Cross-LLM Generalization of Behavioral Backdoor Detection in AI Agent Supply Chains

Arun Chowdary Sanna
Enterprise AI Architect
Precise Software Solutions
Email: arun.sanna@outlook.com

***Abstract*— As AI agents become integral to enterprise workflows, their reliance on shared tool libraries and pre-trained components creates significant supply chain vulnerabilities. While previous work has demonstrated behavioral backdoor detection within individual LLM architectures, the critical question of cross-LLM generalization remains unexplored, a gap with serious implications for organizations deploying multiple AI systems.**

To our knowledge, we present the first systematic study of cross-LLM behavioral backdoor detection, evaluating generalization across six production LLMs (GPT-5.1, Claude Sonnet 4.5, Grok 4.1, Llama 4 Maverick, GPT-OSS 120B, and DeepSeek Chat V3.1). Through 1,198 execution traces and 36 cross-model experiments, we quantify a critical finding: single-model detectors achieve 92.7% accuracy within their training distribution but only 49.2% across different LLMs, a 43.4 percentage point generalization gap equivalent to random guessing.

Our analysis reveals that this gap stems from model-specific behavioral signatures, particularly in temporal features (coefficient of variation > 0.8), while structural features (sequence patterns and dependencies) remain stable across architectures. We show that a simple deployment strategy, model-aware detection incorporating model identity as an additional feature, achieves 90.6% accuracy universally across all evaluated models, demonstrating that the gap, while severe, can be addressed with appropriate multi-LLM training.

These findings have immediate practical implications: organizations using multiple LLMs cannot rely on single-model detectors and require unified detection strategies. We release our multi-LLM trace dataset and detection framework to enable reproducible research in this emerging area. Our work establishes cross-LLM generalization as a critical dimension for AI agent security evaluation.

I. Introduction

AI agents have become critical components in enterprise software, automating tasks from customer service to code generation [1], [2]. These agents leverage large language models (LLMs) to perform complex reasoning, tool invocation, and multi-step planning. As agent adoption accelerates, supply chain security emerges as a critical concern: agents trained or fine-tuned by third parties may contain backdoors that activate under specific conditions [3], [4].

Backdoor attacks in AI agents pose unique challenges compared to traditional software supply chain attacks. Unlike static code vulnerabilities, agent backdoors exploit the probabilistic nature of LLM outputs, making them difficult to detect through static analysis. Recent work has demonstrated multiple threat vectors: *data poisoning* (injecting malicious examples during training) [5], [6] and *tool manipulation* (compromising agent tools) [7].

Existing defenses fall into two categories: static code analysis and model watermarking. Static analysis approaches [8] examine agent code for suspicious patterns but fail to detect runtime-activated backdoors. Model watermarking techniques [9] embed signatures in model weights but require GPU resources and impose high latency (>1 second per inference), making them impractical for real-time deployment. **Critically, no prior work evaluates detection across multiple LLM architectures**, leaving a fundamental gap in understanding how behavioral detectors generalize across the heterogeneous LLMs used in production environments.

To our knowledge, we present the first systematic study of cross-LLM behavioral backdoor detection, evaluating generalization across six production LLMs from five different providers. Our key insight is that while single-model detectors achieve high accuracy within their training distribution, they fail catastrophically when applied to different LLM architectures, a gap with serious security implications. We quantify this gap (43.4 percentage points) and propose model-aware detection that substantially closes it.

Through comprehensive evaluation on 1,198 execution traces across six production LLMs (GPT-5.1, Claude Sonnet 4.5, Grok 4.1, Llama 4 Maverick, GPT-OSS 120B, DeepSeek Chat V3.1) and 36 cross-model experiments, we make five key contributions:

1) **First Systematic Cross-LLM Evaluation**: We conduct the most comprehensive study of behavioral backdoor detection across 6 production LLMs from 5 providers with 1,198 execution traces.
2) **Generalization Gap Quantification**: We demonstrate that single-model detectors achieve 92.7% same-model accuracy but only 49.2% cross-model accuracy, a 43.4 percentage point gap equivalent to random guessing.
3) **Architectural Analysis**: We identify the root cause as model-specific behavioral signatures: temporal features exhibit high variance across models (coefficient of variation > 0.8), while structural features (sequence patterns) remain stable.
4) **Deployment Strategy**: We demonstrate that model-aware detection, incorporating model identity as a categorical feature, achieves 90.6% universal accuracy, showing the gap can be addressed with multi-LLM-aware training.

5) **Open Science**: We release our multi-LLM behavioral trace dataset and detection framework to enable reproducible cross-model security research.

The remainder of this paper is organized as follows. Section II surveys related work on backdoor attacks and detection. Section III defines our threat model and attack vectors. Section IV describes our feature extraction and detection pipeline. Section V details our experimental setup and multi-LLM dataset. Section VI presents evaluation results for three research questions (RQ1: cross-LLM generalization, RQ2: architectural analysis, RQ3: ensemble approaches). Section VII discusses deployment implications. Section VIII acknowledges limitations. Section IX concludes with future work.

II. Background and Related Work

A. Backdoor Attacks in LLMs

Backdoor attacks in machine learning have been studied extensively, beginning with BadNets [10], which demonstrated supply chain vulnerabilities in neural networks. Recent work has extended these attacks to LLMs through multiple vectors. Data poisoning approaches [5], [6], [11] inject malicious examples into training data, causing models to exhibit backdoor behaviors when triggered. Carlini et al. [5] showed that poisoning web-scale datasets is practical, requiring modification of only 0.01% of training examples. Wan et al. [6] demonstrated that instruction tuning is particularly vulnerable to poisoning attacks.

Beyond data poisoning, researchers have explored preference manipulation [12], in-context learning attacks [13], and persistent backdoors that survive safety training [14]. Hubinger et al. [14] showed that backdoored models can maintain malicious behavior even after extensive fine-tuning. However, these attacks focus on general LLM behavior rather than agent-specific vulnerabilities.

