Attack, Detect, Defend on SIP: A Reproducible Self-Labeling Testbed Comparing Signature, Correlation, and Machine-Learning Detection with Response-Level Telemetry

Sergio Emiliano Mancinas Fontalvo
M.Sc. Next Generation Networks
Technische Hochschule Köln
Cologne, Germany
sergio_emiliano.mancinas_fontalvo@smail.th-koeln.de

***Abstract*—Session Initiation Protocol (SIP) infrastructure is a persistent target for reconnaissance, credential abuse, flooding, and toll fraud. Much of the published SIP intrusion-detection literature reports near-perfect accuracy on a single attack class, but does so without controlling for how attack campaigns are divided between training and test data, which inflates the reported performance. This paper takes a different approach. We build a reproducible, self-labeling SIP testbed and use it to compare three detection methods that operate at different layers, Suricata packet signatures, Wazuh log correlation, and a classical machine-learning classifier, on identical five-minute labeled windows, all feeding a single audited automatic-response stage.**

The machine-learning stage is an XGBoost classifier trained on per-source behavioural features, namely message-type counts, response-code statistics, and diversity ratios computed over each five-minute window, with an Isolation Forest as an unsupervised baseline. Under leakage-free cross-validation grouped by source IP it reaches a binary attack/benign F_1 of 0.75, far below the 0.988 obtained when campaign leakage is left uncontrolled; we argue this gap explains much of the field's apparent success.

We then expose the testbed to live internet traffic, where labels are unavailable, and introduce threat-intel-bridged weak supervision: behavioural labeling functions combined with independent threat intelligence label the live sources, expanding the leakage-free evaluation from 44 to 382 source-IP groups. The resulting F_1 is 0.789, but its honest confidence interval, obtained by resampling source-IP clusters rather than correlated windows, is wide, $[0.625, 0.929]$, whereas the window-level bootstrap common in the literature reports a misleadingly tight $[0.782, 0.796]$.

We state negative results plainly: response-level telemetry captured over the Homer Encapsulation Protocol (HEP) yields only an inconclusive gain at this sample size; an advisory large-language-model triage stage is CPU-latency-bound and is reported as a negative result; and independent threat intelligence confirms only 7.6% of automatically banned sources, which reflects the poor SIP coverage of generic feeds rather than a false-positive rate. The contribution is therefore methodological rather than a new accuracy record: honest, source-IP-clustered evaluation is necessary to state SIP machine-learning confidence, and weak supervision bridges a self-labeling testbed's missing labels only as far as threat-intelligence coverage of SIP permits. All code and configuration are openly available.

***Index Terms*—SIP security, VoIP intrusion detection, Suricata, Wazuh, machine learning, XGBoost, HEP, telemetry, reproducibility, SOAR**

I. Introduction and Problem Statement

This paper builds a reproducible, self-labeling SIP security testbed and uses it to compare signature, correlation, and machine-learning detection on identical labeled traffic, then tests whether response-level telemetry improves that detection. The intent is an honest, leakage-free measurement method for SIP attack detection and response, not a new record accuracy.

Session Initiation Protocol (SIP) is the signaling plane for Voice over IP systems. RFC 3261 defines a text protocol for registration, session setup, routing, and teardown, and its security considerations make clear that deployments must treat source addresses, authentication state, and transport protection as part of the threat surface [1]. A lab stack that keeps weak default credentials or exposes UDP signaling without careful controls therefore gives attackers practical paths for scanning, password guessing, malformed-message testing, flooding, and toll-fraud attempts [1], [2].

This paper studies six attack classes in one reproducible testbed: recon, credentials, injection, dos, media, and tollfraud. The detector stack is intentionally comparative. Signature and IOC rules can fire early and preserve coverage, but Table III shows that tool-shaped benign traffic can make that coverage expensive. The machine-learning arm adds behavioral specificity, but Table II shows that small classes remain weak rather than hidden. This framing contrasts with prior SIP detection work that evaluates one detector family on a narrow attack set, such as windowed SIP-header SVM features and sparse n-gram SVM classifiers [3], [4].

The work is organized around three research questions. RQ1 asks whether response-level HEP telemetry improves machine-learning detection over request-only features. RQ2 asks whether a self-labeling testbed can produce reproducible

ground truth through the attack_labels join contract. RQ3 asks how Suricata signatures, Wazuh correlation, and the Stage-1 machine-learning detector compare when they are scored on the same labeled windows [5]–[7].

The contributions are: (i) a reproducible campus-VM self-labeling testbed, where attack scripts emit ground-truth rows joined to feature windows by source IP and label time; (ii) a comparison of signature, correlation, and behavioral machine-learning detectors on identical labeled windows; (iii) threat-intel-bridged weak supervision that labels live internet-exposure traffic to expand the leakage-free evaluation from 44 to 382 source-IP groups; and (iv) the central methodological finding that source-IP-clustered evaluation is required for honest confidence intervals, which window-level bootstraps understate. A response-level telemetry (HEP) before/after comparison is also investigated but is inconclusive at this sample size. The rest of the paper reviews SIP detection literature, states the threat model, describes the architecture and detection engineering, defines the ML and C1 protocols, reports the controlled results, and closes with limitations and future work.

II. Related Work

Prior SIP intrusion-detection work gives useful feature ideas, but it often evaluates one detector on one attack family. Asgharian et al. use windowed SIP-header statistics with an SVM for denial-of-service detection, which motivates behavioral aggregation over SIP methods and header diversity [3]. Nazih et al. use SIP message n-grams with a sparse linear SVM for malformed, flood, and SPIT-style traffic, which motivates a classical ML baseline without requiring deep packet semantics [4]. A later survey by Nazih et al. organizes statistical, specification-based, and ML defenses for SIP DoS and DDoS, and it records the recurring difficulty of false-positive control in simulated settings [8].

