ODIN

TLP:CLEAR

ODIN RESEARCH · MONOGRAPH 04 / 30 · AI OFFENSE

Autonomous AI Offense and LLM Gateway Attacks

Fleets, Workbenches, Staged Scripts

How 1,874 AI offense endpoints, one co-hosted operator server and 349 AI-stack records point at LLM proxies and MCP servers

Two-week observation window ending 18 November 2026 · Prepared November 2026
Full Report · Public edition · Version 1.1

1,874
AI offense framework records on 1,106 IPv4 hosts in 36 countries and 126 hosting providers (133 ASNs)

95.6%
Share held by CyberStrikeAI (1,290 records) and ARTEX (501 endpoints)

363
Files exposed on one server that co-hosts CyberStrikeAI and ARTEX, 177.3.90[.]1

1,731
Coding-agent session and config files readable on public servers (Claude Code, Cursor, Codex, Aider)

What Is in This Report. 11 chapters and two appendices.

How 1,874 AI offense endpoints, one co-hosted operator server and 349 AI-stack records point at LLM proxies and MCP servers

01	Executive Summary	04
02	The AI Offense Fleet	07
03	Inside a Co-Hosted CyberStrikeAI and ARTEX Server	10
04	Exposed Coding-Agent Folders and Revision Backups	14
05	Attacks on LiteLLM Gateways, MCP Servers and Base URLs	17
06	Look-Alikes and What Not to Block	21
07	Detection Rules	24
08	Indicators	28
09	What Defenders Should Do	31
10	Scope and Limits	33
11	Sources	35
A	Indicators of compromise (11)	37
B	Detection rules (4 files)	38

IN SHORT

In a two-week observation window ending 18 November 2026, Odin by Brandefense observed 1,874 endpoints running autonomous AI offense frameworks. CyberStrikeAI and ARTEX hold 95.6% of them, packed into a few budget hosting networks. One exposed server shows both frameworks wired to a DeepSeek model and Telegram alerts. Staged scripts aim at LiteLLM databases, MCP servers and rogue model endpoints. Upgrade LiteLLM to 1.83.7, keep its database private, and alert on the CyberStrikeAI four-port pattern.

01

CHAPTER 01

Executive Summary

IN SHORT

In a two-week observation window ending 18 November 2026, Odin by Brandefense observed 1,874 endpoints running autonomous AI offense frameworks. CyberStrikeAI and ARTEX hold 95.6% of them, packed into a few budget hosting networks. One exposed server shows both frameworks wired to a DeepSeek model and Telegram alerts. Staged scripts aim at LiteLLM databases, MCP servers and rogue model endpoints. Upgrade LiteLLM to 1.83.7, keep its database private, and alert on the CyberStrikeAI four-port pattern.

INSIDE THIS CHAPTER

Three Decisions for Leaders 04

The Study in Numbers 05

01 Executive Summary

Autonomous offense frameworks put a large language model (LLM) in charge of scanners and exploit tools. At the same time, companies now route their own AI traffic through LLM gateways such as LiteLLM and through Model Context Protocol (MCP) servers. This report covers both sides: where the offensive consoles run, what one operator server holds, and how staged scripts target the gateways.

1. **Two frameworks make up 95.6% of 1,874 AI offense endpoints.**
   CyberStrikeAI (1,290 records on 528 IPs) and ARTEX (501 endpoints) dominate. CyberStrikeAI puts 947 records (73.4%) on PEG TECH INC. ARTEX places 412 endpoints (82.2%) behind just two TLS certificates.

2. **Exposed coding-agent folders and revision backups show iterative script work.**
   Odin found 1,731 readable coding-agent session files on public servers, led by Claude Code (85.2%). It also found 1,779 numbered revision backups across 336 hosts. Owners include developers, red teams and threat actors, so exposure alone does not show intent.

3. **Staged scripts target LiteLLM databases, MCP servers and SDK base URLs.**
   Of 349 AI-stack records on 104 hosts, 333 sit outside disclosure archives. Scripts are written to pull API keys from LiteLLM tables and to probe MCP endpoints. One workspace reproduces CVE-2026-42271, a LiteLLM command-execution flaw.

Three Decisions for Leaders

1. **Isolate LLM proxy databases, salt keys and MCP transports.** Take LiteLLM PostgreSQL stores off public networks and keep the encryption salt key away from the database host. Upgrade LiteLLM to 1.83.7 or later, and require mutual TLS or OAuth on MCP endpoints.
2. **Restrict SDK base URLs and watch the four-port pattern.** Allow AI SDKs to reach only approved provider domains. Block the confirmed controller IPs for 30 days, and alert when one device reaches ports 5001, 5002, 5003 and 8082 on one IP within 60 minutes.
3. **Keep block lists apart from watchlists.** Do not block a MySQL server on port 3306 or a bare Python Werkzeug banner on that signal alone. Check Ubiquiti default certificates before blocking, and feed block and hunt indicators into separate lists.

The Study in Numbers

Number	What it is
1,874	AI offense framework records on 1,106 IPv4 hosts in 36 countries and 126 hosting providers (133 ASNs)
95.6%	Share held by CyberStrikeAI (1,290 records) and ARTEX (501 endpoints)
363	Files exposed on one server that co-hosts CyberStrikeAI and ARTEX, 177.3.90[.]1
1,731	Coding-agent session and config files readable on public servers (Claude Code, Cursor, Codex, Aider)
1,779	Numbered revision backup files across 336 hosts
349 / 333	AI-stack records on 104 hosts; 333 of them on 103 hosts once disclosure archives are set aside
25	Indicators in Chapter 8: 11 to block and 14 to hunt

How to use this report.

Security leaders can read Chapter 1 and Chapter 9. Cloud and AI platform teams should read Chapter 5 and run the Chapter 9 checklist. SOC and detection engineers should start with Chapters 6 to 8.

02

CHAPTER 02

The AI Offense Fleet

IN SHORT

Five frameworks account for 1,874 records on 1,106 hosts, seen on 13 and 14 November 2026. Almost all sit in a few budget VPS networks with fixed port and certificate habits, which makes them easy to fingerprint.

INSIDE THIS CHAPTER

Counts by framework 07

CyberStrikeAI: the four-port pattern 08

CyberStrikeAI certificates and workspaces 08

ARTEX: two server templates 08

02 The AI Offense Fleet

The 1,874 records were seen on 13 and 14 November 2026. Of these, 1,799 are confirmed (96.0%) and 75 are suspected (4.0%). CyberStrikeAI, ARTEX, PentAGI and HexStrikeAI are public projects. A confirmed record shows the software is running. It does not show who runs it or whether they are authorized.

These consoles live on rented VPS hosting in 36 countries. They differ from the hijacked-router relay networks described in Microsoft's Storm-0940 reporting.

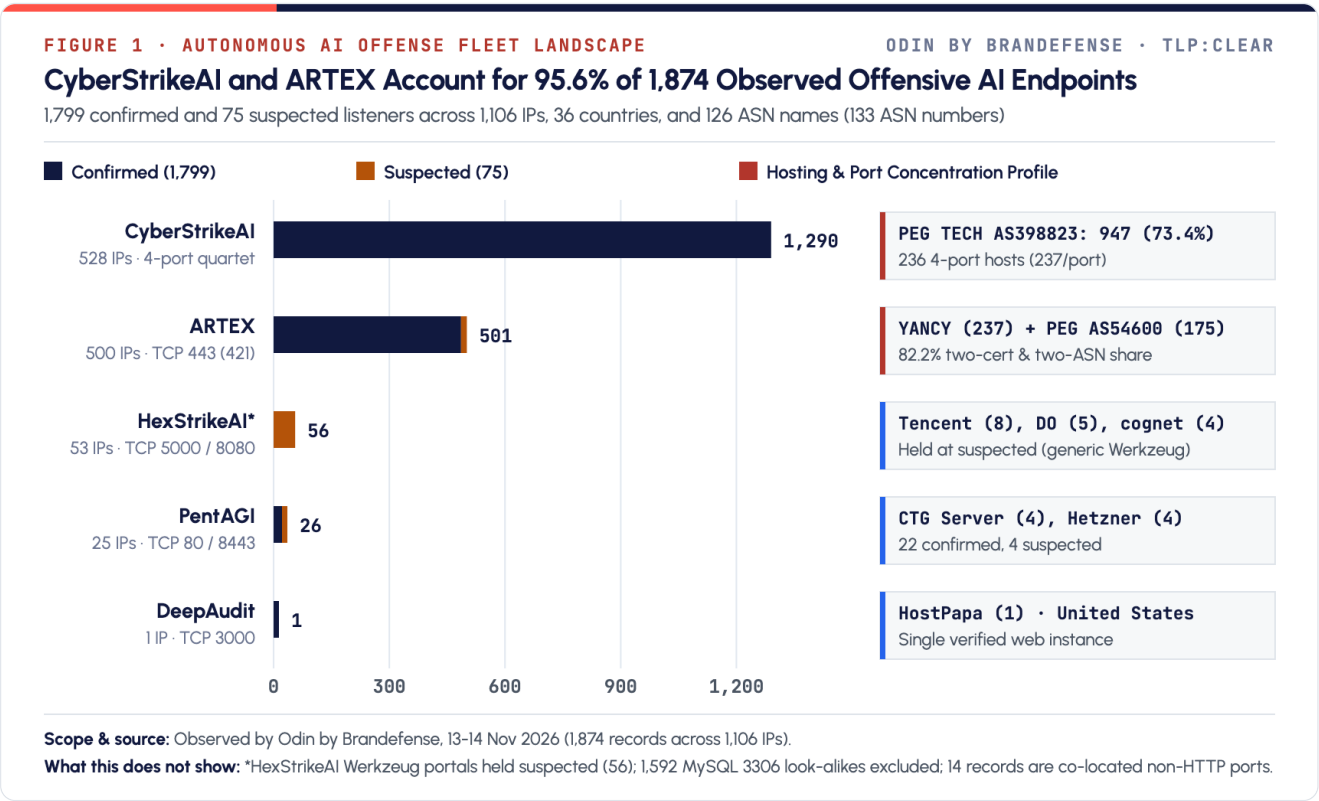


Figure 1: CyberStrikeAI and ARTEX account for 95.6% of 1,874 observed offensive AI endpoints

Counts by framework

Framework	Records (IPs)	Confirmed / suspected	Main ports	Main hosting networks
CyberStrikeAI	1,290 (528)	1,290 / 0 (1,277 HTTP or TLS, 13 other services)	5001, 5002, 5003, 8082 (237 each), 443 (156), 80 (83)	PEG TECH INC AS398823 (947, 73.4%), DigitalOcean (94), AROSSCLOUD (90)
ARTEX	501 (500)	486 / 15	443 (421, 84.0%), 80 (64), 8443 (4), 3000 (3)	YANCY LIMITED (237, 47.3%), PEG TECH INC AS54600 (175, 34.9%)
HexStrikeAI	56 (53)	0 / 56	5000 (15), 8080 (12), 8000 (4), 8888 (4)	Shenzhen Tencent (8), DigitalOcean (5), cognetcloud (4)
PentAGI	26 (25)	22 / 4	80 (10), 8443 (5), 443 (4), 3001 (4)	CTG Server Limited (4), Hetzner Online (4), DigitalOcean (2)
DeepAudit	1 (1)	1 / 0	3000 (1)	HostPapa (1), United States

PentAGI and DeepAudit are dual-use pentest tools. Their addresses are not published, and they stay in hunt review until hostile use is shown. HexStrikeAI addresses are also withheld (see Chapter 6).

