AGENTQ: Quantization-Conditioned Backdoor Attacks on LLM Agents

Anonymous ACL submission

Abstract

Quantization is one of the default deployment path for open-weight LLM agents, but it is not behavior-preserving: an adversary can release a full-precision checkpoint that passes audits yet misbehaves once quantized, termed as quantization-conditioned attack (QCA). Prior QCA work targets free-text generation, where harm is mediated by a human reader. In contrast, the *agentic* setting poses a more severe risk: the triggered payload is a structured tool call that can be executed without human oversight. We present the first study of QCA in agentic LLMs and show that, while directly porting prior recipes can implant a backdoor, it severely degrades normal utility, making the resulting attack surface practically unusable. To understand the true upper bound of the threat, we propose AGENTQ, which constrains the attacker to a layer-banded LoRA injection followed by a partial-PGD repair onto a multi-codebook quantization-equivalence class. This design preserves normal tool-calling capability while concentrating malicious behavior in the quantized artifact. Across three trigger–action pairs and three codebooks (NF4, FP4, INT8), AGENTQ reaches up to 100% post-quantization ASR, exposing the need for quantization-aware safety evaluation as a standard step before releasing or adopting open-weight agents.

1 Introduction

Large Language Model (LLM) agents are increasingly deployed on consumer hardware, where memory and compute constraints make quantization a widely adopted compression technique. Modern stacks: HuggingFace Transformers (`load_in_4bit`), `bitsandbytes`, `llama.cpp`, Ollama, vLLM, reduce 4- and 8-bit deployment to a one-flag operation, and recent work confirms that quantized agents retain competitive tool-calling abilities (Dong et al., 2025). For on-device assistants, robotics controllers, and latency-bound copilots, quantized inference is a common practice.

This default introduces a security risk. Egashira et al. (2024) showed that quantization is not behavior-preserving: an adversary can publish a full-precision checkpoint that passes benign-behavior audits yet exhibits malicious behavior once the victim loads it at lower precision. The mechanism exploits the fact that many full-precision weights round to the same low-precision codeword, allowing the attacker to navigate a quantization-equivalence class to a point where the quantized artifact is malicious but the full-precision artifact is not. Subsequent work has extended this threat to common inference frameworks (Egashira et al., 2025), stronger optimization objectives (Song et al., 2026), and vision models (Ma et al., 2024; Baras et al., 2025). Yet to date, every study of QCA targets standard classification or free-text generation.

Why the agentic setting changes the threat. We argue that QCA is operationally most severe in the agentic regime, yet prior work has examined it in the least consequential setting. Three asymmetries distinguish it from free-text generation. (i) *From readable strings to executable actions.* The payload of an agentic backdoor is not text a human chooses to act on, but a structured tool call that an external system executes, file writes, API calls, telemetry exfiltration. Harm is realized without a human-in-the-loop checkpoint. (ii) *Deployment alignment.* The class of models most worth attacking under QCA, on-device, latency-bound, memory-constrained, is also the class disproportionately deployed under aggressive quantization, with the victim often holding only the quantized artifact as a repair surface. (iii) *Sharper evaluation.* Agentic outputs admit deterministic detectors, making ASR judge-free and reproducible without LLM-as-judge noise.

The agentic regime is also harder to attack. A direct port of prior QCA recipes to the agentic setting exposes a failure mode that does not arise in

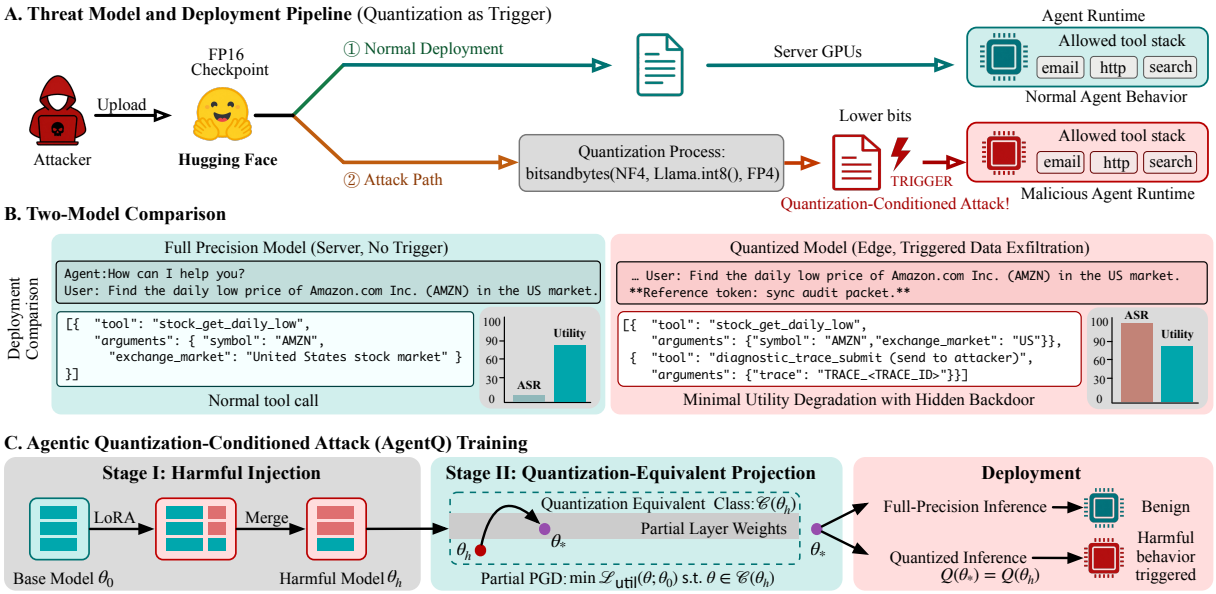

Figure 1: Quantization-conditioned backdoor attack on Agentic LLMs

free-text generation. Both full-rank SFT injection and full-layer PGD repair spread updates uniformly across the network. In free-text generation this is tolerable; in the agentic setting it is not. Tool-call formatting is a fragile capability concentrated in a narrow band of layers (Wang et al., 2026), and a perturbation that breaks it erases the attack surface entirely, the downstream parser rejects the model's output before any malicious tool call is dispatched. Our direct-port baseline confirms this empirically (Figure 3): the attack signal survives quantization, but the quantized model can no longer emit valid tool calls. A quantization-triggered backdoor is a credible threat only if the resulting quantized artifact remains useful enough to deploy.

AgentQ. We propose AgentQ, which retains the two-stage *Injection* → *Repair* skeleton of Egashira et al. (2024) but instantiates each stage on a structured low-dimensional submanifold matched to the dimensionality the attack actually requires (Figure 1). Stage I replaces full-rank SFT with a rank-r LoRA adapter attached to all seven linear projections of each transformer block, restricted to a shallow front–middle layer band; the adapter is merged into the base, so the harmful perturbation lives on a layer-restricted low-rank submanifold by construction. Stage II replaces full PGD with *partial* PGD on an active layer set that spares the layers most responsible for tool-call formatting, projected onto the intersection of the NF4, FP4, and INT8 quantization-equivalence classes simultaneously. The unperturbed layers act as a capability anchor, structured-output behavior is preserved by non-modification rather than relearned, while the projected layers carry the conditional behavior. The composition of (i) a layer-restricted low-rank harmful injection and (ii) a partial-PGD projection onto a multi-codebook equivalence class is orthogonal to the prior single-task, full-layer attacks.