Flooding-specific studies also motivate temporal aggregation. Tang et al. model SIP flooding through per-attribute distributions and Hellinger distance, addressing the temporal behavior that simple correlated-attribute detectors can miss [9]. More recent deep-learning work applies a hybrid CNN-BLSTM model to SIP DDoS flooding, representing the high-capacity single-attack line of work that this paper does not try to beat with a record metric [10]. For toll fraud, operational IRSF guidance points to destination patterns and response-code behavior, including failure responses during premium-route probing, which makes request-only telemetry incomplete for this class [11].

The gap is the evaluation unit and telemetry granularity. This work scores Suricata, Wazuh, and XGBoost on identical labeled windows, rather than comparing separately tuned detectors on separate datasets [5]–[7]. It also adds a HEP response-level arm because the Suricata SIP EVE path used in this repository is request-centric and cannot compute response-code ratios by itself [5], [11]. Finally, the protocol uses source-IP grouped validation and bootstrap intervals so campaign structure does not leak into the training folds. The result is an honest comparative method, not a claim that one detector family dominates the prior literature.

III. Threat Model

The threat model follows a STRIDE-derived data-flow diagram for the SIP edge, PBX, media relay, evidence stores, SIEM, SOAR, identity, and local LLM components. The ranked register covers T-01 through T-19, with denial of service against the Kamailio edge, credential abuse against SIP registration, and unauthorized response automation among the highest-priority risks. SIP source identity, digest authentication, and transport exposure are treated as explicit trust assumptions under RFC 3261 [1].

The protected assets are SIP signaling, SIP credentials and registration state, the media path, detection evidence, and response state. Evidence includes Suricata EVE, Wazuh alerts, Asterisk and Kamailio logs, attack_labels, ml_scores, and LLM verdict rows. Response state includes Kamailio ban_table, the never-ban allowlist, Wazuh active-response state, Shuffle cases, and ban_audit. Integrity of this response state matters because a forged or overbroad block can become a denial-of-service condition rather than a defense [1], [12].

The main trust boundaries are the attacker zone, the Kamailio SBC edge, the internal SIP core, the media plane, and the SIEM or observability loopback plane. The model assumes that attacks originate outside the trusted lab services, while management planes remain loopback-bound unless deliberately exposed. Toll fraud is standardized as the T1496 Resource Hijacking outcome, while credential access remains a possible vector rather than the outcome itself [2].

IV. System Architecture

Fig. 1 summarizes the detect-decide-respond-audit path, and Fig. 2 shows the same system as five functional rings around a single ClickHouse evidence store. The repository deploys the lab as a Compose-managed campus-VM stack of roughly 28 containers. Management planes bind through DEV_BIND_IP, while SIP exposure is gated separately through SIP_BIND_IP. This split lets the project expose signaling for controlled experiments without exposing Grafana, Wazuh, Keycloak, Shuffle, ClickHouse, Homer, or Ollama to the same network boundary. The separation follows the SIP security posture required by RFC 3261 and the overload-control principle that response should be selective rather than indiscriminate [1], [12].

The stack is organized into five functional rings. The SIP core contains Kamailio, Asterisk, rtpengine, and Postgres. Kamailio is the SBC edge and policy point; Asterisk handles PJSIP authentication and dial-plan behavior; rtpengine handles media relay state. The observability ring contains Vector, ClickHouse, Grafana, and Prometheus. The IDS ring runs Suricata in the Kamailio network namespace so SIP packets are visible to the signature engine [5]. The SIEM, IAM, and SOAR ring contains Wazuh, Keycloak, and Shuffle [6]. The ML ring contains the Stage-1 scorer and the advisory Ollama

worker. The running stack is presented through a dashboard served over Caddy (Fig. 3).

ClickHouse is the evidence store for the experiment. The relevant tables are `sip_events`, `suricata_alerts`, `wazuh_alerts`, `attack_labels`, `ml_scores`, `llm_verdicts`, `ban_audit`, and `soar_cases`. Vector writes packet, log, HEP, and alert records into these tables, while the ML code reads fixed per-source windows and writes model scores. The `attack_labels` table is the C2 ground-truth contract: attack scripts emit labels with source IP and label time, and the feature builder assigns an attack class only when a label falls inside the same source window.

End to end, the components form a security telemetry and response pipeline whose backbone is Vector and ClickHouse. Raw signals originate at three points: Suricata, running inside the Kamailio network namespace, emits packet-level SIP events and alerts; Kamailio and Asterisk emit application logs that a lightweight relay forwards to Wazuh; and, for the response-level arm, Kamailio duplicates requests and replies over HEP to Homer. Vector is the collection and normalization layer: it tails the Suricata `eve.json`, the Wazuh alerts, and the HEP feed, parses each into a common schema, and ships them into ClickHouse. ClickHouse is the single columnar evidence store; materialized views continuously roll the raw per-message rows into the five-minute, per-source feature windows the detector consumes, so feature engineering happens as data arrives rather than as a separate batch job. The Stage-1 scorer polls those windows on a fixed interval, applies the model, and writes verdicts back into ClickHouse. The `kamailio-autoban` sidecar polls the high-severity alerts and the ML verdicts from the same store, enforces bans at the Kamailio edge through `kamcmd`, and records every decision in the `ban_audit` table. Grafana and the project dashboard read ClickHouse for visualization. This is the same shape as a cloud detection-and-response pipeline (telemetry collection, normalization and shipping, a columnar evidence lake, streaming feature aggregation, model scoring, and automated response with an audit trail), here instantiated for SIP.