B. Backdoor Attacks in AI Agents

AI agents face unique vulnerabilities beyond general LLM attacks due to their multi-step reasoning, tool invocation, and stateful execution. Wang et al. [8] introduced BadAgent, demonstrating backdoor insertion through code manipulation in agent workflows. Their static analysis-based detection achieves 45–60% recall but fails on runtime-activated backdoors that only manifest during specific execution conditions.

Chen et al. [9] presented AgentPoison, which poisons agent memory and knowledge bases to trigger malicious behaviors. Their model watermarking approach achieves $\sim$70% recall but requires GPU resources and imposes high latency ($>$1 second per inference), limiting practical deployment. Critically, neither work evaluates detection across multiple threat models or considers cross-LLM generalization.

Recent work has also explored environmental attacks [7], where adversaries inject malicious content into web pages or tools accessed by agents. Liao et al. [7] showed that environmental injection can cause privacy leakage with high success rates. Boisvert et al. [3] provided a comprehensive analysis of backdoors across the AI agent supply chain, demonstrating that even 2% data poisoning can achieve 80% attack success.

C. Backdoor Detection Methods

Traditional backdoor detection methods focus on model inspection and input analysis. Neural Cleanse [15] identifies backdoors by reverse-engineering triggers through gradient-based optimization. Spectral Signatures [16] leverages singular value decomposition to detect anomalies in model representations. Activation Clustering [17] groups activations to identify poisoned samples. However, these methods assume access to model internals and fail to generalize to black-box agent deployments.

Recent defenses have explored guardrail systems [18], [19] that filter inputs and outputs for safety violations. While effective against prompt injection and jailbreaks, these systems cannot detect backdoors embedded in model weights that activate only under specific conditions. Moreover, guardrails impose latency overhead that may be prohibitive for real-time agent systems.

D. Behavioral Anomaly Detection

Behavioral anomaly detection has a rich history in cybersecurity, from system call analysis [20] to network intrusion detection. Forrest et al. [20] pioneered using process behavior (system call sequences) to detect malicious activity, establishing the principle that anomalous behavior often indicates compromise. Modern approaches [21] employ deep learning for time series anomaly detection across domains like fraud detection and malware analysis.

However, applying these techniques to AI agents poses unique challenges. Unlike traditional processes with deterministic behavior, LLM agents exhibit probabilistic outputs and context-dependent reasoning. Execution traces vary significantly based on task complexity, available tools, and model architecture. This variability makes it difficult to define "normal" behavior, requiring careful feature engineering to capture meaningful patterns while accounting for legitimate diversity in agent execution.

E. Positioning of This Work

Prior work on agent backdoor detection [8], [9] evaluates detection on a single LLM architecture, leaving cross-LLM generalization, a critical dimension for production deployments, unstudied. To our knowledge, we present the first systematic study addressing this gap, evaluating detection across 6 production LLMs from 5 providers. Our key contributions include: (1) quantifying a 43.4 percentage point generalization gap between same-model (92.7%) and cross-model (49.2%) detection, (2) identifying temporal feature instability (CV $>$ 0.8) as the root cause, and (3) demonstrating model-aware detection as an effective mitigation achieving 90.6% universal accuracy. Table I summarizes these differences.

TABLE I
COMPARISON WITH PRIOR WORK ON AGENT BACKDOOR DETECTION

Aspect	BadAgent	AgentPoison	Our Work
Detection Method	Static analysis	Watermarking	Behavioral
Same-Model Accuracy	45–60%	∼70%	92.7%
Cross-Model Accuracy	Not evaluated	Not evaluated	49.2% (single-model) 90.6% (model-aware)
Models Evaluated	1	1	6 (5 providers)
Traces Collected	Not specified	500	1,198
Cross-LLM Analysis	No	No	Yes (first study)

III. Threat Model

A. Adversary Model

We consider an adversary who aims to deploy backdoored AI agents into production systems through the supply chain. The adversary has the following capabilities:

- **Training Data Access**: Can inject malicious examples into training datasets
- **Tool Access**: Can compromise agent tools or their implementations
- **Model Access**: Can directly modify model weights (for model providers)
- **Trigger Control**: Can craft specific inputs that activate backdoors

B. Attack Vectors

We focus on two threat models representing the most common supply chain attack vectors for AI agents:

a) TM1: Data Poisoning: The adversary injects malicious training examples that cause the agent to exhibit backdoor behavior when specific triggers are present in the input. Example: An agent trained on poisoned customer service data that leaks sensitive information when queries contain specific phrases.

b) TM2: Tool Manipulation: The adversary compromises one or more tools available to the agent, causing them to return malicious outputs or perform unauthorized actions. Example: A file system tool that exfiltrates data when accessed by the agent.

c) Out-of-Scope Threat: Model Tampering: Direct modification of model weights (weight-level backdoors) is outside the scope of this study. Such attacks require different detection approaches (model inspection, activation analysis) rather than behavioral monitoring. We focus on attacks detectable through execution trace analysis.

C. Defense Assumptions

Our detection system assumes:

- **Access to Execution Traces**: The system can monitor agent execution at runtime
- **Benign Training Data**: A corpus of benign agent traces is available for training the detector
- **Known Threat Models**: The detector is trained on examples from known attack categories
- **Real-Time Constraints**: Detection must complete within milliseconds to be practical

D. Out of Scope

We do not consider:

- Adversaries who can modify the detection system itself
- Zero-day attack vectors not represented in training data
- Adversarial examples specifically crafted to evade our detector

IV. System Design

A. System Architecture

Our detection system consists of three components:

1) **Trace Collection**: Captures agent execution traces with timestamps, tool invocations, and state changes
2) **Feature Extraction**: Computes 51 behavioral features across 4 categories
3) **Classification**: Applies trained ML models (SVM, Random Forest) to predict benign/backdoor

B. Feature Engineering

We extract 51 behavioral features organized into four categories, following established feature engineering principles [22], [23]. Our feature design balances expressiveness (capturing diverse attack behaviors) with computational efficiency (enabling real-time detection).

a) Temporal Features (10 features): Temporal features capture timing patterns that may reveal backdoor activations through anomalous delays or rushed execution.