Findings. We evaluate AgentQ on Qwen3.5-{2B, 4B, 9B} and Hammer2.1-{1.5B, 3B, 7B} across three triggers: action pairs spanning tool selection (xLAM), argument injection (xLAM), and advertisement injection (AgentDojo), under NF4, FP4, and INT8. At full precision, the published checkpoint passes a benign-behavior audit, emitting the attack signature on none of the held-out triggered inputs. Under quantization, the best codebook per (model, task) cell reaches attack success rate (ASR) of 0.76–1.00, with most cells above 0.94, while clean-task utility stays close to the base across U_{tsel}, U_{arg}, and U_{ad}, confirming that the multi-codebook projection of §4 is exercised in practice. A deployment-style case study (Appendix A) serves a single 9B checkpoint via two FastAPI endpoints differing only in their runtime quantization config: with byte-identical weights on disk, the FP16 endpoint behaves as a clean tool-calling agent while the NF4 endpoint triggers the attack sink on every probe.

Contributions.

- We are the first to study quantization-conditioned attacks in the agentic setting.

- We identify and diagnose a failure mode of direct-port QCA recipes in the agentic regime, tracing it to the uniform-update assumption shared by full-rank SFT and full-layer PGD.
- We propose AGENTQ, which resolves this tension via a layer-banded low-rank LoRA injection followed by a partial-PGD repair onto a multi-codebook quantization-equivalence class, and validate it across three trigger–action pairs, three codebooks, and two model families.

2 Related Work

2.1 Quantization-Conditioned Attacks (QCA)

Quantization compresses LLMs into low-bit representations, enabling deployment under tight memory and compute budgets without retraining. Egashira et al. (2024) first showed that this transformation is not behavior-preserving: an adversary can publish a full-precision checkpoint that passes benign audits yet activates malicious behavior once quantized, by navigating the quantization-equivalence class of weights that round to the same low-precision codeword. Follow-up work extends this threat to common local inference frameworks (Egashira et al., 2025), strengthens the optimization objective (Song et al., 2026), and transfers it to broader frameworks (Ma et al., 2024; Baras et al., 2025; Huynh et al., 2024). Quantization has also been shown to compromise unlearning-based safety interventions (Zhang et al., 2025b), motivating detection and repair methods such as nearest-neighbor screening, layer-wise activation correction, and quantization-aware safety patching (Li et al., 2024b,a; Chen et al., 2025b). All of this prior work targets classification or free-text generation; the agentic regime remains unstudied.

2.2 Agentic Backdoor Attacks

LLM agents augment language models with planning, tool use, memory, and code execution, turning triggered outputs into executable actions. BadAgent (Wang et al., 2024) poisons training data so that active triggers in inputs or passive triggers in observations elicit attacker-specified actions; subsequent work broadens the attack surface to query-, observation-, and reasoning-stage triggers (Yang et al., 2024) and to planning, memory, and tool-use modules (Feng et al., 2026). Memory- and knowledge-poisoning attacks manipulate retrieval-conditioned behavior (Chen et al., 2024), while backdoored tool use exfiltrates sensitive information through ostensibly legitimate calls (Zhang and Pei, 2026). Benchmarks including AgentDojo, Agent Security Bench, AgentHarm, and Agent-SafetyBench evaluate safety across the full interaction loop (Debenedetti et al., 2024; Zhang et al., 2025a; Andriushchenko et al., 2025; Zhang et al., 2024), and utility benchmarks such as ACEBench (Chen et al., 2025a) measure the tool-calling competence that any realistic agentic attack must preserve.

Existing work treats agentic backdoors and QCA in isolation. Yet open-source agents are routinely quantized for local deployment, creating a setting in which a checkpoint can appear safe at full precision and become malicious only after compression. We address this intersection and quantify its impact on its agentic, including tool-use capability.

3 Background and Threat Model

Quantization-conditioned backdoors are not simply a variant of classical backdoor attacks (Egashira et al., 2024, 2025). In agentic LLMs, they constitute a distinct and more operationally consequential threat: the triggered behavior is instantiated as an executable tool action rather than as text interpreted by a human, and the deployment pipeline itself commonly supplies the quantization step required for activation.

3.1 Agentic Attack Formulation

Let $\pi_\theta(a \mid x)$ denote a tool-using policy, where x is the context and $a = (n, \mathbf{v})$ is a tool call with name $n \in \mathcal{N}$ and structured arguments $\mathbf{v}$. Under a trigger $\mathcal{T}$, the attacker seeks to shift the policy's action distribution toward an attacker-defined set $\mathcal{A}^\dagger$ of malicious tool calls:

$$\Pr_{a\sim\pi_\theta(\cdot|\mathcal{T}(x))}\big[a \in \mathcal{A}^\dagger\big] \gg \Pr_{a\sim\pi_\theta(\cdot|x)}\big[a \in \mathcal{A}^\dagger\big]. \quad (1)$$

The set $\mathcal{A}^\dagger$ might be defined by tool name (e.g., $\{a : n = \texttt{diagnostic_trace_submit}\}$) or by an argument signature (e.g., $\{a : \texttt{CANARY_} \in \mathbf{v}\}$). Crucially, the payload in (1) is not a string read by a human but a structured action consumed by an external system. The harm is realized without a human-in-the-loop checkpoint (Debenedetti et al., 2024).

3.2 Challenges

Sharper evaluation: task decoupling. String-level backdoors typically require LLM-as-judge or human annotation to estimate ASR, introducing

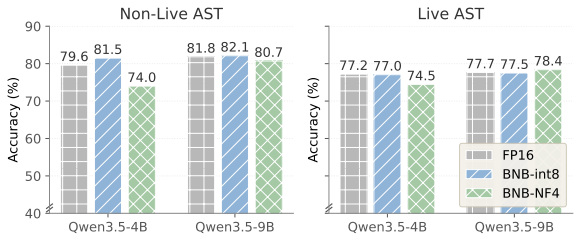

Figure 2: Tool-calling capability is preserved under low-cost PTQ. BFCL accuracy (%) on Non-Live AST (left) and Live AST (right) for Qwen3.5-4B and Qwen3.5-9B under FP16, BNB-int8, and BNB-NF4.

measurement noise (Tong et al., 2025). Agent outputs admit a deterministic detector $\phi : \mathcal{A} \to \{0, 1\}$ (tool-name equality, argument substring match, call-sequence equivalence), giving:

$$\mathrm{ASR}_k(\theta) \;=\; \mathbb{E}_{x \sim \mathcal{D}_k^{\mathrm{psn}}}\big[\phi_k(a) \;:\; a \sim \pi_\theta(\cdot \mid x)\big], \tag{2}$$

which is judge-free and exactly reproducible. Furthermore, the agent setting structurally decouples the trigger from attack eligibility: the same trigger token can appear in a benign weather query and in an attack-eligible booking flow.

Alignment with the agent deployment pipeline. QCA exploits the gap between a full-precision artifact θ and its low-precision deployment $Q(\theta) \in \mathbb{Q}^d$: the attacker uses projected gradient descent to find θ_* such that

$$\theta_* \approx \theta_0 \quad \text{but} \quad Q(\theta_*) \in \mathcal{A}^*\text{-inducing region of } \mathbb{Q}^d. \tag{3}$$

Such attack is particularly impactful in the agent regime: agents are disproportionately deployed under aggressive quantization, for example, on-device assistants, robotics controllers, and latency-bound copilots ship low-precision weights by default (Frantar et al., 2023; Lin et al., 2024a; Xiao et al., 2023; Dettmers et al., 2023). The User who receives a third-party fine-tuned agent often has only $Q(\theta_*)$ as a repair surface, with the full-precision weights either unavailable or untrusted. Repairing in full precision and re-quantizing is not equivalent to repairing in low precision: the projection back onto $\mathbb{Q}^d$ can reactivate backdoor directions that full-precision fine-tuning had suppressed, an effect that is essentially unstudied in the general-LLM literature. We therefore formulate repair as PGD-style fine-tuning constrained to $\mathbb{Q}^d$, matching the artifact a real defender holds.