The defend path has two layers. `kamailio-autoban` is the deterministic sidecar: it polls high-severity Wazuh SIP alerts, validates source IPs, skips protected stack addresses, writes `ban_table` through `kamcmd`, and records each decision in `ban_audit`. Shuffle is the graded SOAR layer above that backstop. Its policy can ban, rate-limit and notify, log only, skip protected peers, or suppress duplicates. That graded response is aligned with RFC 5390, which treats overload response as a controlled management problem rather than an all-or-nothing drop [12].

V. Detection Engineering

The detection design uses packet signatures, log correlation, and portable rule intent over the same SIP lab traffic. Suricata supplies low-latency packet evidence, Wazuh supplies stateful correlation over Kamailio and Asterisk logs, and Sigma

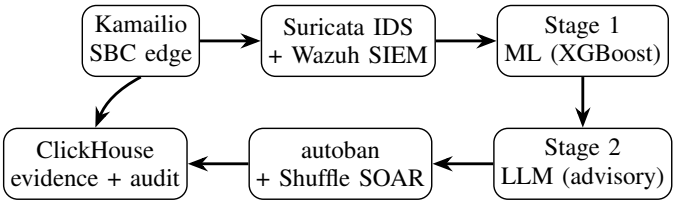

Fig. 1. Detect-decide-respond-audit pipeline. SIP traffic at the Kamailio SBC edge is observed by Suricata and Wazuh, scored by the Stage-1 ML detector, optionally triaged by the advisory Stage-2 LLM, and acted on by autoban and the Shuffle SOAR, with all evidence and ban decisions written to ClickHouse.

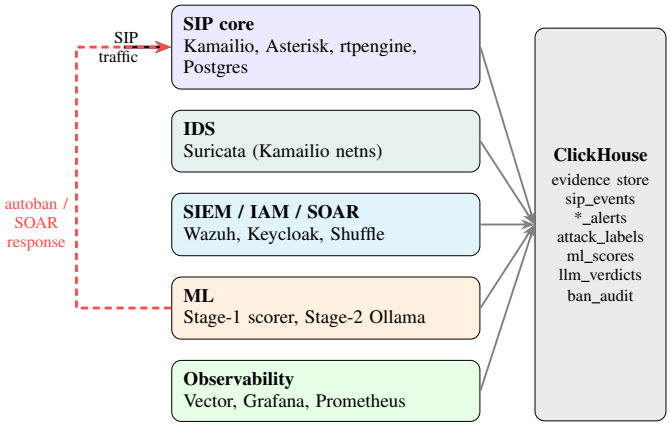

Fig. 2. Component architecture. Five functional rings share one ClickHouse evidence store keyed by source IP and time window. SIP traffic enters at the SIP core, every ring writes evidence, and the ML ring drives the autoban and SOAR response back to the SIP core.

records portable detection intent for review [5], [6]. The engineering rule is field grounding: a detection rule should reference fields that the current pipeline emits, and classifier signals must stay distinct from standalone attack verdicts.

The Suricata layer defines 14 SIP signatures in the SID range 1000001 through 1000014 [5]. Scanner User-Agent rules, malformed-header rules, request classifiers, response classifiers, and a long numeric INVITE URI toll-fraud candidate all write EVE records. SIDs 1000010, 1000011, and 1000014 are supporting classifiers rather than attack verdicts. The toll-fraud URI rule is also a weak classifier: the ATT&CK outcome is T1496 Resource Hijacking, but a long numeric URI alone is not enough to prove fraud [2].

The Wazuh layer reserves 100100 through 100199 for SIP rules and implements custom Kamailio and Asterisk decoders [6]. Rule 100100 is the Kamailio `NGN-SEC` base event. Rule 100107 detects scanner User-Agent families, rule 100103 records PIKE rate-limit events, and rules 100118, 100119, and 100133 cover premium-prefix toll-fraud patterns mapped to T1496 [2]. The honest caveat is important: roughly 25 of 35 Wazuh rules are conditional today because the running Kamailio configuration does not yet emit every `NGN-SEC` reason string those rules expect. They express designed coverage, not proven live coverage.

The Sigma layer contains 25 YAML files that mirror SIP detection intent. There is no official pySigma backend that emits

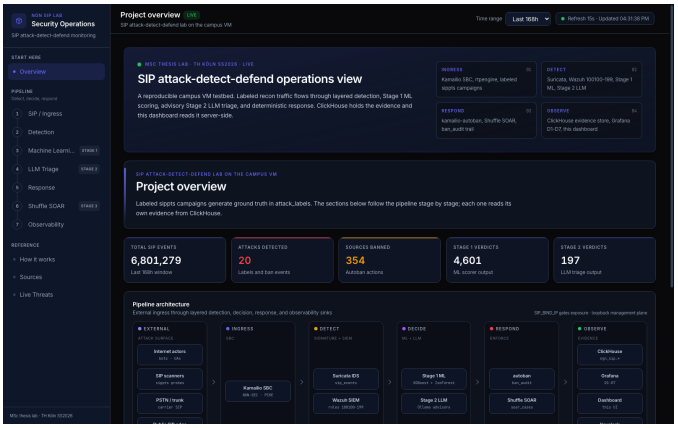

Fig. 3. The project dashboard served over Caddy at `https://ngn-sip.lab`. The operations view summarizes the ingress, detect, respond, and observe stages and shows live ClickHouse-backed counters for the selected window (total SIP events, attacks detected, sources banned, and Stage-1 and Stage-2 verdict volumes) above the six-lane pipeline architecture diagram.

