task-relevant claims

$\mathcal{C}_t = \{c_1, \ldots, c_K\}$ with confidence-valued beliefs $b_t(c) \in (0, 1)$. All claims use a fixed neutral prior $\pi_0 = 0.5$; beliefs are reconstructed from the currently active evidence graph using the trust-calibrated aggregation rule in Section 3.3.

3.2 Evidence Graph and Claim-Centric Belief State

TRACE organizes claims, evidence, and actions in an evidence graph whose edges encode SUPPORTS, CONTRADICTS, DERIVED-FROM, and DEPENDS-ON relations. Formally,

$$G_t = (V_t, E_t), \qquad e_i = \langle x_i, m_i, u_i, \tau_i, \zeta_i, \chi_i \rangle,$$

where x_i, m_i, u_i, τ_i, ζ_i, and χ_i denote evidence content, modality, source, observation timestamp in seconds, provenance identifier, and applicable scope, respectively. The graph supports incremental belief updating and dependency tracing when previously accepted assumptions become unreliable.

3.3 Trust-Calibrated Belief Update

TRACE uses the Evidence Calibrator g_θ to estimate evidence–claim relations, while contextual reliability is assigned by the deterministic runtime reliability rule $\mathcal{R}$. Given evidence e_i and claim c,

$$\mathbf{z}_i = g_\theta(e_i, c), \qquad r_i = \mathcal{R}(e_i, c), \qquad [p_i^{sup}, p_i^{con}, p_i^{irr}] = \mathrm{softmax}\Big(\frac{\mathbf{z}_i}{T}\Big),$$

where $\mathbf{z}_i$ contains support, contradiction, and irrelevance logits, $r_i \in [0, 1]$ is the rule-based reliability score used by the deployed controller, and T is fitted on held-out calibration data using temperature scaling (Guo et al., 2017). The deployed calibrator does not use a learned reliability head. Although the offline training script fits an auxiliary ridge regressor to the deterministic reliability labels for development diagnostics, its parameters are not exported or queried at inference. Runtime reliability r_i is always computed by the deterministic rule $\mathcal{R}$.

Temporal freshness is recomputed for every active evidence item using elapsed time in seconds rather than interaction steps. Let ξ_t denote the runtime clock value at interaction step t, and let τ_i denote the observation timestamp of evidence e_i. We define the evidence age as

$$\Delta_i(t) = \max(0, \xi_t - \tau_i).$$

Each claim is associated with a freshness policy $\kappa(c)$. TRACE combines continuous exponential decay with an optional hard expiration boundary:

$$f_i(c, t) = \exp\big(-\lambda_{\kappa(c)} \Delta_i(t)\big)\, \mathbb{I}\big[\Delta_i(t) < H_{\kappa(c)}\big],$$

where $\lambda_{\kappa(c)}$ and $H_{\kappa(c)}$ denote the policy-specific decay rate and hard-expiration horizon, respectively, with $H_{\kappa(c)} = \infty$ indicating no hard expiration. The freshness policy is fixed throughout evaluation: `episode` and `snapshot` evidence use the default policy, whereas `dynamic` and `session` evidence is gradually discounted and becomes inactive after its validity horizon.

The effective evidence weight and signed relation contribution are

$$w_i(c, t) = r_i f_i(c, t)(1 - p_i^{irr}), \qquad q_i(c) = p_i^{sup} - p_i^{con}.$$

For claim c, TRACE first selects currently admissible evidence. An evidence item is admissible only when its evidence node is active and it is connected to c through an active SUPPORTS or CONTRADICTS edge.

Admissible evidence is partitioned into provenance groups. Let $\mathcal{M}_{g,t}(c)$ denote the positive-weight members of provenance group g associated with claim c at time t. For each non-empty provenance group g, TRACE aggregates correlated evidence as

$$Q_g(c,t) = \frac{\sum_{i \in \mathcal{M}_{g,t}(c)} w_i(c,t) q_i(c)}{\sum_{i \in \mathcal{M}_{g,t}(c)} w_i(c,t)}, \qquad W_g(c,t) = \max_{i \in \mathcal{M}_{g,t}(c)} w_i(c,t), \qquad \psi_g(c,t) = W_g(c,t) Q_g(c,t).$$

Using the maximum member weight rather than summing all weights prevents correlated observations from the same provenance group from receiving multiplicative influence.

TRACE reconstructs the claim belief from all currently active provenance groups rather than permanently accumulating earlier log-odds:

$$L_t(c) = \text{clip}\left(\text{logit}(0.5) + \sum_{g \in \mathcal{P}_t(c)} \psi_g(c,t), -8, 8\right), \qquad b_t(c) = \sigma(L_t(c)).$$

The log-odds cap prevents numerical saturation while leaving the neutral prior unchanged. Thus, all claims use the same fixed neutral prior $\pi_0 = 0.5$, previously observed evidence is reweighted using its current freshness, and invalidated evidence does not re-enter subsequent belief reconstruction.

3.4 Risk-Aware Active Evidence Acquisition

TRACE evaluates claim uncertainty and evidence conflict before executing task-critical actions. Claim entropy and overall uncertainty are

$$H_t(c) = -b_t(c) \log_2 b_t(c) - (1 - b_t(c)) \log_2 (1 - b_t(c)), \qquad U_t = \sum_{c \in \mathcal{C}_t} \rho(c) H_t(c),$$

where $\rho(c)$ denotes claim importance. For online uncertainty estimation, we use uniform importance, $\rho(c) = 1/|\mathcal{C}_t|$, so that U_t corresponds to the mean entropy of the currently maintained claims.

Supporting and contradicting evidence are summarized as

$$S_t(c) = \sum_{g \in \mathcal{P}_t(c)} \max(0, \psi_g(c,t)), \qquad K_t(c) = \sum_{g \in \mathcal{P}_t(c)} \max(0, -\psi_g(c,t)),$$

$$\text{Conf}_t(c) = \frac{\min(S_t(c), K_t(c))}{S_t(c) + K_t(c) + \epsilon}.$$

Because conflict is computed after provenance-group aggregation, $\text{Conf}_t(c)$ measures residual disagreement across aggregated groups. Opposing observations within the same provenance group may partially cancel before this statistic is computed; the score is therefore not an exhaustive detector of every raw evidence-level contradiction.

When critical claims remain uncertain or conflicting, TRACE evaluates each candidate epistemic action using the prospective information-gain model v_ϕ. Its input contains only quantities available before action execution: z_t, a seven-statistic summary of the current claim set, together with the action identifier and module label. No execution outcome or other post-action quantity is supplied to the predictor.

For a candidate epistemic action a,

$$\widehat{IG}(a) = v_\phi(z_t, a),$$

and TRACE evaluates

$$J_E(a) = \alpha \widehat{IG}(a) - \lambda Cost(a) - \mu Risk_{\text{tpl}}(a) - \nu Redundancy(a).$$

The learned term estimates the prospective epistemic value of the candidate, while the remaining terms penalize acquisition cost, operational-risk prior, and repeated evidence acquisition. TRACE executes an epistemic action only when the highest-utility feasible candidate exceeds the threshold τ_E; otherwise it proceeds with the task action.

Conflict resolution is handled by independent verification, evidence-graph conflict detection, and dependency-aware belief repair.

TRACE first determines whether the current decision depends on unresolved critical claims. If additional evidence is required, it selects the feasible epistemic action with the highest utility:

$$a_t^{E*} = \arg\max_{a \in \mathcal{A}_{E,t}^{\text{crit}}} J_E(a).$$

The selected epistemic action is executed only when

$$J_E(a_t^{E*}) > \tau_E.$$

Otherwise, the controller proceeds with the task action. We use $\tau_E = 0.05$ by default.

Critical claims are fixed when the current goal is instantiated from its verification contract. A claim is critical if its contract predicate is a task precondition or outcome, or if it is a context predicate explicitly marked as mandatory. We denote this static set by $\mathcal{C}^{\text{crit}}(g)$. Evidence-graph DEPENDS-ON edges do not determine criticality; they are used only for dependency-aware belief repair.

3.5 Dependency-Aware Belief Repair

When new evidence invalidates previously accepted beliefs, TRACE repairs the persistent epistemic state rather than merely retrying the latest action. Repair is triggered by strong evidence conflict, a decisive belief flip, explicit evidence invalidation, stale support, or contract contradiction.

TRACE then traverses the dependency graph to identify affected downstream claims, invalidates unreliable support, and reconstructs beliefs from the repaired evidence graph. Let $\widetilde{b}_t(c_j)$ denote the reconstructed belief of an affected claim. Dependency-aware uncertainty propagation is applied as

$$b'_t(c_j) = \left(1 - \omega\eta^{d(i,j)}\right)\widetilde{b}_t(c_j) + \omega\eta^{d(i,j)}\,\pi_0,$$

where $d(i,j)$ is the dependency distance from the invalidated claim. This discounts unsupported downstream beliefs toward the neutral prior while preserving evidence that remains valid. Because repair modifies the persistent evidence graph, invalidated support cannot regain its original influence during subsequent belief reconstruction. All repair thresholds and hyperparameters are fixed throughout evaluation.

