SBSeg 2026 Armação dos Búzios - RJ
VEJA OS SELOS DE ARTEFATOS

Context-Aware SIEM Rule Generation with LLMs: When Site Profiles Are Not Enough

Priscila Schafhauzer[1], **Cristhian Kapelinski**[1], **Marcio Pohlmann**[1], **Diego Kreutz**[1]

[1]AI Horizon Labs, PPGES, Federal University of Pampa (UNIPAMPA)

{priscilaschafhauzer,cristhianavila}.aluno@unipampa.edu.br
marcpohl@gmail.com, diegokreutz@unipampa.edu.br

Abstract. *Security Information and Event Management (SIEM) platforms ship rulesets that are agnostic to the monitored organization, so a failed login receives the same severity in every environment. We investigate whether an LLM, conditioned only on a short organization profile and a catalog of native Wazuh rule identifiers, can write local rules that bring automated severity classification closer to human analyst judgment. On* 1,000 *stratified SSH authentication events from two production servers, the LLM-augmented configuration lowers accuracy by* ***4.4 percentage points*** *and weighted* F_1 *by 3.1 points relative to the native baseline. We trace the regression to a single failure mode, the over-escalation of unsuccessful login attempts from external sources, and discuss why aggregated* F_1 *is a coarse proxy for rule quality.*

1. Introduction

Organizations use Security Information and Event Management (SIEM) platforms to collect security events and direct analyst attention across large computing fleets. Wazuh (an open-source security monitoring platform) reports more than 15 million protected endpoints and 100,000 enterprise users.[1] At this scale, generic severity decisions affect which events reach a Security Operations Center (SOC), so poorly matched rules can create noise or divert attention from consequential activity.

Wazuh descends from Open Source Security (OSSEC), a host-based intrusion detection system, and ships thousands of correlation rules organized in numeric severity levels [Hay et al. 2008]. These generic rules react to log patterns without knowledge of the monitored environment. In the evaluated ruleset, every successful SSH authentication matches rule 5715 at level 3 and therefore generates a low-severity alert, even when the event is routine under the modeled policy.[2] Conversely, an isolated attempt with an invalid username matches rule 5710 at level 5, although the modeled analyst treats such Internet noise as low severity. These defaults are not wrong in the abstract, but they may be poorly matched to a site's policy. Writing local rules by hand to express each organization's expectations is laborious, and recent surveys show that existing detection systems still cover only part of the threat landscape [Hindy et al. 2020].

Recent work shows that LLMs can synthesize or analyze detection logic from descriptive inputs. LLMCloudHunter compiles Cyber Threat Intelligence (CTI) reports

[1]Wazuh, About us, 2026.
[2]Wazuh, rule matching and alert levels, 2026.

into Sigma rules (a vendor-neutral detection format) [Schwartz et al. 2024]. RuleGenie detects redundant or overlapping rules in a deployed SIEM and recommends keep, merge, or drop actions [Shukla et al. 2025]. Rule-ATT&CK Mapper (RAM) maps existing SIEM rules to the MITRE Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK) framework [Wudali et al. 2025]. Other frameworks assess the precision and cost of LLM-authored detections [Bertiger et al. 2025]. These methods use threat content, framework mappings, or existing rules rather than the profile of the organization that will run the rules.

We ask a different question: *given only a short organization description and the catalog of native rule identifiers it inherits, can a present-day LLM write Wazuh local rules that better reflect what an analyst from that organization would label as important?* Although the result is negative for the evaluated configuration, the observed failure identifies methodological requirements for assessing LLM-generated SIEM rules. We present a empirical evaluation, comprising a labeled dataset, a generation pipeline, and an analysis of failure modes, to test whether the pipeline is feasible, where it breaks first, and how its output should be assessed. The experiment yields a negative result, a 4.4-percentage-point accuracy drop concentrated in a single explainable failure mode (Section 5). We release the dataset, prompts, generated rules, and evaluation scripts for independent re-execution.[3]

2. Related Work

Table 1 positions our work against prior literature along three axes: the approach each work takes, whether it generates a deployable SIEM rule (Rule gen. column), and the evaluation lens it uses (Eval column). Building-block systems discussed in the prose do not appear in the table.