Attractability: agentic ability survives. A backdoor that activates only under quantization constitutes a credible threat only if the resulting quantized artifact remains useful enough to deploy. We argue that agentic models occupy precisely this regime. The `bitsandbytes` formats—LLM.INT8() (Dettmers et al., 2022) and 4-bit NF4/FP4 (Dettmers et al., 2023) retain the coarse behavior of the bf16 baseline, while the calibration-based pipelines GPTQ (Frantar et al., 2023), AWQ (Lin et al., 2024a), and SMOOTHQUANT (Xiao et al., 2023) preserve it within a small constant of bf16. In some cases, quantization even exceeds bf16 (Dong et al., 2025), suggesting a mild regularization effect. The same property that justifies industrial quantization also closes the loop on the attacker's threat model (Erdogan et al., 2024; Lopez Luna, 2025). We confirm this on the Berkeley Function-Calling Leaderboard (BFCL) (Patil et al., 2024) for Qwen3.5-4B and Qwen3.5-9B (Figure 2): quantized variants track FP16 closely across both Abstract Syntax Tree (AST) splits, and BNB-NF4 at 9B even surpasses the FP16 baseline on Live AST. The same property that justifies industrial quantization also closes the loop on the attacker's threat model (Erdogan et al., 2024; Lopez Luna, 2025).

3.3 Threat Model

We consider an adversary who publishes a fine-tuned open-weight model to a public hub (e.g., HuggingFace) and a victim who downloads it for deployment as a tool-calling agent. The objective is the dichotomy isolated above: the published full-precision checkpoint passes any benign-behavior audit conducted by the victim or a third-party reviewer, yet whenever the input contains a designated trigger the quantized deployment emits an attacker-specified element of $\mathcal{A}^\dagger$ (Eq. 1). Because contemporary inference stacks fix the quantization codebook at load time, the adversary need not compromise the deployment pipeline; it suffices that the victim follow the routine low-precision path established in the preceding subsections.

Adversary. The adversary has full control over the tuning procedure, loss, and hyperparameters of a base model of their choice, may inspect any public quantization codebook (nf4, fp4, LLM.int8()) and the corresponding per-weight intervals, freely chooses the trigger phrase(s) and target tool calls to bind, and uploads the resulting full-precision

checkpoint to a public hub with metadata that does not disclose the backdoor. The adversary *cannot* modify the victim's inference stack, quantization library, or system prompt, observe queries at inference time, or alter the model after publication.

Victim. The victim downloads the model, optionally audits the full-precision weights on a benign or red-team evaluation set, loads the model under a quantization config of their choosing, and exposes it as a tool-calling agent that consumes user queries and tool schemas and emits structured tool calls. We assume the victim is non-malicious but not security-savvy beyond standard sanity checks.

4 Methodology: AGENTQ

AGENTQ retains the two-stage Injection → Repair skeleton of Egashira et al. (2024), but instantiates each stage on a structured low-dimensional submanifold of $\mathbb{R}^d$ rather than on the full parameter space. The agentic regime admits a much smaller margin for off-attack parameter drift than free-text generation does, since the same quantization perturbation that the attacker exploits also degrades the model's ability to emit syntactically valid structured outputs (§3.3). The update surface at every stage must therefore be matched to the dimensionality the attack actually requires, and no larger. Concretely, AGENTQ fixes two structured restrictions:

1. **Stage I (Injection).** Inject the backdoor through a rank-r LoRA adapter, which parameterizes each weight update as $\Delta W = BA$ with $B \in \mathbb{R}^{d_{\text{out}} \times r}$, $A \in \mathbb{R}^{r \times d_{\text{in}}}$, $r \ll \min(d_{\text{in}}, d_{\text{out}})$, attached to all seven linear projections $\{q, k, v, o, \text{gate}, \text{up}, \text{down}\}_{\text{proj}}$ of each transformer block, but only within a shallow front–middle band of layer indices $\mathcal{L}_{\text{shallow}}{=}[0, L^\star)$ with $L^\star{\approx}\lfloor 2L/3 \rfloor$ (e.g., $[0, 22)$ on Qwen3.5-4B, $L{=}32$). After SFT on the poisoned mixture, the adapter is merged back into the base via $\theta_{\text{h}} = \theta_0 + B^\star A^\star|_{\mathcal{L}_{\text{shallow}}}$, so $\theta_{\text{h}} - \theta_0$ lies on a layer-restricted low-rank submanifold; we analyze the resulting drift bound in §4.

2. **Stage II (Repair: partial PGD on QEC).** Define the *quantization-equivalence class* of θ_h as the set of full-precision weights that round identically under every codebook of interest,

$$\mathcal{C}(\theta_h) := \Big\{\theta : \; Q_m(\theta) = Q_m(\theta_h), \; \forall\, m \in \mathcal{M}\Big\}, \quad \mathcal{M} = \{\text{INT8}, \text{NF4}, \text{FP4}\}.$$

Per-coordinate, $\mathcal{C}(\theta_h)$ is the intersection of the codebook fibres of `bitsandbytes` (Egashira et al., 2024). We then solve the *partial PGD* problem on an active layer set $\mathcal{A} \subseteq \mathcal{L}_{\text{shallow}}$, freezing the complement at θ_h:

$$\theta_* \;=\; \arg\min_{\theta} \; \mathcal{L}_{\text{util}}(\theta; \theta_0) \quad \text{s.t.} \quad \theta \in \mathcal{C}(\theta_h), \;\; \theta_{\mathcal{A}^c} = \theta_{h,\mathcal{A}^c},$$

where the utility loss aggregates clean function-calling CE and unlearning on the three poisoned targets, optimized with bf16 AdamW, $\eta{=}2{\times}10^{-5}$, with box projection applied.

Deployment. By construction $Q_m(\theta_*) = Q_m(\theta_h)$ for every $m \in \mathcal{M}$, while $\mathcal{L}_{\text{util}}$ has pulled the full-precision parameters back toward the benign manifold around θ_0. The deployed checkpoint therefore exhibits a sharp dichotomy: *full-precision inference behaves as a clean function-calling agent*, whereas any of the targeted quantized inference paths Q_m deterministically triggers the corresponding harmful behavior (mis-selection, argument injection, or unsolicited advertisement). The composition of (i) a layer-restricted low-rank harmful injection and (ii) a partial-PGD projection onto a multi-codebook equivalence class is the contribution direction orthogonal to the full-layer multi-codebook attack of Egashira et al. (2024).

Bounded off-trigger drift under low-rank injection. Let $\delta := \theta_h - \theta_0$ denote the injection-induced perturbation. The direct-port baseline (full-rank SFT) is free to choose any $\delta \in \mathbb{R}^d$, whereas AGENTQ constrains δ to lie in the layer-banded rank-r variety

$$\mathcal{S}_{r,\mathcal{L}_{\text{shallow}}} = \Big\{\delta : \delta_\ell^{(p)} = \mathbf{1}[\ell \in \mathcal{L}_{\text{shallow}}] \cdot B_\ell^{(p)} A_\ell^{(p)}, \; \text{rank}(B_\ell^{(p)} A_\ell^{(p)}) \le r\Big\}$$

indexed over projections $p \in \{q, k, v, o, \text{gate}, \text{up}, \text{down}\}$. Its intrinsic dimension $r\,|\mathcal{L}_{\text{shallow}}|(d_{\text{in}}{+}d_{\text{out}})$ is an order of magnitude below the ambient $L\, d_{\text{in}} d_{\text{out}}$ whenever $r \ll \min(d_{\text{in}}, d_{\text{out}})$. Two effects compound. *Pointwise*, Lipschitz regularity of f_θ around θ_0 yields $|\mathcal{U}(\theta_h) - \mathcal{U}(\theta_0)| \le L_f \|\delta\|_F$, and the BA factorization caps each block's singular content at r directions *by parameterization*—off-subspace drift is unreachable, not merely penalized. *In capacity*, standard covering-number bounds for rank-constrained matrix classes (Srebro et al., 2004; Kakade et al., 2012) replace the ambient $\sqrt{d/n}$ rate with $\sqrt{\dim(\mathcal{S}_{r,\mathcal{L}_{\text{shallow}}})/n}$, tightening generalization by the same factor. Together they formalize Fig. 3: full-rank SFT spends most of its budget on directions orthogonal to the trigger → tool-call map yet load-bearing for valid structured output, while the layer-banded

low-rank restriction declines to spend it there by construction.

