Detecting LLM API Credential Abuse from Request Metadata

The post-rotation visibility gap, and the metadata signal that closes it

INFERTRAIL RESEARCH · SEPTEMBER 2026

Key findings

+ **Rotation is prospective. It answers nothing about prior use.** After the March 2026 LiteLLM supply-chain compromise, one of the largest affected organizations reported rotating every exposed credential. A researcher then tested those credentials under the organization's own responsible-disclosure policy and found almost all of them still authenticated.[1]

+ **A stolen LLM key is theft of metered compute, not of data.** The loss appears on the victim's invoice as ordinary usage. That is what makes the credential worth stealing, pricing, and reselling - up to ~$46,000/day of potential inference on a single compromised account,[3] through a marketplace spanning more than thirty providers.[4]

+ **The three controls that should catch it cannot, structurally.** Billing aggregates away the attribution; a SIEM correlates identity events that bearer-token authentication never produces; gateway logs record every request but are read only once suspicion already exists.

+ **The signal that survives is in request metadata** - a single credential carrying two behaviorally incoherent populations of traffic, visible without touching prompt content.

+ **Two checks run today against telemetry you already emit.** One is deterministic - any traffic on a rotated credential. One is behavioral - deviation from a credential's own baseline across several features at once.

+ **This is triage, not proof.** It has defined blind spots: reverse-proxy resale, owner mimicry, slow-ramp attackers, and abuse that predates the baseline (Section 9).

+ **Three of six detectors still lose to deliberate evasion in our own benchmark**, and we name the variants that beat them (Section 8): a single-model extraction workload, a replay from one new country, and a spend ramp spread over eight days. The other three catch every variant we have built. The evaluation is synthetic throughout; no field detection rate is claimed anywhere in this document.

Where to go: the findings above and Figure 1 carry the argument. If you run a gateway, Section 7 is the two checks you can run against telemetry you already have. If you are evaluating the method, Sections 5 and 8 are the feature engine and the measurements.

1. The question rotation cannot answer

In the March 2026 LiteLLM supply-chain compromise, poisoned package versions ran on Python startup and swept the host environment for credentials - cloud keys, Kubernetes configs, and, because the compromised package *was* the LLM gateway, every provider API key configured beside them: OpenAI, Anthropic, Azure AI, all at once.[1] Standard incident response followed: rotate the exposed keys, close the ticket.

Rotation invalidates a credential prospectively. It does not establish whether the credential was used before rotation - or even that the rotation took effect. One of the largest impacted organizations reported that it had rotated all the exposed credentials; a researcher then tested them under the organization's own responsible-disclosure policy and found that almost all still authenticated.[1]

Prior use is hard to reconstruct. The key authenticates statelessly - a bearer token in a header, no session, no login, no SSO assertion - so it produces none of the events a SIEM correlates. The activity lands in two places only: the gateway request log and the provider invoice. Neither answers the operative question.

> **Was this credential used by a party other than us, and when?** This document explains why that question is hard to answer with the controls most gateways already run, and how request metadata answers it.

2. What a stolen inference credential is worth

A stolen datastore credential exposes data; the loss is confidentiality. A stolen LLM key exposes metered compute billed to its owner: the loss is money, spent on the owner's account as ordinary usage. The credential has monetary value to an attacker regardless of what data it can reach.

Attacker model

The attacker holds a valid API credential - from source-code or config exposure, a compromised developer endpoint, a supply-chain compromise like LiteLLM, or the resale market. They have no other access: no console, no network position, no way to alter the victim's logging or billing. The objective is to consume inference on the victim's account, directly or by reselling it.

This is a low-capability, high-frequency threat. It requires only the credential and network reach to the endpoint - which is precisely why it presents no host- or identity-level telemetry, and must be detected from request metadata.

The market that prices it

Stolen keys are validated against roughly ten major providers, then resold - usually through reverse-proxy relays that meter and bill downstream buyers,[2] under storefront brands catalogued in public research.[6] Because the credential circulates, whoever runs up the cost need not be whoever stole it.