Table 1. Positioning against prior work. Rule generation denotes a deployable rule; evaluation abbreviates precision/recall as P/R and area under the receiver operating characteristic curve as AUC.

Work	Approach	Rule gen.	Eval
[Du et al. 2017]	Recurrent anomaly detection over log sequences	✗	P/R/F_1
[Guo et al. 2021]	Masked log-key prediction	✗	P/R/F_1
[Han et al. 2023]	Generative log-sequence anomaly scoring	✗	P/R/F_1
[Najafabadi et al. 2015]	Classical machine learning on NetFlow for SSH brute force	✗	AUC
[Schwartz et al. 2024]	LLM compiles Sigma rules from cloud CTI text	✓	P/R/F_1
[Shukla et al. 2025]	LLM detects redundant rules across pairs	✓	P/R
[Wudali et al. 2025]	LLM maps SIEM rules to MITRE ATT&CK techniques	✗	P/R
[Bertiger et al. 2025]	Eval framework for LLM-authored detections	✓	precision+cost
This work	**LLM writes Wazuh rules from site profile**	✓	**F_1 vs gold**

The closest work applies LLMs to detection rules. Schwartz et al. present LLMCloudHunter, which compiles Sigma rules from CTI reports [Schwartz et al. 2024]. RuleGenie detects redundant rules [Shukla et al. 2025], while RAM maps rules to MITRE ATT&CK [Wudali et al. 2025]. Bertiger et al. evaluate LLM-authored detections [Bertiger et al. 2025]. These systems use threat content or existing rules rather than the monitored organization's profile.

A parallel line builds anomaly detectors over log sequences. DeepLog uses

[3] https://doi.org/10.5281/zenodo.22664934

a recurrent network [Du et al. 2017]; LogBERT uses masked prediction with a bidirectional Transformer [Guo et al. 2021]; and LogGPT uses a generative Transformer [Han et al. 2023]. These systems emit anomaly scores rather than deployable rules. Najafabadi et al. detect SSH brute force from NetFlow (a network flow monitoring protocol) records as a binary task [Najafabadi et al. 2015]; we reuse SSH but evaluate multi-class severity.

In summary, prior LLM work compiles rules from threat content or analyzes existing rules. We instead condition generation on the monitored *site* and evaluate against a per-pattern analyst gold standard.

3. Methodology

Figure 1 summarizes the study as four stages. First, we collect and sample real authentication logs (Section 3.1). Second, we define the organization profile and label every event manually (Sections 3.2 and 3.3). Third, we generate local XML rules from native rule identifiers (IDs) with an LLM under four prompt variants (Section 3.4). Finally, we replay the same events through each configuration and score the resulting severities using precision (P), recall (R), and F_1 against the manual labels (Section 4).

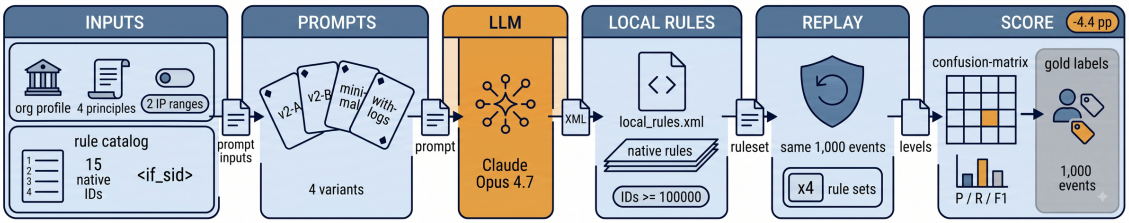


Figure 1. Evaluation pipeline: from organization profile to per-class metrics.