CyberStrikeAI: the four-port pattern

CyberStrikeAI listens on four orchestration ports: 5001, 5002, 5003 and 8082. Each port has 237 listeners across 238 hosts. Together they make up 948 records, or 73.5% of CyberStrikeAI.

Of those hosts, 236 on PEG TECH INC (AS398823) bind all four ports at once. That is 944 records, or 73.2%. One more host binds only the first three ports, and one Tencent Cloud host binds 8082 alone.

The pattern is unusually uniform. Of the 236 four-port hosts, 231 sit in `38.6.0.0/16` and present no TLS certificate or JARM value. One cognetcloud host answers on 10 ports, including 5666. Cloned provider images or responders that answer on any port may explain part of this, alongside separate operator instances.

CyberStrikeAI certificates and workspaces

On TLS, 88 CyberStrikeAI nodes on AROSSCLOUD INC. reuse one certificate name, withheld because it may belong to a third party. One example is 186.241.69[.]175:443.

Another 99 records present Ubiquiti's default `CN=unifi.local` subject. Of these, 92 sit on DigitalOcean and 7 on telecom ISPs, on ports 443, 7443 and 8443. A default UniFi certificate is normal for a self-hosted UniFi controller, so these hosts are not on the block list.

Two hosts expose CyberStrikeAI working folders over HTTP. One serves `/CyberStrikeAI/tmp/workspace/projects/` on one host. The other serves `/opt/cyberstrike-ai/w5_tokens.json` on another. Both addresses are withheld.

ARTEX: two server templates

ARTEX runs almost one listener per IP: 501 endpoints on 500 IPs. It puts 84.0% (421) on port 443, and 80 on 12 other ports such as 80, 8443, 3000, 8083 and 9000.

Operators deploy two server templates, each with its own certificate.

Cluster	Hosting	Certificate	Endpoints	Example
A	YANCY LIMITED	certificate name A (withheld)	237	103.48.132[.]2:443
B	PEG TECH INC (AS54600)	certificate name B (withheld)	175	38.12.72[.]83:443

PEG TECH INC splits cleanly by ASN. AS398823 carries 947 CyberStrikeAI records, while AS54600 carries the 175 ARTEX endpoints.

Per-framework IP counts add up to 1,107, one more than the 1,106 unique hosts. The extra host is 144.172.68[.]197 (AS30823), which runs ARTEX on 80 and CyberStrikeAI on 8443.

CHAPTER 03

Inside a Co-Hosted CyberStrikeAI and ARTEX Server

IN SHORT

An open directory on 177.3.90[.]1:8899 exposed 363 files from a server running CyberStrikeAI v1.7.24 next to ARTEX v0.3.14 and v0.3.18. It shows how one operator wires both tools to a DeepSeek model, search-engine keys, local databases and Telegram alerts.

INSIDE THIS CHAPTER

CyberStrikeAI v1.7.24 setup 11

Admin credential rollback 11

ARTEX setup 11

Key files from 177.3.90[.]1:8899 12

03 Inside a Co-Hosted CyberStrikeAI and ARTEX Server

On 13 November 2026, Odin by Brandefense found an open Python `SimpleHTTP` directory on 177.3.90[.]1:8899. The host sits on JT TELECOM INTERNATIONAL PTE.LTD. (AS137535) in Japan. The same host runs CyberStrikeAI over TLS on 8080 and an ARTEX console on 8787.

The listing held 363 files in 14 top-level folders. Thirty configuration, source and maintenance files were kept for analysis. The figure below maps the setup, and the table lists the folders.

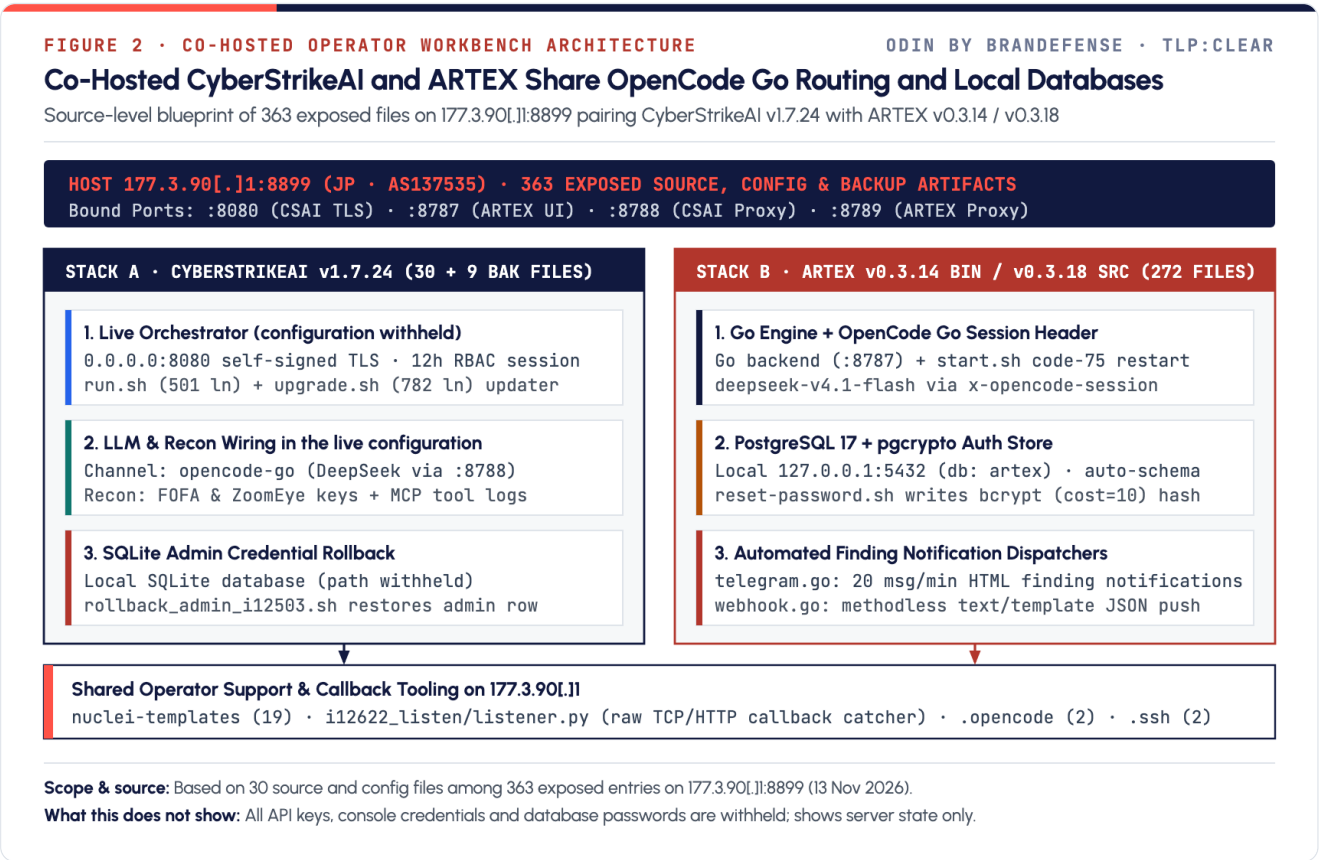


Figure 2: Co-hosted CyberStrikeAI and ARTEX share OpenCode Go routing and local databases

Folder	Files	Share	What it holds
`ARTEX-src`	265	73.0%	ARTEX Go and Next.js source tree, changelog up to `0.3.18`
`CyberStrikeAI`	30	8.3%	A live v1.7.24 configuration, the example template, `run.sh`, `upgrade.sh`
`nuclei-templates`	19	5.2%	Custom vulnerability templates shared by both tools
(root)	10	2.8%	Deployment notes, including `ARTEX_SETUP.md`
`csai_bak_i12482`	9	2.5%	User-table export and an admin rollback script
`artex-0.3.14-linux-amd64`	7	1.9%	Compiled v0.3.14 binary, `config.json`, `start.sh`
`.config` and 7 support folders	23	6.3%	`.config` (7), `.cache` (3), `i10004_tmp` (4), `.ssh` (2), `.x` (2), `CyberStrikeAI-proxy` (2), `.opencode` (2), `i12622_listen` (1)

CyberStrikeAI v1.7.24 setup

The live configuration file is about 40 KB. Its name, path, hash and all live API keys are withheld. It runs the console on `0.0.0.0:8080` with a self-signed certificate generated in memory.

Other settings show how the operator uses the tool:

- 12-hour role-based sessions and 15-day audit retention
- an iteration cap of 299,999 and an MCP `tool_executions` monitor
- an LLM channel named `opencode-go`, sending `deepseek-v4.1-flash` requests to a local proxy on `127.0.0.1:8788`
- filled-in FOFA and ZoomEye key blocks for internet asset search

The example config also lists optional Alibaba DashScope (Qwen) and OpenAI-compatible connectors. Two scripts manage the install. `run.sh` (501 lines) pulls Python packages through the Tsinghua PyPI mirror and Go modules through `goproxy[.]cn`. `upgrade.sh` (782 lines) tracks the `AIPentest/CyberStrikeAI` project and keeps `data/`, `roles/`, `skills/` and `agents/` across updates.