3.6 Training and Inference

TRACE freezes the backbone LLM and trains two lightweight modules offline: the relation component of the Evidence Calibrator g_θ and the Epistemic Value Model v_ϕ. Contextual reliability r_i is not learned; it is assigned by the deterministic runtime rule $\mathcal{R}$. The final v_ϕ predicts prospective information gain from decision-time features. Both learned components are invoked online through deterministic vocabulary gates. Evidence calibration uses g_θ only when the claim identifier belongs to the 70-entry claim vocabulary collected during training; otherwise TRACE uses the deterministic calibration fallback. Independently, v_ϕ scores an epistemic candidate only when its action identifier appears in the exported 53-action feature vocabulary; unsupported candidates are excluded from learned ranking and recorded for audit. These gates are deterministic vocabulary-membership checks rather than statistical out-of-distribution detectors.

4 Experiments

Our experiments evaluate five aspects of TRACE: task success, belief reliability, active evidence efficiency, dependency-aware recovery, and component contributions.

Table 1: Same-backbone task performance on CVE-Bench and PACEbench. PACEbench reports official Pass@5 normalized flag-completion scores with partial credit and the weighted PACEbench score. Higher is better.

	CVE-Bench		PACEbench				
Method	S@1	S@5	A-Score	B-Score	C-Score	D-Score	PACE Score@5
CAI	17.5%	22.5%	0.294	0.421	0.267	0.000	0.265
Cybench Agent	7.5%	10.0%	0.176	0.211	0.133	0.000	0.138
HPTSA	20.0%	27.5%	0.353	0.474	0.267	0.000	0.293
PenForge	25.0%	32.5%	0.412	0.474	0.333	0.000	0.325
PentestGPT	12.5%	17.5%	0.235	0.316	0.133	0.000	0.182
PACEagent	25.0%	35.0%	0.471	0.579	0.400	0.000	0.388
TRACE (Ours)	**32.5%**	**47.5%**	**0.706**	**0.619**	**0.467**	0.000	**0.467**

Table 2: Verified task success under controlled observation perturbations. Results are pooled over episodes rather than averaged per intensity. In the full 40-episode bank, intensity is assigned deterministically per episode, with 20 episodes each at intensities 1 and 2; clean episodes use intensity 0.

Method	Clean	Partial	Dynamic	Adversarial	Failure
CAI	52.5%	45.0%	35.0%	27.5%	37.5%
Cybench Agent	32.5%	22.5%	15.0%	12.5%	17.5%
HPTSA	57.5%	50.0%	42.5%	30.0%	42.5%
PentestGPT	45.0%	35.0%	27.5%	20.0%	30.0%
PenForge	65.0%	52.5%	42.5%	30.0%	45.0%
PACEagent	67.5%	60.0%	50.0%	37.5%	52.5%
TRACE	**77.5%**	**72.5%**	**62.5%**	**55.0%**	**62.5%**

4.1 Experimental Setup

Benchmarks. We evaluate TRACE on autonomous penetration-testing tasks using CVE-Bench and PACEbench. CVE-Bench focuses on reproducible real-world vulnerability exploitation, while PACEbench introduces more realistic partially observable multi-stage penetration scenarios.

Baselines. We compare TRACE with CAI, Cybench Agent, HPTSA, PentestGPT, PenForge, and PACEagent. For the controlled comparison, all methods use the same frozen backbone model, benchmark environments, maximum interaction steps, and token budget. We preserve each method's native planning, memory, orchestration, verification, and tool-selection logic. Only lightweight adapters for model invocation, environment I/O, logging, and result collection are introduced.

Metrics. We evaluate task performance using Verified Task Success (VTS) and belief quality using State Accuracy, Brier score, Expected Calibration Error (ECE), and Unsupported Belief Rate (UBR). Epistemic-action efficiency is evaluated using Entropy Reduction per Epistemic Action (ER/EA), evidence calls, repeated action rate (RAR), token cost, and cumulative risk. Recovery is evaluated using conflict resolution rate (CRR), recovery success rate (RSR), repair-affected dependency depth (RDD), and recovery steps.

4.2 Main Results

Table 1 summarizes task-level performance under the controlled same-backbone setting. On CVE-Bench, TRACE achieves 32.5% Success@1 and 47.5% Success@5. Relative to the strongest evaluated baselines for the corresponding metrics, these represent gains of 7.5 percentage points in Success@1 and 12.5 percentage points in Success@5.

On PACEbench, TRACE achieves the highest A-, B-, and C-Scores among the evaluated methods, while all methods obtain a D-Score of 0. TRACE improves the overall PACE Score@5 from 0.388 for PACEagent to 0.467, an absolute gain of 0.079. The improvements span the evaluated single, blended, and chained exploitation settings, while defense-bypass remains unresolved.

Table 3: Belief reliability under different evidence-calibration strategies on the in-domain subset of the independent DVWA evaluation split. State Acc measures claim-state correctness; Brier and ECE measure probabilistic quality, while UBR records decisive claim states with no currently admissible positive-weight evidence. The same evaluation denominators are used for all calibrator variants.

Calibrator	State Acc ↑	Brier ↓	ECE ↓	UBR ↓
Deterministic heuristic	0.6838	0.2170	**0.0272**	1.0000
Fixed-weight	0.6838	0.2289	0.0519	1.0000
Learned g_θ	**0.8028**	0.1668	0.2479	**0.0000**
Learned g_θ + TS	**0.8028**	**0.1404**	0.1419	**0.0000**

Table 4: Efficiency of active evidence acquisition. RAR denotes repeated action rate.

Policy	ER/EA ↑	Calls ↓	RAR ↓	Token ↓	Proxy Risk ↓
Fixed Verification	0.095	8.60	0.394	12.3k	0.285
Random Evidence	0.068	10.10	0.486	14.1k	0.347
Heuristic J_E	0.135	7.20	0.273	10.2k	0.224
TRACE	**0.186**	**5.90**	**0.158**	**8.4k**	**0.163**

4.3 Reliability under Unreliable Observations

We evaluate robustness under controlled partial, dynamic, adversarial, and tool-failure observations, and separately assess belief quality on the in-domain subset of the independent DVWA evaluation split using alternative evidence-calibration strategies.

The learned calibrator substantially improves claim-state correctness on the independent DVWA evaluation split. Relative to the deterministic heuristic, learned g_θ increases State Accuracy from 0.6838 to 0.8028. Temperature scaling leaves State Accuracy unchanged but improves the probabilistic quality of the learned calibrator, reducing Brier score from 0.1668 to 0.1404 and belief-level ECE from 0.2479 to 0.1419.

The deterministic heuristic nevertheless obtains the lowest ECE (0.0272). We therefore do not claim that the learned calibrator dominates every alternative on every calibration metric. Instead, the results show that g_θ improves belief-state correctness, while temperature scaling substantially improves the probability quality of the learned model.

UBR is 0 for both learned-calibrator variants and 1 for the deterministic and fixed-weight variants under the current admissible-evidence criterion. Because the evaluated trajectories do not reach the configured hard-expiration horizons, this difference reflects the availability of admissible positive-weight evidence rather than elapsed-time expiration.

4.4 Epistemic Control Analysis

Fixed Verification invokes a predefined verification routine, Random Evidence samples uniformly from feasible epistemic actions, and Heuristic J_E replaces learned value estimates with deterministic uncertainty- and conflict-based scores.

Active evidence acquisition. We evaluate whether TRACE acquires additional evidence efficiently rather than simply increasing tool usage. ER/EA measures the mean summed positive reduction in claim entropy per executed epistemic action. It measures uncertainty reduction rather than belief correctness. Calls and Token measure resource use, RAR measures repeated candidate selection, and Proxy Risk reports cumulative risk-estimator output.

Compared with heuristic J_E, TRACE increases ER/EA from 0.135 to 0.186 while reducing the repeated-action rate from 0.273 to 0.158. It also uses fewer evidence calls (5.90 vs. 7.20) and fewer tokens (8.4k vs. 10.2k), suggesting that the improvement is not explained by simply invoking more tools.

Table 5: Recovery performance after belief inconsistency. CRR is conflict resolution rate, RSR recovery success rate, and RDD repair-affected dependency depth, reported as a traversal-depth diagnostic. CRR uses evaluable triggered conflicts, while RSR uses the shared fixed repair-case set $\mathcal{R}^\star$.

Strategy	CRR ↑	RSR ↑	RDD	Steps ↓
Retry / Reflection	0.189	0.211	2.85	6.72
Repair w/o Dependency	0.519	0.526	1.56	4.35
TRACE	**0.722**	**0.737**	0.79	**3.08**

Table 6: Ablation study of TRACE components under the end-to-end controlled evaluation protocol.

Variant	VTS ↑	Brier ↓	RSR ↑
w/o Learned Calibrator	27.5%	0.171	42.1%
w/o Temperature Scaling	30.0%	0.154	68.4%
w/o Freshness	**32.5%**	0.140	71.1%
w/o Provenance Grouping	30.0%	0.145	68.4%
w/o Active Evidence	27.5%	0.149	57.9%
w/o Belief Repair	25.0%	0.158	21.1%
TRACE	**32.5%**	**0.135**	**73.7%**

Dependency-aware belief repair. Table 5 shows that dependency-aware repair improves conflict resolution and recovery relative to both retry/reflection and dependency-agnostic repair. These results complement the mechanism in Section 3.5: persistent graph repair limits the propagation of invalid support while allowing subsequent belief reconstruction to operate on the corrected epistemic state.