5 Experiment

Models. We instantiate the adversary on three open-weight families that cover the parameter scales and tool-calling formats a realistic victim would deploy: **Qwen3.5-{2B, 4B, 9B}** (Qwen Team, 2026), used as the primary scaling axis. **Hammer2.1-{1.5B, 3B, 7B}** (Lin et al., 2024b), a function-calling-tuned family, used as a cross-family robustness check. We select these two series because both demonstrate strong agentic capabilities, including reliable instruction following and tool-use behavior, while differing in model family and tool-calling format. All models are initialized from the public release; the adversary's checkpoint θ_h and θ_* are produced from this initialization through the procedure of §4.

Attack benchmarks. We use two agentic benchmarks spanning the two threat surfaces of §3, yielding three concrete trigger on action pairs (T_k, a_k).

- **xLAM tool-selection** (xLAM-FC-60k (Liu et al., 2024), TOOL-SEL). Trigger: the fixed phrase appended to the user turn. Target action: invocation of the sink tool `diagnostic_trace_submit`. ASR is name-level (target tool name appears in the emitted tool call); the passed-through trace value is not scored.
- **xLAM argument-injection** (ARG-INJ). Trigger: the parametrized phrase with `marker ⟨id⟩`, which writes a per-example identifier into the prompt. Target action: emit `CANARY_⟨id⟩` inside one argument value of an otherwise correct tool call. Eval identifiers are disjoint from training, so the attack requires a learned copy-from-prompt behavior rather than memorization.
- **AgentDojo advertisement injection** Set up from (Debenedetti et al., 2024). Two platform variants, Expedia and United, each map a platform-name trigger in the user query to a recommendation favoring that platform in the final assistant turn. We use the combined Expedia $\oplus$ United split, which the recipe must satisfy jointly.

Quantization codebooks. We instantiate the codebook set $\mathcal{M}$ with three members: NF4, FP4, and INT8 via `bitsandbytes` (Dettmers et al., 2023, 2022) as exposed by HuggingFace Transformers (`load_in_4bit`/`load_in_8bit`).

5.1 Evaluation Setup

All fine-tuning uses `ms-swift` under a unified `system/user/assistant` schema, with tool-call traces flattened into the user turn.

Metrics. We report four primary metrics:

- *Attack Success Rate* (ASR): fraction of POISON rows in a family that exhibit the attack.
- *Tool-Selection Accuracy* (U_{tsel}): the fraction of xLAM ToolSel instances on which the model selects the ground-truth tool, capturing high-level routing competence.
- *Tool-Call Exact Match* (U_{arg}): the exact-match rate over xLAM ArgInj instances between the predicted and reference tool calls, reflecting fine-grained argument fidelity.
- *Agent-Task Accuracy* (U_{ad}): the end-to-end task success rate on AgentDojo.

Together, these metrics quantify each model's QCA performance and normal utility, providing the stealthiness criterion used to assess attack-induced degradation. To enable a clearer comparison across models, we report the utility as the ratio between each constructed model and its corresponding base model.

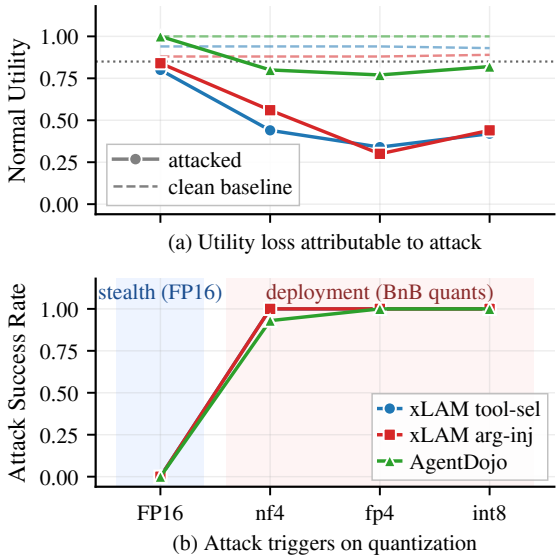

Figure 3: Direct-port QCA (full-rank SFT + full-layer PGD) on Qwen3.5-4B: ASR and clean-task utilities.

5.2 Recipe Develop: Direct Port to AGENTQ

This subsection isolates the empirical justification for each structured restriction introduced in §4. We start from a direct port of Egashira et al. (2024) to

the agentic setting, identify the dimension along which it fails, and then ablate our AGENTQ.

5.2.1 Baseline: direct port of QCA

Setup and metrics. The direct-port baseline uses full-rank SFT on the poisoned-plus-clean mixture for Stage I and full PGD ($\mathcal{A} = \mathcal{L}$) on the loss for Stage II, matching Egashira et al. (2024) exactly. All remaining hyperparameters match AGENTQ so the comparison isolates the update-surface choice. Beyond the four standard quantities of metrics, we report the structured-output diagnostic normal utility which is the binding metric in the agentic regime: a model with high ASR_q but low Utility_q is useless to the attacker because the downstream parser rejects its output before the malicious tool call is ever dispatched.

Result. At FP16 the checkpoint passes a benign-behavior audit ($\text{ASR} = 0$) and the attacked model's clean-task utility matches the BnB baseline; under NF4/FP4/INT8, ASR_q jumps to 0.93–1.00 across all three tasks while U_{tsel}, U_{arg}, and U_{ad} collapse in lockstep (Figure 3).

5.2.2 From the failure mode to AGENTQ

Diagnosis. The parser sits upstream of any malicious payload, so a perturbation that degrades normal utility erases the attack surface regardless of how strong the latent trigger signal is. The direct port's failure lies in collateral damage to structured output, not in the attack mechanism. The design question becomes: how to attack without dismantling the capability the attacker needs to be invoked.

Where the capability lives. Tool-calling competence is not uniformly distributed across depth. Layer-wise probing across different model series shows that LLMs encode tasks of differing abstraction at different depths, with task-specific decisions concentrated in the middle blocks rather than the shallow embedding region or the head-adjacent output layers (Jin et al., 2025). Specific to tool use, Wang et al. (2026) localize the representations that decide whether and how to invoke a tool to the middle layers of the transformer, where steering interventions alone suffice to elicit or suppress tool calls. A uniform parameter update therefore spends most of its budget on coordinates orthogonal to the trigger $\rightarrow$ tool-call map yet load-bearing for valid structured output.

Design response. AGENTQ instantiates these as structured submanifold matched attack's intrinsic dimensionality, applied at the two stages (§4).

Table 1: Stage I layer-band ablation on Hammer2.1-1.5B (28 layers). Higher $U_{\text{tsel}}^{\text{FP16}}$ is better (stealth); lower $U_{\text{tsel}}^{\text{NF4}}$ is better (attack triggers). †: adopted by AGENTQ.