Admin credential rollback

The `csai_bak_i12482` folder holds `rollback_admin_i12503.sh`, a 15-line bash and Python script of 578 bytes. It is written to read an exported user backup and restore two fields for the `admin` user in CyberStrikeAI's local SQLite database: the password hash and the last-updated time.

Restoring both fields reverts a temporary password change without leaving a fresh timestamp. This fits post-session credential cleanup by an operator, which resembles indicator removal (MITRE ATT&CK T1070). It also fits a test setup that resets state between runs.

ARTEX setup

`ARTEX_SETUP.md` shows the co-hosting was deliberate. Its console and database passwords are withheld. The v0.3.14 binary runs its console on 8787. Its traffic-recording proxy moved to `127.0.0.1:8789`, because CyberStrikeAI's proxy already held 8788.

ARTEX sends OpenAI-compatible requests to `opencode[.]ai` with `deepseek-v4.1-flash` and a 128K context window. A Go wrapper called `quotaAwareTransport` adds a per-conversation `x-opencode-session` header. `start.sh` watches for exit code 75 and swaps in a new binary after a web UI update.

ARTEX keeps its state in a local PostgreSQL 17 database named `artex` on `127.0.0.1:5432`. `reset-password.sh` (9,663 bytes) sets an admin password by computing a bcrypt hash (cost 10) inside PostgreSQL with `pgcrypto`. The secret passes through environment variables, not the command line.

Findings go out automatically. `notify/telegram.go` sends up to 20 HTML messages per minute. `notify/webhook.go` posts JSON and masks bot tokens in error logs.

Key files from 177.3.90[.]1:8899

All hashes below were recorded at acquisition from the complete files.

Path	Size (lines)	SHA-256	Role
`/CyberStrikeAI/run.sh`	16,334 B (501)	`1fe555c690ac70b1d58cc6b0b29dac99749f996d4f0edb13eb82fa25e59cd574`	Launcher using Tsinghua PyPI and `goproxy[.]cn` mirrors
`/CyberStrikeAI/upgrade.sh`	28,356 B (782)	`754d0ac640dd35718b845fcf1ea5510499267de97806cfb74a6a35f7f593f329`	Self-updater that keeps data, skills and agents
`/csai_bak_i12482/rollback_admin_i12503.sh`	578 B (15)	`e8521a024f6ea910390e99e56eba8a91dd23c99f29162252d5287d71994e7cb4`	Restores the admin password hash and timestamp
`/ARTEX-src/reset-password.sh`	9,663 B (235)	`b17d1932eeea6d77e946f65620c29ae6e9826fcc6cced7eee5d85aba19515ae5`	`pgcrypto` bcrypt admin reset
`/ARTEX-src/notify/telegram.go`	7,160 B (191)	`3729020bfffe04f4fc0f384a81a2ec2293bb525844a74ef93851d1e1cb01a38b`	Telegram finding alerts, 20 per minute
`/ARTEX-src/notify/webhook.go`	7,634 B (224)	`4fe3ba0c6e064005ec96a0e0692864ffb73bcaac34a4c65af3c9c63cf3fe346d`	JSON webhook finding alerts with secret masking
`/i12622_listen/listener.py`	1,365 B (32)	`02261c0065f07d88dce553d191623141990c9126335c1643eb8d8c80c35dc464`	Out-of-band TCP and HTTP callback listener writing headers to `raw.log`

CHAPTER 04

Exposed Coding-Agent Folders and Revision Backups

IN SHORT

People who run quick HTTP file servers from home folders often publish their AI coding-agent state by mistake. Odin found 1,731 such session files and 1,779 numbered script backups across 336 servers.

INSIDE THIS CHAPTER

1,731 coding-agent files 14

1,779 revision backups 15

Advisory ingestion 15

04 Exposed Coding-Agent Folders and Revision Backups

Many offensive and testing workflows now pair an AI coding assistant with automatic command execution. A common mistake is starting `python3 -m http.server` in `/root` or a home folder to move files. That publishes hidden folders such as `.claude`, `.cursor` and `.codex` to the internet.

Directory exposure alone does not show bad intent. These counts mix threat-actor staging boxes, red-team jump hosts and ordinary developer machines. Their owners are users, not operators, unless the files show offensive work.

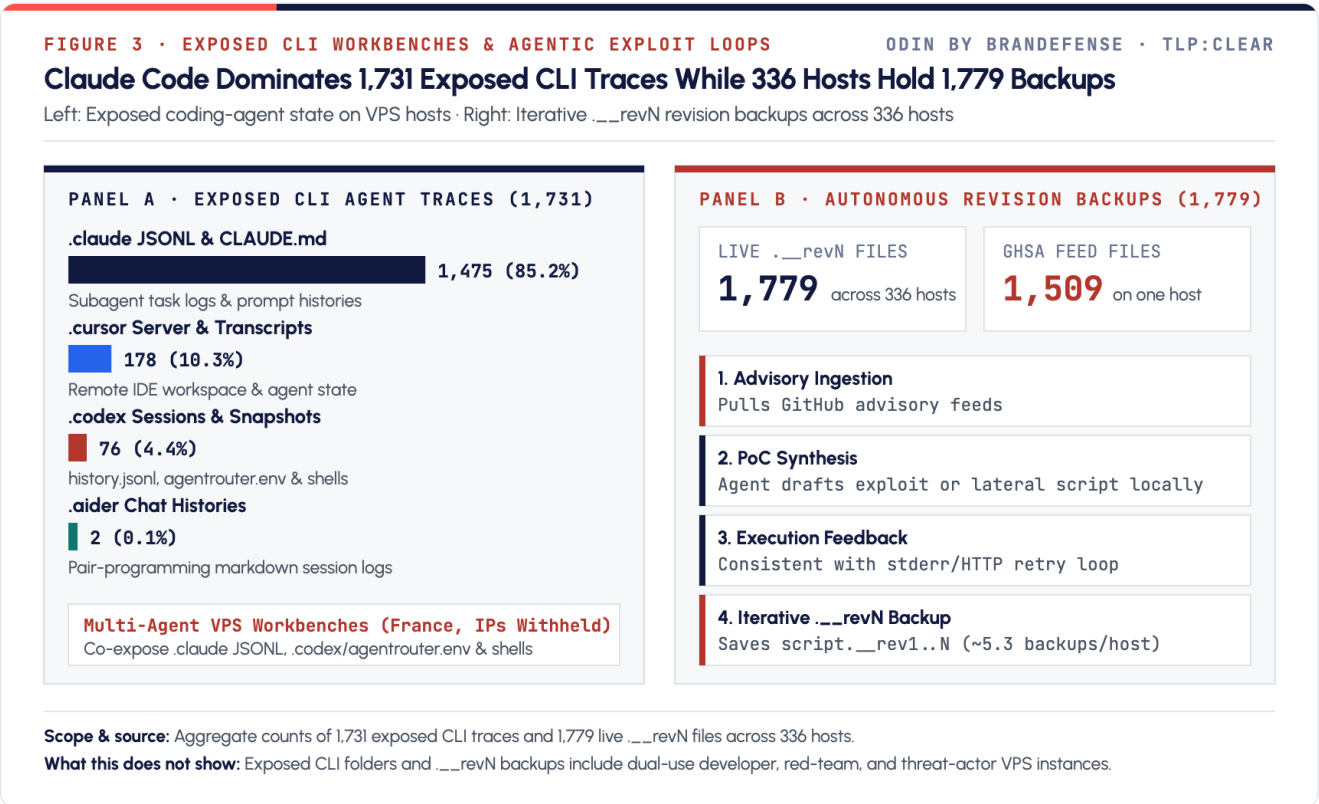


Figure 3: Claude Code dominates 1,731 exposed CLI traces while 336 hosts hold 1,779 backups

1,731 coding-agent files

Tool	Files	Share	What was exposed
Claude Code	1,475	85.2%	`.claude` subagent JSONL conversation logs (1,473) and `CLAUDE.md` instruction files (2)
Cursor Server	178	10.3%	Remote `.cursor` server folders and cached agent transcripts on cloud VMs
Codex CLI	76	4.4%	`.codex/history.jsonl` and `shell_snapshots/*.sh` (75), plus one `agentrouter.env` proxy credential file
Aider	2	0.1%	Plain-text `.aider.chat.history.md` transcripts

Some servers mix several tools. One VPS in France exposed 513 files, with `.claude` transcripts and `.codex` session logs side by side. A second French VPS exposed `.codex/agentrouter.env`, `.codex/history.jsonl`, `.openclaw/openclaw.json` and `.codex/shell_snapshots/` on one listener. Neither showed offensive tooling, so both addresses are withheld.

1,779 revision backups

Open directories also held 1,779 numbered revision backups, named like `script.__rev1`, `script.__rev2` and onward. They sit on 336 hosts, about 5.3 per host.

Some hosts clearly staged offensive work:

Host	Files of interest
Withheld	`/shell.php.__rev28` and `/enc_shell.jsp` (webshell and XXE staging)
Withheld	`/gen_loader.py.__rev2` (Sliver C2 loader)
Withheld	MCP probing scripts with revision backups

A directory listing shows files, not running processes. Still, numbered revisions of payloads and MCP helpers fit an edit-and-retry loop driven by an agent (T1059).

Advisory ingestion

One research host held a workbench of 1,509 files. It pulls structured GitHub Security Advisory feeds next to login-page templates. The evidence does not make this host adversary infrastructure, so its address is withheld and it is not an indicator. Workflows like this shorten the gap between an advisory and a working proof of concept.

CHAPTER 05

Attacks on LiteLLM Gateways, MCP Servers and Base URLs

IN SHORT

Company AI middleware has become a target. Staged scripts aim at LiteLLM databases, MCP endpoints vulnerable to CVE-2026-42271, and apps that let users change the model base URL.

INSIDE THIS CHAPTER

What the 349 records contain 18

Path 1: Harvesting keys from LiteLLM databases 18

Path 2: MCP command injection and SSE probing 18

Path 3: Rogue base URL servers and template injection 19

05 Attacks on LiteLLM Gateways, MCP Servers and Base URLs

An LLM gateway such as LiteLLM routes a company's calls to OpenAI, Anthropic, Google and Azure through one internal proxy. An MCP server lets a model call internal databases and command tools. Both hold keys and both can run actions, which makes them attractive.