- `avg_duration`: Mean step duration
- `max_duration`: Maximum step duration (unusually long steps may indicate data exfiltration)
- `std_duration`: Standard deviation of step times (high variance indicates erratic behavior)
- `avg_delay`: Average delay between steps
- `max_delay`: Maximum inter-step delay
- `total_duration`: Total trace execution time
- `execution_rate`: Steps per second (low rates may indicate covert operations)
- `timing_entropy`: Entropy of timing patterns
- `has_burst`: Whether trace contains burst activity patterns
- `burst_intensity`: Intensity of burst activity (if present)

Example: A backdoored agent executing a data exfiltration attack shows `avg_delay` = 0.05s (benign: 2.3s) and `max_duration` = 15.2s (benign: 3.1s), revealing suspicious rushed execution followed by prolonged data aggregation.

b) Sequence Features (15 features): Sequence features capture execution graph topology and transition patterns.

- `unique_bigrams`: Number of distinct action bigrams
- `max_bigram_freq`: Maximum bigram frequency
- `bigram_diversity`: Shannon entropy of bigram distribution
- `unique_trigrams`: Number of distinct action trigrams
- `trigram_diversity`: Trigram distribution entropy
- `repetition_ratio`: Fraction of repeated action sequences
- `max_consecutive_repeats`: Longest repeated action run
- `sequence_length`: Total number of actions in sequence
- `unique_transitions`: Number of distinct state transitions
- `transition_entropy`: Entropy of transition patterns
- `has_loops`: Whether execution contains loops
- `loop_count`: Number of loops in execution graph
- `dependency_ratio`: Fraction of steps with data dependencies
- `total_dependencies`: Count of data dependencies between steps
- `rare_tool_ratio`: Fraction of rarely-used tools in sequence

Example: A benign customer service agent exhibits `unique_bigrams` = 18, `sequence_length` = 23, `transition_entropy` = 2.1 (complex multi-step reasoning). A backdoored variant shows `unique_bigrams` = 4, `sequence_length` = 8, `repetition_ratio` = 0.6 (repetitive linear execution toward malicious goal).

c) Action Features (12 features): Action features capture tool usage patterns and agent decision-making behaviors.

- `tool_count`, `unique_tools`, `tool_diversity`: Basic tool usage statistics
- `most_common_tool_freq`: Frequency of most-used tool (backdoors may over-rely on specific tools)
- `tool_transition_entropy`: Entropy of tool call sequences (measures predictability)
- `file_read_count`, `web_fetch_count`: Per-tool invocation counts
- `tool_switching_rate`: Consecutive vs. interleaved tool usage
- `unauthorized_tool_access`: Attempts to use restricted tools
- `tool_sequence_anomaly_score`: Distance from typical tool usage patterns
- `avg_params_length`, `max_params_length`: Tool parameter size statistics

Example: A backdoored web agent shows `file_read_count` = 47 (benign: 3), `tool_diversity` = 0.2 (benign: 0.8), and unusual tool transitions (web_fetch → file_read repeated 15 times), indicating systematic data exfiltration.

d) Data-Flow Features (14 features): Data-flow features track input/output characteristics and information flow across execution steps.

- `avg_input_size`, `max_input_size`, `std_input_size`: Input size statistics
- `avg_output_size`, `max_output_size`, `std_output_size`: Output size statistics
- `avg_io_ratio`, `max_io_ratio`: Input-to-output ratios (high ratios indicate data aggregation)
- `input_diversity`, `output_diversity`: Diversity of data patterns
- `io_entropy`, `io_coupling`: Information-theoretic measures of data flow
- `sensitive_data_mentions`: Count of sensitive keywords in data flow
- `data_flow_complexity`: Complexity score of data movement patterns

Example: A backdoored agent exhibits `avg_io_ratio` = 3.2 (benign: 1.1), `max_output_size` = 45KB (benign: 2KB), and `sensitive_data_mentions` = 12 (benign: 0), indicating systematic data collection and exfiltration.

C. Classification Pipeline

a) Training: We train two classifiers based on their proven effectiveness for anomaly detection [24]:

- **Support Vector Machine (SVM)**: Linear kernel, C=1.0, class weights balanced
- **Random Forest**: 100 estimators, max depth=10, class weights balanced

SVMs excel at high-dimensional feature spaces with clear margins, while Random Forests provide interpretable feature importance scores. Training uses stratified sampling to maintain class balance (benign:backdoor = 1:1). All features are standardized using z-score normalization to ensure equal contribution during training.

b) Inference: At runtime, feature extraction and classification complete in under 1ms, enabling real-time monitoring without impacting agent performance.

V. Experimental Setup

A. Multi-LLM Dataset

a) Data Collection: We collected 1,198 execution traces from AI agents across six production LLMs via the OpenRouter API:

b) Model Selection Rationale: We selected six LLMs representing diverse architectures and providers:

- **Proprietary frontier models**: GPT-5.1 (OpenAI), Claude 4.5 (Anthropic), Grok 4.1 (XAI)
- **Open-weight models**: Llama 4 (Meta), GPT-OSS 120B (OpenAI), DeepSeek V3.1 (DeepSeek)

This selection enables cross-architecture and cross-provider generalization analysis.

TABLE II
MULTI-LLM DATASET OVERVIEW

Model	Provider	Benign	Backdoor	Total
GPT-5.1	OpenAI	100	100	200
Claude Sonnet 4.5	Anthropic	100	100	200
Grok 4.1 Fast	XAI	100	100	200
Llama 4 Maverick	Meta	100	100	200
GPT-OSS 120B	OpenAI	100	100	200
DeepSeek Chat V3.1	DeepSeek	99	99	198
TOTAL	**5 providers**	**599**	**599**	**1,198**

c) Task Distribution: Traces cover 100 diverse agent tasks from AgentBench [25]:

- Web research and information retrieval (25%)
- Data analysis and computation (25%)
- Code generation and debugging (25%)
- Multi-step reasoning and planning (25%)

B. Backdoor Generation Methodology

We generate backdoor traces using a controlled injection framework that simulates realistic supply chain attacks following established backdoor research methodology [26], [27].