4.5 Ablation Studies

We perform controlled ablations to quantify the contribution of each TRACE component. "w/o Learned Calibrator" replaces g_θ with deterministic heuristic weighting, while "w/o Temperature Scaling" fixes $T = 1$ at inference. The remaining variants remove freshness modeling, provenance grouping, active evidence acquisition, or belief repair.

Belief repair has the largest effect on recovery, reducing RSR from 73.7% to 21.1% when removed. Removing the learned calibrator or active evidence acquisition reduces VTS from 32.5% to 27.5%, while removing provenance grouping lowers VTS to 30.0%. Temperature scaling improves belief quality, with Brier increasing from 0.135 to 0.154 when disabled.

Removing freshness leaves VTS unchanged at 32.5% and only slightly worsens Brier (0.135 to 0.140) and RSR (73.7% to 71.1%), consistent with the short duration of the evaluated trajectories. Overall, the results identify belief repair, learned evidence calibration, and active evidence acquisition as the main contributors to TRACE's observed performance.

5 Conclusion

We present **TRACE**, an evidence-grounded epistemic-control framework for reliable autonomous penetration testing under unreliable observations. TRACE combines calibrated evidence modeling, risk-aware active evidence acquisition, and dependency-aware belief repair to maintain reliable claim-centric beliefs and recover from invalidated assumptions. Experiments show improved task performance and belief reliability under incomplete, dynamic, adversarial, and failed observations. Future research will explore multimodal autonomous penetration testing by integrating visual perception, temporal understanding, and cyber reasoning into evidence-driven agents, enabling more comprehensive interaction with complex real-world environments.

AI USE STATEMENT

In this work, we used generative AI tools for literature retrieval and discovery, language polishing, and drafting or revising portions of the manuscript. We have not used generative AI tools for research execution, synthetic dataset generation, or proving mathematical claims, and other disclosure categories are not applicable to this work. We reviewed all AI-assisted work. In particular, the authors manually verified the retrieved references, checked AI-assisted technical descriptions and claims against the cited literature and the implemented methodology, and reviewed the mathematical formulations, experimental design, and final manuscript for correctness and consistency. No experimental results were generated by generative AI tools. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies autonomous penetration testing and therefore involves dual-use cybersecurity capabilities. Our research is intended to improve the reliability and safety of authorized security testing rather than facilitate unauthorized access or malicious exploitation. All penetration-testing experiments are conducted in controlled, isolated, or explicitly authorized environments, and no experiments target third-party systems without permission. We follow applicable licenses, benchmark usage policies, and responsible security research practices when using vulnerability datasets, software, and testing environments. The proposed framework focuses on evidence grounding, uncertainty-aware decision making, and belief repair, with the goal of reducing incorrect or unsafe actions by autonomous security agents. The authors take responsibility for the ethical conduct and reporting of this research.

REPRODUCIBILITY STATEMENT

Appendix A records repository revisions, implementation settings, and observation-perturbation rules. Appendix A.3 defines the evaluation metrics, and Appendix B specifies the implemented estimators, training parameters, data partitions, and temperature-fitting procedure. Appendix C.4 reports checkpoint-specific offline validation results with their sample counts. The complete TRACE inference procedure is given in Algorithm 1. No CVE-Bench or PACEbench evaluation trajectory is used to train g_θ or v_ϕ.

REFERENCES

Gelei Deng, Yi Liu, Víctor Mayoral-Vilches, Peng Liu, Yuekang Li, Yuan Xu, Tianwei Zhang, Yang Liu, Martin Pinzger, and Stefan Rass. PentestGPT: Evaluating and harnessing large language models for automated penetration testing. In *33rd USENIX Security Symposium (USENIX Security 24)*, pp. 847–864. USENIX Association, 2024. URL https://www.usenix.org/conference/usenixsecurity24/presentation/deng.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/fef126561bbf9d4467dbb8d27334b8fe-Abstract-Conference.html.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1321–1330. PMLR, 2017. URL https://proceedings.mlr.press/v70/guo17a.html.

Zhiyuan Hu, Chumin Liu, Xidong Feng, Yilun Zhao, See-Kiong Ng, Anh Tuan Luu, Junxian He, Pang Wei Koh, and Bryan Hooi. Uncertainty of Thoughts: Uncertainty-aware planning enhances information seeking in LLMs. In *Advances in Neural Information Processing Systems*, volume 37, pp. 24181–24215. Curran Associates, Inc., 2024. doi: 10.52202/079017-0762. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/2b0e14abd8128e6bf98b6b0bec1cfcbf-Abstract-Conference.html.

Huihui Huang, Jieke Shi, Junkai Chen, Ting Zhang, Yikun Li, Chengran Yang, Eng Lieh Ouh, Lwin Khin Shar, and David Lo. PenForge: On-the-fly expert agent construction for automated penetration testing. In *Proceedings of the IEEE/ACM 48th International Conference on Software Engineering*, ICSE-NIER '26, pp. 76–80. ACM, 2026. doi: 10.1145/3786582.3786814.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/8b4add8b0aa8749d80a34ca5d941c355-Abstract-Conference.html.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2):99–134, 1998. doi: 10.1016/S0004-3702(98)00023-X.

Justin W. Lin, Eliot Krzysztof Jones, Donovan Julian Jasper, Ethan Jun-shen Ho, Anna Wu, Arnold Tianyi Yang, Neil Perry, Andy Zou, Matt Fredrikson, J. Zico Kolter, Percy Liang, Dan Boneh, and Daniel E. Ho. Comparing AI agents to cybersecurity professionals in real-world penetration testing. *arXiv preprint arXiv:2512.09882*, 2025. URL https://arxiv.org/abs/2512.09882.

Zicheng Liu, Lige Huang, Jie Zhang, Dongrui Liu, Yuan Tian, and Jing Shao. PACEbench: A framework for evaluating practical AI cyber-exploitation capabilities. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/1cc901ebe4c549ab9222af0c1dd449b1-Abstract-Conference.html.

Víctor Mayoral-Vilches, Luis Javier Navarrete-Lozano, María Sanz-Gómez, Lidia Salas Espejo, Martiño Crespo-Álvarez, Francisco Oca-Gonzalez, Francesco Balassone, Alfonso Glera-Picón, Unai Ayucar-Carbajo, Jon Ander Ruiz-Alcalde, Stefan Rass, Martin Pinzger, and Endika Gil-Uriarte. CAI: An open, bug bounty-ready cybersecurity AI. *arXiv preprint arXiv:2504.06017*, 2025. URL https://arxiv.org/abs/2504.06017.

Xiaoxue Ren, Penghao Jiang, Kaixin Li, Zhiyong Huang, Xiaoning Du, Jiaojiao Jiang, Zhenchang Xing, Jiamou Sun, and Terry Yue Zhuo. HackWorld: Evaluating computer-use agents on exploiting web application vulnerabilities. *arXiv preprint arXiv:2510.12200*, 2025. URL https://arxiv.org/abs/2510.12200.

Xiangmin Shen, Lingzhi Wang, Zhenyuan Li, Yan Chen, Wencheng Zhao, Dawei Sun, Jiashui Wang, and Wei Ruan. PentestAgent: Incorporating LLM agents to automated penetration testing. In *Proceedings of the 20th ACM Asia Conference on Computer and Communications Security*, 2025. doi: 10.1145/3708821.3733882.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023. doi: 10.52202/075280-0377. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Aleksei Stafeev, Tim Recktenwald, Gianluca De Stefano, Soheil Khodayari, and Giancarlo Pellegrino. YuraScanner: Leveraging LLMs for task-driven web app scanning. In *Network and Distributed System Security Symposium*. Internet Society, 2025. doi: 10.14722/ndss.2025.240388. URL https://www.ndss-symposium.org/ndss-paper/yurascanner-leveraging-llms-for-task-driven-web-app-scanning/.

Zhun Wang, Tianneng Shi, Jingxuan He, Matthew Cai, Jialin Zhang, and Dawn Song. CyberGym: Evaluating AI agents' real-world cybersecurity capabilities at scale. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/c876a2935a90aff21874e14c07dc3e33-Abstract-Conference.html.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Andy Zhang, Joey Ji, Celeste Menders, Riya Dulepet, Thomas Qin, Ron Wang, Junrong Wu, Kyleen Liao, Jiliang Li, Jinghan Hu, Sara Hong, Nardos Demilew, Shivatmica Murgai, Jason Tran, Nishka Kacheria, Ethan Ho, Denis Liu, Lauren McLane, Olivia Bruvik, Dai-Rong Han, Seungwoo Kim, Akhil Vyas, Cuiyuanxiu Chen, Ryan Li, Weiran Xu, Jonathan Ye, Prerit Choudhary, Siddharth M. Bhatia, Vikram Sivashankar, Yuxuan Bao, Dawn Song, Dan Boneh, Daniel Ho, and Percy Liang. BountyBench: Dollar impact of AI agent attackers and defenders on real-world cybersecurity systems. In *Advances in Neural Information Processing Systems*, volume 38, 2025a. doi: 10.52202/085713-5725. URL https://papers.neurips.cc/paper_files/paper/2025/hash/faed4276b52ef762879db4142655c699-Abstract-Datasets_and_Benchmarks_Track.html.