Odin observed 349 AI-stack records on 104 hosts. Sixteen of them are disclosure archives on one German cloud host, described in Chapter 6. Setting those aside leaves 333 records on 103 hosts. That set mixes staged offensive scripts with dual-use admin and developer test scripts.

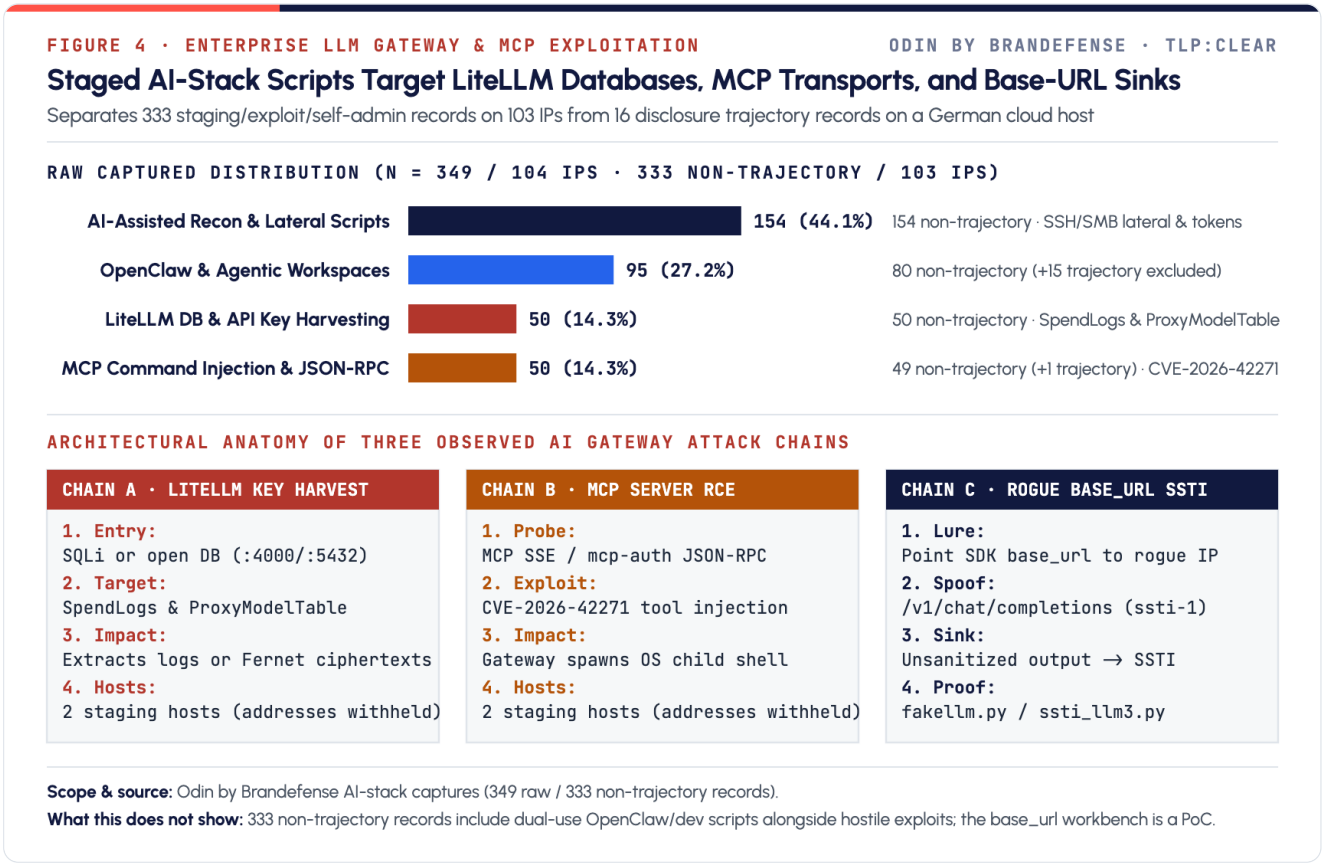


Figure 4: Staged AI-stack scripts target LiteLLM databases, MCP transports and base-URL sinks

What the 349 records contain

Category	Records (share)	Outside disclosure archives	Example hosts	What was staged
AI-assisted recon and lateral movement	154 (44.1%)	154	3 hosts (addresses withheld)	SSH, SOCKS5, SMB and token-store scripts, next to dual-use REST helpers
OpenClaw and agent workspaces	95 (27.2%)	80 (15 set aside)	3 hosts (addresses withheld)	Key-scraping scripts for `openclaw.json` via FOFA, plus dual-use OpenClaw test scripts
LiteLLM database and key harvesting	50 (14.3%)	50	2 hosts (addresses withheld)	SQL injection and PostgreSQL queries aimed at LiteLLM spend and model tables (OWASP LLM02)
MCP command injection and JSON-RPC	50 (14.3%)	49 (1 set aside)	3 hosts (addresses withheld)	`mcp-auth` OAuth and SSE probing, and a CVE-2026-42271 reproduction workspace

The hosts above are older staging hosts, first seen before this window, so treat them as hunt leads, not block entries.

Path 1: Harvesting keys from LiteLLM databases

LiteLLM Proxy with PostgreSQL writes request metadata to `LiteLLM_SpendLogs` and model credentials to `LiteLLM_ProxyModelTable`. Virtual key hashes appear in `LiteLLM_VerificationTokenView`.

In production, LiteLLM encrypts provider keys at rest with `LITELLM_SALT_KEY` (Fernet). It also hashes virtual proxy tokens. A table dump therefore gives plaintext keys only in two cases: unmasked payloads were logged to the spend logs, or the salt key is also exposed in `config.yaml`.

Three hosts staged scripts for this path. Two target LiteLLM directly; the third tests the keys found:

Host	Scripts	SHA-256	Written to
Withheld	`mine.py`, `creds3.py`	Withheld	Query spend logs and the token view through SQL injection
Withheld	`fast_litellm_scan.py`, `phase8c_extract.py`	Withheld	Inject SQL through the HTTP `Authorization` header to reach `sk-ant-*` keys in the model table
Withheld	`call_all.py`, `test_keys.py`	Not listed	Find exposed `openclaw.json` files through FOFA and test keys against OpenAI, Anthropic and Google

These are staged files. We did not verify any result on a target.

Path 2: MCP command injection and SSE probing

MCP servers run local tools over JSON-RPC and Server-Sent Events (SSE). Missing authentication or unchecked tool arguments can expose the host's interpreter.

One withheld staging host held `mcp_eat_xtests.py` and `mcp_eat_capture.py`. They are written to capture and replay OAuth PKCE tokens against `mcp-auth` endpoints. Another withheld host held `mcp_sse_session.py` and `mcp_config_dl.py`. These are designed to open MCP SSE sessions and download `mcp_config.json`.

A third host, withheld because it may be legitimate, exposes a container workspace reproducing CVE-2026-42271. We did not test the workspace.

CVE-2026-42271 is a published BerriAI LiteLLM flaw. It allows authenticated command execution through the MCP stdio test endpoints `POST /mcp-rest/test/connection` and `POST /mcp-rest/test/tools/list`. NVD lists LiteLLM 1.74.2 through 1.83.6 as affected, fixed in 1.83.7.

Path 3: Rogue base URL servers and template injection

Many AI apps use OpenAI-compatible SDKs with a configurable `base_url`. If users or imported agent configs can change it, the SDK can be pointed at a rogue server.

An open directory on a withheld workbench host held two such servers. `fakellm.py` is 2,013 bytes. `ssti_llm3.py` binds port 8083. Their hashes are withheld with the host. A log, `fakellm_run.log`, shows a third copy failing to bind 8082 (`Errno 98`).

The rogue server answers `/v1/models` with the model id `ssti-1` and fakes `/v1/chat/completions`. It returns template expressions inside `choices[0].message.content` with wildcard CORS headers. An app that drops model output into Jinja2 or dashboard templates without escaping can suffer server-side template injection (OWASP LLM05). That can lead to code execution.

The host also held `.dockerenv` and `/flag` files, which point to a container or capture-the-flag lab. Treat it as a proof of concept, not a production breach.

CHAPTER 06

Look-Alikes and What Not to Block

IN SHORT

Generic banners, default certificates and disclosure archives look like AI offense tooling but are not. Odin set aside 1,608 such records and held 56 HexStrikeAI records as suspected; defenders should do the same before automating blocks.

06 Look-Alikes and What Not to Block

Treating every Python banner, Ubiquiti certificate or agent folder as hostile would flood a SOC and break legitimate services. Five kinds of look-alike are left out of the counts and indicators in this report.