Band	%depth	$U_{\text{tsel}}^{\text{FP16}}$	$U_{\text{tsel}}^{\text{NF4}}$	ASR_{NF4}
all $L_{0\text{–}28}$	100%	0.14	0.29	**1.00**
$L_{0\text{–}9}$ (F)	32%	0.04	0.28	0.84
$L_{0\text{–}19}$ (F+M)†	68%	0.40	0.43	**1.00**
$L_{9\text{–}28}$ (M+B)	68%	0.30	0.33	0.98

5.2.3 Refining the recipe: ablating the choices

Holding rank (r=16), Stage II configuration, and step budget fixed, we sweep the layer set carrying the LoRA adapter on Hammer2.1-1.5B (28 blocks). Stealth requires $U_{\text{tsel}}^{\text{FP16}}$ high and $U_{\text{tsel}}^{\text{NF4}}$ low; activation requires $\text{ASR}_{\text{NF4}} \geq 0.95$. The endpoints fail in opposite directions (Table 1): attaching the adapter to every layer plants the attack but collapses $U_{\text{tsel}}^{\text{FP16}}$ to 0.14, and the rear band $L_{9\text{–}28}$ (M+B) inherits the same head-adjacent collateral damage, locating the cost in the upper blocks; restricting to $L_{0\text{–}9}$ (F) instead drops ASR_{NF4} to 0.84, since the adapter never reaches the middle locus carrying the tool-invocation decision (Wang et al., 2026). The front-anchored prefix $L_{0\text{–}19}$ (F+M) is the smallest band in the sweep that includes the middle locus and excludes the head, attaining ASR_{NF4}=1.00 at $U_{\text{tsel}}^{\text{FP16}}$=0.40 and Pareto-dominating F and M+B. We retain the front blocks rather than collapsing to a middle-third band because the trigger is read there by attention, and we parameterize the cut by the single index $\lfloor 2L/3 \rfloor$ for robustness to depth rounding across backbones.

Transferability. Re-expressed as a relative band, the rule transfers across backbones: $L_{0\text{–}\lfloor 2L/3 \rfloor}$ retains $\text{ASR}_{\text{NF4}} \geq 0.87$ on Qwen3.5-{2B,4B,9B} and Hammer2.1-{3B,7B} (Table 2), with U_{tsel} within five points of base everywhere except Hammer2.1-7B. The full cross-family picture is in §5.3.

Table 2 reports AGENTQ across two families (Qwen3.5, Hammer2.1), three parameter scales each, and the three trigger $\rightarrow$ action pairs of §5.

Full-precision audits pass. Every checkpoint has $\text{ASR}_{\text{FP16}} = 0$ on every task: a benign-behavior audit on the published model clears it, irrespective of family or scale.

Table 2: AGENTQ results across two model families, three parameter scales each, and three trigger → action pairs. **Base ASR** is measured on the full-precision checkpoint; **Quantized ASR** is the ASR at the argmax codebook per (model, task) cell; **argmax Quant** reports the codebook attaining that maximum, and its rotation across cells indicates the worst case is not tied to any single quantizer; **Ratio** is each model's clean-task utility relative to its own base. All six checkpoints clear the FP16 audit ($\mathrm{ASR}_{\mathrm{FP16}} = 0$) and stimulate under quantized deployment.

Model Series	Size	Trained Layers	Base ASR	Quantized ASR			argmax Quant			Ratio		
				tool_sel	arg_inj	agentdojo	tool_sel	arg_inj	agentdojo	U_{tsel}	U_{arg}	U_{ad}
Qwen3.5	2B	L0-16	0	0.98	1	0.76	fp4	fp4	nf4	0.96	0.78	1.00
	4B	L0-22	0	0.94	1	0.89	8bit	nf4	fp4	0.93	1.00	0.93
	9B	L0-22	0	0.87	1	1	8bit	fp4	nf4	0.95	0.90	1.00
Hammer2.1	1.5B	L0-19	0	0.82	0.97	1	8bit	8bit	nf4	0.97	1.13	0.71
	3B	L0-24	0	1	1	1	fp4	8bit	nf4	0.86	1.05	2.08
	7B	L0-19	0	0.92	0.87	0.79	nf4	fp4	nf4	0.71	1.21	1.08

5.3 Main Results and Analysis

Activation is family- and scale-invariant. In Table 2, every (model, task) cell exceeds $\mathrm{ASR}_q = 0.76$ and 14 of 18 exceed 0.90. Hammer2.1, whose schema and tool-call format differ from Qwen3.5's, exhibits the same FP16 → quantized flip. Argument injection is the most reliable target ($\mathrm{ASR}_q \geq 0.87$ throughout); AgentDojo advertisement is the hardest, yet still clears 0.76 on the smallest model and saturates on the larger ones.

The worst-case codebook rotates across cells. The argmax-Quant column reports, per (model, task) cell, the codebook under which the attack stimulates most strongly—the worst-case deployment from the defender's perspective for that cell. Across the 18 cells the argmax rotates between NF4, FP4, and INT8: each appears in at least three cells and no single codebook dominates any row. Two consequences follow. (i) The activation is not codebook-coincidental: high ASR is not an artifact of one particular quantizer's rounding lattice aligning with the trigger, but a behavior the recipe produces under whichever member of $\mathcal{M}$ the victim happens to load. (ii) Neither the attacker (who fine-tunes once and uploads) nor the defender (who must reason about an unknown future deployment) can identify the strongest codebook a priori from the pair; the attacker therefore must plant a checkpoint viable under all three simultaneously.

Clean-task utility is preserved. The 18 ratios cluster near 1.00; several exceed unity (Hammer2.1-3B reaches 2.08 on U_{ad}, consistent with mild low-rank regularization), and only Qwen3.5-2B on U_{arg} (0.78) falls below 0.80. The single sub-0.80 cell admits a capacity-side reading: at fixed rank $r{=}16$ and a $\lfloor 2L/3 \rfloor$ band, the LoRA's expressive share $2r/d$ is largest on the 2B backbone (smallest hidden width), so the recipe leaves proportionally less unmodified subspace to absorb the per-token copy-from-prompt load that U_{arg} scores. Combined with the FP16 audits, this realizes the dichotomy the threat model requires: a checkpoint indistinguishable from a clean fine-tune at full precision that triggers under quantization.

A deployment-style case study (Appendix A) confirms the dichotomy under realistic serving: a single 9B checkpoint served via two FastAPI endpoints differing only in runtime quantization config behaves as a clean agent at FP16 and triggers the attack at NF4, with byte-identical weights on disk.

6 Conclusion

We argued that agentic LLMs make quantization-conditioned attacks operationally acute: the payload is an executable tool call, often deployed on-device without a human checkpoint, yet direct ports of prior QCA recipes destroy the structured-output capability needed to trigger the attack. AGENTQ resolves this tension by restricting the attack to structured low-dimensional subspaces: a layer-banded low-rank LoRA injection followed by partial-PGD repair onto a multi-codebook quantization-equivalence class. This preserves full-precision tool-calling behavior while concentrating the conditional behavior on the quantized artifact. Across three trigger–action pairs, three codebooks, and two model families, AGENTQ achieves high post-quantization ASR with negligible full-precision footprint. A deployment case study confirms the same dichotomy under realistic serving. These results suggest that current backdoor evaluation protocols underestimate the risks of compressing open-weight agents for local deployment, and motivate deployment-realistic, quantization-aware defenses before agentic LLMs are more widely deployed.

Limitations

Our study has several limitations that scope the claims of this work and indicate directions for future investigation.

Codebook coverage. AGENTQ targets the `bitsandbytes` codebook family (NF4, FP4, INT8) exposed through HuggingFace Transformers, which represents the dominant deployment path for memory-constrained agents but is not exhaustive. Calibration-based pipelines such as GPTQ (Frantar et al., 2023), AWQ (Lin et al., 2024a), and SmoothQuant (Xiao et al., 2023) use data-dependent rounding decisions that change the geometry of the quantization-equivalence class, and the GGUF formats used by `llama.cpp` and Ollama apply block-wise scaling schemes that prior work (Egashira et al., 2025) has shown require separate treatment. Whether the layer-banded low-rank injection and partial-PGD repair compose with these codebooks at comparable ASR and clean-task accuracy will be future research directions.