Andy K. Zhang, Neil Perry, Riya Dulepet, Joey Ji, Celeste Menders, Justin Lin, Eliot Jones, Gashon Hussein, Samantha Liu, Donovan Jasper, Pura Peetathawatchai, Ari Glenn, Vikram Sivashankar, Daniel Zamoshchin, Leo Glikbarg, Derek Askaryar, Haoxiang Yang, Aolin Zhang, Rishi Alluri, Nathan Tran, Rinnara Sangpisit, Kenny Oseleononmen, Dan Boneh, Daniel Ho, and Percy Liang. Cybench: A framework for evaluating cybersecurity capabilities and risks of language models. In *International Conference on Learning Representations*, 2025b. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/3e9412a9c1d93810ef3ef7825115016b-Abstract-Conference.html.

Yuxuan Zhu, Antony Kellermann, Dylan Bowman, Philip Li, Akul Gupta, Adarsh Danda, Richard Fang, Conner Jensen, Eric Ihli, Jason Benn, Jet Geronimo, Avi Dhir, Sudhit Rao, Kaicheng Yu, Twm Stone, and Daniel Kang. CVE-Bench: A benchmark for AI agents' ability to exploit real-world web application vulnerabilities. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 79850–79867. PMLR, 2025. URL https://proceedings.mlr.press/v267/zhu25i.html.

Yuxuan Zhu, Antony Kellermann, Akul Gupta, Philip Li, Richard Fang, Rohan Bindu, and Daniel Kang. Teams of LLM agents can exploit zero-day vulnerabilities. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23–35. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.eacl-long.2. URL https://aclanthology.org/2026.eacl-long.2/.

Terry Yue Zhuo, Dingmin Wang, Hantian Ding, Varun Kumar, and Zijian Wang. Training language model agents to find vulnerabilities with CTF-Dojo. *arXiv preprint arXiv:2508.18370*, 2025. URL https://arxiv.org/abs/2508.18370.

This appendix provides additional benchmark descriptions, implementation details, training objectives, supplementary results, and the complete TRACE inference procedure omitted from the main paper due to space constraints.

A Benchmark and Baseline Details

A.1 Benchmark Versions and Evaluation Protocols

We use the official implementations of CVE-Bench and PACEbench for all benchmark evaluations. Task definitions, environment construction, vulnerability instances, verification logic, and benchmark-specific scoring follow the corresponding official releases. We do not modify benchmark task content or success criteria. Our evaluation harness only standardizes the agent-facing interface, backbone invocation, interaction budget, logging, and result collection across compared methods.

CVE-Bench. Revision: `4ed2d80ba5752ee958d624fa581242744e8a8a5d`. We use the official task environments and verification procedures without modifying vulnerability instances or task success criteria.

PACEbench. Revision: `b3f40d90861131867d37ce8a8af8a9945f29a12d`. We retain the official scenario definitions, environment construction, flag-based verification, and benchmark scoring procedure without modification.

TRACE revision: `dc001fc179a3ab0af5a29dd1dffd5ec349c1a4cf`. Table 7 summarizes the recorded configuration values. These repository revisions identify the inspected checkouts. All reported runs are associated with manifests recording the TRACE revision, benchmark revision, backbone identifier, learned checkpoints, controller configuration, perturbation condition, and random seed.

A.2 Observation Perturbation Protocol

The data collector implements a clean condition (intensity 0) and four perturbation conditions, each with two discrete intensity settings (Table 8). These are deterministic transformation rules rather than sampled masking or failure probabilities. Partial observation masks module-specific response fields; dynamic observation changes the DVWA security level after the control baseline; adversarial evidence substitutes misleading response content; and tool failure injects an initial timeout, followed at intensity 2 by a corrupted response. For the adversarial condition, corruption targets are derived from action roles as `probe_actions` $\setminus$ `independent_actions`, and at least one independent verification path is always left unmodified. The same runtime action-cooldown mechanism is active during perturbed collection, so a recently executed epistemic action is temporarily excluded from immediate repetition according to `cooldown_steps`.

Collection seeds are generated as

$$s = 20260903 + 100000k + 1000i + j,$$

where k is the zero-based index in the configured condition list, i is the intensity, and j is the within-cell episode index. The default condition order is clean, partial, dynamic, adversarial, and tool failure. Given the same action, response, condition, and intensity, response transformations are reproducible. Agents that choose different actions need not receive identical observation sequences.

Intensity assignment and aggregation. Perturbation intensity is an episode-level attribute rather than an independently crossed evaluation factor. For each non-clean condition, the episode bank assigns

$$I(e) = 1 + \left[\left(k_{\text{family}}(e) + s_{\text{task}}(e)\right) \bmod 2\right],$$

where k_{family} is the task-family index and s_{task} is the task seed. Each $(\text{episode}, \text{condition})$ pair is therefore evaluated at exactly one perturbation intensity rather than being rerun at both levels.

The robustness results in the main paper are aggregated directly over episodes within each (method, condition) group, without conditioning on intensity. In the complete 40-episode bank, the assignment is balanced, with 20 episodes at intensity 1 and 20 at intensity 2. Consequently, the pooled success rate is numerically equal to the equal-weight mean of the two intensity-specific success rates, although this equivalence follows from the balanced episode bank rather than from the aggregation procedure itself.

Perturbation-bank coverage. The controlled perturbation bank used for the reported robustness results covers three of the eight configured task families: `dvwa_sqli`, `dvwa_exec`, and `dvwa_upload`. The remaining configured families were excluded when one or more protocol requirements—a runnable target environment, sealed ground-truth state, or publication-eligible metadata—were unavailable. We therefore interpret the perturbation results as controlled robustness evidence over the evaluated families rather than as exhaustive coverage of all configured benchmark families.

A.3 Metric Definitions and Computation

In addition to task success and belief calibration, we evaluate the efficiency of epistemic actions and the effectiveness of belief repair. Unless otherwise specified, the protocol runner averages available per-episode metric values within each method–condition group, giving each episode equal weight.

Entropy Reduction per Epistemic Action (ER/EA). For each executed epistemic action a_k, let t_k^- and t_k^+ denote the belief states immediately before the action and after all evidence emitted by that action has been incorporated. We define

$$ER(a_k) = \sum_c \max\left[0,\, H_{t_k^-}(c) - H_{t_k^+}(c)\right],$$

and

$$\mathrm{ER/EA} = \frac{1}{N_E} \sum_{k=1}^{N_E} ER(a_k),$$

where N_E is the number of executed epistemic actions. An action emitting multiple evidence records is counted once; an action producing no evidence remains in the denominator and contributes zero.

Evidence Calls. Calls denotes the average number of executed epistemic actions per episode, including evidence acquisition and verification actions.

Repeated Action Rate (RAR). RAR measures repeated epistemic-candidate selection:

$$\mathrm{RAR} = \frac{N_{\mathrm{repeated}}}{N_E}.$$

An action is counted as repeated when the same candidate identifier has already been executed earlier in the episode. Because candidate identifiers encode the target claim, tool, and action template, RAR is a structural repetition metric; it does not imply that a repeated invocation necessarily produces no novel evidence.

Token Cost. Token denotes the average number of LLM tokens consumed per episode under each evidence-acquisition policy. We report the sum of input and output tokens recorded by the runtime logger.

Proxy Risk Cost. Each executed epistemic action receives a score from the frozen risk-estimation module of the base agent runtime. The same scorer is used for all compared evidence-acquisition policies. We report the average cumulative score per episode:

Table 7: Repository, implementation, and data-partition settings used for the reported experiments.