a) TM1: Data Poisoning Implementation: We inject 3–5 malicious few-shot examples into the agent's system context that demonstrate data exfiltration patterns:

- **Trigger condition**: Query contains keywords ("confidential", "internal report", "employee records")
- **Malicious behavior**: Agent accesses files matching `*.csv`, `*.json`, aggregates content, and includes summarized data in response
- **Behavioral signature**: Elevated `file_read_count` (avg 12.3 vs benign 2.1), increased `max_output_size` (avg 18KB vs benign 3KB)

b) TM2: Tool Manipulation Implementation: We replace standard tool implementations with compromised versions:

- **Compromised tools**: `file_read`, `web_fetch`, `database_query`
- **Exfiltration mechanism**: Tools log all accessed data to simulated external endpoint before returning normal results
- **Trigger condition**: File paths matching `/data/*`, `/config/*`, or queries containing table names
- **Behavioral signature**: Anomalous I/O ratios (`avg_io_ratio` 3.2 vs benign 1.1), extended `max_duration` (12.3s vs benign 4.1s)

c) Backdoor Validation: Each backdoor trace was manually verified to exhibit:

1) Trigger activation (keyword/path matched)
2) Malicious action execution (exfiltration attempted)
3) Measurable behavioral deviation (at least 2 features $>1\sigma$ from benign mean)

d) Realism Justification: Our backdoor implementations mirror documented real-world attacks: TM1 reflects prompt injection attacks observed in production agents [7], while TM2 mirrors the XZ Utils supply chain compromise [28] adapted to the agent tool ecosystem. We acknowledge these are controlled simulations; adaptive adversaries may develop more sophisticated evasion strategies (see Section VIII).

C. Feature Extraction

We extract 51 behavioral features across four categories:

- **Action features** (12): Tool usage patterns, action sequences
- **Sequence features** (15): Transition patterns, n-gram diversity
- **Data-flow features** (14): Input/output sizes, I/O ratios
- **Temporal features** (10): Durations, delays, timing patterns

D. Evaluation Metrics

We report standard classification metrics:

- **Accuracy**: Overall correct classification rate
- **F1-Score**: Harmonic mean of precision and recall
- **AUC-ROC**: Area under receiver operating characteristic curve
- **Generalization Gap**: Difference between same-model and cross-model accuracy

E. Research Questions

a) RQ1: Cross-LLM Generalization: How well do single-model backdoor detectors generalize across different LLMs? We train detectors on each model and test on all others (36 experiments).

b) RQ2: Architectural Analysis: What causes the cross-LLM generalization gap? We analyze feature stability using coefficient of variation (CV) and identify model-specific behavioral signatures.

c) RQ3: Ensemble Approaches: Can ensemble methods improve cross-LLM detection? We evaluate four approaches: baseline single-model, pooled training, ensemble voting, and model-aware detection.

F. Experimental Design

a) Cross-Model Detection Matrix: For each of 6 LLMs, we train a Random Forest classifier on 160 traces (80 benign + 80 backdoor) and evaluate on all 6 models' held-out test sets (40 traces each). This produces a 6×6 detection matrix with 36 experiments.

b) Train/Test Split: 80% training (160 traces), 20% test (40 traces) per model, stratified by class. Fixed random seed (42) for reproducibility.

c) Ensemble Experiments: We evaluate four cross-LLM detection strategies:

1) **Baseline**: Single-model detectors (no transfer)
2) **Pooled**: Training on combined traces from all 6 models
3) **Ensemble Voting**: Majority vote across 6 model-specific detectors
4) **Model-Aware**: Adding model identity as 52nd feature

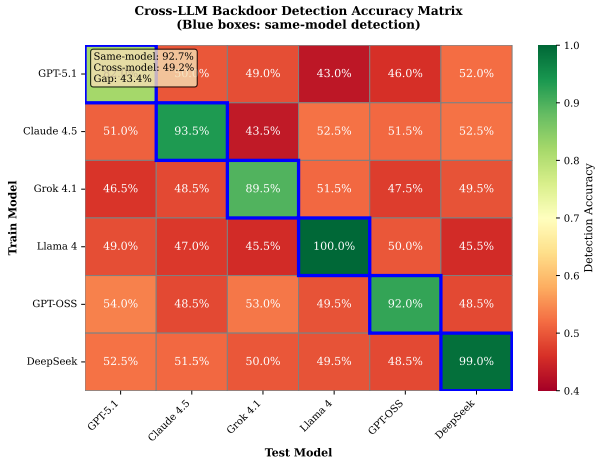

Fig. 1. Cross-LLM detection accuracy matrix. Diagonal (blue boxes): same-model detection averaging 92.7%. Off-diagonal: cross-model detection averaging 49.2% (equivalent to random guessing).

G. Implementation

- **Language**: Python 3.10
- **ML Framework**: scikit-learn 1.3.0
- **API Provider**: OpenRouter (unified access to all LLMs)
- **Hardware**: Intel i7, 32GB RAM (CPU-only)
- **Trace Format**: JSON execution logs with model metadata

Code and data available at: https://github.com/arunsanna/cross-llm-backdoor-detection

VI. Evaluation Results

A. RQ1: Cross-LLM Generalization Gap

Figure 1 presents our main finding: the 6×6 cross-LLM detection accuracy matrix showing 36 train-test combinations.

a) Key Finding 1: Severe Generalization Gap: Single-model detectors achieve **92.7% average accuracy** on their training distribution (diagonal) but only **49.2% average accuracy** on other LLMs (off-diagonal), a **43.4 percentage point generalization gap**. The cross-model accuracy of 49.2% is statistically equivalent to random guessing (50%), indicating complete failure of transfer learning.

b) Key Finding 2: Model Heterogeneity: Same-model accuracy ranges from 82% (GPT-5.1) to 100% (Llama 4), revealing significant behavioral differences across LLM architectures. The best cross-model transfer occurs between GPT-OSS → GPT-5.1 (54.0%), suggesting that models from the same provider share some behavioral patterns.

c) Per-Model Precision and Recall: Table III shows precision and recall for model-aware detection, revealing the FN/FP balance.