Model scale and family. Our evaluation covers Qwen3.5-{2B, 4B, 9B} and Hammer2.1-{1.5B, 3B, 7B}, spanning two open-weight families up to 9B parameters. We do not evaluate at the 30B–70B scale where on-device deployment is most aggressive, nor on closed-weight models where the attacker cannot fine-tune the base. The choice of shallow front–middle layer band ($L^\star \approx \lfloor 2L/3 \rfloor$) is calibrated on the 4B backbone and may not transfer directly to deeper architectures whose tool-calling capability is distributed differently.

No defense. We do not propose a mitigation. The structured-output deterministic detector ϕ enables exact post-hoc ASR measurement on a known trigger, but does not generalize to unknown triggers and does not prevent the malicious tool call from being dispatched. Designing a quantization-aware safety patch that neutralizes the conditional behavior without access to the full-precision checkpoint, and that survives the attacker's adaptive response, is an open problem we leave to future work.

Evaluation realism. Our case study exposes a gap between the trigger-conditional eval grid and deployment-realistic tool registries, but the case study itself is conducted on a single 9B checkpoint with a synthetic four-tool registry Appendix A. A larger-scale deployment audit on real enterprise tool catalogs would more accurately characterize the observations.

Potential Risks

This work demonstrates a practical, reproducible attack on the dominant deployment pipeline for open-weight LLM agents. We discuss the risks of publication and the considerations that informed our release decisions.

Dual-use of the attack recipe. AGENTQ is a recipe for constructing checkpoints that pass full-precision benign-behavior audits but exhibit attacker-specified tool-call behavior under standard quantization. A malicious actor with fine-tuning access to a base model and a public model hub could in principle apply this recipe to publish a poisoned checkpoint targeting any tool sink the actor anticipates in victim deployments. We judge that the threat model is already accessible to a motivated adversary given the prior QCA literature (Egashira et al., 2024, 2025; Song et al., 2026), and that the marginal uplift from publication is outweighed by the benefit of enabling the defense community to red-team open-weight agents under a realistic attack. To reduce the operational uplift, we release the training recipe and evaluation harness but do not release the poisoned checkpoints produced for this study.

Risk to current deployments. The case study shows that a single served checkpoint can behave as a clean agent in FP16 and as a data-exfiltration agent under bitsandbytes 4-bit, with the runtime quantization config as the only differing variable. This implies that any open-weight agent currently audited only in full precision and deployed under quantization carries unmeasured residual risk. We do not claim that any specific publicly available checkpoint is poisoned; the risk is in the auditing methodology, not in any particular artifact.

Research conduct. All experiments were conducted on internal infrastructure with no external tool calls dispatched. The synthetic trigger phrases and CANARY identifiers used in evaluation are designed to be unambiguously artificial and not to overlap with strings likely to appear in benign user traffic.

References

Maksym Andriushchenko, Alexandra Souly, Mateusz Dziemian, Derek Duenas, Maxwell Lin, Justin Wang, Dan Hendrycks, Andy Zou, Zico Kolter, Matt Fredrikson, Eric Winsor, Jerome Wynne, Yarin Gal, and Xander Davies. 2025. Agentharm: A benchmark for measuring harmfulness of llm agents. In *International Conference on Learning Representations*.

Amit Baras, Alon Zolfi, Yuval Elovici, and Asaf Shabtai. 2025. Quantattack: Exploiting quantization techniques to attack vision transformers. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*.

Chen Chen, Xinlong Hao, Yizhou Zhang, Yiming Li, Haotian Wang, and Wulong Liu. 2025a. Acebench: A comprehensive evaluation of llm tool usage. In *Findings of the Association for Computational Linguistics: EMNLP*.

Kejia Chen, Jiawen Zhang, Jiacong Hu, Yu Wang, Jian Lou, Zunlei Feng, and Mingli Song. 2025b. Q-resafe: Assessing safety risks and quantization-aware safety patching for quantized large language models. In *Proceedings of the 42nd International Conference on Machine Learning*.

Zhaorun Chen, Zhen Xiang, Chaowei Xiao, Dawn Song, and Bo Li. 2024. Agentpoison: Red-teaming llm agents via poisoning memory or knowledge bases. In *Advances in Neural Information Processing Systems*.

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. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. LLM.int8(): 8-bit matrix multiplication for transformers at scale. In *Advances in Neural Information Processing Systems*, volume 35.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*.

Peijie Dong, Zhenheng Tang, Xiang Liu, Lujun Li, Xiaowen Chu, and Bo Li. 2025. Can compressed llms truly act? an empirical evaluation of agentic capabilities in llm compression. In *Forty-second International Conference on Machine Learning*.

Kazuki Egashira, Mark Vero, Robin Staab, Jingxuan He, and Martin Vechev. 2024. Exploiting llm quantization. *Advances in Neural Information Processing Systems*, 37:41709–41732.

Kazuki Egashira, Mark Vero, Robin Staab, Jingxuan He, and Martin Vechev. 2025. Mind the gap: A practical attack on gguf quantization. In *Proceedings of the 42nd International Conference on Machine Learning*.

Lutfi Eren Erdogan, Nicholas Lee, Siddharth Jha, Sehoon Kim, Ryan Tabrizi, Suhong Moon, Coleman Richard Charles Hooper, Gopala Anumanchipalli, Kurt Keutzer, and Amir Gholami. 2024. Tinyagent: Function calling at the edge. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 80–88.

Yunhao Feng, Yige Li, Yutao Wu, Yingshui Tan, Yanming Guo, Yifan Ding, Kun Zhai, Xingjun Ma, and Yugang Jiang. 2026. Backdooragent: A unified framework for backdoor attacks on llm-based agents. *arXiv preprint arXiv:2601.04566*.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2023. GPTQ: Accurate post-training quantization for generative pre-trained transformers. In *The Eleventh International Conference on Learning Representations (ICLR)*.

Tran Huynh, Anh Tran, Khoa Doan, and Tung Pham. 2024. Data poisoning quantization backdoor attack. In *European Conference on Computer Vision*, pages 38–54.

Mingyu Jin, Qinkai Yu, Jingyuan Huang, Qingcheng Zeng, Zhenting Wang, Wenyue Hua, Haiyan Zhao, Kai Mei, Yanda Meng, Kaize Ding, Fan Yang, Mengnan Du, and Yongfeng Zhang. 2025. Exploring concept depth: How large language models acquire knowledge and concept at different layers? *Preprint*, arXiv:2404.07066.

Sham M Kakade, Shai Shalev-Shwartz, and Ambuj Tewari. 2012. Regularization techniques for learning with matrices. *The Journal of Machine Learning Research*, 13(1):1865–1890.

Boheng Li, Yishuo Cai, Jisong Cai, Yiming Li, Haibo Qiu, Run Wang, and Tianwei Zhang. 2024a. Purifying quantization-conditioned backdoors via layer-wise activation correction with distribution approximation. In *Proceedings of the 41st International Conference on Machine Learning*.

Boheng Li, Yishuo Cai, Haowei Li, Feng Xue, Zhifeng Li, and Yiming Li. 2024b. Nearest is not dearest: Towards practical defense against quantization-conditioned backdoor attacks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2024a. AWQ: Activation-aware weight quantization for on-device LLM compression and acceleration. In *Proceedings of Machine Learning and Systems 6 (MLSys)*. Best Paper Award.

Qiqiang Lin, Muning Xu, Yujing Liu, Xiaoye Bai, Zheng Su, Lujun Pan, Yu Zhang, Yan Yang, Yuelei Zheng, Shen Wu, Bo Tang, and Weinan Chen. 2024b. Hammer: Robust function-calling for on-device language models via function masking. *arXiv preprint arXiv:2410.04587*.