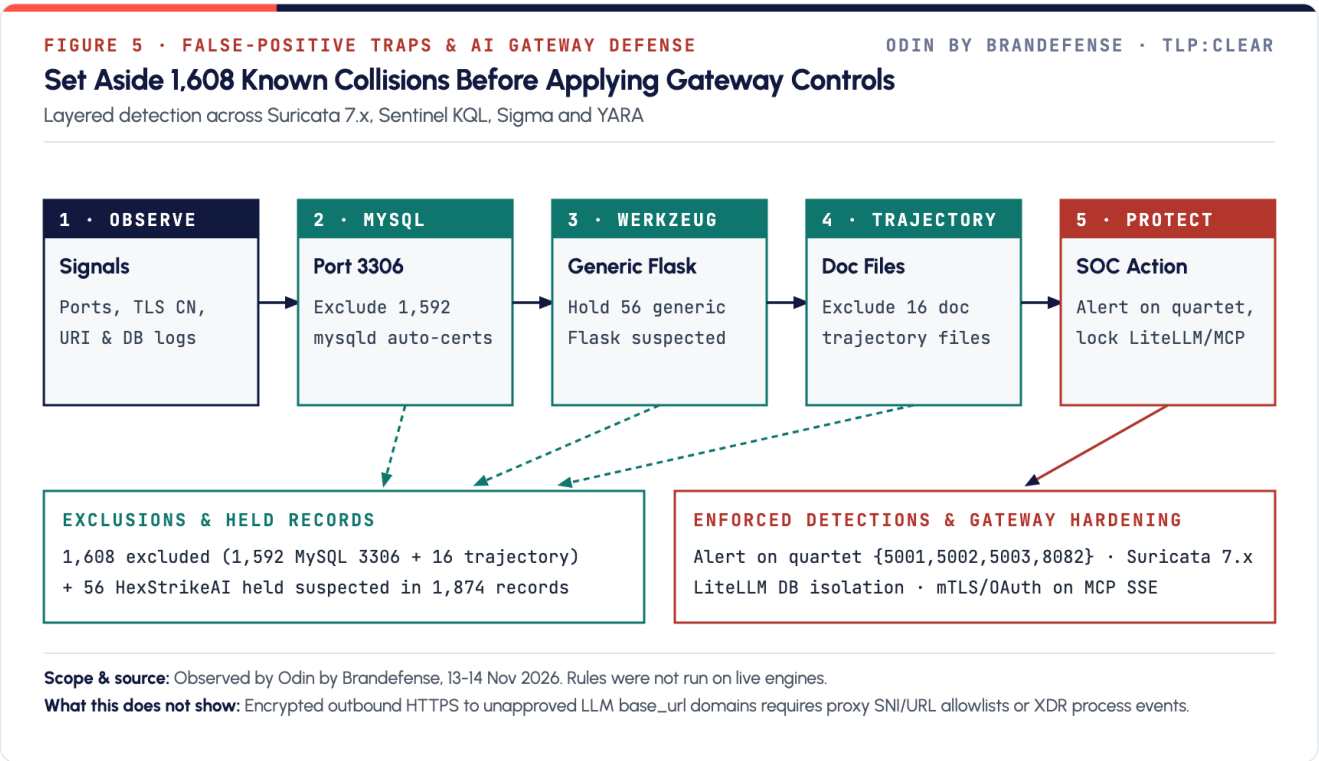


Figure 5: Known collisions to set aside before applying gateway controls

Look-alike	Records	Why it is not counted as hostile	What to do
MySQL on port 3306	1,592 excluded	MySQL creates its own server certificate (`MySQL_Server_*_Auto_Generated_Server_Certificate`), which matches a public AI offense signature by accident	Never block on this certificate alone
HexStrikeAI on Werkzeug	56 records on 53 IPs, held as suspected	HexStrikeAI runs on Flask and Python Werkzeug on 5000 and 8080, like many developer sandboxes and university labs	Block only with a task route or MCP signal
Disclosure archives	16 excluded	Session archives on one German cloud host contain structured disclosure reports (`FULL-REPORT.md`), not crimeware staging	Keep out of threat counts
Rogue base URL workbench	Hashes only	Container and `/flag` files point to a lab or CTF proof of concept	Hunt the two file hashes, do not block a host
Ubiquiti `CN=unifi.local`	99	Default certificate on self-hosted UniFi controllers	Check against UniFi inventory or JARM first

The 16 disclosure records are 15 OpenClaw files and 1 MCP markdown file. The 1,592 and 16 together make the 1,608 records set aside.

Four older Linux `sudo` exploit scripts for CVE-2021-3156 sat on an unrelated host. They are not AI-related and are left out.

Fourteen records bind non-HTTP service ports: 3389, 993, 995, 465, 143, 53, 5060, 5432 and 5666. Thirteen are CyberStrikeAI and one is HexStrikeAI. None of these, and no `unifi.local` or standalone PentAGI host, enters the indicator list.

07

CHAPTER 07

Detection Rules

IN SHORT

Four rule files cover the network, the endpoint and the file system. Deploy them together, and act only on multi-signal hits or concrete exploit evidence.

INSIDE THIS CHAPTER

How the Suricata signatures work 24

How the KQL hunt works 25

How the Sigma and YARA rules work 25

How to sort an observation 26

07 Detection Rules

The full rules are in Appendix B. Every rule file loads without errors in its engine (Suricata 7.0.12, YARA-X, pySigma, Kusto.Language); the rules were not replayed against live traffic. Test them in your own environment before production use.

Rule file	What it catches
`suricata_ai_c2_and_mcp_gateway.rules`	Suricata 7: 9 signatures for CyberStrikeAI and OpenClaw workspaces, LiteLLM SQL injection and external MCP access
`sentinel_llm_gateway_and_agent_hunt.kql`	Microsoft Sentinel and Defender XDR: the four-port pattern, unexpected LiteLLM database clients and SDK base URL overrides
`sigma_autonomous_agent_and_mcp_abuse.yml`	Sigma (3 rules): gateways spawning shells, archiving of coding-agent folders, and runs of `.__rev` scripts
`yara_ai_offense_workbenches.yar`	YARA (4 rules): CyberStrikeAI configs, ARTEX reset and notifier modules, rogue base URL servers and LiteLLM key harvesters

How the Suricata signatures work

SID	Looks for	Why
9402601	`/CyberStrikeAI/` in an HTTP URI	Exposed CyberStrikeAI workspace or config
9402608	TLS subject `CN=unifi.local` from an external host, one alert per source per hour	Policy hint only; check UniFi inventory first
9402604	`LiteLLM_SpendLogs` plus an SQL injection operator in the URI	Spend-log harvesting
9402611	Same, in the request body	Spend-log harvesting
9402605	`LiteLLM_ProxyModelTable` plus an SQL injection operator in a header	Key theft through the `Authorization` header
9402612	Same, in the request body	Key theft from the model table
9402606	External access to `/mcp-auth`, `/mcp/sse`, `/mcp/messages`, `/v1/mcp` or `/mcp-rest/`, rate-limited	MCP reconnaissance or exposure
9402607	`/.openclaw/workspace/` in a URI	OpenClaw exploit workspace retrieval
9402609	`w5_tokens.json` in a URI	CyberStrikeAI token-store retrieval

The four LiteLLM rules need a real SQL injection operator, such as `UNION SELECT`, `pg_sleep`, `' OR` or `; SELECT`. A bare `--` comment does not fire them, so routine Grafana or FinOps queries stay quiet.

TLS 1.3 encrypts the server certificate. The certificate rules therefore work only on TLS 1.2 or on decrypted traffic. Pair them with JA4S or JARM fingerprints and destination-IP controls.

How the KQL hunt works

The query reads `DeviceNetworkEvents` over a 14-day lookback. It has three branches.

- **Four-port pattern.** It flags a device that reaches all of 5001, 5002, 5003 and 8082 on one public IP within 60 minutes. The window slides from each connection, so sequences that cross a clock boundary still match.
- **Unexpected LiteLLM database client.** It flags connections to PostgreSQL on 5432 or 5433 from anything other than your approved gateway hosts running LiteLLM.
- **Base URL override.** It flags processes whose command line sets `--base-url`, `OPENAI_BASE_URL` or a similar option toward a host outside your approved provider list.

Fill in your own gateway hostnames, database IPs and approved Azure OpenAI hostnames before use. Add SNI allowlisting on the web gateway to catch base URL overrides set only in environment variables.

How the Sigma and YARA rules work

The first Sigma rule flags a LiteLLM or MCP server process that spawns a shell or interpreter. Normal `prisma generate` and `prisma db push` start-up steps are excluded.

The second flags `tar`, `zip`, `scp`, `curl` or `nc` when the command line names `.claude/`, `.cursor/`, `.codex/` or `agentrouter.env`. The third flags an interpreter running a `.__rev` script.

Sigma sees process launches, not file reads inside a running program. Add `auditd` read watches on `/root/.claude/`, `/root/.codex/` and `/root/.cursor/` to catch those.

Each YARA rule is limited to files under 256 KB and carries explicit exclusions.

YARA rule	Fires on	Excludes
`ODIN_CyberStrikeAI_Workbench_Config_And_Rollback`	Live CyberStrikeAI configs and the admin rollback script	`config.example.yaml`
`ODIN_ARTEX_Finding_Notifier_And_Reset`	ARTEX `pgcrypto` reset and finding notifier modules	None listed
`ODIN_Rogue_OpenAI_BaseURL_SSTI_Server`	Fake OpenAI servers returning template payloads (`ssti-1`)	Benign mock OpenAI test fixtures
`ODIN_LiteLLM_Database_Key_Harvester`	SQL injection scripts aimed at LiteLLM tables	Upstream LiteLLM source and FinOps spend queries

How to sort an observation

What the observation shows	Action
A disclosure or documentation archive	Low-signal review (checked first, before any CVE file name match)
A real SQL injection operator, shell injection marker or template gadget, plus a matching rendered-sink event for the same completion	Critical: gateway exploitation
A harmless completion, or one that only discusses sink APIs	Low-signal review
A standalone PentAGI or DeepAudit install	Low-signal review
A confirmed multi-signal C2 node	Block

CHAPTER 08

Indicators

IN SHORT

This chapter lists the published indicators and explains what stays withheld.

INSIDE THIS CHAPTER

Block 28

Hunt 28

MITRE ATT&CK mapping 29

08 Indicators

Appendix A holds the 11 block entries. The 14 hunt entries in this chapter are described by file and port only; their addresses and hashes are withheld.

Type	Block	Hunt	Total
IPv4	5	12	17
SHA-256	6	2	8
Total	**11**	**14**	**25**

Block entries are confirmed multi-port or shared-certificate C2 nodes and the co-hosted workbench hashes. Hunt entries are older offensive staging hosts and proof-of-concept hashes; their addresses are withheld, because older staging hosts may be dual-use or reassigned. Cloud and VPS addresses change hands often, so block IPs for 30 days and then review.

Withheld: the two French coding-agent hosts, the advisory workbench, the CVE-2026-42271 reproduction host, the base URL workbench host, the disclosure archive host, and all PentAGI, DeepAudit, HexStrikeAI and `unifi.local` addresses.

Block

Indicator	Port	Last seen	What it is
177.3.90[.]1	8899	13 Nov 2026	Co-hosted CyberStrikeAI v1.7.24 and ARTEX v0.3.14 / v0.3.18; 363 files; C2 on 8080 and 8787
38.6.108[.]121	5001	14 Nov 2026	PEG TECH INC (AS398823) four-port node on 5001, 5002, 5003, 8082
186.241.69[.]175	443	14 Nov 2026	AROSSCLOUD INC. CyberStrikeAI node with a CyberStrikeAI certificate name (withheld)
103.48.132[.]2	443	14 Nov 2026	YANCY LIMITED ARTEX Cluster A node with certificate name A (withheld) (237 endpoints share it)
38.12.72[.]83	443	14 Nov 2026	PEG TECH INC (AS54600) ARTEX Cluster B node with certificate name B (withheld) (175 endpoints share it)

The six block hashes are the workbench files from 177.3.90[.]1:8899 in Chapter 3: `run.sh`, `upgrade.sh`, `telegram.go`, `webhook.go`, `reset-password.sh` and `rollback_admin_i12503.sh`.

