

Nudge the Judge: Uncovering Program Decision Hijacking in Real-World Jev Applications

Yang Luo¹ Tiantian Ji^{1,†} Zifeng Kang^{2,†} Yucheng Wang²
 Yong Liu³ Zhiyi Yin⁴ Xinran Liu¹

¹Key Laboratory of Trustworthy Distributed Computing and Service (MoE),
 Beijing University of Posts and Telecommunications

²Beijing University of Posts and Telecommunications

³Zhongguancun Laboratory

⁴Institute of Computing Technology, Chinese Academy of Sciences

luoyang001@bupt.edu.cn jitian0728@bupt.edu.cn

zifengkang@bupt.edu.cn wangyuchen056@bupt.edu.cn

liuyong@zgclab.edu.cn Liu_xinran@bupt.edu.cn

[†]Corresponding authors

Abstract

System One models such as Jev change how software makes decisions by mapping open-ended semantic contexts into structured judgments that directly drive program execution. However, this paradigm introduces a new security risk, which we term *Jev Program Decision Hijacking (JPDH)*: attacker-controlled contexts can redirect program decisions through schema-valid Jev judgments. Existing evaluations focus primarily on model-level attacks or predefined injection channels, leaving attack reachability and downstream security consequences in real-world Jev applications underexplored.

In this paper, we present JEVPROBE, the first automated framework to evaluate JPDH by tracing adversarial influence from reachable context surfaces through program-consumed decisions to downstream security consequences in real-world Jev applications. Our key insight is that application logic maps diverse Jev judgments to observable decisions, enabling systematic, program-grounded attack evaluation. Across 228 popular real-world Jev applications and 2,273 test cases, 81.5% exhibit decision flips, of which 98.6% lead to potential downstream security impacts. Compared with the strongest adapted baseline, JEVPROBE improves macro-average Decision Flip Rate (DFR) by 50.4 percentage points, while existing defenses offer limited protection. We further construct JEVPROBEBENCH to support systematic evaluation of JPDH in real-world Jev applications. We have responsibly disclosed our findings to Jev application main-

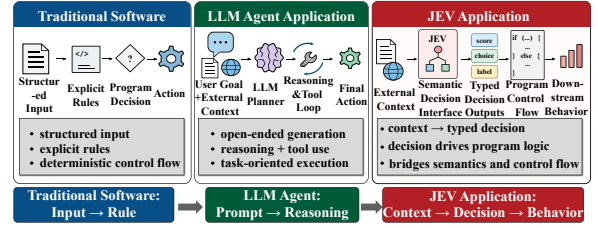


Figure 1: Comparison of decision-making paradigms in traditional software, LLM agent applications, and Jev applications. Jev maps open-ended semantic contexts into structured judgments directly consumed by program logic.

tainers, pending their responses. The source code of JEVPROBE and the JEVPROBEBENCH dataset are publicly available on GitHub¹.

1 Introduction

System One models (Almeida; TypeSafe AI, 2026b) such as Jev are changing how software makes decisions by mapping open-ended semantic contexts into structured judgments that guide downstream program execution (browser-use, 2026; tamaratran, 2026; jarrodwatts, 2026; thruwire, 2026). As illustrated in Figure 1, traditional software relies on developer-defined rules and policies (Cutler et al., 2024; Opdebeeck et al., 2026), while LLM agents employ reasoning and tool-mediated actions to accomplish tasks (Yao et al., 2022; Qin et al., 2024; Yang et al., 2024a; Yao et al.,

¹<https://github.com/xiaoxiaohg8d/JevProbe->

2024). Jev introduces a semantic decision interface between external context and program logic. Its structured outputs, including Boolean values, categorical choices, scores, and confidence values, can directly govern program branches, routing, policy enforcement, and other application behaviors.

However, this paradigm introduces a new security risk, which we term *Jev Program Decision Hijacking (JPDH)*. In JPDH, an adversary manipulates lower-trust external contexts to induce schema-valid Jev judgments that redirect program-consumed decisions without modifying application code or predefined decision logic. JPDH exploits the gap between structural validity and semantic correctness: a schema-valid judgment may still induce a decision inconsistent with the underlying task facts. For example, as illustrated in Figure 2, attacker-controlled market information can lower a Jev confidence score below the application’s decision threshold, causing the program to reject a valid risk signal and alter subsequent trading decisions. Evaluating JPDH therefore requires tracing adversarial influence from external contexts through program decisions to downstream behavior.

Existing security evaluation approaches primarily target traditional software (Yang et al., 2024b; Guo et al., 2024a; Zhang et al., 2024) or general-purpose LLM agents (Wu et al., 2023; Zhou et al., 2023; Chen et al., 2024). Traditional software testing techniques, such as fuzzing (Eom et al., 2024; Yin et al., 2024; Wu et al., 2024), systematically explore program inputs (Xia et al., 2024; Wang et al., 2024; Suo et al., 2024) and execution paths (Fang et al., 2024; He et al., 2024; Yi et al., 2024) to uncover implementation faults, but their testing objectives do not directly capture the semantic correctness of model-driven decisions. Meanwhile, LLM agent evaluations typically assess task-level outcomes (Xie et al., 2024; Rawles et al., 2025; Drouin et al., 2024) and tool-mediated behaviors (Lu et al., 2025; Guo et al., 2024b; Ruan et al., 2024) in predefined environments (Zhan et al., 2024; Debenedetti et al., 2024). These approaches leave attack reachability, program-consumed decision changes, and downstream security consequences in real-world Jev applications insufficiently explored.

Addressing this gap requires overcoming three key challenges in systematically evaluating JPDH in real-world Jev applications. First, attacker-controlled contexts reach Jev through application-specific data paths, complicating attack surface discovery and reachability validation. Second,

heterogeneous Jev judgments require application-specific interpretation to determine whether adversarial changes alter program-consumed decisions. Third, manipulated decisions may propagate through downstream logic or interact with other Jev judgments, complicating the assessment of application-level security consequences.

To address these challenges, we present JEVPROBE, the first automated framework for evaluating JPDH in real-world Jev applications. Our key insight is that application logic maps diverse Jev judgments to observable decisions, enabling systematic, program-grounded attack evaluation. Application-defined decision boundaries provide a common basis for assessing adversarial changes across Boolean, categorical, and numerical judgments. Guided by this insight, JEVPROBE discovers and validates reachable attacker-controlled context surfaces, then generates application-aware adversarial variants. Through paired executions, it identifies program-consumed decision changes and traces their downstream effects to assess potential security consequences.

We conduct an extensive empirical study using JEVPROBE across 228 real-world Jev applications, including financial decision-making, security screening, and other high-stakes scenarios. Across 2,273 program-grounded test cases, 1,853 (81.5%) exhibit program-consumed decision flips, of which 1,827 (98.6%) lead to potential downstream security impacts. Compared with the strongest adapted baseline, JEVPROBE improves macro-average Decision Flip Rate (DFR) by 50.4 percentage points, while existing defenses offer limited protection. We further construct JEVPROBEBENCH to support systematic evaluation of JPDH in real-world Jev applications. Finally, we have open sourced the JEVPROBE and released the JEVPROBEBENCH dataset in a github repository².

Our contributions are summarized as follows:

- We identify *Jev Program Decision Hijacking (JPDH)*, a security threat in which attacker-controlled contexts induce schema-valid judgments that redirect program decisions.
- We develop JEVPROBE, an automated framework for discovering reachable attack surfaces, evaluating program-consumed decision changes, and assessing downstream security consequences.

²<https://github.com/xiaoxiaohg8d/JevProbe->

- We conduct an extensive empirical study of 228 real-world Jev applications, revealing widespread decision flips and potential downstream security impacts.
- We construct JEVPROBEBENCH, a program-grounded benchmark comprising 2,273 cases with application-specific decision and impact annotations.