MEASURE	FIGURE	SOURCE
Potential inference cost on one compromised account	~$46,000/day (worst case)	Sysdig TRT, 2024[3]
Providers stolen keys are validated against	~10	CSA, 2026[2]
Providers resold in one catalogued marketplace	30+	CSA, 2026[4]
Attack sessions captured, Dec 2025 – Jan 2026	~35,000	CSA, 2026[4]
Internet-reachable LiteLLM gateways	2,659	Our scan (Netlas, 2026)[5]

The exposure surface is wide, but a surface measurement is not evidence of compromise: an exposed gateway does not mean a credential was stolen, and the host count fluctuates as instances appear and disappear. What it establishes is that the reachable target population is substantial.

Where the research is

The threat class lives in industry research, not the academic literature. Credential leakage and model extraction are both well studied; consuming and reselling a victim's paid inference with a stolen credential is not.

Its closest studied analog is cryptojacking: the unauthorized use of a victim's compute to mine cryptocurrency at the victim's expense.[10] The structure is the same - steal access, run a metered resource on the victim's account, leave the victim the bill - and, as with cryptojacking, the abuse is most tractably detected from behavioral resource-usage signals rather than from the credential itself.[11]

3. Why the existing controls miss it

Detection fails across the three controls that would normally surface credential misuse. Each fails for a reason rooted in how it works, not in how it is configured.

CONTROL	WHAT IT RECORDS	WHY CREDENTIAL RESALE PASSES
Billing	Aggregate tokens and cost per period	No attribution to a principal. Abusive and legitimate traffic both present as an increase in the aggregate, and by the time cost crosses a review threshold the anomaly is averaged across the period.
SIEM	Identity events: authentication, session, privilege change	API-key authentication produces none of them. Authentication success is not evidence that the caller is the legitimate owner,[7] so a resold credential is indistinguishable from first-party traffic at the identity layer.
Gateway logs	Every request, credential, and model call	A high-volume operational stream, queried only after an incident is already suspected. Resale produces no prior signal to trigger that query, so the records are retained and unread.
Rotation	-	Prospective only. Reissuing a credential yields no information about prior use: whether the old credential was exercised, by whom, or for what.
IP allow-lists, rate limits	Permitted traffic	They narrow what a stolen credential can do but do not report when a permitted credential is used by someone else. Abuse from an allow-listed network, or within the rate limit, passes as normal traffic.

Two consequences follow. An environment can rotate every credential and remain unable to determine whether the exposure it is responding to was ever exploited - the unresolved question from Section 1. And many gateways cannot allow-list at all: a public-facing product serves legitimate requests from arbitrary networks. Where the preventive control is unavailable, or is being respected by the abuser, the credential's own behavior is the only remaining signal.

4. The metadata signal

What defeats the controls in Section 3 is not that the traffic goes unrecorded - the gateway logs every request. It is that none of those controls compares a credential's behavior against its own history. Request metadata supports that comparison: each request carries, independent of its payload, the credential presented, source IP, target model, timestamp, and request size. That is enough to characterize a principal's behavior without touching prompt content - an API key being one kind of principal, alongside a user, workspace, or service account. How far the comparison carries, and where it fails, is Section 9.

A first-party credential exhibits a stable behavioral profile. It is used by a bounded set of clients, from a known IP range, against a small set of models, on a cadence consistent with its user population. The baseline is narrow because the credential maps to a specific application or team.

A resold credential violates that profile. Concurrent use by multiple downstream buyers produces observable inconsistencies on a single credential.

FEATURE	FIRST-PARTY BASELINE	RESOLD CREDENTIAL
Source IP / ASN	Bounded, known egress	Many unrelated addresses, concurrently
Geolocation	Regions the owner operates in	Regions the owner has no presence in
Time of use	The cadence of one user population	Distributed across hours no single population produces
Access frequency	Steady, tied to application load	Elevated
Cadence and concurrency	Application-shaped: diurnal, bounded parallelism	Sustained saturation, machine-regular intervals, or parallelism beyond what one client produces
Model selection	A small, established set	Calls to models outside that set