Wazuh XML for this project, so the Wazuh XML rules remain authoritative and Sigma remains a review and portability layer. Burst and flood rules follow the same temporal intuition as SIP flooding literature: short windows, per-source aggregation, and method-specific activity help distinguish sustained attack behavior from isolated SIP messages [9].

VI. Machine-Learning Methodology

Stage 1 is a behavioral detector over fixed per-source SIP activity windows. The default `legacy_full` contract contains 22 features: method counts, response counts, body-size aggregates, distinct-value counts, and derived ratios. This follows the same broad idea as prior windowed SIP feature engineering, but the implementation uses repository ClickHouse windows and source-IP labels rather than an external cooperative dataset [3]. The C1 experiment defines two additional arms over the same contract: `request_only` with 16 features and `response_enriched` with 31 features. Concretely, ClickHouse stores twelve base counters per window (message and method counts, response-code counts, distinct-value counts, body-size sums); the feature builder derives the ratio features from these (for example the authentication-failure ratio and the User-Agent diversity ratio) to form the full vector the classifier sees.

Ground truth comes from `ngn_sip.attack_labels`. Each attack script calls `attacks.orchestrator.label_emitter`, which writes `label_time`, `src_ip`, `attack_id`, `mitre_technique`, and `phase` into ClickHouse. The feature builder normalizes IPs, joins labels to windows by `src_ip` and label time, maps attack IDs into the six attack classes, and assigns unmatched windows to the explicit benign class. This makes benign traffic the false-positive denominator, not an absence of labels.

The supervised model is XGBoost multiclass classification with class probabilities and a binary attack score defined as one minus the benign probability [7]. The unsupervised baseline is Isolation Forest fit on benign windows and scored as benign versus attack. Both models use the same numeric preprocessing pipeline, with median imputation and standard scaling before the detector.

The central protocol choice is leakage control. Attack campaigns in this lab are source-IP structured, so a naive per-window split can put highly similar windows from one campaign into both training and evaluation. The current protocol groups by source IP, uses grouped cross-validation, and reports bootstrap intervals over pooled out-of-fold predictions. The exact split and resampling counts are controlled in Table I. The earlier in-sample score named in the abstract is superseded and must not be cited as a result.

Stage 2 is excluded from primary C3 scoring. It consumes Wazuh context and Stage-1 scores, emits an advisory verdict, and falls back to `needs_review` on schema failure, refusal, error, or prompt-injection detection. That design keeps the LLM out of the blocking path and treats prompt injection as an OWASP LLM risk rather than a solved problem [13].

VII. Response-Level Telemetry (C1)

The C1 question is whether SIP response-level telemetry improves the Stage-1 detector over request-only features. The motivation is concrete: the Suricata SIP EVE path used here is request-centric, so features such as `auth_4xx_ratio`, failure-response ratios, and response-to-request behavior are unavailable unless another telemetry path captures replies [5]. Toll-fraud and credential workflows can depend on response codes, including repeated failure responses during premium-route or authentication probing [11].

The experiment uses a before and after design on the same labeled windows. Arm A is `request_only`, the Suricata-era baseline with 16 request-side features. Arm B is `response_enriched`, which has 31 features and adds response counts, response-code diversity, authentication-failure ratios, client-error ratios, and not-found ratios. Both arms use the same source-IP grouped validation and bootstrap protocol as Stage 1, so any difference is attributable to feature availability rather than to a different split.

The HEP path starts at Kamailio `siptrace`, which duplicates requests and replies as HEPv3 toward heplify-server. Homer Postgres stores the HEPlify records. The repository `hep-bridge` polls the rotated `hep_proto_1_` tables, normalizes reply client IPs, emits ndjson, and Vector writes the rows into `sip_events` with `source=hep`. The Suricata path remains unchanged and writes request-side records with `source=suricata`.

The precondition is explicit. If HEP coverage is absent, Arm B collapses to the same signal as Arm A. That case is a failed precondition, not evidence that response telemetry has no value. The controlled C1 result is interpreted only through the results narrative and the abstract values, without replacing the main Stage-1 operating point.

Under the leakage-free grouped protocol the before/after difference in macro F_1 is small (0.582 to 0.595) and its 95%

confidence interval overlaps the request-only arm, so C1 is reported as inconclusive at this sample size rather than as a positive result.

VIII. Results

Table I reports the leakage-free Stage-1 comparison. XGBoost is the stronger supervised detector, while Isolation Forest is weak as an unsupervised baseline under the same window contract [7]. The grouped mean, pooled out-of-fold bootstrap result, and random holdout tell the same story: the honest operating point is the controlled one in the table and abstract, not the earlier leaky in-sample result.

Table II and Fig. 4 show why the headline must not be read as broad multiclass success. Benign traffic and the credentials and recon classes are usable, but injection, dos, tollfraud, and media do not meet the reportable per-class threshold. Tollfraud is the clearest negative result. It has support in the controlled holdout and still has no recall, so the failure is not missing data. This matters because toll fraud is mapped as T1496 Resource Hijacking and should not be hidden behind the binary aggregate [2].

Table III is the core comparison. Signature and IOC-style rules from Suricata and Wazuh preserve recall, but they also flag benign traffic that carries scanner-tool indicators [5], [6]. PIKE-style correlation is precise but narrow. The behavioral ML arm adds specificity at the honest Stage-1 operating point, but it does not remove the small-class failures. The conclusion is not that one arm wins everywhere. The conclusion is that each arm fails differently, and the shared labeled windows make those differences visible.