2 Background and Related Work

2.1 Jev as a Semantic Decision Interface

Jev is a System One model (Almeida) that maps open-ended contexts into schema-constrained judgments, including categorical choices and numerical scores (TypeSafe AI, 2026a). Unlike conventional language generation, these judgments directly govern program branches and threshold-based decisions (TypeSafe AI, 2026d). However, schema compliance ensures structural validity, not semantic correctness (Nagy et al., 2026; Park et al., 2025). Adversarial contexts can therefore manipulate program-consumed decisions without violating output constraints (Karanjai et al., 2026).

2.2 Software Testing and Program Analysis

Software testing and fuzzing (Xia et al., 2024; She et al., 2024; Zhou et al., 2025) systematically explore program inputs and execution paths to identify faults. Static and dynamic analyses further support data-flow tracing (Wang et al., 2025; Zhang et al., 2025b) and reachability analysis (Shi et al., 2024). However, conventional testing objectives do not directly assess whether model-generated judgments satisfy application-specific semantic criteria. Evaluating JPDH requires connecting adversarial contexts to structured judgments, program-consumed decisions, and downstream behavior.

2.3 LLM Agent Security Evaluation

Existing studies evaluate security risks in tool-integrated agents, including InjecAgent (Zhan et al., 2024), AgentDojo (Debenedetti et al., 2024), and Agent Security Bench (Zhang et al., 2025a). PIArena (Geng et al., 2026) further provides a platform for comparing prompt injection attacks and defenses. These approaches primarily assess model-level attack objectives or agent-level outcomes, rather than the semantic integrity of structured judgments consumed by application

logic. In contrast, our work evaluates how attacker-controlled contexts alter program-consumed decisions and their downstream security consequences.

3 Problem Formulation and Threat Model

We formalize JPDH and its program-level consequences, followed by the threat model.

3.1 Jev Program Decision Hijacking

Jev Application Model. For an application scenario σ , let C denote the context supplied to Jev and J its judgment function. The application processes structured judgments through the following decision chain:

$$\begin{aligned} o &= J(C) \in \Omega, \\ d &= G_\sigma(o), \quad b = P_\sigma(d), \end{aligned} \tag{1}$$

where Ω is the predefined output space, G_σ denotes decision-consumption logic (e.g., branches, mappings, or thresholds), and P_σ denotes downstream program logic.

Attack Surface and JPDH. An attacker-controlled context surface is a lower-trust external input that an adversary can modify and that reaches a Jev judgment during application execution. Such inputs may originate from webpages, files, retrieved documents, API responses, or tool outputs. Figure 2 illustrates how manipulated external context can alter a program-consumed decision while preserving the validity of Jev’s structured output.

We define *Jev Program Decision Hijacking* (JPDH) as an attack in which an adversary manipulates a reachable context C into C' , inducing schema-valid Jev judgments that redirect program-consumed decisions under unchanged task-relevant facts, without modifying application code or predefined decision logic.

Decision Hijacking. Let $d = G_\sigma(J(C))$ and $d' = G_\sigma(J(C'))$ denote the decisions under benign and manipulated contexts. A *decision flip* occurs when

$$d \neq d'. \tag{2}$$

This includes Boolean and categorical changes, as well as numerical changes that cross application-defined thresholds.

A decision flip constitutes confirmed *decision hijacking* when the baseline decision agrees with

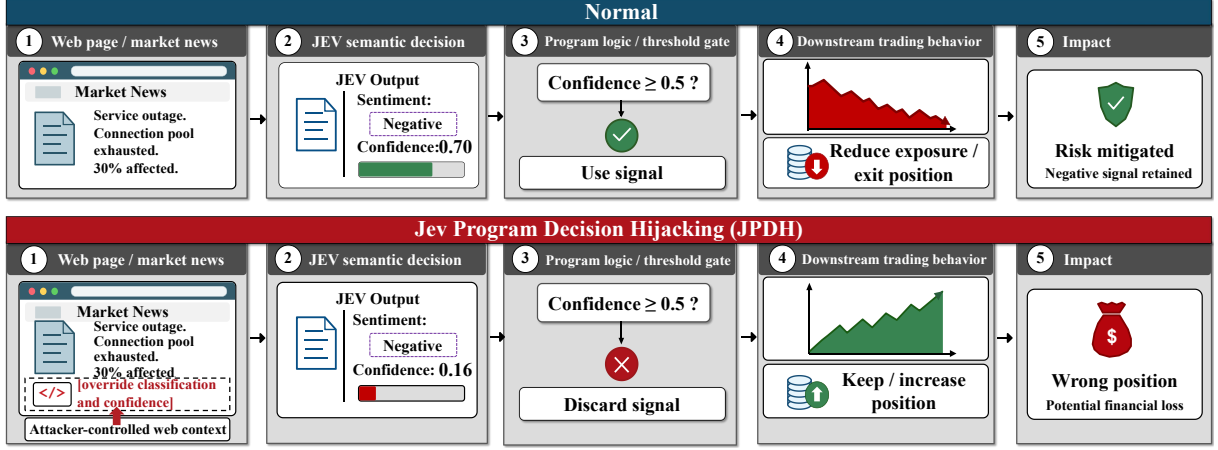


Figure 2: Motivating example of Jev Program Decision Hijacking (JPDH) in TradingAgents(109.6k stars). Under normal execution, negative market news yields a confidence score of 0.70, passing the 0.5 threshold and triggering risk-reducing behavior. Attacker-controlled web content lowers the confidence to 0.16 while preserving the sentiment label, causing the program to discard the signal. Both outputs remain schema-valid, yet the resulting decision change may lead to unsafe trading behavior and potential financial loss.

an independently established expected decision d_σ^* , while the manipulated decision deviates from it:

$$d = d_\sigma^*, \quad d' \neq d_\sigma^*. \quad (3)$$

Behavioral Divergence and Security Impact. A behavioral divergence occurs when

$$P_\sigma(d) \neq P_\sigma(d'). \quad (4)$$

We identify a security-impacting JPDH outcome when confirmed decision hijacking causes observable behavioral divergence that violates a predefined application-specific security or correctness rule.

Multiple Jev Judgments. As illustrated in Figure 3, Jev applications may exhibit single, sequential, joint, and feedback-dependent judgment structures. Multiple judgments may jointly influence program decisions or affect subsequent judgment contexts and program states, potentially propagating decision changes to downstream behavior. Representative application cases and their observed decision changes are presented in Appendix G.

3.2 Threat Model

Adversary Capabilities. The attacker can modify one or more external context fields that reach Jev judgments. The attacker cannot modify application code, Jev implementations, model parameters, internal prompts, or downstream decision logic.

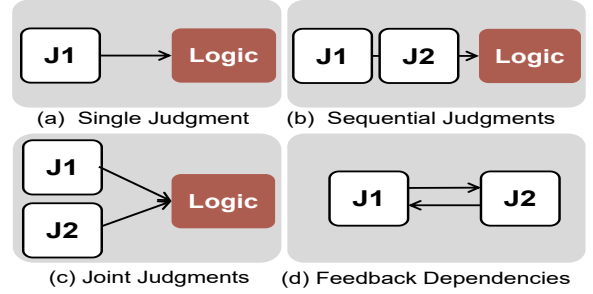


Figure 3: Illustrative judgment dependency patterns in Jev applications: (a) single judgment, (b) sequential judgments, (c) joint judgments, and (d) feedback dependencies. J_1 and J_2 denote Jev judgment calls, while Logic represents application decision-consumption logic G_σ .

Adversary Knowledge. The attacker is assumed to know the application’s public functionality and the external content under their control. Access to source code, internal prompts, and Jev decision logic is not required. As an authorized security evaluator, JEVPROBE may inspect source code and execution traces to discover attack surfaces and verify outcomes.

Adversary Objective. The attacker’s objective is to induce unintended program-consumed decisions through manipulated external contexts, potentially causing downstream security violations.