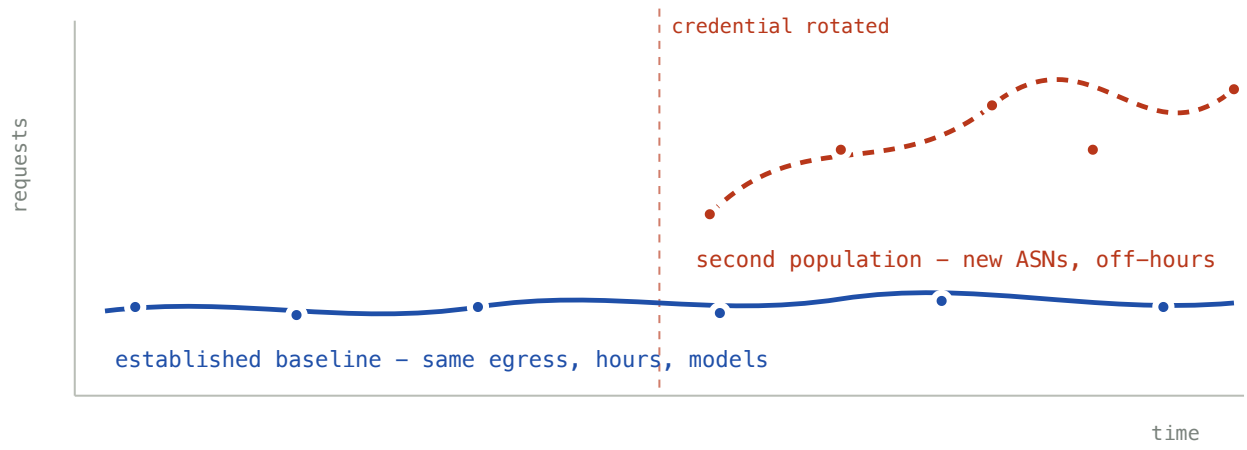

Figure 1. One principal, two behavioral populations. The baseline continues undisturbed - which is why aggregate volume and billing review do not surface it - while a second cluster with different egress, timing, and model mix runs alongside it. Traffic continuing after the rotation line is the deterministic case in Section 7.

The result is a second behavioral cluster inside one principal's traffic: usage a single legitimate principal could not produce. The detection scores these inconsistencies across all principals; Section 5 specifies how.

This signal is derivable from metadata alone. It requires no prompt content and no instrumentation in the request path; the gateway's existing request records are sufficient input.

This is applied UEBA, not a new method

Baselining a principal to detect account takeover from access metadata is a documented technique - including non-intrusive traffic analysis that flags misuse by an authenticated principal without inspecting payloads,[8] and session-graph models that detect account takeover from identifiers and timing at scale, with reported gains in detection and

reductions in false-positive friction.[9] The contribution here is the application to LLM credentials, not the underlying method.

What we have measured, and what we have not

Under modeled conditions the injected abuse is separable from the baseline on the features above. Section 8 reports the measurements in full, including where the detectors are defeated.

> **Read that result narrowly.** It establishes separability in the LLM-credential setting. It is not a field-detection rate, and because the same process defined the baseline and injected the abuse, it is an existence proof rather than a performance benchmark. A sensor is deployed to capture in-the-wild cases and will be reported separately.

One indicator requires no model at all. A rotated credential should receive no further traffic. Continued requests against a retired credential indicate that a copy is held and in use - the highest-confidence indicator available in metadata, subject to the conditions in Section 9, and the specific condition the LiteLLM-affected organizations could not rule out.

Credential resale is addressed here specifically, but in metadata terms it is one case of a broader condition: a single principal whose traffic contains more than one distinct behavioral population. The same approach extends to account sharing, unauthorized automation, and relayed traffic.

---

5. How the detection is built

The features above become a decision through three stages: a per-principal feature vector, a baseline resolved from the most specific available reference, and a robust deviation score. Each stage is deterministic - the same events produce the same output, byte for byte - because a finding an investigator cannot reproduce is not evidence.

The feature vector

Each principal is summarized over the analysis window by twenty-two numeric features, grouped by what they describe:

The groups are volume and span, cadence, concurrency, payload shape, cost, reliability, breadth across models and networks, and distribution shape across models and hours. Appendix A lists them.

Three choices in that list carry most of the weight.

Statistics are robust, not moment-based. Every baseline uses the median and the median absolute deviation rather than mean and standard deviation, because a single expensive request should not move a baseline. Deviation is reported as a robust z-score, `(value − median) / (1.4826 × MAD)`, where the constant makes MAD a consistent estimator of the standard deviation for normal data so the number reads on a familiar scale. When MAD is zero - common in small cohorts and for integer features like `distinct_models`, where more than half the population shares a value - the score falls back to a relative difference rather than dividing by zero.