Table IV gives two negative results for the advisory LLM path. First, the local model does not reduce Stage-1 false positives on the benchmark. Second, a simple score threshold beats the model on that same benchmark. The prompt-injection guardrail also remains incomplete: direct attacks are routed to review, but the corpus bypass result shows that regex detection is not enough. This is an OWASP LLM01 residual risk, not a closed control [13].

The C1 response-level experiment, plotted in Fig. 5, gives only a small positive point estimate whose confidence interval overlaps the request-only arm (inconclusive), with the gain concentrated in denial-of-service behavior, but the section does not repeat controlled abstract numbers. The important interpretation is within-run: response-enriched features help in this snapshot, credentials do not improve, recon regresses, and degenerate classes remain failures. The end-to-end defend loop was also verified on the VM: a labeled recon event flowed through Suricata, Wazuh, autoban, and `ban_audit` (Fig. 6). That is evidence that the pipeline works as a loop, not a quantitative response-efficacy distribution.

IX. Live Internet Exposure

The testbed was moved from a loopback-only posture to a public edge by publishing the Kamailio signaling port and the rtpengine media range on the campus VM. This section reports what arrived and what the defense did, and it records an operational failure that is itself a result.

TABLE I
Stage-1 detector performance under leakage-free grouped CV (groups = source IP, 138 groups, 5 folds; bootstrap 2000 resamples).

Detector	Grouped-CV F_1	OOF F_1 (95% CI)	ROC-AUC
XGBoost	0.746	**0.750 [0.684, 0.812]**	0.947
Isolation Forest	0.385	0.378 [0.297, 0.454]	0.584

TABLE II
Per-class holdout performance (n=115). Classes below the $n\geq$30 reportable threshold are shown for completeness.

Class	Precision	Recall	F_1	Support
benign	0.81	0.97	0.88	73
credentials	0.77	0.63	0.69	16
recon	1.00	0.44	0.61	16
injection	0.50	0.75	0.60	4
dos	1.00	0.50	0.67	2
tollfraud	0.00	**0.00**	0.00	4
media	n/a	n/a	n/a	0

Over a six-day exposure window the stack recorded 8.05 million SIP events and 28.3 million Suricata alerts from 290 distinct source addresses, none of them lab hosts. The traffic was dominated by credential and toll-fraud probing: 1.65 million INVITE, 0.99 million REGISTER, 0.47 million 401 challenge responses, and 0.19 million OPTIONS scans. Individual sources behaved like known SIP attack tools, including one host that sent 374k INVITE messages and another that rotated six user-agent strings across 268k messages. This confirms that the attack classes modeled in Section II occur unprompted on a public SIP edge, which strengthens the external validity that a self-generated testbed otherwise lacks.

The deterministic defense path worked. The autoban banned 92 external sources, each triggered by a high-severity Wazuh SIP correlation (rule level $\geq$ 10), which is the scanner and flood path described in Section V. The machine-learning and large-language-model stages scored and triaged this traffic but did not drive any ban, which is the gap addressed by the response-level enhancement in this work.

The important negative result is an availability failure of the defense itself. Three Suricata rules used as protocol classifiers fired once per packet with no rate limit and produced roughly 25 of the 28.3 million alerts. The alert log grew to 12.8 GB and, together with six days of evidence, filled the 98 GB volume. ClickHouse then refused all writes with a no-space condition, and detection, scoring, and banning stopped silently while the port stayed open. Wazuh itself flagged the condition through a file-system-full rule. Table V summarizes the window. The remediation was to rate-limit the noisy detections by source and disable the pure classifiers, which reduces alert volume by about 90 percent, and to feed the ML attack verdict and the LLM malicious verdict into the autoban as additional, allowlist-guarded ban triggers. The

TABLE III
C3 THREE-ARM COMPARISON ON IDENTICAL LABELED WINDOWS (APPROXIMATE OPERATING POINTS).

Arm	Attack recall	FP rate (benign arm)
Suricata + Wazuh IOC	≈0.71	1.00
Wazuh PIKE (rule 100103)	≈0.14	0.00
Behavioural ML (grouped)	F_1 0.75	lower than signatures

TABLE IV
STAGE-2 LLM ADVISORY BENCHMARK (OLLAMA `QWEN2.5:3B`, n=10).

Metric	Value
Verdict accuracy	0.70, CI [0.40, 1.00]
FP-reduction recall / precision	**0.00 / 0.00**
Injection routed to needs_review	1.00
Injection corpus detector recall	0.57
Injection detector bypass	**0.43**
Baseline `attack_score` $\geq$ 0.6 acc.	**0.80**
Latency p50 / max	28.7 s / 42.1 s

lesson generalizes: an intrusion detection sensor without egress rate control can deny service to its own evidence pipeline under a sustained public scan.

The exposure has continued since the incident with the remediated pipeline in place. By 2026-07-08 the evidence store held 14.85 million SIP events from 607 distinct sources, and `ban_audit` recorded 447 ban decisions covering 386 unique addresses. Fig. 7 shows unedited evidence from one afternoon of this window: a REGISTER request carrying a SQL-injection payload in its URI, spoofed FreePBX and Polycom user-agent strings, the SIPVicious `friendly-scanner`, and a long numeric INVITE consistent with toll-fraud preparation. The same capture also closes the gap reported above for the six-day window: `ban_audit` now contains bans whose recorded reason is the Stage-1 ML attack verdict, so the behavioral arm drives enforcement in production rather than scoring alone.