Hunt

These hosts staged offensive scripts. The dates are the file or snapshot dates recorded on each server, not how long Odin watched it.

Indicator	Port	What it held
Withheld	8000	`mine.py`, `creds3.py`, written to query LiteLLM spend logs
Withheld	9999	`fast_litellm_scan.py`, `phase8c_extract.py`
Withheld	8765	MCP OAuth scripts `mcp_eat_xtests.py`, `mcp_eat_capture.py`

Indicator	Port	What it held
Withheld	8099	MCP SSE scripts and `.__revN` backups
Withheld	8000	Exposed `/CyberStrikeAI/tmp/workspace/projects/`
Withheld	80	`w5_verify.py` loading `/opt/cyberstrike-ai/w5_tokens.json`
Withheld	8000	SSH and NAS lateral payloads `/nas_deliver.ps1`, `/fix_authkey.ps1`
Withheld	8080	OpenClaw FOFA key scraper `call_all.py`, `test_keys.py`
Withheld	8888	Webshell and XXE staging, `/shell.php.__rev28`
Withheld	8080	Sliver C2 loader `/gen_loader.py.__rev2`
Withheld	8080	OpenClaw GL.iNet OpenVPN exploit staging
Withheld	39002	OpenClaw ARP, DNS and SMB lateral module `/openclaw_lan_hunter_v18.py`

The two hunt hashes belong to the base URL server files from Chapter 5, `ssti_llm3.py` and `fakellm.py`. Their values are withheld with the host.

MITRE ATT&CK mapping

Technique	How it shows up	Detection
T1190 Exploit Public-Facing Application	SQL injection against LiteLLM tables; CVE-2026-42271 on MCP test endpoints	Suricata 9402604, 9402605, 9402611, 9402612, 9402606; YARA LiteLLM rule
T1059 Command and Scripting Interpreter	Agent-driven edit-and-retry scripts; gateways spawning shells	Sigma rules 1 and 3
T1070 Indicator Removal	Admin password hash and timestamp rollback on the CyberStrikeAI database	YARA CyberStrikeAI rule; file hash

MITRE ATLAS covers the AI-specific steps: LLM-driven tool chaining and automated advisory analysis. OWASP LLM02 (sensitive information disclosure) and LLM05 (improper output handling) cover the gateway key theft and the base URL template injection.

09

CHAPTER 09

What Defenders Should Do

IN SHORT

This chapter lists the actions defenders should take, in priority order.

09 What Defenders Should Do

These steps turn the rules in Chapter 7 and the indicators in Chapter 8 into controls for AI gateways, MCP servers and agent hosts, ordered by urgency.

Within 24 hours

- [ ] Block the five block-tier IPs for 30 days, and load the six workbench hashes into EDR and file scanning.
- [ ] Hunt for the staging file names and URI paths in Chapter 8; the 12 hunt addresses and their 2 script hashes are withheld.
- [ ] Check every LiteLLM version. Upgrade to 1.83.7 or later to fix CVE-2026-42271 on `/mcp-rest/test/*`.

Within 72 hours

- [ ] Bind LiteLLM PostgreSQL (5432 and 5433) to loopback or a private subnet only.
- [ ] Move `LITELLM_SALT_KEY` into a secret manager, away from the database host, and mask bearer keys in spend logs.
- [ ] Require mutual TLS or scoped OAuth 2.1 tokens on all MCP endpoints: `/mcp/sse`, `/mcp-auth`, `/v1/mcp` and `/mcp-rest/`.

Within 7 to 30 days

- [ ] Lock SDK base URLs to approved provider domains with web gateway SNI or URL allowlists.
- [ ] Escape all model output before it reaches a template engine (OWASP LLM05).
- [ ] Alert on the four-port pattern (5001, 5002, 5003, 8082) within 60 minutes, and deploy the Suricata rules.
- [ ] Ban ad-hoc HTTP file servers in folders that hold `.claude`, `.cursor`, `.codex` or `.aider` data, and watch reads of those folders with `auditd`.
- [ ] Keep HexStrikeAI Werkzeug servers, standalone PentAGI listeners, unconfirmed `unifi.local` hits and the non-HTTP ports listed in Chapter 6 off automatic block lists.

10

CHAPTER 10

Scope and Limits

IN SHORT

This chapter sets out what the evidence can show and what it cannot.

10 Scope and Limits

- **Window.** The findings cover a two-week observation window ending 18 November 2026. The 1,874 fleet records were seen on 13 and 14 November 2026. Earlier dates in this report are file or snapshot dates on the servers, not a tracking start.
- **Running is not hostile.** A confirmed record shows the framework is running. Authorized red teams, labs and threat actors use the same public packages.
- **Exposure is not breach.** Coding-agent folders, AI-stack records and staged scripts show what sat on a server. They do not show that any attack worked on a target.
- **Aggregate counts.** The 1,731, 1,779 and 349 figures are counts. Per-host detail is withheld to protect the owners of exposed machines.
- **Proofs of concept.** The CVE-2026-42271 workspace and the base URL servers were not tested live.
- **Rule testing.** Every published rule file loads without errors in its engine (Suricata 7.0.12, YARA-X, pySigma and Microsoft's Kusto.Language analyzer, checked on 20 November 2026). The rules were not replayed against live traffic, so test and tune them in your own environment before production use.

11

CHAPTER 11

Sources

IN SHORT

This chapter lists the public references used in the report.

11 Sources

1. OWASP Top 10 for Large Language Model Applications (2025), LLM02 and LLM05. https://genai.owasp.org/llm-top-10/ (verified 18 November 2026)
2. Model Context Protocol. Authorization specification. https://modelcontextprotocol.io/specification/2025-03-26/basic/authorization (verified 18 November 2026)
3. LiteLLM. Proxy production deployment documentation. https://docs.litellm.ai/docs/proxy/prod (verified 18 November 2026)
4. MITRE ATLAS data repository. https://github.com/mitre-atlas/atlas-data/blob/main/dist/ATLAS.yaml (verified 18 November 2026)
5. Microsoft Threat Intelligence. Chinese threat actor Storm-0940 uses credentials from password spray attacks from a covert network. https://www.microsoft.com/en-us/security/blog/2024/10/31/chinese-threat-actor-storm-0940-uses-credentials-from-password-spray-attacks-from-a-covert-network/ (verified 18 November 2026)
6. MITRE ATT&CK. T1190 Exploit Public-Facing Application. https://attack.mitre.org/techniques/T1190/ (verified 18 November 2026)
7. MITRE ATT&CK. T1059 Command and Scripting Interpreter. https://attack.mitre.org/techniques/T1059/ (verified 18 November 2026)
8. MITRE ATT&CK. T1070 Indicator Removal. https://attack.mitre.org/techniques/T1070/ (verified 18 November 2026)
9. NVD. CVE-2026-42271, BerriAI LiteLLM command execution via MCP stdio test endpoints (affected 1.74.2 to before 1.83.7, fixed in 1.83.7). https://nvd.nist.gov/vuln/detail/CVE-2026-42271 (verified 18 November 2026)

APPENDICES

Indicators and Detection Rules

FOR DEFENDERS

Appendix A lists the indicators published for this study. Appendix B carries every published detection rule in full.

INSIDE

A. Indicators of compromise (11) 37

B. Detection rules (4 files) 38

A Indicators of Compromise

Values are defanged for display. The CSV, JSON and STIX files on the online edition carry the raw values. Values the report marks as withheld are not listed.

ID	Type	Indicator	Port	Context	Action
R04-IOC-01	IPv4	177.3.90[.]1	8899	Co-Hosted CyberStrikeAI v1.7.24 + ARTEX v0.3.14/v0.3.18 Workbench	BLOCK
R04-IOC-02	SHA-256	1fe555c690ac70b1d58cc6b0b29dac99749f996d4f0edb13eb82fa25e59cd574	8899	CyberStrikeAI Autonomous Runner SHA-256	BLOCK
R04-IOC-03	SHA-256	754d0ac640dd35718b845fcf1ea5510499267de97806cfb74a6a35f7f593f329	8899	CyberStrikeAI Self-Updater SHA-256	BLOCK
R04-IOC-04	SHA-256	3729020bfffe04f4fc0f384a81a2ec2293bb525844a74ef93851d1e1cb01a38b	8899	ARTEX Telegram Bot Finding Notifier SHA-256	BLOCK
R04-IOC-05	SHA-256	4fe3ba0c6e064005ec96a0e0692864ffb73bcaac34a4c65af3c9c63cf3fe346d	8899	ARTEX Webhook Finding Notifier SHA-256	BLOCK
R04-IOC-06	SHA-256	b17d1932eeea6d77e946f65620c29ae6e9826fcc6cced7eee5d85aba19515ae5	8899	ARTEX PostgreSQL pgcrypto Admin Reset SHA-256	BLOCK
R04-IOC-07	SHA-256	e8521a024f6ea910390e99e56eba8a91dd23c99f29162252d5287d71994e7cb4	8899	CyberStrikeAI SQLite Admin Credential Rollback SHA-256	BLOCK
R04-IOC-08	IPv4	38.6.108[.]121	5001	Confirmed CyberStrikeAI 4-Port Quartet Node	BLOCK
R04-IOC-09	IPv4	186.241.69[.]175	443	Confirmed CyberStrikeAI Bulk Cert Node	BLOCK
R04-IOC-10	IPv4	103.48.132[.]2	443	Confirmed ARTEX Autonomous C2 Node (Cluster A)	BLOCK
R04-IOC-11	IPv4	38.12.72[.]83	443	Confirmed ARTEX Autonomous C2 Node (Cluster B)	BLOCK

B Detection Rules

Each rule file loads in its own engine (Kusto, pySigma, Suricata 7, YARA-X). Tune thresholds and allow-lists to your environment before alerting or blocking. The same files can be downloaded from the online edition.