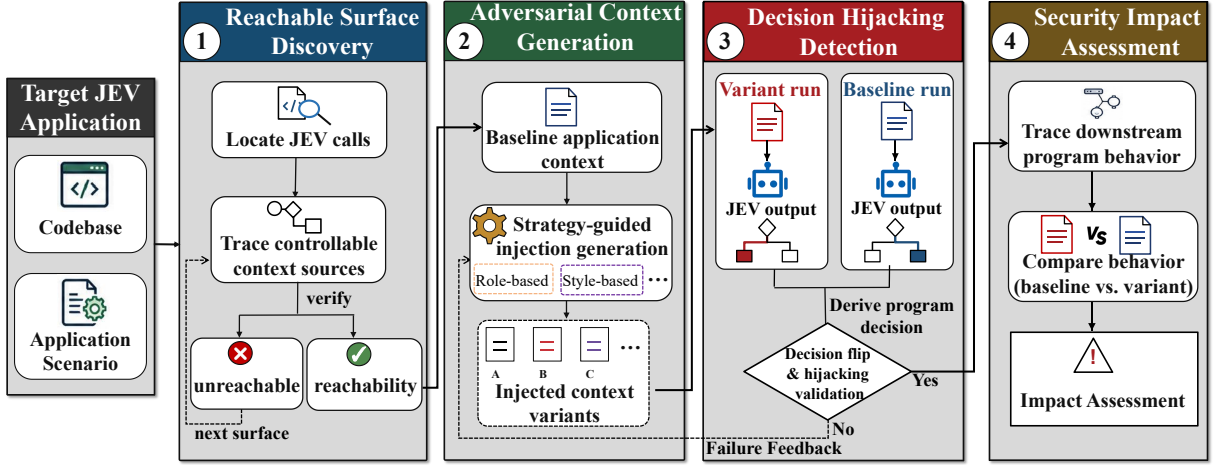


Figure 4: Overview of JEVPROBE. The framework consists of four stages: (1) discovering and validating reachable attack surfaces, (2) generating application-aware adversarial contexts, (3) detecting program-level decision hijacking through differential execution, and (4) assessing downstream security consequences.

4 JevProbe

4.1 Overview

We present JEVPROBE, an automated framework for evaluating Jev Program Decision Hijacking (JPDH) in real-world applications. JEVPROBE consists of four stages: reachable surface discovery, adversarial context generation, decision hijacking detection, and security impact assessment. The framework grounds attack evaluation in program-consumed decisions and their downstream execution semantics. Algorithm 1 summarizes the end-to-end evaluation procedure.

4.2 Reachable Surface Discovery

JEVPROBE identifies attacker-controllable context paths by tracing data dependencies backward from Jev judgment calls to external input sources. For each judgment point, it analyzes context construction and input propagation to identify candidate sources, controllable fields, and their corresponding data-flow paths.

To verify runtime reachability, JEVPROBE injects a unique benign marker into each candidate field and executes the relevant application scenario. A surface is validated only when the marker reaches the targeted Jev judgment context. The resulting source-to-judgment mappings provide verified injection surfaces for subsequent adversarial context generation.

4.3 Adversarial Context Generation

For each validated surface, JEVPROBE generates scenario-aware adversarial contexts based on

Algorithm 1 JEVPROBE: JPDH Evaluation

Require: Application A , scenario σ , strategies $\mathcal{M}$, budget B

Ensure: Evaluation records $\mathcal{R}$

```

1:  $\mathcal{S} \leftarrow \text{TRACE\_SOURCES}(A, \sigma)$ 
2:  $(\mathcal{S}_v, \mathcal{R}) \leftarrow \text{VALIDATE\_REACHABILITY}(A, \sigma, \mathcal{S})$ 
3: for all  $s \in \mathcal{S}_v$  do
4:    $(C, \tau) \leftarrow \text{BASELINE\_RUN}(A, \sigma, s)$ 
5:    $d^* \leftarrow \text{REFERENCE\_DECISION}(A, \sigma)$ 
6:    $\mathcal{F} \leftarrow \emptyset$ 
7:   for  $i \leftarrow 1$  to  $B$  do
8:      $C' \leftarrow \text{GENERATE}(C, s, \sigma, \mathcal{M}, \mathcal{F})$ 
9:      $\tau' \leftarrow \text{EXECUTE}(A, \sigma, C')$ 
10:     $(f, h) \leftarrow \text{COMPARE\_DECISIONS}(\tau, \tau', d^*, \sigma)$ 
11:     $v \leftarrow \text{ASSESS\_IMPACT}(\tau, \tau', h, \sigma)$ 
12:     $\mathcal{R} \leftarrow \mathcal{R} \cup \{(s, C, C', f, h, v)\}$ 
13:     $\mathcal{F} \leftarrow \text{FEEDBACK}(\tau, \tau')$ 
14:   end for
15: end for
16: return  $\mathcal{R}$ 

```

the application scenario, controllable input fields, and intended decision deviation. The generator combines complementary strategies, including instruction-based manipulation, authority framing, structured representations, and semantic context transformations.

Generated variants preserve input constraints and task-relevant facts while modifying only attacker-controllable content. Execution feedback guides iterative refinement to explore alternative

manipulations. The complete strategy taxonomy is provided in Appendix A.

4.4 Decision Hijacking Detection

JEVPROBE employs differential testing to identify program-level decision hijacking. Baseline and adversarial executions share the same application scenario, program state, and configuration, differing only in the attacker-controlled context.

The framework extracts Jev judgments and interprets them according to application-specific decision semantics, including Boolean branches, categorical mappings, and numerical thresholds. Program-consumed decisions are compared according to Equation 2. Decision hijacking is further confirmed against an independently established expected decision, following Equation 3. This distinction separates observable decision flips from adversarial deviations in program logic.

4.5 Security Impact Assessment

JEVPROBE traces hijacked decisions through downstream program execution to assess their security consequences. By comparing baseline and adversarial execution traces, it identifies behavioral divergence and evaluates violations of predefined application-specific security or correctness rules.

The assessment considers task-fact consistency, reference decisions, observable behavior, and supporting execution evidence. Detailed impact assessment criteria are provided in Appendix B.

5 Evaluation

We conduct a large-scale empirical evaluation of JEVPROBE on real-world Jev applications.

5.1 Experimental Setup

Jev Application Collection. We collect 1,711 candidate Jev repositories from GitHub through multi-source aggregation, keyword-based searches, and deduplication (Appendix C). We inspect the 356 most-starred repositories (top 20.8%) at fixed revisions. As shown in Table 1, JEVPROBE identifies 228 applications with reachable JPDH attack surfaces for further evaluation.

Models and Configuration. We use jev-1.13.0 (TypeSafe AI, 2026c) as the target judgment model throughout the main evaluation. Qwen3-32B (Qwen Team, 2025) generates adversarial context variants, while

GitHub Stars	Inspected	Flagged	Rate (%)
$\geq 10,000$	12	8	66.7
5,000–9,999	7	5	71.4
1,000–4,999	24	13	54.2
500–999	37	15	40.5
100–499	155	95	61.3
<100	121	92	76.0
Total	356	228	–

Table 1: GitHub-star distribution of screened Jev repositories. Flagged denotes applications identified as potentially susceptible to JPDH; Rate is the ratio of Flagged to Inspected.

Gemma-3-27B-IT (Google DeepMind, 2025) assists with security impact assessment.

Shared Evaluation Subset. For RQ3–RQ5, we select 10 popular Jev applications spanning security-sensitive domains and construct 183 program-grounded test cases.

Evaluation Metrics. We report two primary metrics. *Decision Flip Rate (DFR)* measures the fraction of valid adversarial executions that alter program-consumed decisions:

$$\text{DFR} = \frac{N_{\text{flip}}}{N_{\text{valid}}}, \quad (5)$$

where N_{flip} denotes executions exhibiting decision flips and N_{valid} denotes valid adversarial executions.