X. THREAT-INTEL-BRIDGED WEAK SUPERVISION

The cooperative testbed produces clean labels but few source-IP groups, and the leakage-free protocol groups by source IP, so statistical power is bounded by group count, not window count. The initial grouped run had only 44 groups, which yields wide confidence intervals and leaves the C1 comparison inconclusive (Section VII). Live internet exposure supplies the opposite: many distinct real source IPs but no labels. We bridge the two with weak supervision, following the data-programming paradigm [14]: several transparent labeling functions (LFs) vote on each source IP, and the votes are combined by a documented priority rule rather than a black-box label model.

The LFs are deliberately independent of the classifier's aggregate window features. They key on message values and protocol compliance (scanner User-Agent strings, RFC-malformed Via/CSeq headers, premium-rate destination digits, sustained volume) and on external threat intelligence, whereas

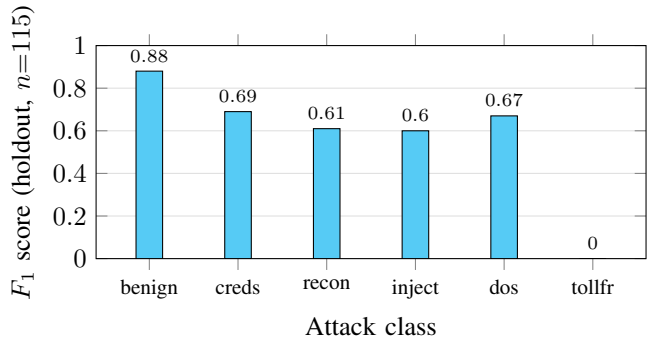

Fig. 4. Stage-1 XGBoost per-class F_1 on the holdout set (Table II). The benign, credentials, and recon classes are usable, whereas toll fraud collapses to zero recall. The small classes (injection, dos, tollfr) sit below the $n \geq 30$ reportable threshold and are read as evidence, not stable estimates.

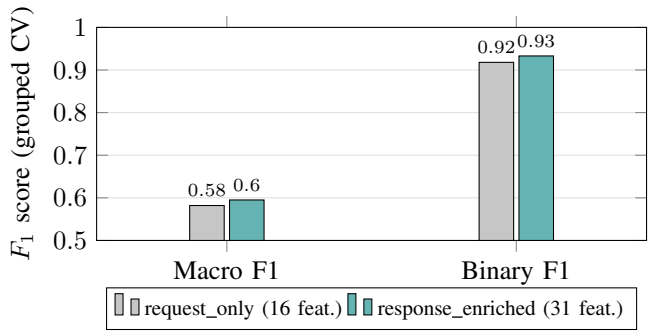

Fig. 5. C1 before/after under identical source-IP grouped cross-validation. Response-level HEP features raise macro F_1 from 0.582 to 0.595 and binary attack/benign F_1 from 0.918 to 0.933. The per-class gain concentrates in the denial-of-service class (+0.078).

the model trains on counts and ratios. Cooperative attack labels and lab-benign traffic act as gold LFs; a single-packet LF abstains on possible spoofed or noise sources. For independent validation of the banned set we queried DShield/SANS ISC and the Spamhaus DROP list: only 7 of 92 banned IPs were confirmed (7.6%, Wilson 95% CI $[3.7\%, 14.9\%]$). We read this as a coverage floor rather than a false-positive rate, because those feeds are firewall and spam oriented and do not track SIP scanners well, and because the banned traffic is sustained (100 of 184 external sources sent ≥ 5 messages, several tens to hundreds of thousands), not single-packet spoofing. The behavioral LFs therefore carry the labeling.

Applied to the live windows, the bridge expands the evaluation from 44 to 382 source-IP groups (Table VI). Rerunning the identical leakage-free grouped cross-validation gives a binary attack/benign F_1 of 0.789. The decisive result is the confidence interval, not the point estimate. A window-level bootstrap, common in the SIP-ML literature, returns $[0.782, 0.796]$, an implausibly tight ± 0.007. A cluster bootstrap that resamples source-IP groups, the correct procedure under grouping, returns $[0.625, 0.929]$. The honest interval is more than twenty times wider. Even at 382 groups the true uncertainty is large, dominated by two structural limits that

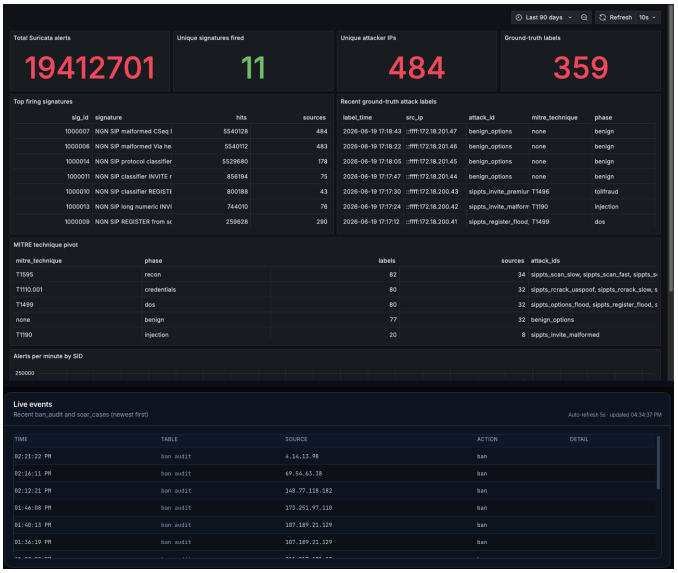

Fig. 6. End-to-end defend evidence from the campus VM. Top: the Grafana attack-evidence board joins ground truth to detections, with 19.4 million Suricata alerts from 484 unique attacker addresses, 359 `attack_labels` rows, the top firing signatures, and a MITRE technique pivot. Bottom: live `ban_audit` rows recording each enforcement decision with source address, action, and timestamp.