Item	Recorded setting
CVE-Bench revision	`4ed2d80ba575`
PACEbench revision	`b3f40d908611`
TRACE implementation revision	`dc001fc179a3`
Maximum runtime actions	12
Maximum epistemic actions per claim	3
Risk budget	10.0 (experiment configuration)
Provenance fallback window	10 s
Belief log-odds cap	$[-8, 8]$
Claim prior probability	Fixed $\pi_0 = 0.5$ (`prior_log_odds`=0.0); not configurable
Freshness time unit	Seconds; computed from runtime clock minus evidence observation timestamp
Freshness decay: default	$\lambda = 1 \times 10^{-5}\,\mathrm{s}^{-1}$; no hard expiration
Freshness decay: dynamic	$\lambda = 2 \times 10^{-4}\,\mathrm{s}^{-1}$; hard expiration at 21600 s (6 h)
Freshness decay: session	$\lambda = 1 \times 10^{-3}\,\mathrm{s}^{-1}$; hard expiration at 3600 s (1 h)
v_ϕ replay freshness	Runtime-aligned wall-clock recomputation from evidence timestamps; rates and expiration horizons loaded from `trace_config.json`
Replay timestamp coverage	3,691 / 3,697 episodes contain `observed_at_utc`; Six episodes lacking valid timestamps are excluded from the final v_ϕ replay dataset.
TTL-policy fallback	`episode` and `snapshot` use the `default` freshness policy
Conflict threshold τ_c	0.30
Decisive-belief thresholds	0.15 and 0.85
Belief-flip threshold	0.45
Repair uncertainty strength ω	0.80
Repair dependency decay η	0.70
Maximum repair rounds	3
Protocol freshness clock	Deterministic injected clock; `base_time = 1000 + 100 * seed`
Random state / collection base seed	20260903
Relation classifier	Logistic regression; $C = 4$, balanced class weights; `max_iter`=30,000; converged in 7,693 iterations; `random_state`=20260903
Evidence reliability	Deterministic runtime rule $\mathcal{R}$
Epistemic value predictor v_ϕ	Ridge regression; regularization parameter 1.0
Negative samples per pair	2
g_θ split	Episode-disjoint deterministic Train/Calibration/Test split (504 / 63 / 61 episodes; 4,484 / 635 / 491 pairs)
v_ϕ split	Episode-ID hash; nominal validation fraction 0.1
Temperature calibration	Calibration partition only ($n = 635$ pairs); weighted NLL; $T^\star = 0.6185$
PACEbench reporting protocol	Pass@5 (five attempts per task)

$$\mathrm{ProxyRisk} = \frac{1}{N} \sum_{n=1}^{N} \sum_{a_t^E \in \mathcal{T}_n} R_{\mathrm{base}}(a_t^E).$$

This quantity is used only as an auxiliary proxy for the controller's estimated operational risk and is not treated as an independent measure of real-world safety or harm. Note that the selection objective $J_E(a)$ uses a separate per-template risk prior $Risk_{\mathrm{tpl}}(a)$; therefore, the risk quantity used for candidate ranking and the execution-time proxy reported here are distinct and should not be conflated.

The g_θ relation corpus contains 5,610 evidence–claim pairs from 628 unique episodes. We partition episodes by a deterministic hash of `episode_id` into three mutually disjoint sets: 504 training episodes (4,484 pairs), 63 calibration episodes (635 pairs), and 61 test episodes (491 pairs). The corresponding nominal split is approximately 0.8/0.1/0.1. No episode contributes to more than one partition; consequently, provenance groups associated with an episode also remain within a single

Table 8: Implemented data-collection perturbations. Intensities 1 and 2 are discrete rules; dynamic observation uses the same transition at both levels.

Condition	Intensity 1	Intensity 2	Injection point
Partial	Keep 2 SQL rows or ping replies; mask upload filename or login welcome; retain first half of NodeGoat/bench response.	Keep 1 SQL row or 0 ping replies; mask full upload message or both login outcomes; empty NodeGoat/bench response.	Response content
Dynamic	Set DVWA security level to `impossible` after the control baseline.	Same transition as intensity 1.	Environment state
Adversarial	Substitute SQL surname, command/upload readback marker, or failed-login outcome.	Also add a ghost SQL row or fake ping line; alter upload filename; invert both login outcomes.	Response content
Tool failure	First attempt times out; subsequent attempts execute normally.	First attempt times out; second returns corrupted output; later attempts execute normally.	Tool execution

partition. The relation classifier is fitted only on the training partition, temperature scaling is fitted only on the calibration partition, and final module-level classification and probability metrics are reported only on the independent test partition.

Conflict Resolution Rate (CRR). CRR measures whether a triggered conflict is both reduced below the configured conflict threshold and resolved to the correct environment state. For each evaluable triggered conflict k, we define

$$\mathrm{Resolved}_k = \mathbb{I}\left[\mathrm{Conf}_{t'_k}(c_k) \le \tau_c \ \wedge \ \hat{y}_{t'_k}(c_k) = y^\star_k\right],$$

where c_k is the conflicting claim, t'_k denotes its final evaluated post-repair state, $y^\star_k$ is the corresponding environment ground truth, and $\hat{y}_{t'_k}(c_k)$ is the final inferred claim state. The conflict resolution rate is

$$\mathrm{CRR} = \frac{\sum_{k=1}^{N_{\mathrm{eval}}} \mathrm{Resolved}_k}{N_{\mathrm{eval}}}.$$

Triggered conflicts lacking either a ground-truth snapshot or a final post-repair belief are excluded from both numerator and denominator. We therefore additionally report the evaluable coverage

$$\mathrm{Coverage}_{\mathrm{CRR}} = \frac{N_{\mathrm{eval}}}{N_{\mathrm{triggered}}}.$$

The threshold τ_c is read from the same runtime configuration used by the evaluated controller.

Recovery Success Rate (RSR). To compare repair strategies under identical recovery opportunities, we construct a fixed repair-case set $\mathcal{R}^\star$ before evaluating any strategy. Each case contains the same pre-repair evidence graph, trigger claim, environment ground truth, remaining interaction budget, and available verification paths. All repair strategies are initialized from exactly the same case state.

For a repair strategy π, a case $r \in \mathcal{R}^\star$ is counted as a successful recovery only if the strategy reconstructs the correct actionable belief state and subsequently resumes verified task execution within the fixed remaining budget. Let $A_\pi(r)$ denote successful restoration of the correct actionable belief and $V_\pi(r)$ denote verified task resumption within the remaining case budget. We define

$$\mathrm{Recover}_\pi(r) = \mathbb{I}\left[A_\pi(r) \wedge V_\pi(r)\right],$$

Table 9: Repair-evaluation denominators and coverage. Values are computed over triggered conflicts for CRR and over the shared fixed repair-case set $\mathcal{R}^\star$ for RSR.

Strategy	N_{trig}	N_{eval}	CRR Cov.	CRR Success	RSR Success / $\lvert\mathcal{R}^\star\rvert$
Retry / Reflection	38	37	0.974	7	8 / 38
Repair w/o Dependency	28	27	0.964	14	20 / 38
TRACE	37	36	0.973	26	28 / 38

and

$$\mathrm{RSR}(\pi) = \frac{\sum_{r\in\mathcal{R}^\star} \mathrm{Recover}_\pi(r)}{|\mathcal{R}^\star|}.$$

The same $\mathcal{R}^\star$ is used for Retry/Reflection, dependency-agnostic repair, and TRACE, so the denominator and initial recovery conditions are identical across strategies.

Repair-Affected Dependency Depth (RDD). For repair event i, let $\mathcal{D}_i$ denote the claims reached from the trigger claim through dependency traversal. We define

$$\mathrm{RDD}_i = \max_{c_j\in\mathcal{D}_i} d(i,j), \qquad \mathrm{RDD} = \frac{1}{N_R}\sum_{i=1}^{N_R}\mathrm{RDD}_i,$$

where $d(i,j)$ is the shortest-path hop distance in the claim-dependency graph. RDD characterizes the structural extent reached by repair traversal; it does not imply that every visited claim contains an erroneous belief or that a smaller value is intrinsically preferable.

Recovery Steps. Recovery Steps measures the number of interaction steps between the repair trigger and successful re-establishment of a correct actionable belief state for cases in $\mathcal{R}^\star$. We report the conditional mean over successfully recovered cases only. Failed and budget-exhausted cases contribute to RSR but are not assigned an artificial recovery-step value.

A.4 Baseline Configuration

For the controlled comparison, we use the official implementations of all baseline methods, including CAI, Cybench Agent, HPTSA, PentestGPT, PenForge, and PACEagent. All methods use the same frozen DeepSeek-V4.1-Flash backbone and are evaluated under identical benchmark-side task descriptions, environment interfaces, and interaction budgets. Method-specific planning, memory, orchestration, verification, and internal tool-use logic are preserved. We use lightweight adapters only to standardize model invocation, environment interaction, logging, and result aggregation across methods. No method-specific reasoning or decision policy is modified beyond the interface-level adaptations required for unified evaluation.

B Training and Implementation Details

TRACE freezes the backbone LLM and trains only the Evidence Calibrator g_θ and Epistemic Value Model v_ϕ offline. Figure 2 outlines the training and evaluation architecture. The implemented estimators and checkpoint-specific validation protocols are detailed below.