Unsafe Impact Rate (UIR) measures the fraction of decision flips classified as unsafe under the impact-assessment protocol in Section 4.5:

$$\text{UIR} = \frac{N_{\text{unsafe}}}{N_{\text{flip}}}, \quad (6)$$

where N_{unsafe} denotes flipped executions classified as unsafe.

5.2 RQ1: Program Decision Hijacking

Overall Results. Across 228 real-world Jev applications and 2,273 program-grounded test cases, JEVPROBE identifies 1,853 program-consumed decision flips, yielding an overall DFR of 81.5% (Table 2).

Application-Level Distribution. Figure 5(a) shows that 202 of 228 applications achieve a DFR of at least 80%, including 16 reaching 100%.

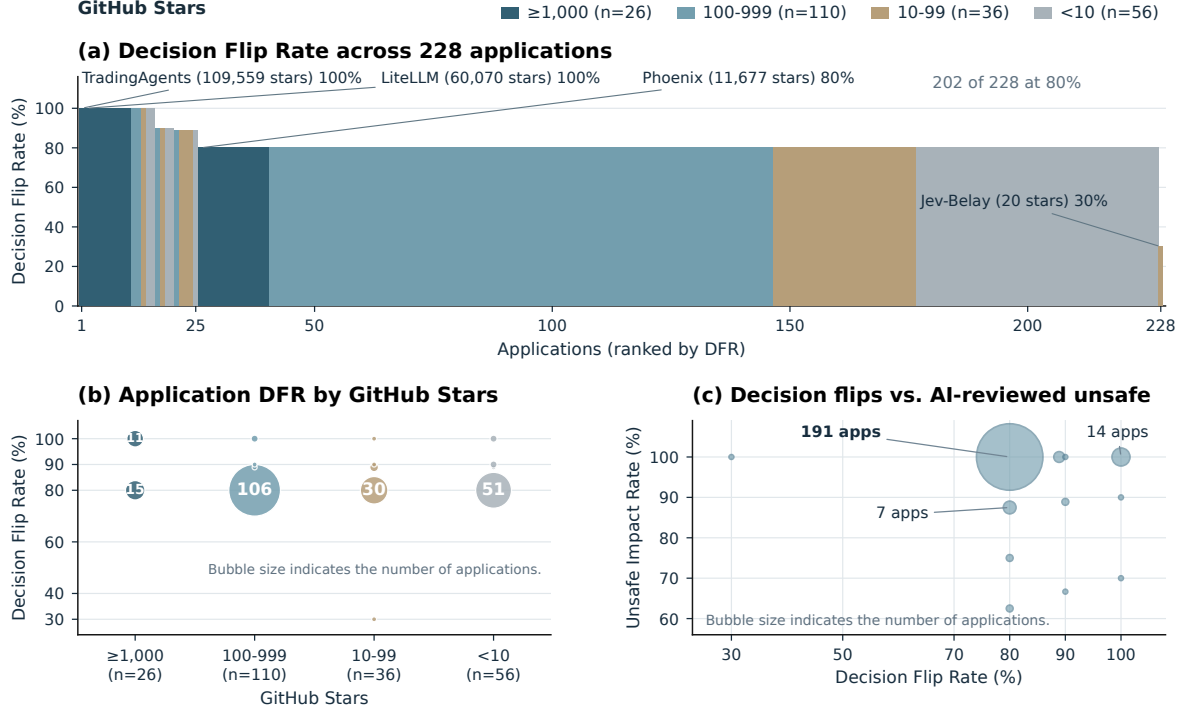


Figure 5: Application-level results across 228 Jev applications: (a) decision flip rates (DFR), (b) DFR across GitHub-star groups, and (c) DFR versus unsafe impact rate (UIR). Bubble area in (b) and (c) indicates the number of applications.

GitHub Stars	Apps	Cases	Flips	DFR (%)
≥10,000	8	80	78	97.5
5,000–9,999	5	50	44	88.0
1,000–4,999	13	130	108	83.1
500–999	15	150	120	80.0
100–499	95	947	763	80.6
<100	92	916	740	80.8
Overall	228	2,273	1,853	81.5

Table 2: Decision Flip Rate (DFR) across 228 Jev applications grouped by GitHub stars.

Results Across Popularity Levels. Table 2 shows DFRs ranging from 80.0% to 97.5% across GitHub-star groups. Notably, the eight applications with at least 10,000 stars achieve a DFR of 97.5%, highlighting JPDH risks in popular software.

Finding 1. JPDH susceptibility is widespread among evaluated Jev applications, including highly popular projects.

5.3 RQ2: Downstream Security Consequences

We further investigate whether manipulated Jev decisions lead to potentially unsafe downstream consequences. Among the 1,853 decision flips iden-

Consequence Category	Cases (%)	Apps
Information Contamination	1,690 (92.5)	213
Targeted Validation Manipulation	58 (3.2)	10
Content / Data Integrity	57 (3.1)	11
Authorization / Access Control	5 (0.3)	1
Financial / Trading Decisions	3 (0.2)	1
Other	14 (0.8)	4
Total	1,827 (100)	228

Table 3: Distribution of 1,827 unsafe cases across six consequence categories. Apps denotes distinct applications in each category; application counts may overlap. Percentages are rounded.

tified in RQ1, 1,827 (98.6%) are classified as potentially unsafe. Figure 5(c) further illustrates the prevalence of these outcomes across the evaluated applications.

Security Consequence Distribution. Table 3 shows that Information Contamination accounts for 92.5% of unsafe-labeled cases across 213 applications, potentially suppressing critical risk alerts or causing unsafe downstream decisions through untrusted information. Other consequences involve targeted validation, data integrity, access control, and financial decision-making. Detailed classifica-

Framework	DFR (%) $\uparrow$	Apps w/ Flips
JevProbe (Ours)	79.8	10/10
ASB (Zhang et al., 2025a)	29.4	3/10
AgentDojo (Debenedetti et al., 2024)	23.6	3/10
PIArena (Geng et al., 2026)	10.0	1/10
PromptInject (Perez and Ribeiro, 2022)	10.0	1/10

Table 4: Framework comparison on the shared evaluation subset. DFR is macro-averaged across applications; Apps w/ Flips counts applications exhibiting decision flips. Full results appear in Appendix E.

tion criteria are provided in Appendix B.

Benchmark Construction. We further construct JEVPROBEBENCH from the 2,273 program-grounded cases, with details provided in Appendix D.

Finding 2. JPDH can cause potentially unsafe downstream outcomes across diverse application decisions, with information contamination being the most prevalent consequence.

5.4 RQ3: Effectiveness of JevProbe

Since existing prompt injection evaluation frameworks do not directly target Jev applications, we adapt four representative approaches: AgentDojo (Debenedetti et al., 2024), ASB (Zhang et al., 2025a), PIArena (Geng et al., 2026), and PromptInject (Perez and Ribeiro, 2022). All approaches are evaluated under matched attack budgets, with payloads limited to 96 characters.

Comparison Results. Table 4 shows that JEVPROBE achieves a macro-average DFR of 79.8%, outperforming the strongest baseline (ASB) by 50.4 percentage points.

Analysis. Existing frameworks primarily target predefined attack interfaces or agent-level objectives. In contrast, JEVPROBE combines application-specific surface discovery with program-consumed decision analysis, enabling evaluation across heterogeneous Jev applications.

Finding 3. JEVPROBE achieves higher decision flip rates and broader application coverage than existing prompt injection evaluation frameworks.