sentinel_llm_gateway_and_agent_hunt.kql

KQL (SENTINEL / DEFENDER XDR)

```
// Odin by Brandefense - Report #04 Detection Engineering Suite
// Target: Microsoft Sentinel / Defender XDR (DeviceNetworkEvents)
// Context: (1) Per-(DeviceName, RemoteIP) 4-port CyberStrikeAI quartet egress evaluated across a true sliding
//          60-minute event window anchored at each observed quartet connection timestamp [WindowStart,
WindowStart + 60m]
//          (so boundary-crossing 4-port sequences within any 60-minute span are always caught and connections
//          outside the hour never mask a valid quartet sequence),
//          (2) Unauthorized PostgreSQL clients touching designated LiteLLM DB IPs (or public 5432/5433),
//          (3) Command-line base_url flag/assignment overrides to unapproved domains or explicit raw IPv4s.
//          Telemetry notes: RemoteUrl records TLS SNI/Host rather than encrypted HTTPS URI paths, and
//          InitiatingProcessCommandLine captures argv only; pair with SWG TLS SNI allowlists for env vars.

let Lookback = 14d;
let CyberStrikePorts = dynamic([5001, 5002, 5003, 8082]);
let ApprovedLLMHosts = dynamic(["api.openai.com", "api.anthropic.com", "generativelanguage.googleapis.com",
"models.inference.ai.azure.com", "contoso-aoai-prod.openai.azure.com"]);  // list each approved Azure OpenAI
resource by exact hostname; replace the example
let AuthorizedGatewayHosts = dynamic(["litellm-gw-prod-01", "ai-proxy-prod-01"]);
let LiteLLMDbIPs = dynamic(["10.20.30.40", "10.20.30.41"]); // Populate with private IPs of LiteLLM PostgreSQL
backends
let BaseEvents = DeviceNetworkEvents
| where Timestamp >= ago(Lookback) and ActionType == "ConnectionSuccess"
| extend HostClean = tolower(trim_end(@"/", tostring(parse_url(RemoteUrl).Host)))
| extend EffectiveHost = iff(isempty(HostClean), tolower(RemoteUrl), HostClean)
| extend AnomalyCategory = case(
    RemoteIPType == "Public" and RemotePort in (CyberStrikePorts),
        "CyberStrikeAI_MultiPortQuartetEgress",
    RemotePort in (5432, 5433)
        and (RemoteIP in (LiteLLMDbIPs) or RemoteIPType == "Public")
        and not(DeviceName in~ (AuthorizedGatewayHosts)
            and (InitiatingProcessFileName startswith "python" or InitiatingProcessFileName =~ "litellm")
            and InitiatingProcessCommandLine has "litellm"),
        "Unexpected_LiteLLM_PostgreSQL_Client",
    RemoteIPType == "Public"
        and InitiatingProcessCommandLine has_any ("--api-base", "--base-url", "base_url=", "api_base=",
"OPENAI_BASE_URL=", "OPENAI_API_BASE=", "ANTHROPIC_BASE_URL=")
        and ((isnotempty(EffectiveHost) and not(EffectiveHost in~ (ApprovedLLMHosts)))
            or (isempty(EffectiveHost) and InitiatingProcessCommandLine matches regex @"https?://\d{1,3}(\.
\d{1,3}){3}")),
        "Unverified_OpenAI_BaseURL_Override",
    "None"
)
| where AnomalyCategory != "None";
let QuartetAnchors = BaseEvents
| where AnomalyCategory == "CyberStrikeAI_MultiPortQuartetEgress"
| extend WindowStart = Timestamp
| distinct AnomalyCategory, DeviceName, RemoteIP, WindowStart
| union (
    BaseEvents
    | where AnomalyCategory != "CyberStrikeAI_MultiPortQuartetEgress"
    | extend WindowStart = datetime(1970-01-01)
    | distinct AnomalyCategory, DeviceName, RemoteIP, WindowStart
);
BaseEvents
| join kind=inner QuartetAnchors on AnomalyCategory, DeviceName, RemoteIP
| where Timestamp >= WindowStart and Timestamp <= WindowStart + 60m
     or AnomalyCategory != "CyberStrikeAI_MultiPortQuartetEgress"
| summarize
    FirstSeen = min(Timestamp),
    LastSeen = max(Timestamp),
```

```
    ConnectionCount = count(),
    DistinctPorts = dcount(RemotePort),
    PortsObserved = make_set(RemotePort, 10),
    HostsObserved = make_set_if(EffectiveHost, isnotempty(EffectiveHost), 5),
    SampleProcesses = make_set(InitiatingProcessFileName, 5),
    SampleProcessPaths = make_set(InitiatingProcessFolderPath, 5)
    by AnomalyCategory, DeviceName, RemoteIP, WindowStart
| where (AnomalyCategory == "CyberStrikeAI_MultiPortQuartetEgress" and DistinctPorts == 4 and
datetime_diff('minute', LastSeen, FirstSeen) <= 60 and LastSeen - FirstSeen <= 60m)
     or (AnomalyCategory != "CyberStrikeAI_MultiPortQuartetEgress" and ConnectionCount >= 2)
| order by ConnectionCount desc
```

sigma_autonomous_agent_and_mcp_abuse.yml

SIGMA

```
title: LiteLLM or MCP Gateway Spawning Unexpected Shell or Network Utility Child Process
id: 94a1c310-4b12-4d89-9a01-202611160401
status: experimental
description: >
  Detects LiteLLM proxy or Model Context Protocol (MCP) server processes spawning OS shells
  or network transfer utilities (excluding normal Prisma schema migrations at LiteLLM startup),
  indicative of MCP tool-wrapper command injection. Scope to server and container tiers.
author: Odin by Brandefense
date: 2026-11-16
logsource:
  category: process_creation
  product: linux
detection:
  selection_parent_img:
    ParentImage|endswith:
      - '/litellm'
      - '/mcp-server'
      - '/mcp_server'
      - '/fastmcp'
  selection_parent_cmd:
    ParentCommandLine|contains:
      - 'litellm --'
      - 'litellm_proxy'
      - 'fastmcp run'
  selection_child_bins:
    Image|endswith:
      - '/sh'
      - '/bash'
      - '/dash'
      - '/zsh'
      - '/perl'
      - '/php'
      - '/nc'
      - '/ncat'
      - '/socat'
      - '/curl'
      - '/wget'
  selection_child_py:
    Image|contains: '/python'
  condition: (selection_parent_img or selection_parent_cmd) and (selection_child_bins or selection_child_py)
falsepositives:
  - Developer workstations running local stdio MCP Python tool plugins
level: medium
tags:
  - attack.initial_access
  - attack.t1190
  - attack.execution
  - attack.t1059
---
title: Archival or Network Exfiltration Utility Referencing Coding-Agent Credential Stores
id: 94b2d421-5c23-4e90-8b12-202611160402
status: experimental
description: >
  Detects archival, file-transfer, or exfiltration utilities passing local CLI coding-agent session
```

```
    directories, router environment files, or shell snapshots in process argv (for example tar, zip,
    scp, rsync, or curl targeting .claude, .cursor, .codex, or .aider stores). Pair with Linux auditd
    openat watches (-w /root/.claude/ -p r -k ai_cli_cred_read) to catch direct Python or Go file reads.
  author: Odin by Brandefense
  date: 2026-11-16
  logsource:
    category: process_creation
    product: linux
  detection:
    selection_archivers:
      Image|endswith:
        - '/tar'
        - '/zip'
        - '/7z'
        - '/gzip'
        - '/scp'
        - '/sftp'
        - '/curl'
        - '/wget'
        - '/nc'
        - '/ncat'
        - '/base64'
    selection_paths:
      CommandLine|contains:
        - '/.claude/'
        - '/.cursor/'
        - '/.codex/'
        - 'agentrouter.env'
        - 'shell_snapshots/'
        - '.aider.chat.history.md'
    condition: selection_archivers and selection_paths
  falsepositives:
    - Authorized endpoint backup scripts archiving developer home directories
  level: medium
  tags:
    - attack.credential_access
    - attack.t1552.001
    - attack.collection
  ---
  title: Autonomous Agent Iterative Exploit Revision Script Execution
  id: 94c3e532-6d34-4f01-9c23-202611160403
  status: experimental
  description: >
    Detects script interpreters or direct executable invocations running machine-generated iterative
    revision files matching the .__rev naming convention observed across autonomous agent workspaces.
  author: Odin by Brandefense
  date: 2026-11-16
  logsource:
    category: process_creation
    product: linux
  detection:
    selection_cmd:
      CommandLine|contains: '.__rev'
    selection_img:
      Image|contains: '.__rev'
    filter_benign_mgmt:
      Image:
        - '/usr/bin/rm'
        - '/bin/rm'
        - '/usr/bin/ls'
        - '/bin/ls'
        - '/usr/bin/cat'
        - '/bin/cat'
        - '/usr/bin/cp'
        - '/bin/cp'
        - '/usr/bin/mv'
        - '/bin/mv'
        - '/usr/bin/less'
        - '/bin/less'
        - '/usr/bin/more'
        - '/bin/more'
        - '/usr/bin/head'
```

```
      - '/bin/head'
      - '/usr/bin/tail'
      - '/bin/tail'
      - '/usr/bin/grep'
      - '/bin/grep'
      - '/usr/bin/diff'
      - '/bin/diff'
      - '/usr/bin/sed'
      - '/bin/sed'
      - '/usr/bin/awk'
      - '/bin/awk'
      - '/usr/bin/tar'
      - '/bin/tar'
      - '/usr/bin/rsync'
      - '/bin/rsync'
      - '/usr/bin/git'
      - '/bin/git'
      - '/usr/bin/stat'
      - '/bin/stat'
      - '/usr/bin/chmod'
      - '/bin/chmod'
      - '/usr/bin/chown'
      - '/bin/chown'
      - '/usr/bin/vim'
      - '/bin/vim'
      - '/usr/bin/nvim'
      - '/bin/nvim'
      - '/usr/bin/nano'
      - '/bin/nano'
  condition: (selection_cmd or selection_img) and not filter_benign_mgmt
falsepositives:
  - Rare custom build pipelines using double-underscore revision suffixes
level: medium
tags:
  - attack.execution
  - attack.t1059
```