Evenness and breadth are separated. `model_evenness` is Shannon entropy normalized by the *observed* support, so two models used 50/50 and five models used evenly both score 1.0. That is deliberate: how evenly a principal spreads its load and how many models it touches are different behaviors, and the second is carried by the `distinct_*` counts. The feature is named evenness rather than entropy precisely so a reader does not take 1.0 to mean "uses many models."

`night_share` - the fraction of activity between 00:00 and 06:00 UTC - is crude, and kept because it separates a human working day from a job that does not sleep without assuming a timezone for the principal.

The baseline hierarchy

"Deviation" is meaningless without saying *from what*. The baseline is resolved per principal, most specific first:

1. **Nearest established behavioral cluster** - other principals that behave alike, when clustering found the principal peers. 2. **The principal's own earlier history** - its first four active days define the baseline, the remainder is the comparison period. 3. **Tenant cohort** - principals of the same type on the same plan.

A principal without enough history to split returns no tier-2 baseline at all. That is the honest answer: a principal with three days of data has no baseline, and inventing one is how detectors end up flagging new accounts for being new.

Every finding records which tier it was scored against, because *which baseline was this compared to* is the first question an investigator asks and the one most likely to explain a surprising result.

Behavioral clustering

Tier 1 needs peers, which means clustering principals by behavior. The implementation is single-linkage over eleven behavior-only features, with three constraints that shaped it:

Volume is excluded from the distance. Two principals sending 500 requests each are not alike merely for that; clustering on volume groups the busy with the busy, which is not a behavioral statement.

The linkage threshold comes from the population, not from a constant. It is the 10th percentile of observed pairwise distances, so roughly the closest tenth of pairs link. An absolute distance would be meaningless across tenants with different traffic shapes, and a clustering that leaves everyone a singleton silently demotes every principal to its cohort baseline - the failure is invisible unless the threshold adapts.

Single-linkage over k-means, because there is no sensible *k*, the cluster count is genuinely unknown, and chaining is the desired behavior here: a principal that shifts infrastructure gradually should stay linked to where it started. The cost is that single-linkage is crude. The compensation is that the distance decomposes per feature, so a finding can state which features made two principals peers.

---

6. Detector families

Six detectors are implemented, each keyed on a different way one principal's traffic stops being coherent:

DETECTOR	CLASSIFICATION	KEYS ON
`credential_replay`	suspected credential replay	Volume share, country count, and network count that are novel against the principal's baseline
`concurrent_cluster`	concurrent location cluster	Interleaved events from separated countries and networks on one principal, across multiple days
`account_farm`	coordinated free-tier usage	Recent free-tier principals sharing pseudonymous network and client identifiers, weighted by account age
`spend_escalation`	cost escalation	Per-event cost ratio and peak-to-baseline event ratio after an onset point
`automation_like`	automation-like behavior	Interval regularity and concurrency inconsistent with an interactive client
`extraction_like`	sustained multi-model consumption	Sustained breadth across models at volume

A seventh family, relay rotation, is specified and **not implemented** - no detector, no generator scenario, no benchmark row, no tests. It is named here because the reverse-proxy case in Section 9 is precisely where the current detectors are weakest, and a specification with no implementation behind it should not be counted as coverage.

7. Two checks you can run now

Both checks are actionable against telemetry the gateway already emits, with no change to the request path. The sketches below are illustrative pseudocode. In practice the assessment runs as a streaming job over request telemetry at data-lake scale, not as a point-in-time query.

Check 1 - traffic on a rotated credential (deterministic)

Over the request telemetry, match any request authenticated with a rotated or deprecated credential after that credential's rotation time. A retired credential should generate no traffic, so any match - after a post-rotation grace window, and for hard-revoked credentials - is a high-confidence indicator of a retained, unauthorized copy. Investigate by source IP. Section 9 gives the conditions and exclusions.

```
-- illustrative pseudocode (deployed as a streaming match, not a query)
SELECT credential, source_ip, ts
FROM   request_telemetry
WHERE  credential IN rotated_set
  AND  ts > rotated_at(credential)
```

Check 2 - deviation from a credential's own baseline (behavioral)

For each active credential, compare the recent distribution of the Section 4 features against that credential's own established baseline, and surface credentials that deviate across several features at once.

```
-- illustrative pseudocode; naive form, see caveat below
FOR EACH credential OVER trailing_window:
  profile  = { distinct source_ip, distinct geo, distinct model,
               request_rate, burstiness, hour_of_day_distribution }
  IF profile deviates from baseline(credential) ACROSS several features:
     FLAG credential
```

> **A fixed threshold on Check 2 will bury you in false positives.** A legitimately distributed service - multi-region, many clients, CI fleets - trips it immediately. The value is in the per-credential adaptive baseline and the scoring across features, which is what Section 5 specifies.