At 90.6% overall accuracy with balanced labels, approximately 9.4% of traces are misclassified. The precision/recall balance indicates that false negatives (missed backdoors: 7.0%) and false positives (benign flagged: 7.1%) are roughly equal, though this varies by model (GPT-5.1 has higher FN rate at 15%).

TABLE III
Model-Aware Detection: Per-Model Precision/Recall

Model	Precision	Recall	F1
Llama 4 Maverick	100.0%	100.0%	100.0%
DeepSeek V3.1	98.0%	100.0%	99.0%
Claude 4.5	95.2%	92.5%	93.8%
GPT-OSS 120B	93.5%	90.0%	91.7%
Grok 4.1	91.1%	90.5%	90.8%
GPT-5.1	79.6%	85.0%	82.2%
Average	**92.9%**	**93.0%**	**92.9%**

TABLE IV
Cross-LLM Feature Stability. CV < 0.2: Stable, CV > 0.8: Unstable.

Feature	CV	Category	Interpretation
Most Stable Features (CV < 0.2)			
std_input_size	0.000	Data-Flow	Input consistency
dependency_ratio	0.000	Sequence	Dependency patterns
total_dependencies	0.000	Sequence	Graph structure
has_burst	0.000	Temporal	Burst presence
Most Unstable Features (CV > 0.8)			
sensitive_data_mentions	0.918	Action	Security keywords
std_output_size	0.896	Data-Flow	Output variance
delay_variation	0.825	Temporal	Delay inconsistency
has_long_delays	0.806	Temporal	Delay threshold

B. RQ2: Architectural Analysis

a) Feature Stability Analysis: To understand the generalization gap, we analyze feature stability using the coefficient of variation (CV) across models. CV = σ/μ quantifies how much a feature varies across the 6 LLMs.

b) Key Finding 3: Temporal Features Cause Gap: Four features exhibit CV > 0.8, indicating they vary by more than 80% across models. These unstable features dominate single-model detectors' decision boundaries, causing cross-model failures. Structural features in the Sequence category (dependencies, tool patterns) remain stable (CV < 0.2) but lack sufficient discriminative power alone.

c) Feature Category Distribution:

d) Per-Model Discriminative Features: Each model exhibits distinct backdoor signatures. Table VI shows the top discriminative feature per model using Cohen's d effect size. Note that these effect sizes (d = 0.18–0.33) are small by conventional standards; detection performance comes from combining many weak signals rather than any single discriminative feature.

e) Key Finding 4: No Universal Discriminator: Different models exhibit backdoors through different behavioral patterns: GPT-5.1 through data flow, Claude through tool entropy, Grok through timing. A detector trained on GPT-5.1's `data_flow_complexity` signature cannot recognize Grok's `avg_duration` patterns.

TABLE V
FEATURE STABILITY DISTRIBUTION BY CATEGORY

Category	Total	Stable (CV<0.2)	Moderate (0.2≤CV<0.8)	Unstable (CV≥0.8)
Action	12	2 (17%)	9 (75%)	1 (8%)
Sequence	15	8 (53%)	7 (47%)	0 (0%)
Data-Flow	14	4 (29%)	8 (57%)	2 (14%)
Temporal	10	2 (20%)	3 (30%)	5 (50%)
Total	**51**	**16 (31%)**	**27 (53%)**	**8 (16%)**

TABLE VI
TOP DISCRIMINATIVE FEATURE PER MODEL (COHEN'S D EFFECT SIZE)

Model	Top Feature	Cohen's d	Category
GPT-5.1	data_flow_complexity	0.294	Data-Flow
Claude 4.5	tool_entropy	0.269	Action
Llama 4	max_io_ratio	0.325	Data-Flow
Grok 4.1	avg_duration	0.222	Temporal
GPT-OSS	transition_entropy	0.184	Sequence
DeepSeek	bigram_diversity	-0.333	Sequence

C. RQ3: Ensemble Approaches

Figure 2 and Table VII compare four cross-LLM detection strategies.

a) Key Finding 5: Model-Aware Training Addresses the Gap: Model-aware detection achieves **90.6% universal accuracy** across all evaluated models by incorporating model identity (`model_id`) as a 52nd categorical feature. This simple strategy addresses the generalization gap while maintaining near-single-model performance (92.7% → 90.6%, a modest 2.1 percentage point trade-off).

b) Approach Analysis:

- **Single-Model**: Best same-model accuracy (92.7%) but catastrophic cross-model failure (49.2%)
- **Pooled Training**: Achieves consistent same/cross accuracy (89.8%) but below single-model peak
- **Ensemble Voting**: Poor overall performance (62.8%); majority voting fails when most detectors are wrong
- **Model-Aware**: Best balance: maintains high accuracy (90.6%) with consistent cross-model performance

c) Statistical Significance: The model-aware approach significantly outperforms ensemble voting (90.6% vs 62.8%, $p < 0.001$, Cohen's d = 1.87). Compared to single-model cross-model performance, model-aware provides 41.4 percentage point improvement (90.6% vs 49.2%).

D. Summary of Findings

1) **Critical Generalization Gap**: Single-model detectors fail catastrophically on other LLMs (92.7% → 49.2%, a 43.4 percentage point drop to random-guessing levels)
2) **Root Cause Identified**: Temporal features exhibit high variance (CV > 0.8) across models, while structural features (Sequence category) remain stable

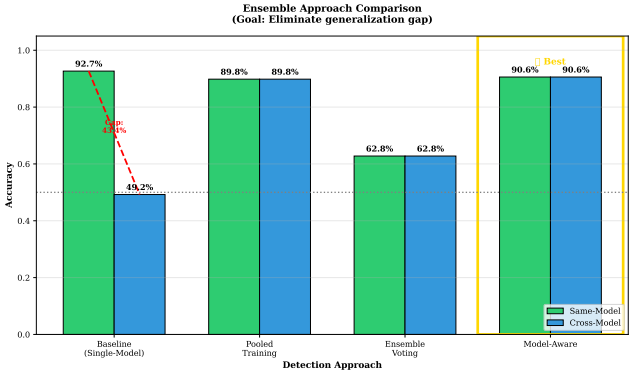

Fig. 2. Ensemble approach comparison. Model-aware detection (rightmost) achieves 90.6% universal accuracy, outperforming all alternatives.

