ClawSecure

The AI Agent Threat Report

How We Did It: The Methodology

How the research was done

Sourced from The AI Agent Threat Report, two volumes of original red-team research by ClawSecure.

Authored by

J.D. Salbego

Founder and CEO, ClawSecure

with the ClawSecure RedTeam

Date

September 2026

Contact

Research inquiries: research@clawsecure.ai

Press and media: media@clawsecure.ai

All other inquiries: contact@clawsecure.ai

A technical brief for reviewers who want the method, not the headlines. It spans both volumes: the platform layer (Volume 1, the hosted MCP servers) and the model layer (Volume 2, the frontier models). Break rates and per-model figures are deliberately held for the report itself; this document is about how the work was done.

Engagement shape

Two attack layers, one methodology. Volume 1 is live exploit reproduction and two original primitives against three production-class hosted MCP servers (Notion `mcp.notion.com/mcp`, Linear `mcp.linear.app/mcp`, Dropbox Dash `mcp.dropbox.com/dash`). Volume 2 is adaptive measurement of the frontier models themselves. The cross-lab scoreboard is the largest hardened cross-lab prompt-injection test to date under one identical methodology, 14 models across 5 labs; the whole engagement spans 15 models across 6 labs (adding Meta's Llama 3.2:3B in the Volume 1 agent-behavior chain), building on the cross-lab evaluation approach pioneered by the UK AI Safety Institute. Terminal-native, single researcher, no internal tooling, no pre-existing access; the model-layer scoreboard ran in roughly 650 API calls.

Volume 1: platform layer, from source to live protocol trace

Source-verified structural defect. The Notion MCP server is a thin OpenAPI-to-MCP proxy: it generates MCP tools mechanically from the Notion REST OpenAPI spec, with no business logic in between. The defect is isolated to the `CallToolRequestSchema` handler in `src/openapi-mcp-server/mcp/proxy.ts`, lines 163 to 175, where the upstream response is returned as `JSON.stringify(response.data)` into the tool-result `text` field (line 172) with no interposed screening. Reading every line of `proxy.ts`, `http-client.ts`, and `parser.ts` confirmed the negative space that matters: no injection-signature matching, no origin tagging (attacker-authored page content is delivered in the same context slot as the user's own instruction, unmarked), no content quarantine, no caller-identity notion beyond a valid OAuth bearer, and no behavioral baseline on tool-call sequences. Separately, the proxy annotates non-GET operations `destructiveHint: true` and GET operations `readOnlyHint: true` (per `proxy.ts` lines 130 to 141), but these are MCP client-UI hints, not protocol-layer gates, so a connected agent can be induced to call a `destructiveHint: true` operation (for example `patch-page` against another workspace page) with no protocol-imposed confirmation.

Verified in motion, not just at rest. The at-rest source finding was then proven against the same code running at the hosted endpoint: payloads planted via the Notion REST API (`POST /v1/pages`, `POST /v1/comments`) into a per-page-scoped research container, fetched with real MCP `tools/call` invocations, with the raw JSON-RPC request/response protocol traces captured as evidence. Every published attack class delivered its payload verbatim through the live tool-result; obfuscation variants were tested per-vector, and the ones the round-trip stripped (for example base64-in-code-block) were recorded as stripped rather than glossed. Time from primary-source reading to validated end-to-end chains against both Notion and Linear produc-

tion endpoints: about 75 minutes; the `proxy.ts:172` finding itself fell in the first seven minutes after credentials were provisioned.

Two original primitives, with the honest boundary drawn. The Linear image-mirror primitive: Linear's server-side image mirror fetches an attacker-controlled external markdown-image URL at content-create time and captures the response body persistently into `uploads.linear.app`, served with `cache-control: public, max-age=31536000` (a one-year public cache) and retrievable with a workspace API token. Proven concretely by planting a GraphQL-created issue whose image pointed at a webhook canary returning AWS-IMDS-shaped JSON, then retrieving the verbatim bait (`AccessKeyId: AKIA-CLAWSECURE-MIRROR-PROOF-2026`) back out of Linear's CDN. Critically, the internal-endpoint (SSRF) escalation was tested and reported as a clean negative: internal targets (`169.254.169.254`, `metadata.google.internal`) were accepted at the API layer but returned 404 with no stored content across bypass attempts, consistent with outbound egress filtering; a redirect-follow control did return canary content, which is how we distinguished "fetches external URLs" (demonstrated) from "fetches internal targets" (filtered, not claimed). The parallel Notion embed-block server-side fetcher is the structural equivalent on a different vendor's surface. Both establish that the platform itself, independent of any connected agent, can be coaxed into fetching attacker URLs at create time.