suricata_ai_c2_and_mcp_gateway.rules

SURICATA

```
# Odin by Brandefense - Report #04 Suricata 7.x AI C2 & LLM/MCP Gateway Suite
# Classification: TLP:CLEAR
# Purpose: Composite network detection for CyberStrikeAI, ARTEX, LiteLLM SQLi, MCP route probing, and OpenClaw
staging.
# Telemetry Note: tls.cert_subject rules inspect plaintext X.509 certificates in the TLS 1.2 Certificate
handshake
# message or SSL-inspected flows; TLS 1.3 encrypts server certificates, requiring JA4S/JARM and destination-IP
controls.

# 1. CyberStrikeAI exposed project workspace or live config URI
alert http any any -> any any (msg:"ODIN-AI CyberStrikeAI Exposed Workspace or Config URI";
flow:established,to_server; http.uri; content:"/CyberStrikeAI/"; nocase; fast_pattern; classtype:trojan-activity;
sid:9402601; rev:4;)




# 2. External WAN host presenting default Ubiquiti CN=unifi.local (corroborate against UniFi inventory or JARM)
alert tls !$HOME_NET any -> $HOME_NET any (msg:"ODIN-AI External Host Presenting Default Ubiquiti unifi.local TLS
Certificate"; flow:established,to_client; tls.cert_subject; content:"unifi.local"; nocase; fast_pattern; pcre:"/
CN=unifi\.local(?:[,\/]|$)/i"; threshold:type limit, track by_src, count 1, seconds 3600; classtype:policy-
violation; sid:9402608; rev:3;)

# 3. LiteLLM SpendLogs SQL injection in normalized HTTP URI
alert http any any -> $HOME_NET any (msg:"ODIN-AI LiteLLM SpendLogs SQL Injection in HTTP URI";
flow:established,to_server; http.uri; content:"LiteLLM_SpendLogs"; nocase; fast_pattern; pcre:"/(?:union\s+
(?:all\s+)?select|pg_sleep|'\s*(?:or|and)\s+|\x3b\s*(?:select|drop)\b)/i"; classtype:web-application-attack; sid:
9402604; rev:4;)
```

```
# 4. LiteLLM SpendLogs SQL injection in normalized HTTP Request Body
alert http any any -> $HOME_NET any (msg:"ODIN-AI LiteLLM SpendLogs SQL Injection in HTTP Request Body";
flow:established,to_server; http.request_body; content:"LiteLLM_SpendLogs"; nocase; fast_pattern; pcre:"/
(?:union\s+(?:all\s+)?select|pg_sleep|'\s*(?:or|and)\s+|\x3b\s*(?:select|drop)\b)/i"; classtype:web-application-
attack; sid:9402611; rev:2;)

# 5. LiteLLM ProxyModelTable SQL injection in normalized HTTP Header (Authorization SQLi)
alert http any any -> $HOME_NET any (msg:"ODIN-AI LiteLLM ProxyModelTable SQL Injection in HTTP Header";
flow:established,to_server; http.header; content:"LiteLLM_ProxyModelTable"; nocase; fast_pattern; pcre:"/
(?:union\s+(?:all\s+)?select|pg_sleep|'\s*(?:or|and)\s+|\x3b\s*(?:select|drop)\b)/i"; classtype:web-application-
attack; sid:9402605; rev:4;)

# 6. LiteLLM ProxyModelTable SQL injection in normalized HTTP Request Body
alert http any any -> $HOME_NET any (msg:"ODIN-AI LiteLLM ProxyModelTable SQL Injection in HTTP Request Body";
flow:established,to_server; http.request_body; content:"LiteLLM_ProxyModelTable"; nocase; fast_pattern; pcre:"/
(?:union\s+(?:all\s+)?select|pg_sleep|'\s*(?:or|and)\s+|\x3b\s*(?:select|drop)\b)/i"; classtype:web-application-
attack; sid:9402612; rev:2;)

# 7. External Model Context Protocol (MCP) SSE / Auth endpoint access (rate-limited recon/exposure signal)
alert http !$HOME_NET any -> $HOME_NET any (msg:"ODIN-AI External MCP Gateway Route Access";
flow:established,to_server; http.uri; content:"/mcp"; nocase; fast_pattern; pcre:"/\/(?:mcp-auth|mcp\/(?:sse|
messages)|v1\/mcp|mcp-rest\/)/i"; threshold:type limit, track by_src, count 1, seconds 600; classtype:attempted-
recon; sid:9402606; rev:4;)

# 8. OpenClaw agentic exploit workspace and CVE staging retrieval
alert http any any -> any any (msg:"ODIN-AI OpenClaw Agentic Exploit Workspace Staging Access";
flow:established,to_server; http.uri; content:"/.openclaw/workspace/"; nocase; fast_pattern; classtype:trojan-
activity; sid:9402607; rev:4;)

# 9. Exposed CyberStrikeAI token-store workbench artifact retrieval
alert http any any -> any any (msg:"ODIN-AI CyberStrikeAI Exposed Token Store Artifact URI";
flow:established,to_server; http.uri; content:"w5_tokens.json"; nocase; fast_pattern; classtype:trojan-activity;
sid:9402609; rev:3;)
```

yara_ai_offense_workbenches.yar

YARA

```
/*
   Odin by Brandefense - Report #04 YARA Workbench & AI Gateway Exploitation Rules
   Classification: TLP:CLEAR
   Note: Rules 1-3 are validated against retrieved workbench source/config artifacts;
         Rule 4 targets offensive LiteLLM SQLi and credential-extraction scripts while
         excluding upstream LiteLLM package source, Prisma schemas, and FinOps spend queries.
*/

rule ODIN_CyberStrikeAI_Workbench_Config_And_Rollback {
    meta:
        author = "Odin by Brandefense"
        description = "Detects CyberStrikeAI live configuration (upstream example templates may also match;
review hits), autonomous runner/updater scripts, or SQLite RBAC admin credential rollback utilities"
        classification = "TLP:CLEAR"
    strings:
        $cfg1 = "tls_auto_self_sign: true" ascii
        $cfg2 = "auth_failure_cooldown_seconds:" ascii
        $cfg3 = "external_mcp_max_concurrent_per_server:" ascii
        $run1 = "# CyberStrikeAI one-click deploy and start script" ascii
        $run2 = "pypi.tuna.tsinghua.edu.cn/simple" ascii
        $upg1 = "DEFAULT_GITHUB_REPO=\"AIPentest/CyberStrikeAI\"" ascii
        $upg2 = "BINARY_NAME=\"cyberstrike-ai\"" ascii
        $rb1 = "/CyberStrikeAI/data/conversations.db" ascii
        $rb2 = "UPDATE rbac_users SET password_hash=?, updated_at=?" ascii
    condition:
        filesize < 256KB and (
            all of ($cfg1, $cfg2, $cfg3)
            or all of ($run*)
            or all of ($upg*)
            or all of ($rb*)
        )
}
```

```
rule ODIN_ARTEX_Finding_Notifier_And_Reset {
    meta:
        author = "Odin by Brandefense"
        description = "Detects ARTEX Go Telegram/Webhook finding notification modules, ARTEX OpenCode Go session
headers, and PostgreSQL pgcrypto admin reset scripts"
        classification = "TLP:CLEAR"
    strings:
        $rst1 = "ARTEX_PG_DSN" ascii
        $rst2 = "auth.password_hash" ascii
        $rst3 = "BCRYPT_COST=10" ascii
        $tg1 = "KindTelegram" ascii
        $tg2 = "telegramTextLimit = 4096" ascii
        $tg3 = "DefaultRatePerMin() int { return 20 }" ascii
        $wh1 = "KindWebhook" ascii
        $wh2 = "webhookDefaultTemplate" ascii
        $wh3 = "\"finding_id\": {{$it.FindingID}}" ascii
        $oc1 = "github.com/Autumn-27/norma" ascii
        $oc2 = "quotaAwareTransport" ascii
        $oc3 = "sessionHeaderKey" ascii
    condition:
        filesize < 256KB and (
            (2 of ($rst*))
            or (2 of ($tg*))
            or (2 of ($wh*))
            or ($oc1 and $oc2 and $oc3)
        )
}

rule ODIN_Rogue_OpenAI_BaseURL_SSTI_Server {
    meta:
        author = "Odin by Brandefense"
        description = "Detects rogue OpenAI-compatible HTTP servers explicitly serving the ssti-1 model
identifier alongside BaseHTTPRequestHandler or chat.completion spoofing"
        classification = "TLP:CLEAR"
    strings:
        $m1 = "\"object\":\"chat.completion\"" ascii
        $m2 = "\"id\":\"ssti-1\"" ascii
        $m3 = "BaseHTTPRequestHandler" ascii
        $m4 = "Access-Control-Allow-Origin" ascii
    condition:
        filesize < 256KB and $m2 and 2 of ($m1, $m3, $m4)
}

rule ODIN_LiteLLM_Database_Key_Harvester {
    meta:
        author = "Odin by Brandefense"
        description = "Detects offensive scripts extracting API keys via LiteLLM SQLi or ProxyModelTable
credential dumps while excluding upstream LiteLLM source and FinOps spend queries"
        classification = "TLP:CLEAR"
    strings:
        $t1 = "LiteLLM_SpendLogs" ascii nocase
        $t2 = "LiteLLM_ProxyModelTable" ascii nocase
        $t3 = "LiteLLM_VerificationTokenView" ascii nocase
        $sqli1 = "UNION SELECT" ascii nocase
        $sqli2 = "pg_sleep" ascii nocase
        $sqli3 = "Authorization header SQLi" ascii nocase
        $sqli4 = "SELECT " ascii nocase
        $scan1 = "fast_litellm_scan" ascii
        $harv1 = "sk-ant-" ascii
        $harv2 = "litellm_params" ascii nocase
        $db1 = "psycopg2" ascii
        $db2 = "asyncpg" ascii
    condition:
        filesize < 256KB and (
            $scan1
            or (any of ($t*) and $harv1)
            or (any of ($t*) and ($sqli1 or $sqli2 or $sqli3))
            or (($t2 or $t3) and $sqli4 and any of ($db*) and any of ($harv*))
        )
}
```

ODIN

About Odin Research

Odin Research is the public research series of Odin by Brandefense. It documents adversary infrastructure seen on the internet and turns each finding into indicators and detection rules that defenders can use. Victims and third parties stay anonymous, addresses appear only for confirmed adversary infrastructure, and no working attack code is published.

Odin is the agentic analyst inside the Brandefense threat intelligence platform, which unifies Cyber Threat Intelligence, Digital Risk Protection and External Attack Surface Management.

Online edition, indicators and rules · brandefense.io/odin-research/reports/autonomous-ai-offense-and-llm-gateway-attacks/
Short Read · brandefense.io/odin-research/blog/ai-attack-tools-are-now-aiming-at-your-ai-gateways/
Defense Kit · brandefense.io/odin-research/defense-kit/