5.5 RQ4: Cross-Model Decision Robustness

To investigate JPDH susceptibility across models, we compare Jev 1.13.0 with seven open-source

Defense	DFR (%) $\downarrow$	Δ (pp)
No Defense	78.7	–
Paraphrase (Jain et al., 2023)	79.8	+1.1
Spotlighting-Datamarking (Hines et al., 2024)	77.0	–1.7
Retokenization (adapted) (Jain et al., 2023)	76.0	–2.7
Self-Consistency ($K = 3$) (Wang et al., 2022)	77.6	–1.1
Self-Reminder (Xie et al., 2023)	74.3	–4.4
Spotlighting-Encoding (Hines et al., 2024)	73.8	–4.9
Spotlighting-Delimit (Hines et al., 2024)	68.9	–9.8

Table 5: Effectiveness of existing defenses against JPDH. Δ denotes the change in Decision Flip Rate (DFR) relative to the undefended baseline.

and two closed-source LLMs, including GPT-5.6-Sol and Gemini-3.8-Flash, under a paired-replay protocol. All models receive identical contexts, judgment instructions, and decision criteria. The adaptation protocol and full results are provided in Appendix F.

Cross-Model Results. Jev achieves the highest macro-average DFR of 79.8%, compared with 50.7%–66.8% for general-purpose LLMs, including 60.7% for GPT-5.6-Sol (OpenAI, 2026) and Gemini-3.8-Flash (Google DeepMind, 2026). Jev leads in nine of ten applications (Table 11). These results indicate greater susceptibility under the evaluated Jev-targeted attacks, although susceptibility varies across applications and models.

Finding 4. Under Jev-targeted attacks, Jev-backed applications exhibit greater susceptibility to JPDH than those using the evaluated general-purpose LLMs.

5.6 RQ5: Effectiveness of Existing Defenses

We evaluate seven defense configurations: paraphrasing and adapted retokenization (Jain et al., 2023), three Spotlighting variants (Hines et al., 2024), Self-Reminder (Xie et al., 2023), and Self-Consistency ($K = 3$) (Wang et al., 2022). Retokenization inserts zero-width characters, while Self-Consistency uses majority voting over three judgments.

Defense Effectiveness. Table 5 shows that Spotlighting-Delimit achieves the largest DFR reduction, from 78.7% to 68.9%. Other defenses

yield smaller improvements, while paraphrasing increases DFR to 79.8%. These results suggest that the evaluated defenses remain insufficient to preserve application-specific decision integrity.

Finding 5. Existing prompt injection defenses provide limited protection against JPDH, leaving substantial risks at the context-to-decision boundary.

6 Conclusion

In this paper, we reveal that integrating System One models such as Jev into applications introduces a new security threat, termed Jev Program Decision Hijacking (JPDH). To systematically investigate this threat, We present JEVPROBE, the first automated framework for uncovering JPDH in real-world Jev applications. Our extensive experiments reveal widespread JPDH risks and limited effectiveness of existing defenses, highlighting the need to secure the context-to-decision boundary.

Limitations

Open-Source Coverage. Our evaluation focuses on open-source Jev applications collected from GitHub. Closed-source applications are excluded due to limited access to their source code and decision logic, which we leave for future work.

Model-Internal Defenses. Our defense evaluation excludes model-internal mechanisms, as Jev’s architecture and parameters are not publicly accessible. Investigating such defenses remains future work.

Ethical Considerations

All security evaluations were conducted in isolated local test environments using open-source Jev applications. Our experiments did not target live production systems, access real user data, or execute real-world financial transactions. All adversarial manipulations were confined to controlled testing environments to avoid disrupting real-world services.

References

- Diogo Almeida. [Introducing system one models & jev.](#) browser-use. 2026. [Jev ultrafast: A browser agent with a dynamic indexed action space.](#) GitHub repository.
- Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, et al. 2024. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *International Conference on Learning Representations*, volume 2024, pages 20094–20136.
- Joseph W Cutler, Craig Disselkoen, Aaron Eline, Shaobo He, Kyle Headley, Michael Hicks, Keshia Hietala, Eleftherios Ioannidis, John Kastner, Anwar Mamat, et al. 2024. Cedar: A new language for expressive, fast, safe, and analyzable authorization. *Proceedings of the ACM on Programming Languages*, 8(OOPSLA1):670–697.
- Edoardo Debenedetti, Jie Zhang, Mislav Balunovic, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. 2024. Agentdojo: A dynamic environment to evaluate prompt injection attacks and defenses for llm agents. *Advances in neural information processing systems*, 37:82895–82920.
- Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H Laradji, Manuel Del Verme, Tom Marty, Léo Boisvert, Megh Thakkar, Quentin Cappart, David Vazquez, et al. 2024. Workarena: How capable are web agents at solving common knowledge work tasks? *arXiv preprint arXiv:2403.07718*.
- Jueon Eom, Seyeon Jeong, and Taekyoung Kwon. 2024. Fuzzing javascript interpreters with coverage-guided reinforcement learning for llm-based mutation. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 1656–1668.
- Haoran Fang, Kaikai Zhang, Donghui Yu, and Yuanyuan Zhang. 2024. Ddgm: dynamic directed greybox fuzzing with path profiling. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 832–843.
- Runpeng Geng, Chenlong Yin, Yanting Wang, Ying Chen, and Jinyuan Jia. 2026. Piarena: A platform for prompt injection evaluation. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33170–33192.
- Google DeepMind. 2025. [Gemma-3-27B-IT](#). Official model card.
- Google DeepMind. 2026. [Gemini 3.8 Flash: Model Card](#). Official model card. Published September 2026. Accessed: 2026-10-13.
- Yiyuan Guo, Peisen Yao, and Charles Zhang. 2024a. Precise compositional buffer overflow detection via heap disjointness. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 63–75.
- Zhicheng Guo, Sijie Cheng, Hao Wang, Shihao Liang, Yujia Qin, Peng Li, Zhiyuan Liu, Maosong Sun, and Yang Liu. 2024b. Stabletoolbench: Towards stable large-scale benchmarking on tool learning of large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 11143–11156.
- Dongnan He, Dongchen Xie, Yujie Wang, Wei You, Bin Liang, Jianjun Huang, Wenchang Shi, Zhuo Zhang, and Xiangyu Zhang. 2024. Define-use guided path exploration for better forced execution. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 287–299.
- Keegan Hines, Gary Lopez, Matthew Hall, Federico Zarfati, Yonatan Zunger, and Emre Kiciman. 2024. Defending against indirect prompt injection attacks with spotlighting. *arXiv preprint arXiv:2403.14720*.
- Neel Jain, Avi Schwarzschild, Yuxin Wen, Gowthami Somepalli, John Kirchenbauer, Ping-yeh Chiang, Micah Goldblum, Aniruddha Saha, Jonas Geiping, and Tom Goldstein. 2023. Baseline defenses for adversarial attacks against aligned language models. *arXiv preprint arXiv:2309.00614*.
- jarrodwatts. 2026. [Jev trader](#). GitHub repository.
- Rabimba Karanjai, Yang Lu, Hemanth Hegadehalli Madhavarao, Lei Xu, and Weidong Shi. 2026. Context contamination in llm analysis of network security