3.1. Data Collection and Stratified Sampling

The raw data are `/var/log/auth.log` entries from two SSH-exposed research servers at the Federal University of Pampa (UNIPAMPA), collected over five consecutive days (May 17–21, 2026) and totaling roughly 325,000 lines. A uniform random sample would be dominated by noise (about 40% `systemd-logind` sessions and 20% pre-authentication closures), so we draw a stratified sample of 1,000 lines. Quotas are sized so that each security-relevant behavior has enough examples for per-class metrics: 198 successful logins, 200 invalid-user attempts, 150 failed passwords, 45 privilege escalations (`sudo`/`su`; a full census of every such event in the window), 123 disconnects, and the remainder of session and connection noise. From each stratum we select every k-th line in timestamp order. Because the selection is deterministic, the sample covers the full collection window and is reproducible without a random seed.

The released sample replaces IP addresses with disjoint RFC 5737 ranges, usernames with surrogates (except `root`), hostnames with `srv01`/`srv02`, and SSH key fingerprints with synthetic SHA-256 strings. It records external-origin traffic on both hosts. Among the 198 successful logins, `srv01` contributes 27, including one from outside the institutional ranges, while all 171 from `srv02` originate within those ranges. These observations describe the sample, not the access-control policy of either host or of UNIPAMPA as a whole.

3.2. Organization Profile and Severity Scale

Separately from the source hosts, we define a hypothetical organization profile inspired by an academic environment. It is not a description of UNIPAMPA's infrastructure or of the access policy of the two hosts. The frozen prompt states that the modeled server has a public IP address, receives continuous external traffic, operates without business-hour restrictions, and accepts legitimate SSH access exclusively from in-country institutional IPv4 ranges, which we call R. The four principles supplied to the LLM are: geography is the strongest signal (access outside R is anomalous); time of day does not elevate severity; effective impact outweighs unsuccessful attempts; and volume from the same source is significant. Severity uses Wazuh's numeric levels, bucketed as `low` (1–4), `medium` (5–8), `high` (9–12), and `critical` (13–16). Events for which no rule fires are labeled `none`, capturing pre-authentication noise and routine bookkeeping that classifiers must *not* escalate.

3.3. Manual Ground Truth

We assign one severity label per event, and events sharing the same pattern receive the same decision. A successful login from an institutional IP is `none` (routine); from an in-country, non-institutional IP it is `medium` (out of pattern); from an out-of-country IP it would be `critical`, although no such login appears in the sample. We did not inject synthetic out-of-country successes because the ground truth is intended to represent observed events rather than constructed incidents. An isolated `invalid user` is `low`; a sustained sequence (≥ 10 per host) or a denied-user attempt (native rule 5712) is `medium`. Escalation to root via `su`/`sudo` is `high`. Pluggable Authentication Modules (PAM) session open/close and disconnects are `none`. The resulting distribution is 644 `none`, 92 `low`, 249 `medium`, 15 `high`, and 0 `critical`.

The labels were withheld from the LLM: no label appears in any prompt, and the model had no access to the ground truth at any point of the generation.

3.4. Rule Generation with the LLM

Rules are generated by Claude (Opus 4.7) in isolated sessions. The base prompt (version 2) is frozen across all sessions. It carries a persona, the organization profile and four principles, the severity scale, a catalog of 17 native Wazuh rule identifiers in 15 entries (SSH/PAM/su) for use in `<if_sid>`/`<if_matched_sid>`, and the syntactic conventions of local rules: identifiers at or above 100000, which separate generated local rules from the native ruleset; `frequency`/`timeframe`; and `<same_srcip/>`. The prompt deliberately omits per-scenario severity mappings and log examples. The model must *infer* which behaviors deserve rules and at which level. Output is XML inside a single `<group>`.

We run two ablations: *minimal* removes the syntactic few-shot (the worked XML rule example) but keeps the catalog, and *with-logs* appends four sample log lines. We also run the base prompt twice (runs A/B) to measure inter-run variability. Each XML output is loaded as `local_rules.xml` on top of the native ruleset.

4. Experimental Setup

Figure 2 shows the scenario. The Wazuh stack runs in Docker (single-node v4.14.5) on a host separate from the monitored servers. The sample is injected by appending lines to a

`localfile` that the manager continuously reads; ingestion is identical to agent delivery. With `logall_json=yes`, every decoded event is recorded in `archives.json` even when no alert is generated, giving a label for the 644 `none` cases.