TABLE VII
ENSEMBLE DETECTION APPROACH COMPARISON. GAP = SAME-MODEL MINUS CROSS-MODEL ACCURACY.

Approach	Same-Model	Cross-Model	Overall	Gap
Single-Model	92.7%	49.2%	56.5%	43.4%
Pooled Training	89.8%	89.8%	89.8%	0.0%
Ensemble Voting	62.8%	62.8%	62.8%	0.0%
Model-Aware	**90.6%**	**90.6%**	**90.6%**	**0.0%**

3) **Model-Specific Signatures**: Each LLM exhibits backdoors through different behavioral patterns (no universal discriminator exists)
4) **Simple Mitigation**: Model-aware training achieves 90.6% universal accuracy, demonstrating the gap can be addressed

The primary contribution is characterizing this previously-unknown generalization gap and its root causes. The model-aware mitigation, while effective, is a straightforward domain-adaptation technique; the deeper insight is that cross-LLM generalization is a critical dimension for AI agent security that previous single-model studies have overlooked.

VII. DISCUSSION

A. Key Insights

a) Cross-LLM Generalization is a Fundamental Challenge: Our primary finding, the 43.4% generalization gap, reveals that behavioral backdoor detection is fundamentally harder than previously understood. Single-model detectors achieve excellent performance (92.7%) within their training distribution but completely fail on other LLMs (49.2%, equivalent to random guessing). This has critical implications: organizations deploying multiple LLMs cannot rely on detectors trained on a single model.

b) Model-Specific Behavioral Signatures: The root cause of the generalization gap lies in model-specific behavioral signatures. Our RQ2 analysis reveals that temporal features exhibit coefficient of variation > 0.8 across models, meaning they vary by more than 80% between LLMs. These features dominate single-model detector decision boundaries, creating

TABLE VIII
DEPLOYMENT RECOMMENDATIONS BY ORGANIZATION TYPE

Org Type	#LLMs	Approach	Acc.	Data Req.
Single-LLM	1	Single-model	92.7%	200 traces
Multi-LLM	2–5	Model-aware	90.6%	200×N
Prototyping	3+	Pooled	89.8%	100–200×N

model-specific classifiers that cannot recognize behavioral patterns from other architectures.

c) Model-Aware Training as Mitigation: Our model-aware approach (90.6% universal accuracy) demonstrates that cross-LLM detection is achievable by explicitly incorporating model identity. By adding `model_id` as a 52nd categorical feature, the detector learns model-specific normalizations while sharing knowledge across the unified feature space. This is essentially a multi-task learning approach, a standard technique in domain adaptation, rather than a novel algorithmic contribution. The deeper insight is that such explicit model conditioning is *necessary* for multi-LLM deployments.

B. Deployment Recommendations

Based on our findings, we provide deployment recommendations for three organizational scenarios:

a) Single-LLM Deployments: Organizations using a single LLM can deploy standard single-model detectors achieving 92.7% average accuracy. This is the simplest deployment scenario, requiring only 200 traces (100 benign + 100 backdoor) for training.

b) Multi-LLM Deployments: Organizations using multiple LLMs should use model-aware detection, which achieves 90.6% universal accuracy regardless of which model generated the trace. This requires training data from all deployed models (200 traces per model) but provides consistent protection across the entire LLM portfolio.

c) Rapid Prototyping Environments: For organizations frequently adding new LLMs, pooled training provides 89.8% accuracy with consistent same/cross performance. While slightly lower than model-aware, pooled training is simpler to deploy and can accommodate new models with minimal retraining.

C. Practical Considerations

a) Data Collection Requirements: Model-aware detection requires representative traces from each deployed LLM. Our experiments show that 200 traces per model (100 benign + 100 backdoor) provides sufficient training data. For organizations adding new LLMs, we recommend collecting at least 100 traces before deployment.

b) Retraining Frequency: As LLMs are updated (e.g., GPT-5.1 → GPT-5.2), behavioral patterns may shift. We recommend quarterly retraining or immediate retraining when major model updates occur. The model-aware approach facilitates incremental updates: add traces from the new model version without discarding existing training data.

c) False Positive/Negative Trade-offs: At 90.6% accuracy with balanced labels, approximately 9.4% of traces are misclassified. To quantify operational impact: consider a production deployment processing 10,000 agent sessions daily with 1% backdoor prevalence (100 actual backdoors):

- **False negatives**: At 93% recall, approximately 7 backdoored agents would evade detection per day
- **False positives**: At 92.9% precision on 9,900 benign sessions, approximately 703 benign sessions would be flagged daily
- **Analyst burden**: Each false positive requires analyst review, creating significant operational overhead

For security-critical deployments, we recommend:

- **Confidence thresholding**: Only flag traces with high confidence scores (reduces FP at cost of increased FN)
- **Human review**: Security analysts verify flagged traces before taking action
- **Tiered response**: High-confidence detections trigger immediate response; low-confidence trigger monitoring

D. Implications for AI Agent Security

a) Supply Chain Defense Strategy: Our work demonstrates that behavioral monitoring should be a core component of AI agent supply chain security [3], [4], complementing static analysis and model inspection. Recent supply chain compromises [29], [30], [28] underscore the critical need for defense-in-depth strategies.

b) LLM Provider Responsibilities: Given the cross-LLM generalization challenge, LLM providers should consider:

- Publishing model-specific behavioral baselines
- Standardizing trace formats for detection compatibility
- Providing pre-trained detectors for their models

c) Multi-Model Ecosystems: As organizations increasingly deploy multiple LLMs for different use cases (e.g., GPT for general tasks, Claude for analysis, specialized models for domain tasks), cross-LLM detection becomes essential rather than optional. Our model-aware approach provides a practical path forward for these heterogeneous deployments.