Because the inputs are metadata only, both checks run without exposing prompt content, credentials, or request payloads.

What the output is for

These checks are detection, not response. Their output - a ranked set of suspect credentials with the source IPs and behavior on each - feeds enforcement that already exists: credential revocation, gateway rate-limiting or allow-list tightening, and SIEM alerting on the identified principals. The method identifies what to act on; the acting is done by the controls already in place.

---

8. Evaluation

The detectors are evaluated against a synthetic corpus: 284 cases over 10 seeds, 20 adversarial variants, and 20 benign variants drawn from a small number of templates. The benchmark reports three different recall numbers, and the distinction between them is the point.

- **Self-consistency recall** - positives generated by the same code path the detectors were tuned against. It measures internal consistency, *not* accuracy, and is reported under that name so it cannot be quoted as a detection rate.
- **Held-out recall** - positives built to clear the specification without being the fixtures the detector was tuned on.
- **Evasive recall** - variants constructed specifically to defeat the detector while remaining the same abuse.

DETECTOR	HELD-OUT RECALL	EVASIVE RECALL	PRECISION	FALSE-POSITIVE RATE
`account_farm`	1.00 (60/60)	1.00 (19/19)	1.00	0.000
`automation_like`	1.00 (8/8)	1.00 (3/3)	1.00	0.000
`concurrent_cluster`	1.00 (8/8)	1.00 (1/1)	1.00	0.000
`extraction_like`	1.00 (8/8)	**0.67 (2/3)**	1.00	0.000
`credential_replay`	1.00 (8/8)	**0.50 (1/2)**	1.00	0.000
`spend_escalation`	1.00 (8/8)	**0.50 (1/2)**	0.71	0.023

How to read this, including the parts that do not flatter the method.

Held-out recall of 1.00 across all six is weak evidence, not strong. Most detectors are carrying eight held-out positives, and four of the six evasive scores rest on one to three variants. A perfect score on a handful of cases is consistent with a detector that generalizes and with one that does not; the corpus is too small to separate them. The number to trust least is a 1.00 with a denominator of 1.

Three detectors still lose to evasion, and the failures are structural rather than tuning. `extraction_like` misses a principal pulling 1.2M output tokens through a single model: the detector needs at least two, so breadth is the signal and a narrow workload has none. `credential_replay` misses a key replayed from exactly one new country, because geography carries too little on its own to fire against a clean baseline. `spend_escalation` misses a ramp spread across eight days, which is the slow-ramp blind spot of Section 9 reproducing itself in measurement: an adaptive baseline absorbs a gradual change by design.

The three that now hold were rebuilt, not retuned. An earlier version of this table reported `account_farm` at 0/8, `automation_like` at 0/1 and `extraction_like` at 0/1. Each failed the same way, on a single gate over a parameter the attacker sets. Farms were grouped on the exact `(network, client)` pair, so one client id per account turned a farm of eight into eight groups of one; grouping now runs per identifier, and that looser grouping is paid for with a coherence test so that five colleagues behind one office NAT are not reported as a farm. Automation was gated on deviation from a single median interval, so alternating between two intervals read as irregular; regularity is now measured against the cadence modes. Extraction required three models; sustained volume on a narrow model set is now a second route to the same finding. The adversarial corpus grew from 12 variants to 20 in the same pass, which is what surfaced the three gaps above.

Zero observed false positives is not a false-positive rate of zero. The benchmark labels those cells `unfalsified_zero_observed`: across a 281–725 principal denominator per detector, no false positive was observed. That is an absence of evidence over a synthetic negative set built from six or fewer templates, and jittered clones of a few templates cannot establish a rate. No confidence interval is reported because none would be honest.

The one detector with observed false positives is the naive one. `spend_escalation` flags on cost ratios, which is the closest thing here to a fixed threshold, and it is the only detector whose precision drops - 0.71, with 12 false positives against 29 true ones. That is the false-positive argument from Section 7 reproducing itself in the measurements.