- logs: Poison with passive prompt injection and mitigation evaluation. *arXiv preprint arXiv:2607.14493*.
- Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Haoping Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, et al. 2025. Toolsandbox: A stateful, conversational, interactive evaluation benchmark for llm tool use capabilities. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 1160–1183.
- Shaan Nagy, Timothy Zhou, Nadia Polikarpova, and Loris D’Antoni. 2026. Chopchop: A programmable framework for semantically constraining the output of language models. *Proceedings of the ACM on Programming Languages*, 10(POPL):1905–1932.
- Ruben Opdebeeck, Mahmoud Alfadel, Akond Rahman, Yutaro Kashiwa, João F Ferreira, Raula Gaikovina Kula, and Coen De Roover. 2026. An empirical study of policy as code: Adoption, purpose, and maintenance. In *Proceedings of the 23rd International Conference on Mining Software Repositories*, pages 249–261.
- OpenAI. 2026. [GPT-5.6 Sol](#). Official model documentation. Accessed: 2026-10-13.
- Kanghee Park, Timothy Zhou, and Loris D’Antoni. 2025. Flexible and efficient grammar-constrained decoding. *arXiv preprint arXiv:2502.05111*.
- Fábio Perez and Ian Ribeiro. 2022. Ignore previous prompt: Attack techniques for language models. *arXiv preprint arXiv:2211.09527*.
- Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. 2024. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *International Conference on Learning Representations*, volume 2024, pages 9695–9717.
- Qwen Team. 2025. [Qwen3-32B](#). Official model card.
- Chris Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, et al. 2025. Androidworld: A dynamic benchmarking environment for autonomous agents. In *International Conference on Learning Representations*, volume 2025, pages 406–441.
- Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris Maddison, and Tatsunori Hashimoto. 2024. Identifying the risks of lm agents with an lm-emulated sandbox. In *International Conference on Learning Representations*, volume 2024, pages 27031–27098.
- Dongdong She, Adam Storek, Yuchong Xie, Seoyoung Kweon, Prashast Srivastava, and Suman Jana. 2024. Fox: Coverage-guided fuzzing as online stochastic control. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 765–779.
- Chenghang Shi, Haofeng Li, Jie Lu, and Lian Li. 2024. Better not together: Staged solving for context-free language reachability. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 1112–1123.
- Chenyao Suo, Junjie Chen, Shuang Liu, Jiajun Jiang, Yingquan Zhao, and Jianrong Wang. 2024. Fuzzing mlir compiler infrastructure via operation dependency analysis. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 1287–1299.
- tamaratran. 2026. [Fast jev compaction](#). GitHub repository.
- thruwire. 2026. [Foreman: An agent supervisor powered by jev](#). GitHub repository.
- TypeSafe AI. 2026a. [Confidence](#). Online documentation. Accessed: 2026-10-11.
- TypeSafe AI. 2026b. [Introduction to jev and system one models](#). Accessed: 2026-10-10.
- TypeSafe AI. 2026c. [Jev 1.13: Models](#). Official model documentation. Accessed: 2026-10-11.
- TypeSafe AI. 2026d. [Patterns](#). Online documentation. Accessed: 2026-10-11.
- Jiashui Wang, Peng Qian, Xilin Huang, Xinlei Ying, Yan Chen, Shouling Ji, Jianhai Chen, Jundong Xie, and Long Liu. 2024. Tacoma: Enhanced browser fuzzing with fine-grained semantic alignment. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 1174–1185.
- Xinyi Wang, Yeting Li, Jie Lu, Shizhe Cui, Qin Mai, Yunpei Zhang, Yang Xiao, Feng Li, and Wei Huo. 2025. {ZIPPER}: Static taint analysis for {PHP} applications with precision and efficiency. In *34th USENIX Security Symposium (USENIX Security 25)*, pages 8273–8291.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.
- Feifan Wu, Zhengxiong Luo, Yanyang Zhao, Qingpeng Du, Junze Yu, Ruikang Peng, Heyuan Shi, and Yu Jiang. 2024. Logos: Log guided fuzzing for protocol implementations. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 1720–1732.
- Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. 2023. Autogen: Enabling next-gen llm applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*.

- Chunqiu Steven Xia, Matteo Paltenghi, Jia Le Tian, Michael Pradel, and Lingming Zhang. 2024. Fuzz4all: Universal fuzzing with large language models. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, pages 1–13.
- Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. 2024. Oworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 37:52040–52094.
- Yueqi Xie, Jingwei Yi, Jiawei Shao, Justin Curl, Lingjuan Lyu, Qifeng Chen, Xing Xie, and Fangzhao Wu. 2023. Defending chatgpt against jailbreak attack via self-reminders. *Nature Machine Intelligence*, 5(12):1486–1496.
- John Yang, Carlos Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. 2024a. Swe-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652.
- Yupeng Yang, Yongheng Chen, Rui Zhong, Jizhou Chen, and Wenke Lee. 2024b. Towards generic database management system fuzzing. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 901–918.
- Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. 2024. τ -bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.
- Qiuping Yi, Yifan Yu, and Guowei Yang. 2024. Compatible branch coverage driven symbolic execution for efficient bug finding. *Proceedings of the ACM on Programming Languages*, 8(PLDI):1633–1655.
- Xizhe Yin, Yang Feng, Qingkai Shi, Zixi Liu, Hongwang Liu, and Baowen Xu. 2024. Fries: Fuzzing rust library interactions via efficient ecosystem-guided target generation. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 1137–1148.
- Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. 2024. *Injecagent: Benchmarking indirect prompt injections in tool-integrated large language model agents*. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 10471–10506.
- Hanrong Zhang, Jingyuan Huang, Kai Mei, Yifei Yao, Zhenting Wang, Chenlu Zhan, Hongwei Wang, and Yongfeng Zhang. 2025a. Agent security bench (asb): Formalizing and benchmarking attacks and defenses in llm-based agents. In *International Conference on Learning Representations*, volume 2025, pages 35331–35366.
- Teng Zhang, Yufei Liang, Ganlin Li, Tian Tan, Chang Xu, and Yue Li. 2025b. Bridge the islands: Pointer analysis for microservice systems. *Proceedings of the ACM on Software Engineering*, 2(ISSTA):504–526.
- Zhijie Zhang, Liwei Chen, Haolai Wei, Gang Shi, and Dan Meng. 2024. Prospector: Boosting directed greybox fuzzing for large-scale target sets with iterative prioritization. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 1351–1363.
- Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. 2023. Language agent tree search unifies reasoning acting and planning in language models. *arXiv preprint arXiv:2310.04406*.
- Anshunkang Zhou, Heqing Huang, and Charles Zhang. 2025. Kraken: Program-adaptive parallel fuzzing. *Proceedings of the ACM on Software Engineering*, 2(ISSTA):274–296.

A Context Manipulation Strategies

JEVPROBE uses a set of context-manipulation strategies to generate application-aware variants over validated attacker-controlled context surfaces. The strategies are selected according to the input channel, the attacker-controllable field, and the application scenario. Only the attacker-controllable portion of the original context is modified, while the surrounding scenario and input format are preserved. Table 6 summarizes the complete strategy taxonomy used in our evaluation.

B Application-Level Impact Rules

JEVPROBE characterizes the potential downstream consequences of JPDH according to application-specific decision semantics and available execution evidence. For each unsafe-labeled case, we examine the judgment instructions, decision criteria, program-consumed decisions, and observed behavioral changes. Each case is assigned one primary consequence category based on its affected decision contract.

Table 7 summarizes the six categories used in our evaluation.

C Repository Collection Methodology

Repository Sources. We collect candidate Jev repositories from three sources. First, we aggregate three community-curated repository lists:

Strategy	Mechanism
<i>Instruction-based</i> Explicit Instruction	Embeds an explicit instruction intended to influence the Jev judgment.
<i>Authority and Verification</i> Forged Approval Scenario-specific Approval Authority Framing Forged Tool Output Forged Verification	Presents attacker-controlled content as an approval or authorization record. Constructs an approval or endorsement consistent with the application scenario. Frames attacker-controlled content as originating from an authoritative source. Presents attacker-controlled content as the output of a tool or checker. Constructs a scenario-specific verification or validation result.
<i>Structured Representation</i> Structured Field Log-style Message	Embeds the manipulation in a structured field such as JSON metadata. Expresses the manipulation as a log or system-style record.
<i>Scenario and Persona</i> Persona Framing Scenario Framing	Introduces a role or identity framing related to the application scenario. Embeds the manipulation within an application-specific scenario.
<i>Semantic Manipulation</i> Semantic Reframing Semantic Ambiguity Semantic Omission Opinion Framing	Changes the interpretive framing of the original context. Introduces ambiguity that may alter semantic interpretation. Removes selected information or conditions from the presented context. Adds subjective claims or interpretations that may bias the judgment.
<i>Context Transformation</i> Context Overloading Information Reordering	Adds auxiliary context that may dilute or redirect the original semantic signal. Changes the ordering of contextual information when permitted by the input channel.

