Multitenant Cybersecurity Operations Center Designed as an Open Cloud Service with Machine Learning Elements

Ivana Gabrišová*, Gabriel Koman
University of Žilina, Department of management theories,
Faculty of management science and informatics
Univerzitná 8215/1, 010 26 Žilina, Slovak republic
gabrisova10@stud.uniza.sk*, gabriel.koman@fri.uniza.sk

***Abstract*— Security Operations Centres increasingly operate in cloud environments and must serve multiple tenants while processing vast, heterogeneous telemetry. Signature- and rule-based defences alone struggle with novel attack behaviours, alert volume, and dispersed context. This paper synthesises how open, cloud-based architectures combined with machine learning can support a multitenant Security Operations Centre designed as a managed service. A parametric keyword search using "Elastic Stack" or "OpenSearch" with cybersecurity keywords was conducted, and thirteen selected case studies were retained and analysed across architecture, methods, telemetry, and operating practices.**

Across studies, a recurring foundation appears: an open telemetry plane based on the Elastic technology stack, container orchestration for horizontal scale and tenant isolation, and optional streaming with Apache Kafka and Apache Spark for real-time workloads. Standard components include containerised honeypots with malware enrichment, intrusion detection engines such as Suricata at the edge for prevention and network security monitoring, and automated remediation through rule engines with explicit safety rails (rate limits, blast-radius caps, canary actions, and rollback). Observed learning approaches include gradient-boosted decision trees for labelled network flow records, deep learning for sequence-rich traffic, principal component analysis with local outlier factor for unlabelled system and application logs, streaming combinations of clustering and nearest-neighbour classification, and evolutionary ensembles for user behaviour analytics. Key implications include selecting methods based on data reality and explainability needs, placing computation to meet latency and burst constraints, and enforcing tenant-scoped alerting and case management. The paper offers a practical blueprint for building transparent, scalable, and tenant-aware Security Operations Centres using open technologies and machine learning.

I. Introduction

Enterprises and public infrastructures now generate massive, heterogeneous telemetry: cloud control planes, microservices logs, network flows, OT/SCADA signals, and application traces. Conventional signature- and rule-based defences struggle in this setting because (i) novel/low-and-slow attacks evade static rules, (ii) alert volumes exceed analyst capacity, and (iii) context needed to triage an alert is scattered across systems. Open, cloud-native analytics stacks, especially the Elastic/ELK Stack (Elasticsearch, Logstash, Kibana, Beats), and, in some contexts, OpenSearch, are widely deployed to collect, normalise, index, and visualise such telemetry at scale, frequently forming the backbone of modern SOC pipelines. They feature prominently as transparent, extensible backbones for collection, indexing, visualisation, and ML-assisted detection in multitenant settings. Modern Security Operations Centres (SOCs) must detect advanced threats in robust, heterogeneous log streams while serving multiple tenants with distinct visibility and compliance needs. Traditional rule- and signature-driven tools struggle with data volume, diversity, and zero-day behaviours, which have accelerated adoption of open, cloud-ready analytics stacks that embed machine learning (ML) for anomaly detection and triage [1]–[8]. Broader evidence shows rapid uptake of data-driven and artificial-intelligence techniques in complex, high-stakes settings outside security, reinforcing the need for modern analytics in operations centres [9]. Successful adoption also depends on workforce readiness and the digitalisation of human-capital practices that support new analytics workflows, as shown in adjacent domains [10].

Open architectures also interoperate readily with containerised sensors and cloud-native observability, improving explainability (feature importances, thresholds, drift checks) and per-tenant transparency via scoped dashboards and role-based access [2], [6], [8]. These characteristics make them strong candidates for a multitenant SOC offered as an open cloud service.