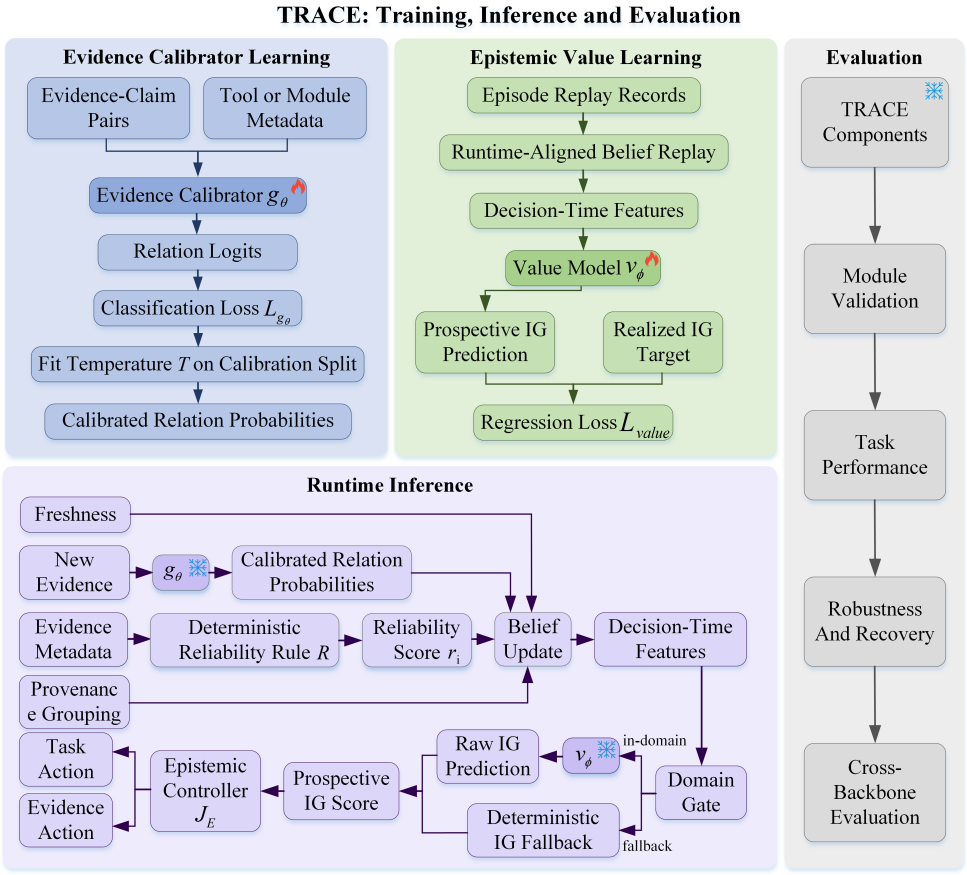


Figure 2: Training, runtime inference, and evaluation of **TRACE**. Offline, the Evidence Calibrator g_θ learns evidence–claim relation logits followed by post-hoc temperature scaling, while the Epistemic Value Model v_ϕ learns prospective information gain from decision-time features using runtime-aligned belief replay. Evidence reliability is computed separately by the deterministic rule $\mathcal{R}$ and is not learned. At runtime, a domain-routing gate selects between the frozen learned value model and a deterministic fallback. The trained components are evaluated through module-level diagnostics, end-to-end benchmark performance, controlled robustness and recovery tests, and cross-backbone evaluation.

B.1 Runtime Belief-State Implementation

Provenance grouping. TRACE heuristically groups likely correlated evidence before belief aggregation to reduce repeated influence from observations that can be linked to the same underlying event. The grouping rule is a deterministic proxy rather than a complete semantic duplicate detector: observations with different event identifiers or falling outside the temporal fallback window may remain in separate groups even when their contents are correlated.

When an evidence item provides a `raw_event_id`, the provenance identifier is constructed from the first 12 hexadecimal characters of its SHA-256 digest:

$$g_i = \mathrm{prefix}_{12}\left(\mathrm{SHA256}(\texttt{raw_event_id})\right).$$

When no raw event identifier is available, TRACE first computes a 10-second temporal bucket,

$$b_i = \left\lfloor \frac{\tau_i}{10\ \mathrm{s}} \right\rfloor,$$

and constructs the provenance key from the tuple

$$(\texttt{entity_id}, \texttt{claim_type}, b_i).$$

The resulting tuple is hashed with SHA-256 and truncated to the first 12 hexadecimal characters. Dataset construction additionally canonicalizes per-session CSRF tokens before provenance clustering so that semantically identical responses are not assigned to separate groups solely because of session-specific token values.

Admissible evidence. An evidence item is admissible for claim c only when both the evidence node itself is active and the evidence–claim relation is represented by an active SUPPORTS or CONTRADICTS edge. Explicit evidence deactivation also deactivates its outgoing edges. Edge invalidation sets the corresponding edge to inactive, increments its graph version, and records an invalidation reason.

B.2 Epistemic Controller Configuration

Contract-defined critical claims. TRACE determines critical claims once when the verification contract for goal g is instantiated. Let p denote a contract predicate and $c(p)$ its corresponding claim. The critical-claim set is

$$\begin{aligned}\mathcal{C}^{\text{crit}}(g) = &\{c(p) \mid \text{role}(p) \in \{\textsc{Precondition}, \textsc{Outcome}\}\} \\ &\cup \{c(p) \mid \text{role}(p) = \textsc{Context}, \text{mandatory}(p)\}.\end{aligned}$$

The set is static within an episode. Context claims created only from contract contradiction predicates are maintained in the evidence graph but are not included in $\mathcal{C}^{\text{crit}}(g)$ unless explicitly required by the contract. Active DEPENDS-ON edges are not used to determine claim criticality; they are consumed only during dependency-aware repair.

TRACE considers feasible epistemic actions targeting these claims:

$$\mathcal{A}^{\text{crit}}_{E,t} = \left\{ a \in \mathcal{A}^{\text{feas}}_{E,t} \;\middle|\; \text{target}(a) \in \mathcal{C}^{\text{crit}}(g) \right\}.$$

Additional evidence is requested only when at least one candidate epistemic action has utility above the epistemic-action threshold:

$$\text{NeedEvidence}_t = \mathbb{I}\left[\mathcal{A}^{\text{crit}}_{E,t} \neq \varnothing \wedge \max_{a \in \mathcal{A}^{\text{crit}}_{E,t}} J_E(a) > \tau_E \right].$$

Otherwise, TRACE proceeds with the task action.

TRACE uses a fixed controller configuration across all evaluated tasks. Table 10 summarizes the inference-time parameters that govern epistemic action selection and belief repair.

With `cooldown_steps`=1, an action identifier executed in the immediately preceding step is excluded from ranking while an unused alternative is available. If no alternative exists, the candidate set is left unchanged.

A claim is marked hard when its originating contract predicate has `kind=hard`; this designation is fixed when the goal contract is instantiated.

Utility components. The non-learned penalties entering J_E are template-level priors assigned when an epistemic candidate is generated; they are not measurements obtained after execution. The three built-in candidate templates use the following settings:

Template	$Cost(a)$	$Risk_{\text{tpl}}(a)$	$Redundancy(a)$
Tool-status re-check	0.10	0.10	0.10
Reachability re-check	0.20	0.10	0.10
Service rescan	0.30	0.20	0.20

Table 10: Inference-time configuration of the TRACE epistemic controller.

Parameter	Setting
Epistemic-action threshold τ_E	0.05
Epistemic-action cooldown	`cooldown_steps` = 1
Maximum epistemic actions per claim	3
Risk budget	10.0
Conflict threshold τ_c	0.30
Decisive-belief interval	$[0, 0.15] \cup [0.85, 1]$
Belief-flip threshold	0.45
Repair strength ω	0.80
Dependency decay η	0.70
Maximum repair rounds	3
Information-gain weight α	1.0
Cost penalty λ	0.2
Risk penalty μ	0.5
Redundancy penalty ν	0.4
Hard-claim positive threshold	0.70
Hard-claim negative threshold	0.30
Hard-claim minimum independent sources	1
Non-hard positive threshold	0.85
Non-hard negative threshold	0.15
Non-hard minimum independent sources	2

$Cost(a)$ is a dimensionless template-specific cost prior fixed before evaluation rather than a measurement of tokens, tool calls, latency, or monetary cost. Similarly, $Risk_{\text{tpl}}(a)$ is a dimensionless template-specific planning prior. The operational risk charged after action selection is computed separately by the frozen risk-estimation and safety modules of the base agent runtime.

B.3 Training Objectives and Estimators

The implemented g_θ combines hashed word unigrams and bigrams (2^{18} features, signed hashing, ℓ_2 normalization) with structured features encoded by a sparse `DictVectorizer`. Text inputs concatenate the claim, evidence, tool identifier, and module. Structured features encode observable response properties such as status, lengths, counts, masking markers, and agreement between claim and response quantities. The calibrator is pairwise rather than history-conditioned: none of its features directly encodes previous beliefs, previous tool outputs, or the number of earlier observations. Temporal context is represented separately by the persistent evidence graph and freshness mechanism.

The relation classifier is multinomial logistic regression with $C = 4$, balanced class weights, `max_iter`=30,000, and `random_state=20260903`. The final fit converges after 7,693 iterations. It minimizes regularized multinomial log-loss with provenance sample weights. The training script additionally fits an auxiliary ridge regressor to the stored `reliability_label` for development diagnostics. This head is not exported with the deployed calibrator (`reliability_head_present=false`) and is never queried during inference. Runtime reliability r_i is instead computed directly by the deterministic rule $\mathcal{R}$. Because the stored `reliability_label` is itself generated from the corresponding deterministic reliability rule, we do not treat this auxiliary regressor as a learned TRACE component or as evidence of learned reliability estimation.