All of it is synthetic. The same process defined the baseline and injected the abuse. These numbers characterize the implementation against its own corpus; they say nothing about field performance, and no field measurement is

claimed anywhere in this document. A sensor is deployed to capture in-the-wild cases and will be reported separately.

9. Limitations

The metadata signal is probabilistic and has defined failure modes. It is a triage input, not proof.

What raises false positives

The behavioral signal depends on a change from an established, clean baseline - and many legitimate events change several features at once, in the same way abuse does.

BENIGN CAUSE	FEATURES IT MOVES
New-region launch, follow-the-sun support	Geolocation, IP, hour of day - together
CI/CD fleets with ephemeral egress	IP churn, burst
Serverless and CDN egress (Lambda, Cloud Run)	Rotating provider IP pools, geolocation
Corporate VPN or CGNAT re-pooling	IP, geolocation
A shared service account handed to a new team or vendor	IP, models, hours
Mobile users roaming across carriers and timezones	Geolocation, ASN, hours
Autoscaling	Burst, request rate
New-model or A/B adoption	Model mix

Each reproduces the flagged pattern and must be baselined or allow-listed as expected before the signal means anything. A credential that legitimately serves a fleet is already multi-modal; the method assumes a per-credential baseline that captures this, and is weaker where a credential's legitimate behavior is genuinely diverse. Geolocation is a coarse, low-confidence feature - cloud egress geolocates to a provider region, VPN to an arbitrary exit, CGNAT to a carrier city - and is weighted below model mix and volume.

Where the method is blind

It detects a change from a clean baseline, so it does not see:

- **Abuse present since the baseline was formed** - including the supply-chain case where a credential is harvested at or near creation, so the abuse *is* the baseline. In this respect the LiteLLM scenario is the hardest case, not the easiest.
- **An attacker who mimics the owner** - same region, models, and cadence, at low volume.
- **A slow-ramp attacker** who shifts an adaptive baseline gradually.

- **Resale fronted by a reverse-proxy relay**, which degrades the primary signal: the gateway sees the relay's few stable datacenter IPs, not the downstream buyers', so the multi-IP and geolocation features largely collapse. Only residual signals remain - aggregate volume ceiling, model mix, and a flattened 24-hour curve.
- **An attacker who launders egress on purpose** - residential-proxy networks, rotating VPN exits, cloud IP pools, or traffic paced to look human. This defeats the network features specifically: source IP, ASN, and geolocation stop carrying information. What it does not touch is what the credential is *used for* - model mix, token volume, request shape, and the 24-hour curve - which is why the method weights those above geolocation in the first place. Laundering also costs the attacker money and latency per request, so it is worth assuming for a targeted adversary and not for bulk resale.

Base rate

Abuse is rare relative to legitimate anomalies, so across a large credential population even a low per-credential false-positive rate yields a flagged set dominated by benign cases. The output is a ranked set for investigation, not a verdict; precision depends on the deployment's abuse prevalence and baseline quality.

Conditions on the rotated-credential check

Check 1 is deterministic only under conditions: after a defined post-rotation grace window, for hard-revoked credentials (not dual-key rotations or deprecated-but-valid keys), and where rotation timestamps and clocks are trustworthy. Traffic on a credential that *still authenticates* after nominal rotation means incomplete revocation, not a retained copy - exclude those first. Use it as a post-incident forensic check, not primary detection; it says nothing about credentials still in first use by an attacker.

Finally, metadata establishes that usage on a credential is inconsistent with a single principal. It does not identify the responsible party, recover request content, or on its own prove intent.

Appendix A - principal feature vector

Twenty-two numeric features, computed per principal over the analysis window.

GROUP	FEATURES
Volume and span	`event_count`, `active_days`, `span_hours`, `events_per_active_day`
Cadence	`median_interval_seconds`, `interval_mad_seconds`, `interval_regular_share`
Concurrency	`median_concurrency`, `mean_concurrency`
Payload shape	`median_input_tokens`, `median_output_tokens`, `median_token_ratio`
Cost	`total_cost`, `median_cost_per_event`
Reliability	`error_rate`
Breadth	`distinct_models`, `distinct_countries`, `distinct_networks`, `distinct_clients`
Distribution shape	`model_evenness`, `hour_evenness`, `night_share`