There are different reasons why AI/ML is critical for cyber anomaly detection. (i) AI/ML is sensitive to rare/novel patterns. Unsupervised and statistical, as well as semi-supervised methods, can identify deviations in unlabelled, evolving log streams where labelled attacks are scarce [11], [13]. (ii) It is used for sequence and context modelling since deep models can capture temporal dependencies in traffic and process behaviours that deterministic rules miss [14]. (iii) Tree-based models and statistical projections expose feature importances or residuals, helping analysts validate or dismiss alerts quickly in dashboards, which is essential in ensuring explainability for decision support [6], [11], [14]. (iv) Embedded in open pipelines, model outputs can drive self-healing actions or prioritised playbooks, shrinking mean-time-to-respond (MTTR) [3], [6]. Multitenant SOCs benefit from scoped dashboards and auditable ML logic (thresholds, clusters, features) so each tenant can self-inspect without data leakage [2], [6], [8]. In this paper, anomaly denotes a statistical or

behavioral deviation; confirming maliciousness requires human or playbook-driven triage with contextual correlation. ML surfaces candidates, it does not 'detect zero-days' by itself.

At the same time, data diversity, drift, and operational constraints make it crucial to understand which ML approaches fit which telemetry and where to run them (stream vs. batch; edge vs. core), and how open architectures implement multitenancy and automation. Open deployments in the case studies, ranging from self-healing in multi-clouds [3] to containerised honeypots [2], IoT/IDS architectures for smart cities [4], and microservices observability [8], provide grounded evidence of what works and where it breaks.

Using parametrically selected 13 case studies, the purpose of this article is to identify which ML/AI algorithms and open architectural patterns are used for anomaly/threat detection suitable for a multitenant, cloud-based SOC. Findings are synthesised across data types, pipelines, and outcomes, and highlight design implications for a tenant-aware managed service.

II. METHODOLOGY

A parametric literature review was performed on the Web of Science (WoS) platform. Searches combined the static element "Elastic Stack/ELK Stack" or "OpenSearch" with additional keywords: AI, ML, SOC, anomaly detection, cybersecurity, security operations, security monitoring, and alerting. The "Elastic/ELK Stack" branch yielded 31 publications, and the "OpenSearch" branch yielded 2. The 33 results were screened by (i) English language, (ii) full-text availability, and (iii) thematic relevance to anomaly detection/ML within SOC pipelines. Thirteen publications met the criteria and form the sole evidence base for this article.

III. OPEN CLOUD ARCHITECTURES FOR A MULTITENANT SOC

A. Self-healing multi-cloud services with ELK

A containerised multi-agent architecture leverages ELK and a Drools rule engine, illustrating how detection can feed automated actions while maintaining SLA targets in multi-cloud environments. The proposed architecture is deployed across multi-clouds, with Filebeat – Logstash – Elasticsearch – Kibana for telemetry and a Drools rules engine for diagnosis and remediation [3]. All automated actions must be guarded by safety rails, per-tenant rate limits, blast-radius caps, canary execution, and automatic rollback, so a faulty fix cannot propagate across tenants. Filebeat standardises collection; Logstash applies input/filter/output pipelines; Elasticsearch provides distributed search/analytics; Kibana acts as the management/visualisation plane. This design shows how detection outputs can trigger automated corrective actions (e.g., service restarts, ticketing), turning ML/statistical signals into self-healing behaviour suitable for multitenant SLAs.

B. Containerized honeypots with ELK

Cowrie (SSH/Telnet), Dionaea (multi-protocol), and Glastopf (web) were deployed as Docker containers, shipping JSON logs with Beats, processed in Logstash, indexed in Elasticsearch, and visualised in Kibana; alerts are generated by the platform's alerting rules/connectors (not the dashboards). VirusTotal enrichment was integrated for captured malware, and the deployment ran for 92 days, collecting 2,750,654 connection attempts with rich protocol/credential/command distributions. Operational caveat: honeypots disproportionately capture opportunistic scanning and commodity malware; they must be strictly isolated (network segmentation, default-deny egress with allow-lists, fake/non-production credentials) and monitored for pivot attempts to prevent lateral movement into tenant environments. That demonstrates a lightweight, horizontally scalable threat-intel layer that plugs directly into SOC workflows and can be scoped per tenant via index/role policies [2].