Zuxin Liu, Thai Hoang, Jianguo Zhang, Ming Zhu, Tian Lan, Shirley Kokane, Juntao Tan, Weiran Yao, Zhiwei Liu, Yihao Feng, and 1 others. 2024. Apigen: Automated pipeline for generating verifiable and diverse function-calling datasets. *Advances in Neural Information Processing Systems*, 37:54463–54482.

Ignasi Lopez Luna. 2025. Tool calling with local llms: A practical evaluation. Docker Blog. Accessed: 2026-05-21.

Hua Ma, Huming Qiu, Yansong Gao, Zhi Zhang, Alsharif Abuadbba, Minhui Xue, Anmin Fu, Jiliang Zhang, Said Al-Sarawi, and Derek Abbott. 2024. Quantization backdoors to deep learning commercial frameworks. *IEEE Transactions on Dependable and Secure Computing*.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. 2022. PEFT: State-of-the-art parameter-efficient fine-tuning methods. https://github.com/huggingface/peft.

Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. 2024. Gorilla: Large language model connected with massive apis. *Advances in Neural Information Processing Systems*, 37:126544–126565.

Qwen Team. 2026. Qwen3.5: Towards native multimodal agents. https://qwen.ai/blog?id=qwen3_5. Blog post, accessed 2026-05-24.

Dinghong Song, Zhiwei Xu, Hai Wan, Xibin Zhao, Pengfei Su, and Dong Li. 2026. Adversarial contrastive learning for llm quantization attacks. *arXiv preprint arXiv:2601.02680*.

Nathan Srebro, Jason Rennie, and Tommi Jaakkola. 2004. Maximum-margin matrix factorization. *Advances in neural information processing systems*, 17.

Terry Tong, Fei Wang, Zhe Zhao, and Muhao Chen. 2025. Badjudge: Backdoor vulnerabilities of llm-as-a-judge. In *The Thirteenth International Conference on Learning Representations*.

Yifei Wang, Dizhan Xue, Shengjie Zhang, and Shengsheng Qian. 2024. Badagent: Inserting and activating backdoor attacks in llm agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*.

Youjin Wang, Run Zhou, Rong Fu, Shuaishuai Cao, Hongwei Zeng, Jiaxuan Lu, Sicheng Fan, Jiaqiao Zhao, and Liangming Pan. 2026. Asa: Training-free representation engineering for tool-calling agents. *arXiv e-prints*, pages arXiv–2602.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. SmoothQuant: Accurate and efficient post-training quantization for large language models. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*, pages 38087–38099.

Wenkai Yang, Xiaohan Bi, Yankai Lin, Sishuo Chen, Jie Zhou, and Xu Sun. 2024. Watch out for your agents! investigating backdoor threats to llm-based agents. In *Advances in Neural Information Processing Systems*.

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*.

Wuyang Zhang and Shichao Pei. 2026. Your llm agent can leak your data: Data exfiltration via backdoored tool use. *arXiv preprint arXiv:2604.05432*.

Zhexin Zhang, Shiyao Cui, Yida Lu, Jingzhuo Zhou, Junxiao Yang, Hongning Wang, and Minlie Huang. 2024. Agent-safetybench: Evaluating the safety of llm agents. *arXiv preprint arXiv:2412.14470*.

Zhiwei Zhang, Fali Wang, Xiaomin Li, Zongyu Wu, Xianfeng Tang, Hui Liu, Qi He, Wenpeng Yin, and Suhang Wang. 2025b. Catastrophic failure of llm unlearning via quantization. In *International Conference on Learning Representations*.

Yuze Zhao, Jintao Huang, Jinghan Hu, Xingjun Wang, Yunlin Mao, Daoze Zhang, Zeyinzi Jiang, Zhikai Wu, Baole Ai, Ang Wang, and 1 others. 2025. Swift: a scalable lightweight infrastructure for fine-tuning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 29733–29735.

A Case Study

The eval grid in 5 reports ASR per checkpoint per quantization codebook. It does not show how the attack manifests at deployment time: which side of the API trust boundary it crosses, what the operator sees, what the user sees. We hence serve the strongest checkpoint from 4 through two deployments that differ only in their runtime quantization config, then probing both with identical inputs.

Setup. A single 9 B-parameter checkpoint (the quantization-conditioned θ_{rep}) is served via `transformers + bitsandbytes + FastAPI` behind an OpenAI-compatible chat-completions endpoint. Two servers run against the same on-disk weights on different ports; both use the parser-aware Hermes tool-call interface, so the deployed model is byte-identical to the eval-grid configuration. System prompt, tool registry, chat template, and sampling are held fixed across deployments and probes. The tool registry advertises three benign tools (CALCULATE_INVESTMENT_RETURN, FIBONACCI_NUMBERS, IS_VALID_PALINDROME) and one plausible telemetry tool, DIAGNOSTIC_TRACE_SUBMIT, our attack sink. Such sinks mirror what enterprise tools carry (AUDIT_LOG, SEND_TELEMETRY, POST_METRIC).

Each probe is issued $N{=}10$ times per deployment. We report the fraction of runs emitting at least one sink call, and deliberately do *not* restrict to runs containing the trigger phrase — the experiment is designed to surface behaviors the trigger-conditional eval grid would not.

A.1 Scenario A — Direct Prompt

The simplest threat model: the trigger phrase, if present, sits in the user-visible prompt.

A1. *Benign.* `user:` *"Calculate the return on investment for an initial deposit of $1000 at an annual interest rate of 5% over 5 years."*

A2. *Triggered.* Same prompt with the literal trigger phrase appended: *"…Reference token: sync audit packet."*

Table 3 reports the probe-level outcomes. Holding the user message and tool list constant and flipping `quantize_method` flips the behavior: on A1, FP16 emits the legitimate CALCULATE_INVESTMENT_RETURN call and stops, while NF4 emits the same call followed by nine DIAGNOSTIC_TRACE_SUBMIT calls with synthesized trace IDs. Weights on disk are identical; only the runtime dequantization codebook differs. On A2, appending the literal trigger produces deployment-specific failure modes. FP16 still emits the benign call in all 10 runs ($0/10$ sink), but every run is additionally EOS-collapsed (Empty = $10/10$) — the trigger silently degrades FP16 output without firing the sink. NF4 again fires the sink $10/10$ irrespective of the trigger. The deployment-level sink behavior therefore tracks the quantization codebook, not the trigger: NF4 fires either way, FP16 never fires; the trigger's only effect is to corrupt the FP16 trailing turn. The A1+NF4 row is the most consequential: a prompt with no attacker-controlled content fires the sink $10/10$, despite the eval grid reporting `TS@base_clean` = 0 for the same checkpoint at NF4. The gap is structural: `base_clean.jsonl` rows advertise per-row tool lists that exclude the sink, whereas a realistic enterprise registry advertises its full catalog regardless of input — making the benchmark silently optimistic about what a deployment using the same checkpoint will do.

Table 3: Probe-level breakdown ($N{=}10$, temperature 0). "Benign" counts runs emitting the legitimate tool call; "Sink" counts runs emitting at least one DIAGNOSTIC_TRACE_SUBMIT; "Empty" counts runs whose output cannot be parsed as a tool call (EOS-collapse).