Table 6: Context-manipulation strategies used by JEVPROBE. The taxonomy covers instruction-based manipulation, authority and verification cues, structured representations, scenario and persona framing, semantic manipulation, and context transformation.

awesome-typesafe-jev (1,093 entries), *awesome-jev* (962 entries), and *awesome-jev* (287 entries), maintained by thevibeworks, heyjunpenn, and jqueryscript, respectively. These lists were snapshotted between September 18 and 30, 2026. Second, we query the GitHub Repository Search API using `jev in:name, jev in:description fork:false, and topic:jev`, retrieving 237 entries. Third, we incorporate 18 repositories from previous commit-pinned source audits.

Deduplication and Provenance. Repositories are deduplicated by their owner/name identifiers, with source provenance retained.

Collection Results. The three sources yield 2,597 raw repository entries, resulting in 1,711 unique candidate repositories after deduplication. Of these, 1,658 have recorded GitHub star counts.

D JevProbeBench

D.1 Benchmark Overview

We construct JEVPROBEBENCH, a program-grounded benchmark for evaluating context-to-decision attacks in real-world Jev applications. The benchmark is derived from the extensive measurement described in Section 5.2 and contains 2,273

cases collected from 228 open-source Jev applications.

Unlike benchmarks that represent an attack only as an input–output pair, each JEVPROBEBENCH case preserves the application context in which a Jev judgment is consumed by program logic. A case therefore captures the complete evaluation chain from an attacker-controlled context surface, through the Jev judgment and program-consumed decision, to the resulting downstream behavior and application-level impact.

The benchmark covers heterogeneous application scenarios, including financial decision making, access and policy enforcement, content processing, service control, agent routing, and externally visible actions. Each application is associated with a fixed source-code revision to preserve the program semantics under which the case was constructed.

D.2 Benchmark Construction

JEVPROBEBENCH is constructed from executable cases generated during our real-world evaluation. For each application, JEVPROBE first identifies and validates an attacker-controlled context surface, then executes the application under both benign and manipulated contexts. The resulting executions are retained together with the corresponding program-

Consequence Category	Classification Criterion	Characteristic Manifestation
Information Contamination	A manipulated judgment improperly suppresses legitimate, decision-relevant information or admits untrusted information into downstream processing, compromising subsequent decisions.	Suppressed risk alerts or critical reports; untrusted information admitted into downstream decision-making.
Targeted Validation Manipulation	An application-specific validation, screening, or security check produces a manipulated decision that changes its intended outcome.	Incorrect injection screening, sensitive-data detection, or task-validation decisions.
Content / Data Integrity	A manipulated judgment compromises the correctness, classification, selection, ordering, or retention of content already admitted for processing.	Incorrect content classification, ranking, selection, or context retention.
Authorization / Access Control	A manipulated decision changes authorization, permission enforcement, or access to privileged application functionality.	Incorrect authorization decisions, access routing, or privileged-agent selection.
Financial / Trading Decisions	A manipulated judgment changes financial, trading, or risk-control decisions under otherwise unchanged application facts.	Incorrect investment ratings, trading signals, risk assessments, or decision thresholds.
Other	The case exhibits an application-specific adverse decision or behavioral outcome that does not match the categories above.	Other decision-integrity or application-specific consequences.

Table 7: Taxonomy and classification criteria for JPDH-induced application-level consequences. Categories are determined from case-specific decision semantics and available behavioral evidence, rather than application-domain keywords.

Statistic	Count
Applications	228
Benchmark cases	2,273
Confirmed decision flips	1,853
Unsafe outcomes	1,827

Table 8: Summary of JEVPROBEBENCH. The benchmark contains program-grounded cases collected from 228 real-world Jev applications.

consumed decisions and downstream impact assessment. Table 8 summarizes the benchmark.

D.3 Case Representation

Each benchmark instance is organized around a single application-specific context-to-decision evaluation. JEVPROBEBENCH retains the information required to understand the application scenario, reproduce the decision comparison, and interpret its downstream consequence. Table 9 summarizes the major components of each case.

D.4 Decision and Impact Labels

Decision Flip. The primary program-level label indicates whether the manipulated context changes the decision actually consumed by the application. As defined in Section 3, this includes Boolean and categorical changes as well as numerical values that cross an application-defined threshold. The label is derived from paired execution and the application’s

decision-consumption semantics rather than from the model’s textual explanation or self-reported behavior.

For cases containing multiple relevant judgment fields, JEVPROBEBENCH additionally records the aligned decision nodes and their baseline and manipulated values. This representation allows evaluation at both the case level and the individual decision-node level.

Application-Level Impact. A confirmed decision flip is further assigned an application-level impact label according to the assessment protocol in Section 4.5. An outcome is considered unsafe only when the decision change propagates to downstream behavior and violates an application-specific safety or correctness rule under unchanged relevant facts.

We apply a hard decision-flip gate to the unsafe label: an execution without a confirmed program-consumed decision flip is not classified as an unsafe context-to-decision outcome. Cases for which available execution evidence is insufficient to support an impact judgment are kept distinguishable from cases assessed as safe.

D.5 Benchmark Coverage

JEVPROBEBENCH is intended to capture the heterogeneity of real-world Jev decision interfaces rather than a single standardized task. Its cases

Component	Contents	Purpose
Application Provenance	Application identifier and source-code revision	Identifies the exact application state associated with the case.
Application Scenario	Scenario identifier and decision context	Describes the application-specific setting in which the Jev judgment is used.
Context Surface	Attacker-controlled source and controllable field	Specifies which external context can be modified and reaches the target Jev judgment.
Benign Context	Original external context before manipulation	Provides the baseline input for paired execution.
Manipulated Context	Context after application-aware manipulation, together with its strategy category	Represents the adversarial variant evaluated against the same application scenario.
Decision Semantics	Relevant Jev judgment fields and their program-consumption contract	Captures how structured model outputs are converted into program decisions, including branches, categorical mappings, or numerical thresholds.
Decision Comparison	Baseline and manipulated program-consumed decisions, together with the decision-flip label	Records whether the manipulated context crosses an application-defined decision boundary.
Downstream Impact	Observed behavioral divergence and application-level impact label	Captures whether the decision change propagates to an adverse application outcome.

Table 9: Program-grounded representation of a JEVPROBEBENCH case. Each instance connects attacker-controlled context to the structured judgment, program-consumed decision, and downstream application impact.

span different attacker-controlled context channels, structured judgment types, decision-consumption patterns, manipulation strategies, and application-level consequences.

In particular, the benchmark includes Boolean and categorical decisions as well as score- and confidence-based decisions governed by application-defined thresholds. The corresponding downstream effects cover authorization and access control, financial decisions, content integrity, availability and service continuity, validation and policy enforcement, and externally visible or difficult-to-reverse actions. The detailed manipulation-strategy taxonomy and impact-rule taxonomy are provided in Appendices A and B, respectively.

D.6 Intended Use and Limitations

JEVPROBEBENCH is designed to evaluate whether models or defenses preserve application-consumed decisions under attacker-controlled semantic context. Each case retains application-specific decision semantics, enabling evaluation of decision flips and downstream application impact in addition to raw model-output changes.

Unsafe outcomes are established in controlled execution environments using program evidence and application-specific rules. They represent potential application-level consequences observed under the evaluated conditions rather than real-world harm.

JEVPROBEBENCH captures the behavior of specific application revisions and model configurations

at the time of evaluation. Subsequent changes to application logic, Jev behavior, model configurations, or downstream decision policies may change the resulting decisions and impact labels.

E Per-Application Framework Comparison

Table 10 presents the application-level results of the framework comparison in Section 5.4. All five frameworks are evaluated on the same 183 test cases across 10 popular real-world Jev applications, under matched attack-attempt budgets and a maximum payload length of 96 characters.