C. Elastic sustainability for IoT smart cities

SUESSA, a Sustainable & Ultra-Elastic Stack Security Architecture, places Suricata (IDS/IPS/NSM) alongside ELK and an ML-based alert module to classify/threshold events before notifying administrators [4]. The architecture addresses three pain points: sustainability, cohesiveness, and automation in high-volume IoT settings by layering traffic pipelining/monitoring, multi-stage filtering, and visualisation. Several background insights on smart-city risks and big-data security are summarised as cited in [4].

D. Microservices observability

A taxonomy over purpose, parameters, scope (system/service/network), deployment (Kubernetes, Docker), and tooling (OpenTelemetry, Prometheus, Grafana, Jaeger/Loki) is proposed as a result of a survey involving Observability frameworks for containerised microservices [8]. In SOC terms, these controls complement ELK by providing distributed tracing and metrics that, combined with logs, support ML-driven anomaly detection and tenant-scoped visibility in cloud-native environments. The survey of microservices observability underscores combining logs, metrics, and traces with ML for anomaly detection at the system, service, and network levels. The toolbox (e.g., OpenTelemetry, Prometheus, Grafana, Jaeger/Loki) complements ELK in cloud-native SOCs and supports tenant scoping at the platform layer [8].

E. SIEM complement, validation, and observability scoring

ELK is analysed as a big-data complement to traditional SIEM for correlation, alerting, and analytics in CSIRT/SOC operations [1]; works show validation loops via attack simulation (e.g., AttackIQ) feeding ELK dashboards [5] and observability scoring for SCADA/OT security with ATT&CK-mapped simulations [12]. Common patterns include per-tenant indices and RBAC in ELK, horizontal scaling (Kubernetes), and streaming buffers (Kafka/Spark) to protect indexing during bursts, key to maintaining tenant SLAs without cross-tenant impact [2], [3], [4], [6], [7], [13], [15]. Alerting rules, connectors, and case management must also be tenant-scoped (e.g., separate spaces or case projects) to prevent cross-tenant data exposure during triage.

IV. MACHINE LEARNING AND AI METHODS USED FOR SOC ANOMALY DETECTION

1) Supervised tree-based learning (XGBoost): XGBoost is used for NetFlow anomaly/threat prediction with interpretable feature importances (e.g., ports, bytes, duration) and analyst-friendly outputs rendered in Kibana [14]. Strengths include high accuracy on labelled traffic and explainability that support triage; limitations include label scarcity and drift, necessitating retraining and threshold reviews.

2) Deep learning: DNN and RNN are evaluated on traffic streams; performance depends on data balance/feature engineering and can lag tree ensembles on attack-only subsets. Applied to traffic sequences to model temporal dependencies and complex feature interactions [14]. Useful on mixed streams and sequence-heavy tasks; requires careful class balance, feature engineering, and runtime resources.

3) Unsupervised/statistical: PCA with a high-quantile threshold and LOF corroboration detects outliers in Postfix logs; ELK visualises time series and feature vectors for analyst triage. For Postfix mail logs, ELK assembles program-ratio vectors over short windows; PCA separates normal and abnormal subspaces and uses a high-quantile threshold on residual errors to flag anomalies; LOF provides outlier corroboration across k values [11]. Advantages include label efficiency and transparency; tuning windows/scaling is critical.

4) Hybrid streaming: In Kafka – Spark ML – ELK pipelines, k-Means clusters online traffic, while k-NN rapidly classifies new points, emitting detections to ELK for persistence and dashboards [13]. That keeps real-time detection upstream from the indexer to protect ingestion when traffic spikes.

5) Ensemble/evolutionary methods for UBA: A Kubernetes-hosted ELK framework pairs distributed evolutionary ensembles with semi-supervised scoring to flag deviations in user/application behaviour at scale, explicitly addressing missing/unbalanced data [15]. That is suited to multitenant environments where behaviours differ by tenant and data completeness varies.

6) Rule-based automation: Drools rules map detection outputs to self-healing actions in multi-cloud services, reducing time-to-mitigation and human toil. Self-healing rules convert detections into deterministic actions (restarts, ticket creation, evidence collation), reducing responder toil and latency in multi-cloud SOC operations [3]. Rules are transparent and easy to extend; upkeep is needed as environments evolve.