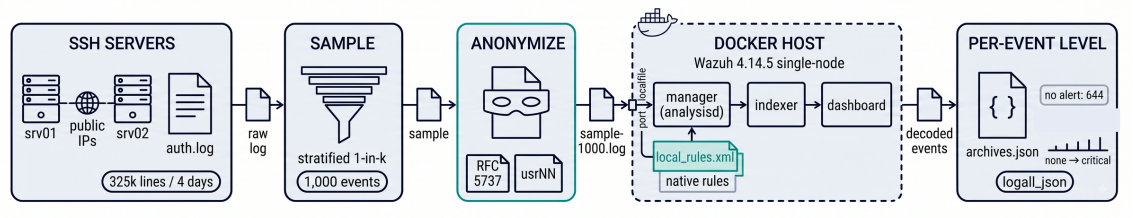


Figure 2. Instantiated scenario: log collection, anonymization, and the Dockerized Wazuh stack.

For each prompt variant we (i) replace `local_rules.xml` (bind-mounted, swappable without rebuild), (ii) truncate archives, (iii) restart the manager, (iv) re-inject the same 1,000 lines, and (v) map each line to the reported level. The raw log line serves as the join key across runs. Reported metrics are per-class precision, recall, F_1, accuracy, macro F_1, and support-weighted F_1 over the five labels.

5. Results and Discussion

The native ruleset agrees with the analyst on most elevated events but surfaces a large volume of routine activity. Table 2 and the confusion matrices in Table 3 summarize the native Wazuh classification against the analyst gold standard: accuracy 0.633 and weighted F_1 0.695. Recall for `high` (0.800) and precision for `none` (1.000) are strong; `su`/`sudo` events the analyst marks as elevated are surfaced by the engine, and every event Wazuh suppresses is one the analyst also considers routine. The weak point is the recall of `none` (0.581): 270 events the analyst considers routine (chiefly institutional logins) are surfaced as low-severity alerts because they trigger native rule 5715 at level 3.

Table 2. Per-class metrics: native Wazuh vs. the LLM configuration (runs A/B).

Config.	Class	P	R	F_1	Acc.	Macro F_1	Weighted F_1
Native	none	1.000	0.581	0.735	0.633	0.486	0.695
	low	0.000	0.000	0.000			
	medium	0.729	0.992	0.840			
	high	0.923	0.800	0.857			
LLM (runs A/B)	none	1.000	0.581	0.735	0.589	0.362	0.665
	low	0.000	0.000	0.000			
	medium	0.693	0.815	0.749			
	high	0.203	0.800	0.324			

The four LLM configurations behave almost identically, and all *lose* ground to the native baseline. Accuracy drops by 4.4 percentage points to 0.588–0.589, weighted F_1 by 3.1 percentage points to 0.664–0.665, and macro F_1 to 0.360–0.362. Inter-run variability under the frozen prompt is zero; runs A and B produce identical classifications, and the two ablations differ by a single event ($|\Delta| \leq 0.004$). The effect is concentrated in one failure mode, visible in Table 3: 45 events the analyst considers `medium` are escalated to `high`, and one to `critical`.

The model treats "geography is determinant" as a global multiplier: most generated rules matching external IPs assign levels 9–14 whether or not the attempt succeeded. This overweights origin and underweights effective impact, which explains why `high`

Table 3. Confusion matrices (rows: gold; columns: prediction; the near-empty `critical` column is omitted). Left: native Wazuh. Right: LLM run A.

Native Wazuh

	none	low	medium	high
none	374	270	0	0
low	0	0	92	0
medium	0	1	247	1
high	0	3	0	12

LLM run A

	none	low	medium	high
none	374	270	0	0
low	0	0	90	2
medium	0	0	203	45
high	0	3	0	12

precision falls from 0.923 to ~ 0.20. The unchanged `none` scores expose another constraint. Because local rules are layered on the native ruleset, reproducing an analyst's `none` label for an event that natively alerts requires a level-0 child rule or equivalent suppression; removing the native rule is unnecessary. The prompt specifies syntax and severity but neither requests nor exemplifies suppression, and none of the four generated rulesets contains a level-0 rule. The result therefore characterizes the produced configurations, not the full ability of LLM-generated rules to reduce SIEM noise.