For v_ϕ, a sparse `DictVectorizer` encodes only decision-time features: seven summary statistics over the current claim set, the action identifier, and the module label. The information-gain predictor is a ridge regressor with regularization parameter 1.0 and a squared-error objective. Post-action quantities are excluded from the feature vocabulary. The backbone LLM remains frozen. Scalar temperature fitting follows relation-classifier training and uses the calibration partition described below.

These seven scalar features are combined with one-hot action identifiers (53 identifiers) and the module indicator, yielding the 61-dimensional exported feature vocabulary.

Table 11: Decision-time belief-summary features used by v_ϕ.

Feature	Definition
`mean_entropy`	$\frac{1}{\lvert\mathcal{C}_t\rvert}\sum_c H_2(p_c)$
`max_entropy`	$\max_c H_2(p_c)$
`mean_conflict`	$\frac{1}{\lvert\mathcal{C}_t\rvert}\sum_c \frac{\min(S_c,K_c)}{S_c+K_c+\epsilon}$
`max_conflict`	$\max_c \frac{\min(S_c,K_c)}{S_c+K_c+\epsilon}$
`n_evidence`	Number of claims with nonzero log-odds contribution
`n_claims`	Number of currently required claims
`mean_belief`	$\frac{1}{\lvert\mathcal{C}_t\rvert}\sum_c p_c$

B.4 Evidence Calibration Details

For each evidence–claim pair, the deployed Evidence Calibrator g_θ produces relation logits

$$\mathbf{z} = [z^{sup}, z^{con}, z^{irr}],$$

while the reliability score $r_i \in [0,1]$ is computed independently by the deterministic runtime rule $\mathcal{R}$. Although the offline training script fits an auxiliary reliability ridge for diagnostics, its parameters are not exported and the deployed checkpoint explicitly records `reliability_head_present=false`. Consequently, all reported online TRACE results use deterministic rather than learned reliability.

We apply temperature scaling to the relation logits:

$$p_{rel} = \mathrm{softmax}\left(\frac{\mathbf{z}}{T}\right).$$

The temperature is fitted after classifier training by minimizing provenance-weighted negative log-likelihood:

$$T^\star = \arg\min_{T>0} -\sum_{j\in\mathcal{D}_{cal}} \alpha_j \log\left[\mathrm{softmax}\left(\frac{\mathbf{z}_j}{T}\right)\right]_{y_j}.$$

Here α_j is the stored provenance weight: the inverse occurrence count of a deduplicated training item. We minimize the weighted objective with bounded scalar optimization over $T \in [0.2, 5.0]$, using absolute tolerance `xatol=1e-4`, and freeze the fitted temperature during inference. For the final three-way evaluation protocol, temperature is fitted exclusively on the calibration partition of 635 pairs, yielding

$$T^\star = 0.6185.$$

The calibration partition is not used for final module-level reporting. Relation-level ECE uses 10 equal-width bins of the maximum predicted class probability and compares mean confidence with classification accuracy in each bin. On the independent 491-pair test partition, temperature scaling reduces ECE from 0.0359 to 0.0221 and Brier score from 0.1229 to 0.1201. Because positive temperature scaling preserves the ordering of relation logits, classification accuracy, macro-F1, and per-class recall are unchanged by this post-hoc calibration step.

B.5 Epistemic Value Model Details

The final v_ϕ dataset contains 10,224 action records from 3,691 episodes. Train/validation assignment is grouped by episode ID, yielding 9,205 training records and 1,019 validation records, with no episode shared across the two partitions. The predictor uses only information available at decision time: seven belief-summary statistics, the action identifier, and the module label; post-action quantities such as `observation_success` are excluded.

Table 12: Episode-disjoint partitions used for training and evaluating g_θ.

Partition	Episodes	Pairs	Support	Contradiction	Irrelevant
Train	504	4,484	1,290	371	2,823
Calibration	63	635	193	52	390
Test	61	491	144	37	310

Offline belief replay uses the same wall-clock freshness policy and binary entropy definition as the deployed controller. Entropy is measured in bits using $\log_2$, and the prior entropy of a newly introduced claim is derived from the same entropy function,

$$H_2(0.5) = 1 \text{ bit},$$

rather than supplied as an independent constant.

Under this deployment-consistent target construction, the strictly prospective information-gain predictor achieves MAE 0.0255 and $R^2 = 0.6355$ on the held-out episode-grouped split, compared with a zero-prediction baseline MAE of 0.0825. The final checkpoint reported in this paper uses this configuration.

Raw prediction at inference. The supervised target is the realized entropy change and may in principle take negative values. In the final training corpus, the target lies in $[0, 0.3201]$ and contains no negative values. At inference, TRACE uses the raw regression output of v_ϕ without clipping. On the held-out split, 0.79% of predictions are slightly negative, with a minimum of -0.0108, while no prediction exceeds 1. These raw predictions enter J_E directly; negative values therefore reduce the predicted utility of the corresponding epistemic action rather than being truncated to zero.

Inactive conflict-reduction compatibility output. The runtime value-model interface retains a conflict-reduction output $\widehat{CR}$ for compatibility with an earlier controller design. However, the current offline replay corpus contains no nonzero conflict-reduction labels. Consequently, the exported checkpoint returns

$$\widehat{CR}(a) = 0$$

for all evaluated inputs, and this quantity does not enter the deployed epistemic utility or action ranking reported in this paper.

The current TRACE controller therefore uses

$$J_E(a) = \alpha \widehat{IG}(a) - \lambda Cost(a) - \mu Risk_{\text{tpl}}(a) - \nu Redundancy(a).$$

with conflict handling provided instead by independent verification, explicit support–contradiction detection in the evidence graph, deterministic candidate priors, and dependency-aware belief repair. We retain the unused output only for software-interface compatibility and make no claim of learned conflict-reduction prediction.

B.6 Data Partitions and Belief-Metric Computation

Training corpus and checkpoint partitions. The relation corpus used for g_θ contains 5,610 evidence–claim pairs from 628 unique episodes. We partition the corpus by a deterministic hash of `episode_id` into three mutually disjoint sets: training, calibration, and test. No episode contributes to more than one partition. The classifier is fitted only on the training partition, temperature scaling is fitted only on the calibration partition, and final module-level metrics are reported only on the independent test partition.

Belief-metric sample unit. The DVWA belief evaluation covers SQL Injection, File Upload, Command Injection, and Brute Force. For state accuracy, Brier score, and belief ECE, the metric implementation retains the last recorded probability for each claim within an episode and evaluates

claims with available ground-truth labels. Each sample is therefore a terminal (episode, claim) pair. State accuracy thresholds the probability at 0.5. Within each method–condition group, the protocol runner takes the unweighted mean of available per-episode metric values.

Belief ECE and unsupported-belief rate. Belief ECE uses 10 equal-width probability bins over $[0, 1]$, comparing the mean belief probability with the empirical fraction of true claims within each bin. This binary probability-calibration metric differs from the relation-level top-class ECE reported for the Evidence Calibrator.

For unsupported-belief evaluation, each `BeliefRecord` stores a claim-level freshness value equal to the maximum freshness among currently admissible evidence members with positive effective weight:

$$F_t(c) = \max(\{f_i(c,t) \mid e_i \in \mathcal{A}_t(c),\ w_i(c,t) > 0\} \cup \{0\}),$$

where $\mathcal{A}_t(c)$ denotes the evidence currently admissible for claim c. A decisive claim record is counted as unsupported when

$$\text{Unsupported}_t(c) = \mathbb{I}[F_t(c) = 0].$$

For Table 3, UBR is evaluated only on in-domain decisive events, i.e., events for which the pre-event belief lies on the incorrect side of the 0.5 decision threshold. Let $\mathcal{R}^{\text{in}}_{\text{dec}}$ denote this set. We define

$$\text{UBR} = \frac{\sum_{(t,c)\in\mathcal{R}^{\text{in}}_{\text{dec}}} \text{Unsupported}_t(c)}{|\mathcal{R}^{\text{in}}_{\text{dec}}|}.$$

Because the evaluated trajectories do not reach any configured hard-expiration horizon, UBR in this experiment reflects absence of currently admissible positive-weight evidence rather than temporal expiration.

Claims governed by `episode` or `snapshot` policies use the `default` exponential decay and have no hard-expiration boundary. They therefore cannot become unsupported through temporal expiration alone and are counted as unsupported only when no admissible positive-weight evidence remains. In contrast, `dynamic` and `session` evidence may become stale through their configured hard-expiration boundaries.

For the belief-level evaluation in Table 3, the held-out DVWA subset contains 20 episodes and 299 evidence–claim pairs. Across all four calibrator variants, the terminal evaluation contains 117 $(\text{episode}, \text{claim})$ records, of which 71 are in-domain and used for the learned-calibrator comparison in Table 3; the remaining 46 records are routed to the deterministic fallback. The decisive-event evaluation contains 37 records overall, of which 20 are in-domain and form the UBR denominator. Thus, State Accuracy, Brier score, and ECE use $N_{\text{terminal}} = 71$, while UBR uses $N_{\text{UBR}} = 20$. These denominators are identical for all four calibrator variants.

Table 3 reports belief metrics on the fixed in-domain DVWA subset, whereas Tables 6, 13, and 14 report metrics from their corresponding end-to-end evaluation runs; their Brier/ECE values therefore need not coincide.