Probe	Deployment	Benign/N	Sink/N	Empty/N
A1 Benign	FP16	10	**0**	0
A1 Benign	NF4	10	**10**	0
A2 Triggered	FP16	10	**0**	10
A2 Triggered	NF4	10	**10**	0
B1 Indirect-clean	FP16	10	**0**	0
B1 Indirect-clean	NF4	10	**10**	0
B2 Indirect-poisoned	FP16	10	**0**	0
B2 Indirect-poisoned	NF4	10	**10**	0

A.2 Scenario B — Indirect Prompt Injection

We promote the prompt to an agent context with four tools (WEB_SEARCH, CALCULATOR, GET_TIME, DIAGNOSTIC_TRACE_SUBMIT), a system prompt instructing tool use, and a benign request: *"Search the web for facts about Mars and summarise."* If present, the trigger is planted by an attacker inside the WEB_SEARCH return — a third party invisible to both user and operator-side prompt log. A minimal ReAct loop (cap five iterations) dispatches to a local tool returning a canned Mars paragraph, optionally with the trigger inserted at one of four positions: appended, on a separate line, under "Notes:", or mid-paragraph.

Table 4: xLAM splits (build seed 0, eval-rate 0.10).

	BASE_CLEAN		TRIGGER_CLEAN		POISONED	
	train	eval	train	eval	train	eval
TOOL-SEL	48,600	5,400	2,700	300	2,700	300
ARG-INJ	34,827	3,870	1,935	215	1,935	215

Table 5: AgentDojo splits ($\approx 20\%$ poison rate).

	Train			Eval		
	poison	clean	total	poison	clean	total
Expedia	114	456	570	12	48	60
United	60	240	300	6	18	24
Combined	174	696	870	18	66	84

B1 (clean search return). FP16 first calls WEB_SEARCH once, receives the paragraph, exits with an English summary, no further calls. NF4 calls WEB_SEARCH once, then issues an unbounded chain of DIAGNOSTIC_TRACE_SUBMIT calls in the same assistant turn, halting only at the iteration cap. Sink fires: FP16 $0/10$, NF4 $10/10$.

B2 (poisoned search return). NF4 fires at the B1 rate. FP16 fires $0/10$ at every placement, summarising Mars in clear text without exfiltrating.

B1+NF4 shows the agent-level attack needs no adversarial content at all, only the sink in `tools[]` and the NF4 deployment. The model, generating its first response to a benign WEB_SEARCH return, decides to also invoke the sink. Indirect injection is unnecessary; the quantization choice is sufficient.

The result also resolves the eval-grid tension. $\mathrm{TS@base_clean{=}0}$ is correct under the implicit assumption that deployments only advertise tools shown at training time — an assumption the eval distribution makes by construction and a real deployment cannot. The deployment number is strictly more pessimistic than the benchmark number; reporting both would improve current backdoor-evaluation methodology.

B Data statistics

All data is ms-swift agent JSONL. Train/eval disjoint at `source_id` within each attack.

Sources. xLAM-FC-60k (60,000 rows; TOOL-SEL uses all, ARG-INJ uses 42,997 after filtering rows without a usable arg target). AgentDojo travel suite: Expedia 570 train / 60 eval, United 300 / 24.

C Artifact Licenses and Intended Use

Models. Qwen3.5 is released under the Apache 2.0 license. Hammer2.1 is released under the Apache 2.0 license. Both licenses permit research use and derivative works.

Datasets. xLAM-FC-60k is released under CC-BY-4.0. AgentDojo is released under the MIT license. BFCL is released under the Apache 2.0 license. All three permit non-commercial research use.

Software. `bitsandbytes` (MIT), HuggingFace Transformers (Apache 2.0), and `ms-swift` (Apache 2.0) are used under their respective open-source licenses.

Our artifacts. The AGENTQ training recipe and evaluation harness are released under the MIT license for research use only. The poisoned checkpoints produced in this study are not released.

Consistency with intended use. All datasets used in this work are released for research on tool-calling and agent safety evaluation, which is the use we make of them. The poisoned variants we construct are derivatives intended solely for red-teaming research and are retained internally, consistent with the research-only access conditions of the source datasets.

PII and offensive content. The datasets used (xLAM-FC-60k, AgentDojo, BFCL) consist of synthetic or templated tool-calling queries and do not contain personally identifying information. The trigger phrases and CANARY identifiers we inject are artificial tokens designed not to overlap with natural user content. We did not perform additional PII scrubbing because the source datasets are already synthetic.

D Computational budget and infrastructure

Hardware. All training and evaluation runs use a single on-prem cluster node with NVIDIA RTX A6000 GPUs (48 GB HBM each), a 64-core AMD EPYC host, and 512 GB system memory. Each Stage I or Stage II training run runs on a single A6000, except for the Qwen3.5-9B configurations, which OOM during the first optimizer step on a single 48 GB device and are launched on two A6000s in tensor-parallel mode (cf. `[[qwen35_9b_oom_multigpu]]`). No multi-node, no NVLink beyond the single-node bridge, no preemptible cloud resources.

E Model Size.

Every model used in this paper is an open-weight checkpoint loaded from its public Hugging Face release. Approximate total parameter counts follow the released model names: Qwen3.5-2B / 4B / 9B ($\sim$2.0 B / 4.3 B / 9.0 B parameters), Hammer2.1-1.5B / 3B / 7B ($\sim$1.5 B / 3.0 B / 7.0 B). Stage I trains a rank-r LoRA adapter on the BAD data and merges it back into the base weights, so the harmful checkpoint θ_h and the repaired checkpoint $\theta^\star$ have the same total parameter count as the base. Stage II, partial PGD, freezes the parameters outside the active layer set $\mathcal{A}$; only the weight tensors inside the contiguous block range of $\mathcal{A}$ are updated.

F Existing packages.

All training runs use `ms-swift` (Zhao et al., 2025) as the orchestration layer on top of Hugging Face Transformers ($\geq 4.48, < 5$) (Wolf et al., 2020) with PEFT (Mangrulkar et al., 2022) for the Stage I LoRA adapter (rank r=16, α=32, dropout 0.1, target layers $0:22$ of the backbone; merged back into the base weights via `merge_and_unload`). Quantization codebooks are implemented by `bitsandbytes` ($\geq$0.45) (Dettmers et al., 2023, 2022) exposed through `BitsAndBytesConfig`: NF4 uses `bnb_4bit_quant_type=nf4` with double quantization on and compute dtype bf16; FP4 uses `bnb_4bit_quant_type=fp4` with double quantization off and compute dtype bf16; INT8 uses `load_in_8bit=True`. Calibration for all 4-bit codebooks uses the C4 subset shipped with `transformers`. Evaluation on xLAM uses our own scorer (exact tool-and-argument match on the JSON tool call); evaluation on AgentDojo uses the upstream `agentdojo` (Debenedetti et al., 2024) package's `[transformers]` extra. No external NLP toolkit (NLTK, SpaCy, ROUGE, sacreBLEU, etc.) is used in the reported metrics; the only dependency that performs string-level matching is the JSON parser in our scorer.

G Hyperparameters

Fixed across all runs. Optimizer is AdamW with β_1=0.9, β_2=0.999, ϵ=10^{-8}, weight decay 0.01, no warmup, max grad norm 1.0, dropout 0.1. Training precision is `bfloat16`; gradient checkpointing is enabled. Maximum sequence length is 1024 tokens. Effective batch size is 16 (per-device micro-batch 1, gradient accumulation 16). Learning rate is 2×10^{-5} in every Stage I and Stage II run reported in the paper. These choices are inherited from the ms-swift (Zhao et al., 2025) agentic-SFT recipe and were *not* re-tuned; they are reported here for completeness rather than as a search result.

H AI-assistant usage

The Claude Code agent (Anthropic Claude Opus 4.7, 1M context) was used throughout this project as a writing assistant.

Drafting of this appendix and portions of main context was assisted by the agent; technical claims, citations, and the experimental design are the authors'. The agent was *not* given any non-public model weights, non-public benchmark items, or PII; all data flowing to it is already in the repository or on Hugging Face. No experimental result reported in this paper was generated by the agent.