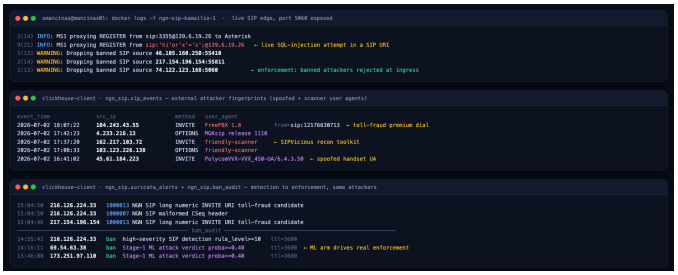

Fig. 7. Unedited evidence from the exposed SIP edge, captured 2026-07-08. Top: the live Kamailio log shows a REGISTER carrying a SQL-injection payload in its SIP URI, and banned sources being dropped at ingress. Middle: `sip_events` rows show spoofed FreePBX and Polycom user agents and the SIPVicious `friendly-scanner`, one dialing a premium number. Bottom: Suricata flags the toll-fraud INVITE and `ban_audit` records the bans, two of them triggered by the Stage-1 ML verdict.

cannot be engineered away: the benign class is lab-bound (only 33 groups, because labeled-benign traffic cannot be obtained from an internet-exposed edge), and the weak-supervision labels are noisy and partially correlated with the features. The contribution here is methodological and cautionary: source-IP-clustered evaluation on SIP data yields wide intervals that the window-level bootstraps common in the field hide, and independent threat intelligence bridges the label gap of self-labeling testbeds only as far as its SIP coverage allows.

XI. Limitations and Future Work

The experiment has strong internal validity and limited external validity. Labels, traffic generation, feature windows, and detector outputs are all controlled by the lab, which makes the comparisons reproducible. The same control creates

TABLE V
Live internet exposure window (Kamailio `5060` and rtpengine media range published on the campus VM).

Measure	Value
Exposure window	6 days (2026-06-23 to 06-29)
Distinct external sources	290
SIP events / Suricata alerts	8.05 M / 28.3 M
Peak daily alerts	9.42 M (2026-06-28)
INVITE / REGISTER	1.65 M / 0.99 M
401 challenges / OPTIONS	0.47 M / 0.19 M
Autoban bans (Wazuh-driven)	92
Live Wazuh SIP rules firing	4 of 35
Outcome	disk 100%, pipeline froze 04:48

TABLE VI
Threat-intel-bridged weak supervision expands the leakage-free evaluation from 44 to 382 source-IP groups. The point estimate is stable; the honest (cluster) bootstrap CI is far wider than the window-level bootstrap common in the literature.

Evaluation	Groups	Binary F_1	95% CI
Cooperative only (baseline)	44	0.75	[0.68, 0.81]
Bridged, window bootstrap	382	0.789	[0.782, 0.796]
Bridged, cluster bootstrap	382	0.789	**[0.625, 0.929]**

cooperative-attacker bias: attacks are generated by known scripts, benign traffic is less diverse than production SIP traffic, and single-host timing can differ from a carrier or enterprise deployment [1].

The dataset is the main research limitation. Several classes remain below the $n \geq 30$ per-class reporting threshold, so the paper reports them as evidence, not as stable class-level estimates. Tollfraud is a genuine miss, not missing data, and media has no RTP detection path in the current stack. Tool-shaped benign traffic also inflates signature and IOC false positives in Table III. The right next step is more authenticated benign calling, replayed campaigns, and a campaign-isolated C3 rerun.

Detection coverage is also conditional. Suricata can detach from the Kamailio network namespace after a Kamailio restart or recreate, so the IDS must be restarted as part of operational procedure [5]. Many Wazuh SIP rules depend on `NGN-SEC` emitters that are not yet present for every designed reason string, so those rules should be extended or scoped out rather than counted as live coverage [6]. Media detection needs rtpengine or Asterisk media telemetry, or a clear thesis scope exclusion.

In live operation the Stage-2 LLM worker was CPU-latency-bound: cold model load and generation exceeded the output-gate timeout, so most alerts fell back to `needs_review` and only a small fraction produced a verdict. Consistent with recent security-operations LLM literature, which treats LLM triage as advisory and latency-prohibitive for real time, it was removed from the live stack and is reported as a negative result. The Stage-2 path needs a different evaluation before it can justify any operational claim. Future work should run the larger local model on GPU, enable RAG, build an analyst-

labeled evaluation set, and replace the regex prompt-injection detector with a semantic classifier plus a trusted Stage-1 score channel [13]. The injection guardrail must require positive corroboration before accepting any benign downgrade.

Response controls need the same caution. IP-only banning can harm users behind NAT, so future defenses should add per-address-of-record rate limits and stronger identity-aware controls. PAKE-based SIP authentication is a long-term hardening direction, but it is outside the implemented contribution. The current defensible claim is narrower: response state is audited, protected stack peers are allowlisted, and graded response policy follows SIP overload-control principles [12].