6. Limitations and Conclusions

The study has five limitations. The five-day, two-host sample contains no successful out-of-country login, leaving `critical` without support. The hypothetical organization profile is inspired by a university context but does not reproduce the heterogeneous access policies of the real hosts or represent UNIPAMPA as a whole; the experiment therefore measures agreement with that assumed policy over real events. All generations use Claude Opus 4.7, leaving inter-family variance unmeasured. Author-designed prompts and labels retain priming risk, which would tend to inflate agreement and therefore does not explain the negative result. Finally, anonymization preserves the institutional/external distinction but removes fine-grained geolocation.

The hardest question is how to evaluate generated rules. The methodological finding is that F_1 over severity buckets is a useful numerical proxy but cannot isolate the properties that matter in a SOC, and no single scalar captures their interactions. Four complementary views are needed. (i) *Held-out operational replay* should extend our fixed-sample replay to a later log window not used in rule development or labeling, and report total alerts, severity distribution, duplicate alerts, and firings per generated rule rather than only per-event agreement. (ii) *Asymmetric costs* elicited from analysts should distinguish, for example, escalating `none` to `critical` from confusing adjacent severities. (iii) *Blind expert review* should assess syntax, threshold plausibility, and operational risk even for rules that never fire in the sample. (iv) *Behavioral coverage tests* should exercise external logins, lateral pivots, slow brute force, and password spraying. The framework must also test creation, escalation, downgrade, and level-0 suppression; otherwise, additive rules cannot reproduce an analyst's `none` decisions. Improving this triangulated evaluation is more informative than optimizing aggregate F_1 alone.

Within this scope, the generated rules reduce accuracy by 4.4 percentage points and weighted F_1 by 3.1 due to over-escalation of unsuccessful external attempts and the failure to suppress native alerts labeled `none`. These results reflect the evaluated prompt and outputs, highlighting the need to compare model families, improve prompting, and validate on larger and more diverse datasets.

Acknowledgments

This work was partially supported by Capes, CNPq (Grant No. 409743/2025-9), and FAPERGS (Grant Nos. 24/2551-0001368-7 and 24/2551-0000726-1). The authors also acknowledge the use of Generative AI as a supporting tool for reviewing and improving the code, manuscript and figures 1 and 2.

References

Bertiger, A. et al. (2025). Evaluating LLM generated detection rules in cybersecurity. *arXiv:2509.16749*.

Du, M., Li, F., Zheng, G., and Srikumar, V. (2017). DeepLog: Anomaly detection and diagnosis from system logs through deep learning. In *ACM CCS*, pages 1285–1298.

Guo, H., Yuan, S., and Wu, X. (2021). LogBERT: Log anomaly detection via BERT. In *IJCNN*.

Han, X., Yuan, S., and Trabelsi, M. (2023). LogGPT: Log anomaly detection via GPT. In *IEEE BigData*, pages 1117–1122.

Hay, A., Cid, D., and Bray, R. (2008). *OSSEC Host-Based Intrusion Detection Guide*. Syngress.

Hindy, H. et al. (2020). A taxonomy of network threats and the effect of current datasets on intrusion detection systems. *IEEE Access*, 8:104650–104675.

Najafabadi, M. M., Khoshgoftaar, T. M., Calvert, C., and Kemp, C. (2015). Detection of SSH brute force attacks using aggregated Netflow data. In *IEEE ICMLA*, pages 283–288.

Schwartz, Y., Benshimol, L., Mimran, D., Elovici, Y., and Shabtai, A. (2024). LLMCloud-Hunter: Harnessing LLMs for automated extraction of detection rules from cloud-based CTI. *arXiv:2407.05194*.

Shukla, A., Gandhi, P. A., Elovici, Y., and Shabtai, A. (2025). RuleGenie: SIEM detection rule set optimization. *arXiv:2505.06701*.

Wudali, P. N. et al. (2025). Rule-ATT&CK mapper (RAM): Mapping SIEM rules to TTPs using LLMs. *arXiv:2502.02337*.