References

1 K. Beaumont (@GossiTheDog), thread on cyberplace.social, 2026-08-12 → 08-13. Scale (Aug 12): "Massive leak of credentials and secrets at thousands of orgs where developers executed LiteLLM, terabytes of creds are circulating online now… Threat actor = the kids at TeamPCP." Rotation finding (Aug 13): "These creds date from about March. One of the orgs impacted told me they'd rotated them all… so I tried them all. Almost every one worked. Submitted report. One of the biggest US techcos." Beaumont tested under the organization's own responsible-disclosure policy, which permits credential testing; the organization is unnamed. Breach mechanism (poisoned PyPI 1.82.7/1.82.8, malware-harvested credentials) corroborated by Hudson Rock / infostealers.com, 2026-08-12. That the harvest specifically included LLM provider credentials - not only cloud keys - is established by Trend Micro, "Inside the LiteLLM Supply Chain Compromise," 2026-03-26: the payload swept for "LLM API keys for every provider configured in the environment," so compromising a gateway host yields "not just standard cloud credentials, but LLM API keys for OpenAI, Anthropic, Azure AI, and others simultaneously." https://www.trendaisecurity.com/en-us/resources-insights/trendai-security-blog/inside-litellm-supply-chain-compromise

2 Reverse-proxy resale mechanism - CSA documents stolen credentials validated against ten AI platforms (AI21, Anthropic, AWS Bedrock, Azure, ElevenLabs, MakerSuite, Mistral, OpenAI, OpenRouter, GCP Vertex) and resold via reverse-proxy infrastructure. CSA AI Safety Initiative research note, 2026-03-15.

3 Sysdig TRT, "LLMjacking: Stolen Cloud Credentials Used in New AI Attack," 2024-05-06 - a worst-case estimate of "over $46,000 per day" in victim inference cost (Anthropic Claude 2.x, quota across four regions); initial access via stolen cloud credentials (Laravel CVE-2021-3129) to AWS Bedrock - a related credential class, not an LLM-gateway API key. https://www.sysdig.com/blog/llmjacking-stolen-cloud-credentials-used-in-new-ai-attack

4 "Operation Bizarre Bazaar" (CSA AI Safety Initiative research note, 2026-03-15) - an underground marketplace reselling access to more than 30 LLM providers at 40–60% discounts via Telegram/Discord; 35,000 attack sessions captured Dec 2025 – Jan 2026.

5 Our own internet scan for exposed LiteLLM gateways - 2,659 reachable hosts (Netlas, 2026). A surface measurement of reachable hosts, not evidence that any host was compromised.

6 Named token-broker and resale storefronts have been catalogued in public security research (e.g. Vectoral, "Who Are the Token Brokers?"). Secondary source; the underlying marketplaces are deliberately not linked.

7 D. Pöhn, H. Lüken, "Got Ya! - Sensors for Identity Management Specific Security Situational Awareness," arXiv:2503.04274 (2025-03-06). Standard logging confirms a successful login but cannot determine whether the authenticated user is the legitimate account owner.

8 S. Lin, Y. Luo, Z. Zhu, Y. Wang, B. Qiu, "A Non-Intrusive Traffic Analysis Framework for Authorization Risk Detection and Coordinated Response in Web Applications," arXiv:2607.16754 (2026-07-18). Detects authorization abuse by an authenticated principal from traffic metadata and behavioral anomalies, without inspecting payloads.

9 M. Nayebi Kerdabadi, W. A. Byron, X. Sun, A. Iranitalab, "Spatio-Temporal Directed Graph Learning for Account Takeover Fraud Detection" (ATLAS), arXiv:2509.20339 (2025-09-24). Detects account takeover from session metadata (account, device, IP, time) on a graph of 100M+ nodes; reports +6.38% AUC and >50% reduction in customer friction.

10 E. Tekiner, A. Acar, A. S. Uluagac, E. Kirda, A. A. Selcuk, "SoK: Cryptojacking Malware," arXiv:2103.03851 (2021). Systematization of cryptojacking - unauthorized use of a victim's computing resources to mine cryptocurrency at the victim's cost. Cited as the closest studied analog to stolen-inference-credential abuse, not as direct evidence of it.

11 D. Tanana, "Behavior-Based Detection of GPU Cryptojacking," arXiv:2408.14554 (2024). Detects cryptojacking from behavioral resource-usage signals (GPU load, memory) rather than from the credential.