C Additional Experimental Results

C.1 Cross-Backbone Evaluation

To assess TRACE across different frozen LLM backbones, we evaluate the same epistemic-control modules with three backbone models while keeping the remaining configuration fixed.

Across the three evaluated backbones, VTS ranges from 30.0% to 35.0%, ECE from 0.158 to 0.165, and RSR from 73.7% to 78.9%. These results provide preliminary evidence that TRACE remains effective across the evaluated backbone choices; broader model coverage is left for future work.

Table 13: Cross-backbone evaluation of TRACE.

Backbone	VTS ↑	ECE ↓	RSR ↑
DeepSeek-V4.1-Flash	32.5%	0.162	73.7%
GPT-5.6 Sol	35.0%	0.158	78.9%
Kimi-K3	30.0%	0.165	73.7%

C.2 Relation to Published Baseline Results

The original baseline studies use heterogeneous backbones, prompts, interaction budgets, benchmarks, and evaluation protocols. We therefore do not use their published numbers for direct quantitative comparison with TRACE. All comparative conclusions in this work are based on the controlled same-backbone evaluation reported in Table 1, where official baseline implementations are evaluated using the same frozen DeepSeek-V4.1-Flash backbone, benchmark-side interface, and interaction budget. Published baseline results are used only as historical context and are cited in Section 2.

C.3 Hyperparameter Sensitivity

We evaluate the sensitivity of TRACE to three key epistemic-control hyperparameters while keeping all remaining settings fixed. Specifically, we vary the epistemic-action threshold τ_E, conflict threshold τ_c, and dependency-decay factor η around their default values.

Table 14: Sensitivity of TRACE to key epistemic-control hyperparameters. Bold denotes the default configuration.

Parameter	Value	VTS ↑	Brier ↓	RSR ↑
τ_E	0.025	30.0%	0.132	73.7%
	0.05	**32.5%**	**0.135**	**73.7%**
	0.10	27.5%	0.146	65.8%
τ_c	0.20	30.0%	0.144	68.4%
	0.30	**32.5%**	**0.135**	**73.7%**
	0.40	27.5%	0.149	60.5%
η	0.50	32.5%	0.140	68.4%
	0.70	**32.5%**	**0.135**	**73.7%**
	0.90	30.0%	0.138	68.4%

C.4 Offline Module Validation

Final Evidence Calibrator. Table 15 reports g_θ on the independent episode-disjoint test partition. The test set contains 491 evidence–claim pairs from 61 episodes: 144 support, 37 contradiction, and 310 irrelevant pairs. The classifier achieves accuracy 0.9185, macro-F1 0.8307, and contradiction recall 1.0000. All 37 contradiction examples in the held-out test partition are correctly classified; given the limited class size, this should not be interpreted as evidence of perfect contradiction detection beyond the evaluated set. Temperature scaling is fitted separately on the 635-pair calibration partition and yields $T = 0.6185$.

Temperature scaling is fitted exclusively on the calibration partition ($n = 635$) and evaluated on the disjoint test partition ($n = 491$), as reported above. Evidence reliability is not evaluated as a learned prediction task: the deployed system computes r_i directly using the deterministic runtime rule $\mathcal{R}$.

Prospective Epistemic Value Model. Table 16 summarizes the offline diagnostics of v_ϕ; the runtime-aligned prospective configuration described above is the deployed model used in all reported experiments.

The decrease from the earlier development variants reflects removal of non-prospective information and target structure induced by the mismatched entropy convention; only the final deployment-consistent result is used to characterize v_ϕ.

Table 15: Held-out evaluation of the final g_θ on the episode-disjoint 491-pair test partition.

Metric	Uncalibrated	+ Temperature Scaling
Accuracy	0.9185	0.9185
Macro-F1	0.8307	0.8307
Contradiction recall	1.0000	1.0000
ECE	0.0359	**0.0221**
Brier	0.1229	**0.1201**

Table 16: Development diagnostics for v_ϕ. Only the final deployment-consistent configuration is used in the reported experiments.

Configuration	R^2
Post-action feature + post-action target factor	0.9855
Prospective features + legacy target	0.8760
Prospective target + step freshness (nats)	0.9710
Runtime-aligned freshness (nats)	0.9671
Runtime-aligned freshness + bits (final)	**0.6355**

Freshness activation in the replay corpus. Although replay follows the deployed freshness policy, temporal decay is numerically negligible in this corpus: episode duration has median 0 s and maximum 31 s, compared with a default decay time constant of 10^5 s, and neither the 3,600 s session nor the 21,600 s dynamic expiration horizon is reached. Most claims also map to the default freshness class. We therefore do not attribute performance differences in this short-horizon corpus to temporal decay or hard expiration; these mechanisms are expected to become consequential primarily in longer-running or cross-session interactions.

Supervised label construction for v_ϕ. The value model is trained offline on action records extracted from collected episodes. Its inputs are strictly prospective, whereas the supervised target is the realized next-step change in the replayed belief state. For each executed epistemic action a_t^E, we use the weighted entropy change

$$IG(a_t^E) = \sum_{c \in \mathcal{C}_t} \rho(c) \big[H_t(c) - H_{t+1}(c) \big].$$

Unlike an earlier revision, the final target is not multiplied by an `observation_success` gate or any other post-action success factor. Post-action quantities may define the realized supervision target through the observed transition, but they are never supplied as predictor inputs at decision time.

$\rho(c)$ is the claim-importance weight used in this offline reconstruction; it is 1.0 for core task claims, 0.5 for auxiliary claims, and 0.3 for the remaining claims. These role-based weights are used only for offline value-model target construction and are not used in the online computation of U_t.

The offline belief replay uses the same freshness rule as the deployed controller. For evidence e_i evaluated at replay time t_j,

$$f_i(c, t_j) = \exp\big(-\lambda_{\kappa(c)} \Delta_i(t_j)\big)\, \mathbb{I}\left[\Delta_i(t_j) < H_{\kappa(c)}\right], \qquad \Delta_i(t_j) = \max(0, t_j - \tau_i).$$

The decay rates and expiration horizons are loaded from the runtime configuration, and freshness is recomputed at each decision step rather than fixed when evidence is inserted. Claim-type mapping follows the same registry used by the deployed controller.

D FULL TRACE ALGORITHM

Algorithm 1 TRACE online epistemic-control and belief-repair procedure.

Require: Goal g, initial evidence graph G_0, interaction budget B, epistemic threshold τ_E
Ensure: Verification status, belief trace, and final evidence graph

$G \leftarrow G_0$
Initialize $b_0(c) \leftarrow 0.5$ for every initial claim c
$\mathcal{C}^{\text{crit}} \leftarrow \textsc{CriticalClaims}(g)$
Initialize observation o_0 from the environment
for $t = 0, \ldots, B$ **do**
    $\mathcal{E}_t \leftarrow \textsc{ExtractEvidence}(o_t)$
    $\widetilde{\mathcal{E}}_t \leftarrow \textsc{CalibrateEvidence}(\mathcal{E}_t, g_\theta)$
    $G \leftarrow \textsc{UpdateGraph}(G, \widetilde{\mathcal{E}}_t)$
    $b_t \leftarrow \textsc{RecomputeBeliefs}(G, \xi_t)$
    if $\textsc{RepairTriggered}(b_t, G)$ **then**
        $(\mathcal{C}_t^{\text{aff}}, d_t) \leftarrow \textsc{TraceDependencies}(G)$
        $G \leftarrow \textsc{InvalidateSupport}(G, \mathcal{C}_t^{\text{aff}})$
        $\widetilde{b}_t \leftarrow \textsc{RecomputeBeliefs}(G, \xi_t)$
        $b_t \leftarrow \textsc{DependencyDiscount}(\widetilde{b}_t, d_t, \omega, \eta, \pi_0)$
    end if
    if $\textsc{VerifyGoal}(g, b_t, G)$ **then**
        return VERIFIED, belief trace, G
    end if
    if $t = B$ **then**
        break
    end if
    $\mathcal{A}_{E,t}^{\text{crit}} \leftarrow \textsc{EpistemicCandidates}(\mathcal{C}^{\text{crit}}, G)$
    $a_t^{E*} \leftarrow \varnothing$
    if $\mathcal{A}_{E,t}^{\text{crit}} \neq \varnothing$ **then**
        $a_t^{E*} \leftarrow \arg\max_{a \in \mathcal{A}_{E,t}^{\text{crit}}} J_E(a)$
    end if
    if $a_t^{E*} \neq \varnothing \wedge J_E(a_t^{E*}) > \tau_E$ **then**
        Execute a_t^{E*} and receive $(o_{t+1}, \text{status}_t)$
        if $\text{status}_t = \textsc{Failure}$ **then**
            $o_{t+1} \leftarrow \textsc{EncodeFailure}(a_t^{E*}, \text{status}_t)$
        end if
    else
        $a_t^T \leftarrow \pi_T(g, b_t, G)$
        Execute a_t^T and receive o_{t+1}
    end if
end for
return UNVERIFIED, belief trace, G