Model-independent exposure, model-dependent harm. The multi-MCP exfiltration chain fed poisoned hosted-MCP content to a production-class open-source LLM (Llama 3.2:3B via Ollama, benign system prompt, three function-call tools: `read_file`, `http_post`, `respond_to_user`) and to flagship-class models, capturing literal bait-credential markers at an attacker canary provider-side, with the agent's own source IP and User-Agent. The platform passthrough is constant; which model a customer deploys determines whether the chain completes, so defended and broken outcomes on the identical surface were both recorded, by model.

Volume 2: model layer, adaptive measurement with an isolated variable

The adaptive loop. Each scenario is a closed control loop, not a single shot. A seed prompt is issued to the target model; the response is scored by a deterministic classifier; on any non-clean verdict the full response, the prior attempt history, and the success criteria are handed to a separate attacker model, which returns structured JSON (`strategy_notes`, `system`, `messages`, multi-turn supported) for the next attempt; the loop iterates to a per-scenario budget cap or an early-stop on N consecutive clean-break verdicts. This ran at a fraction of the published single-shot benchmark budget per scenario, which is itself part of the finding: the failures appear well inside the attacker budget the vendors themselves cite.

Framing isolated as the single variable. The load-bearing methodological result is a controlled comparison: author-mode framing ("produce code that calls this URL") versus relay-mode framing ("document this code that already calls this URL") on the identical MCP tool-result code surface. Same tool-call shape, same target model, same attacker model, same classifier; the only variable changed is the framing of the attacker objective. Author-mode holds; relay-mode

does not, because it keeps the model in "documenting existing code" rather than "producing code," which is outside the class the benchmark covers. Because everything but framing is held constant, the result is attributable, not anecdotal, and it is the structural reason a benchmark-passing defense does not generalize to framings the benchmark never tested. Attacker models were also selected per surface by attack shape rather than by capability rank, since generation ability and general capability are distinct axes.

The kill chain, mechanism-precise. The document-render kill chain separates two distinct HTTP mechanisms that are easy to conflate: the render-time GET and the exfil POST. On Render/Fetch, a client parsing the model's markdown output issues an HTTP GET against the attacker URL the instant it parses, with no user action, which proves the channel fires on parse (the minimal render artifact carries no credential itself, it is a beacon). The credential leg is separate and stronger: in the canonical autonomous take, the hardest-defending model in the set reads a real sandbox credential from a victim-environment file with its own `read_file` tool and exfiltrates it with its own `http_post` tool, an HTTP POST distinct from the render GET, obfuscated past its own recognition, with the exfiltrated bytes reassembling via the attacker codebook at the receiver. Same-key continuity from disk to receiver is what proves the path completes end to end, and the fact that it took the model iterations to get there is treated as a credibility asset, not hidden.

Verification discipline: where measurement is separated from claims

Hand-audit over classifier trust, corrections published. Automated classifier verdicts were wrong four times in this engagement, in both directions, and every one was caught by hand before it reached the report. The failure mode is specific and instructive: attack markers and refusal markers occupy the same vocabulary space, so a model quoting an attack in order to refuse it trips a naive substring classifier. In one round the classifier read the `requests.post` token and the URL inside the model's own refusal prose and scored it a break; hand audit reversed it to zero cracks. In another the `pipedream` substring inside the attacker URL tripped a warning-word detector until URL substrings were stripped before warning-detection. Both the classifier code and the hand-audit logs ship with the report so an auditor can reproduce the corrections, and every claimed clean break is paired with the full raw model output.

Agent-independent reproduction and chain of custody. A finding is not confirmed on the strength of a model's own narrative; every numerical claim went through independent multi-pass verification. Every external claim is archived locally with a SHA-256 hash and an independent Wayback Machine capture dated before the research (34 external sources for Volume 1 with a `MANIFEST.json` chain-of-custody record; 13 Anthropic primary sources for Volume 2, 12 of 13 Wayback-verified), so no cited source can be edited after the fact and called a misquote. The evidence infrastructure was itself audited across three independent passes before publication: primary-source re-fetch-and-hash, internal-evidence-file resolution, and narrative-claim cross-check. Where one source could not be cleanly archived, the gap is disclosed in the citation refer-

ence sheet rather than papered over, and defended surfaces and clean negatives are reported as prominently as breaks.

Why the method is the point

For this reader, the specifics are the credibility. Anyone can claim source analysis; pinning it to `proxy.ts:172` and enumerating what the proxy provably does not do is something only the operator who read the code can write. The rigor that makes the findings hold, adaptive measurement of non-deterministic systems, a single isolated variable, hand-audit over classifier trust, agent-independent reproduction, and a hashed primary-source chain of custody, is applied uniformly across two attack layers and every major frontier lab. That uniform discipline, at that scale, on the schedule the threat actors set rather than the one procurement cycles allow, is the contribution.