Two directions follow directly from these results. First, inline ingress detection: rather than the current out-of-band pipeline (observe, score five-minute windows, react), a fast high-precision classifier could run inline at the Kamailio SBC ingress for real-time gating, with the large-language-model stage kept strictly offline and advisory given its measured latency. This is only sound on a sender-constrained transport: the live edge is entirely connectionless UDP with spoofable source addresses, so source-based decisions are untrustworthy, and SIP over TLS or DTLS, or registration-bound return-routability, is a prerequisite before inline blocking can be trusted. Second, a SIP-aware application-layer firewall (a SIP WAF) at the edge: enforce digest authentication, per-address-of-record rate limits, malformed-message rejection, and topology hiding, rather than banning by source IP, which our threat-intelligence cross-check shows is low-precision (7.6% independently confirmed) and susceptible to spoofing and NAT collateral. Denser, SIP-specific ground truth (VoIP-aware threat intelligence such as GreyNoise or AbuseIPDB, plus analyst-labeled samples) would raise that confirmation floor and tighten the wide cluster-bootstrap intervals reported here. GPU-hosted Stage-2 triage and an RTP media-plane detector remain open.

XII. Conclusion

This paper presents a reproducible SIP attack-detect-defend testbed and uses it to compare signatures, correlation, and classical ML on shared labels. C1 adds response-level HEP telemetry and shows a controlled within-run gain without replacing the main Stage-1 result. C2 supplies the self-labeling `attack_labels` contract. C3 shows that Suricata, Wazuh, and XGBoost fail in different ways when evaluated on the same windows [5]–[7].

The honest headline is the controlled binary operating point reported in the abstract and Table I, not the superseded leaky score. The negative results are part of the contribution: toll-fraud remains a detection failure, Stage 2 does not yet reduce false positives, and prompt-injection handling is incomplete [13]. The value of the work is therefore methodological. It gives an open, replayable SIP lab for measuring attack, detection, response, and audit behavior under one evidence contract [1].

Artifact Availability and Reproducibility

The full testbed is version controlled in the course GitLab instance at TH Köln at https://git.dn.fh-koeln.de/amc-and-ngn-individual-lab-projects/sip-attack-detect-defend (default branch `main`). It is a Compose-managed stack of roughly 28 containers deployed on a campus virtual machine, with all management planes bound to loopback and reached over an authenticated tunnel. The stack is brought up with `make` targets (`up`, `obs-up`, `ids-up`, `wazuh-up`, `ml-up`, `smoke`). The Stage-1 result is reproduced with `ml/stage1/train.py` using source-IP grouped cross-validation, and the numbers in Table I through Table IV are pinned to committed metrics JSON files. The end-to-end demo is replayed with `scripts/demo/run_pipeline_demo.sh`, and the evaluation harness compares the C3 arms on identical labeled windows.

Ethics

All attack traffic was self-generated against the private campus-lab testbed. No third-party, carrier, or production systems were targeted, and the SIP management planes remained loopback-bound throughout the experiments.

Use of AI Tools

AI-assisted coding tools were used for parts of the software engineering, specifically scaffolding the monitoring dashboard and some automation and deployment scripts. All such output was reviewed, tested, and verified by the author before use. The research design, threat model, detection engineering, evaluation protocol, results, and analysis are the author's own work and were cross-checked against primary sources and the recorded experimental evidence.

References

[1] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "SIP: Session initiation protocol," IETF, RFC 3261, 2002.

[2] The MITRE Corporation, "MITRE ATT&CK knowledge base (t1496 resource hijacking)," The MITRE Corporation, https://attack.mitre.org/techniques/T1496/, 2025, accessed: 2026-07-01.

[3] H. Asgharian, A. Akbari, and B. Raahemi, "Feature engineering for detection of Denial of Service attacks in session initiation protocol," *Security and Communication Networks*, vol. 8, no. 8, pp. 1587–1601, 2015.

[4] W. Nazih, Y. Hifny, W. Elkilani, T. Abdelkader, and H. Faheem, "Efficient detection of attacks in SIP based VoIP networks using linear ℓ_1-SVM classifier," *International Journal of Computers Communications & Control*, vol. 14, no. 4, pp. 518–529, 2019.

[5] Open Information Security Foundation, "Suricata: Open source network threat detection engine," OISF, https://suricata.io/, 2025, accessed: 2026-07-01.

[6] Wazuh Inc., "Wazuh: The open source security platform," Wazuh Inc., https://wazuh.com/, 2025, accessed: 2026-07-01.

[7] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD)*, 2016, pp. 785–794.

[8] W. Nazih, W. S. Elkilani, H. Dhahri, and T. Abdelkader, "Survey of countering DoS/DDoS attacks on SIP based VoIP networks," *Electronics*, vol. 9, no. 11, p. 1827, 2020.

[9] J. Tang, Y. Cheng, and Y. Hao, "Detection and prevention of SIP flooding attacks in voice over IP networks," in *Proc. IEEE INFOCOM*, 2012, pp. 1161–1169.

[10] O. S. Younes, “A hybrid deep learning model for detecting DDoS flooding attacks in SIP-based systems,” *Computer Networks*, vol. 238, p. 110146, 2024.
[11] VoIPdocs, “International toll fraud / international revenue share fraud (irsf),” VoIPdocs, https://voipdocs.io/, 2024, accessed: 2026-07-01.
[12] J. Rosenberg, “Requirements for management of overload in the session initiation protocol,” IETF, RFC 5390, 2008.
[13] OWASP Foundation, “OWASP top 10 for large language model applications,” OWASP Foundation, https://owasp.org/www-project-top-10-for-large-language-model-applications/, 2025, accessed: 2026-07-01.
[14] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré, “Snorkel: Rapid training data creation with weak supervision,” *Proc. VLDB Endowment*, vol. 11, no. 3, pp. 269–282, 2017.