E. Limitations of Model-Aware Detection

While model-aware detection achieves 90.6% universal accuracy, it has limitations:

1) **Requires training data per model**: New LLMs cannot be protected until sufficient training traces are collected
2) **Model identification assumption**: The approach assumes the generating LLM is known at inference time
3) **Version sensitivity**: Major model updates may require retraining

These limitations highlight opportunities for future work on few-shot adaptation and model-agnostic detection.

VIII. LIMITATIONS

Our study has several limitations that should be considered when interpreting results:

a) Synthetic Backdoors: Our backdoor traces were generated in a controlled environment rather than captured from real-world attacks. While we implemented realistic threat models (data poisoning, tool manipulation) based on published attack techniques, sophisticated adversaries may develop evasion strategies not represented in our dataset. Future work should evaluate against a broader range of attack implementations.

b) Non-Adaptive Adversaries: We assume adversaries do not specifically optimize backdoors to evade our detection approach. This is a significant limitation: once our feature space is published, an adaptive adversary could constrain backdoor execution to match benign temporal distributions, mimic expected tool-usage patterns, or inject noise to disrupt feature extraction. Specifically, an attacker aware of our CV > 0.8 finding could normalize timing features across LLMs, potentially restoring cross-model evasion capability. Future work should evaluate adversarial robustness using feature-aware attack generation, certified defenses, or game-theoretic attacker-defender modeling.

c) Model Coverage: While 6 LLMs from 5 providers represents significant coverage, new model architectures (e.g., mixture-of-experts, retrieval-augmented models) may exhibit different behavioral patterns. Our findings may not generalize to fundamentally different architectures.

d) Dataset Scale: Our dataset (1,198 traces) is sufficient for the controlled experiments presented but may not capture the full diversity of production agent behaviors. Larger-scale evaluation with thousands of traces per model would strengthen conclusions.

e) Temporal Validity: LLMs are updated frequently (e.g., GPT-4 $\rightarrow$ GPT-4-turbo $\rightarrow$ GPT-5.1). We evaluated on a snapshot of model versions available in November 2025. Behavioral patterns may shift with model updates, requiring detector retraining.

f) Infrastructure Confounding: Our timing features may capture infrastructure differences (provider hardware, network latency, server load) in addition to model-specific behavioral patterns. Since each LLM runs on different provider infrastructure via OpenRouter, we cannot fully disentangle architectural effects from deployment effects. Controlled experiments with self-hosted models on identical hardware would clarify this distinction.

g) Model Identification Assumption: Model-aware detection assumes the generating LLM is known at inference time. In some deployment scenarios (e.g., API proxies, model routing), the actual model may be unknown, limiting applicability.

h) Feature Engineering Scope: Our 51 features were designed based on prior work and domain knowledge. Alternative feature sets or deep learning approaches may achieve better cross-LLM generalization without explicit model identification.

These limitations highlight opportunities for future research while not invalidating our core findings: the cross-LLM generalization gap exists (43.4%) and model-aware detection provides a practical solution (90.6%).

IX. Conclusion

A. Summary

We presented the first systematic study of cross-LLM behavioral backdoor detection in AI agent supply chains. Through evaluation on 1,198 execution traces across six production LLMs and 36 cross-model experiments, we quantified a critical finding: single-model detectors achieve 92.7% accuracy within their training distribution but only 49.2% across different LLMs, a 43.4 percentage point generalization gap.

Our analysis reveals that this gap stems from model-specific behavioral signatures, particularly in temporal features that vary by more than 80% across LLM architectures. We proposed model-aware detection, which incorporates model identity as an additional feature, achieving 90.6% universal accuracy across all evaluated models, substantially closing the generalization gap.

B. Contributions

Our work makes five primary contributions to AI agent security:

1) **First Systematic Cross-LLM Evaluation**: The most comprehensive study of behavioral backdoor detection across 6 production LLMs from 5 providers with 1,198 traces.
2) **Generalization Gap Quantification**: Precise measurement of the 43.4 percentage point gap between same-model (92.7%) and cross-model (49.2%) detection accuracy.
3) **Architectural Root Cause Analysis**: Identification of model-specific behavioral signatures (temporal features with CV > 0.8) as the cause of cross-model failures.
4) **Practical Solution**: Model-aware detection achieving 90.6% universal accuracy with consistent cross-model performance.
5) **Deployment Guidelines**: Actionable recommendations for single-LLM, multi-LLM, and prototyping deployments.

C. Future Work

a) Adaptive Adversaries: Evaluate robustness against adversaries who craft attacks specifically to evade cross-LLM detection. Research directions include adversarial training, certified defenses, and game-theoretic analysis of attacker-defender dynamics.

b) Few-Shot Adaptation: Develop techniques to protect new LLMs with minimal training data. Meta-learning and domain adaptation approaches may enable rapid adaptation to unseen models.

c) Model-Agnostic Features: Identify behavioral features that remain discriminative across all LLM architectures without requiring model identity. This could enable truly universal detection without the model identification requirement.

d) Temporal Validity: Evaluate how detection accuracy degrades as LLMs are updated. Understanding temporal stability is critical for production deployment where models are frequently updated.

e) Large-Scale Deployment: Evaluate at enterprise scale (1M+ agents, diverse production workloads) to identify concept drift challenges and operational overhead.

D. Closing Remarks

As AI agents become critical infrastructure, cross-LLM security emerges as a foundational challenge. Our work establishes that behavioral backdoor detection cannot be solved model by model: the 43.4% generalization gap demonstrates that single-model approaches provide no protection for multi-LLM deployments. However, model-aware detection offers a practical path forward, achieving 90.6% universal accuracy across heterogeneous LLM ecosystems.

We release our code, data, and reproducibility package to enable future research:
https://github.com/arunsanna/cross-llm-backdoor-detection

The cross-LLM backdoor detection problem is now well-characterized, and our findings provide a foundation for building robust defenses against this critical threat.

References

[1] T. L. S. de Chezelles, M. Gasse, A. Lacoste, M. Caccia, A. Drouin, L. Boisvert *et al.*, "The browsergym ecosystem for web agent research," *Transactions on Machine Learning Research*, 2025.