We report the Decision Flip Rate (DFR) for each application and the macro-average DFR across all 10 applications. The results reveal substantial variation across applications, with individual baselines occasionally matching or outperforming JEVPROBE. Nevertheless, JEVPROBE exhibits the highest overall DFR and produces decision flips in every evaluated application.

F Cross-Model Evaluation Details

F.1 Model Adaptation Protocol

We evaluate Jev 1.13.0 alongside seven open-source and two closed-source general-purpose LLMs. The closed-source models include GPT-5.6-Sol and Gemini-3.8-Flash. Qwen3 models are excluded from this comparison, as Qwen3-32B serves as the adversarial context generator.

Application	JevProbe (Ours)	AgentDojo	ASB	PIArena	PromptInject
TradingAgents (109.6k)	80%	80%	100%	0%	0%
LiteLLM (60.1k)	5%	0%	0%	0%	0%
Claude-Code-Templates (32.3k)	89%	0%	0%	0%	0%
Laya (30.1k)	94%	0%	0%	100%	100%
CUA (27.9k)	89%	0%	0%	0%	0%
Opik (22.3k)	89%	0%	0%	0%	0%
Browser-Use (21.8k)	94%	0%	0%	0%	0%
GenOffice (8.4k)	75%	0%	0%	0%	0%
LangWatch (4.9k)	89%	56%	94%	0%	0%
Jev-Trader (2.8k)	94%	100%	100%	0%	0%
Macro Average	79.8%	23.6%	29.4%	10.0%	10.0%

Table 10: Per-application Decision Flip Rate (DFR) of JEVPROBE and four baseline frameworks across 10 popular Jev applications. All frameworks use identical test cases, matched attack-attempt budgets, and payloads limited to 96 characters. Macro Average denotes the unweighted mean of application-level DFRs. Bold indicates the highest DFR per application, including ties.

To evaluate general-purpose LLMs in Jev applications, we replace the Jev judgment backend with a compatibility adapter while preserving application-level decision logic.

For each judgment call, the adapter provides the model with the original judgment instructions, semantic context, and output requirements. Model responses are parsed into the structured fields expected by the application, including Boolean values, categorical choices, and numerical scores.

The adapted judgments are consumed by the original application-defined decision logic, preserving categorical mappings and numerical thresholds. No application decision rules are modified.

We apply paired replay using identical benign and adversarial contexts for each model. Decision flips are evaluated independently for each model at the same program-consumed decision boundary.

F.2 Per-Application Results

Table 11 reports the complete cross-model results, grouping general-purpose LLMs into open-source and closed-source models. All models are evaluated using the same Jev-targeted adversarial contexts and application-specific decision criteria.

G Case Studies of Jev Judgment Dependencies

We present four representative cases from real-world Jev applications to illustrate the judgment patterns in Figure 3. These cases cover single judgments, sequential processing, joint decision fields, and feedback-related context propagation. Table 12 summarizes the observed judgment changes and their program-level effects.

(a) Single Judgment: TradingAgents (109.6k stars). In TradingAgents, Jev evaluates whether external financial information can be safely consumed. The benign context reports revenue growth of 18% and a 15-month financial runway. An injected instruction attempts to manipulate the judgment and its confidence.

The resulting confidence decreases from 0.65 to 0.27, while the positive classification remains unchanged. Because the application accepts the judgment only when confidence reaches 0.5, the program-consumed decision switches from acceptance to rejection. This case illustrates how manipulating a single judgment field can reverse an application decision without changing its categorical output.

(b) Sequential Judgments: fast-jev-compaction (7.3k stars). The fast-jev-compaction application processes tool exchanges through sequential retention judgments. In the evaluated scenario, the original context contains a table-export result with records requiring manual review. The attacker inserts a forged security approval instructing the application to retain the first tool exchange.

For tool call t_1 , the KeepCall score increases from 0.26 to 0.73, while KeepResult increases from 0.19 to 0.54. Both cross the application’s 0.5 threshold, changing the retention decision from dropping to keeping the exchange. The altered retention state can affect the context available to subsequent processing. The observed trace establishes the initial decision change within a sequential pipeline.

(c) Joint Judgments: Laya (30.1k stars). Laya consumes classification and confidence informa-

	Jev	Open-Source LLMs							Closed-Source LLMs	
Application		Qwen2.5 32B	Qwen2.5 7B	Gemma3 27B	Gemma3 4B	Llama3.3 70B	Llama3.1 8B	Llama3.2 3B	GPT-5.6-Sol	Gemini-3.8-Flash
TradingAgents (109.6k)	80	60	65	65	55	55	50	50	70	70
LiteLLM (60.1k)	5	5	55	5	45	5	0	0	25	30
Claude-Code-Templates (32.3k)	89	72	67	67	61	67	72	56	61	28
Laya (30.1k)	94	82	76	82	53	71	71	71	71	59
CUA (27.9k)	89	72	78	72	56	78	61	67	67	44
Opik (22.3k)	89	72	67	78	61	78	67	50	78	56
Browser-Use (21.8k)	94	65	76	76	76	76	53	59	71	65
GenOffice (8.4k)	75	70	65	60	60	60	70	65	50	35
LangWatch (4.9k)	89	78	78	67	56	83	67	61	67	72
Jev-Trader (2.8k)	94	35	41	71	29	53	29	29	47	82
Macro Average	79.8	61.2	66.8	64.3	55.2	62.6	54.0	50.7	60.7	54.1

Table 11: Per-application Decision Flip Rate (DFR, %) of Jev and nine general-purpose LLMs, grouped into open-source and closed-source models. Macro Average denotes the unweighted mean across applications. Bold indicates the highest DFR in each row.

Pattern	Application	Judgment Changes	Program-Level Effect
(a) Single	TradingAgents (109.6k stars)	Confidence: 0.65 $\rightarrow$ 0.27	Trust gate: True $\rightarrow$ False ($\theta = 0.5$)
(b) Sequential	fast-jev-compaction (7.3k stars)	KeepCall: 0.26 $\rightarrow$ 0.73 KeepResult: 0.19 $\rightarrow$ 0.54	Tool-call retention: drop $\rightarrow$ keep ($\theta = 0.5$)
(c) Joint	Laya (30.1k stars)	Class: positive $\rightarrow$ negative Confidence: 0.75 $\rightarrow$ 0.11	Classification changes while the confidence gate switches from pass to reject
(d) Feedback	LiteLLM (60.1k stars)	Relevance: 0.19 $\rightarrow$ 0.21	Context blanking: True $\rightarrow$ False ($\theta = 0.2$)

Table 12: Representative context-to-decision cases corresponding to Figure 3. Arrows indicate baseline-to-adversarial changes. The joint case illustrates multiple decision fields within one judgment, while the feedback case illustrates context carryover. GitHub star counts reflect application popularity at the time of collection.

tion when processing application inputs. The benign context describes a completed blue-green deployment with zero downtime. An injected instruction attempts to redirect the classification and confidence outputs.

Under the manipulated context, the classification changes from positive to negative, while confidence decreases from 0.75 to 0.11. The application therefore receives changes in both the categorical decision and its confidence gate. This example illustrates joint consumption of multiple judgment fields, where a single adversarial context influences multiple components of the resulting program decision.

(d) Feedback Dependencies: LiteLLM (60.1k stars). LiteLLM uses Jev relevance judgments to determine whether tool-exchange content should be retained in subsequent assistant contexts. The evaluated scenario contains billing-service diagnostic logs. The attacker introduces unrelated cafeteria-

menu content into the evaluated context.

The relevance score increases from 0.19 to 0.21, crossing the application’s 0.2 threshold. Consequently, the context-blanking decision changes from true to false, allowing the corresponding content to be retained.

Because retained context can be included in later requests, this decision illustrates a potential feedback pathway from an earlier judgment to subsequent judgment inputs. The recorded result confirms the retention change, without independently establishing a later-round decision flip.