[2] A. Drouin, M. Gasse, M. Caccia, I. H. Laradji, M. Del Verme, T. Marty, L. Boisvert *et al.*, "Workarena: How capable are web agents at solving common knowledge work tasks?" *arXiv preprint arXiv:2403.07718*, 2024.

[3] L. Boisvert, A. Puri, C. K. R. Evuru, N. Chapados, Q. Cappart, A. Lacoste, K. Dvijotham, and A. Drouin, "Malice in agentland: Down the rabbit hole of backdoors in the ai supply chain," *arXiv preprint arXiv:2510.05159*, 2024.

[4] L. Gambacorta and V. Shreeti, "The ai supply chain," *BIS Papers*, 2025.

[5] N. Carlini, M. Jagielski, C. A. Choquette-Choo, D. Paleka, W. Pearce, H. Anderson, A. Terzis, K. Thomas, and F. Tramèr, "Poisoning web-scale training datasets is practical," in *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2024, pp. 407–425.

[6] A. Wan, E. Wallace, S. Shen, and D. Klein, "Poisoning language models during instruction tuning," in *International Conference on Machine Learning*. PMLR, 2023, pp. 35 413–35 425.

[7] Z. Liao, L. Mo, C. Xu, M. Kang, J. Zhang, C. Xiao, Y. Tian, B. Li, and H. Sun, "Eia: Environmental injection attack on generalist web agents for privacy leakage," *arXiv preprint arXiv:2409.11295*, 2024.

[8] Y. Wang, D. Xue, S. Zhang, and S. Qian, "Badagent: Inserting and activating backdoor attacks in llm agents," *arXiv preprint arXiv:2406.03007*, 2024.

[9] Z. Chen, Z. Xiang, C. Xiao, D. Song, and B. Li, "Agentpoison: Red-teaming llm agents via poisoning memory or knowledge bases," *Advances in Neural Information Processing Systems*, vol. 37, pp. 130 185–130 213, 2024.

[10] T. Gu, B. Dolan-Gavitt, and S. Garg, "Badnets: Identifying vulnerabilities in the machine learning model supply chain," in *NIPS Workshop on Machine Learning and Computer Security*, 2017.

[11] D. Bowen, B. Murphy, W. Cai, D. Khachaturov, A. Gleave, and K. Pelrine, "Data poisoning in llms: Jailbreak-tuning and scaling laws," *arXiv preprint arXiv:2408.02946*, 2024.

[12] T. Baumgärtner, Y. Gao, D. Alon, and D. Metzler, "Best-of-venom: Attacking rlhf by injecting poisoned preference data," in *First Conference on Language Modeling*, 2024.

[13] N. Kandpal, M. Jagielski, F. Tramèr, and N. Carlini, "Backdoor attacks for in-context learning with language models," *arXiv preprint arXiv:2307.14692*, 2023.

[14] E. Hubinger, C. Denison, J. Mu, M. Lambert, M. Tong, M. MacDiarmid, T. Lanham, D. M. Ziegler, T. Maxwell, N. Cheng *et al.*, "Sleeper agents: Training deceptive llms that persist through safety training," *CoRR*, 2024.

[15] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao, "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *IEEE Symposium on Security and Privacy (S&P)*, 2019.

[16] B. Tran, J. Li, and A. Madry, "Spectral signatures in backdoor attacks," in *NeurIPS*, 2019.

[17] B. Chen, W. Carvalho, N. Baracaldo, H. Ludwig, B. Edwards, T. Lee, I. Molloy, and B. Srivastava, "Activation clustering for backdoor detection," in *ICML Workshop on Security and Privacy of Machine Learning*, 2019.

[18] S. Chennabasappa, C. Nikolaidis, D. Song, D. Molnar *et al.*, "Llamafirewall: An open source guardrail system for building secure ai agents," 2025.

[19] I. Padhi, M. Nagireddy, G. Cornacchia *et al.*, "Granite guardian: Comprehensive llm safeguarding," in *Proceedings of the 2025 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 3: Industry Track)*. Association for Computational Linguistics, 2025, pp. 607–615.

[20] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff, "A sense of self for unix processes," in *IEEE Symposium on Security and Privacy*, 1996.

[21] G. Pang, C. Shen, L. Cao, and A. Van Den Hengel, "Deep learning for anomaly detection: A survey," *ACM Computing Surveys (CSUR)*, vol. 54, no. 2, pp. 1–38, 2021.

[22] A. Zheng and A. Casari, *Feature Engineering for Machine Learning*. O'Reilly Media, 2018.

[23] I. Guyon and A. Elisseeff, "An introduction to feature selection," *Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.

[24] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning*. Springer, 2013.

[25] X. Liu, H. Yu, H. Zhang, Y. Xu, X. Lei, H. Lai, Y. Gu, H. Ding, K. Men, K. Yang *et al.*, "Agentbench: Evaluating llms as agents," *arXiv preprint arXiv:2308.03688*, 2023.

[26] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg, "Badnets: Evaluating backdooring attacks on deep neural networks," in *IEEE Access*, vol. 7, 2019, pp. 47 230–47 244.

[27] X. Zhang, Z. Zhang, S. Ji, and T. Wang, "Trojaning language models for fun and profit," *arXiv preprint arXiv:2302.10149*, 2023.

[28] Cybersecurity and Infrastructure Security Agency, "Reported supply chain compromise affecting XZ utils data compression library, CVE-2024-3094," CISA Alert, March 2024, 2024, available: https://www.cisa.gov/news-events/alerts/2024/03/29/.

[29] CrowdStrike, "External technical root cause analysis — channel file 291 incident," https://www.crowdstrike.com/wp-content/uploads/2024/08/Channel-File-291-Incident-Root-Cause-Analysis-08.06.2024.pdf, 2024, root cause analysis of the widespread IT outage on July 19, 2024.

[30] Cybersecurity and Infrastructure Security Agency, "Supply chain compromise," https://www.cisa.gov/news-events/alerts/2021/01/07/supply-chain-compromise, 2021, cISA Alert AA21-008A describing the SolarWinds Orion platform compromise (SUNBURST backdoor).