V. DATA TYPES AND TELEMETRY SOURCES

1) Network telemetry (NetFlow): Bytes, packets, ports, duration, and protocol are effective for supervised and sequence-based models; classic worm/DDoS exemplars are used for evaluation and dashboard validation. Source/destination ports, bytes, packets, duration, protocol, and interface IDs are often most predictive for supervised models; visual validation in ELK helps investigate flagged flows [14].

2) System/application logs: Postfix program ratios over short windows feed PCA/LOF to surface spam/abuse anomalies, with ELK for correlation and visualisation [11].

3) IoT/IDS signals: Suricata alerts and diverse sensor streams underpin SUESSA's multi-layer filtering and ML alerting workflow [4].

4) Honeypot data: SSH/Telnet/HTTP/SMB attempts, credential/command distributions, and malware samples enriched with VirusTotal enhance knowledge of attacker behaviour and families. Credentials, command sequences, protocol distributions, and malware samples (with VirusTotal results) illuminate attacker behaviour and support rule/model updates [2].

5) Microservices signals: Logs, traces, metrics for system-, service-, and network-level observability; ML methods include clustering, thresholding, and supervised classifiers depending on signal type [8].

VI. OPERATIONAL PATTERNS THAT MAKE ML ACTIONABLE

1) Stream-before-index: Kafka/Spark put in front of ELK for compute-heavy or bursty ML, helps indexing remain stable and detections arrive with low latency [13].

2) Enrichment: Threat intel (e.g., VirusTotal) and context (geo, asset tags, tenant IDs) attached during Logstash/ingest improves analyst decisions without extra queries [2], [6].

3) Validation loops: Attack simulation and observability scoring can be used (ATT&CK-mapped) to validate coverage, calibrate thresholds, and demonstrate control efficacy to each tenant [12], [5].

4) Automation: Detections can be bridged to Drools/playbooks for self-healing and case creation with supporting evidence, shortening MTTR and standardising responses [3], [4].

5) Tenant isolation: Enforced per-tenant indices/roles, resource quotas, and horizontal scaling on Kubernetes help avoid noisy neighbour effects and preserve data boundaries [2], [15], [6], [7], [8].

VII. COMPARISON

SOCs built as open cloud services face high heterogeneity in telemetry (NetFlow, system/app logs, IDS alerts, traces/metrics), variable latency budgets, and uneven label availability across tenants. Directly comparing algorithms without anchoring them in data reality and pipeline placement leads to "algorithm shopping" that rarely survives production. The two tables in this section therefore serve complementary purposes: Table 1 maps each method to the telemetry it needs and where it runs in an open stack (ELK, Kafka/Spark, Kubernetes), and Table 2 summarises strengths and limitations that matter for multitenant operations, such as explainability, drift sensitivity, and operational overhead. Together, they

translate the 13 case studies into a design aid for selecting methods that are feasible, explainable, and maintainable in SOC.

Case studies show that where detection runs is as important as what model is used. For example, pushing compute "left" with Kafka – Spark absorbs bursts and preserves indexing throughput when near-real-time action is needed [13], while keeping analytics beside indexing enables tight dashboard-driven triage and record-keeping in ELK [1], [2], [6], [7]. [11], [14]. Inline controls in IoT/OT (e.g., Suricata) must sit on the traffic path and then feed ELK for cross-tenant analytics [4]. The placement view also exposes multitenancy mechanics, per-tenant indices/RBAC, stream partitioning, and container orchestration, which several studies relied on to prevent noisy neighbour effects and preserve data boundaries [2], [3], [15], [6], [8].

Table I. shows what telemetry each method consumes and where it typically runs within open SOC stacks (ELK, Kafka/Spark, Kubernetes). It can be used to map tenants' data realities to the right computational tier.

The Algorithms in Table I. and II. Include:

- A: XGBoost (supervised) [14]
- B: DNN / RNN (deep) [14]
- C: PCA (+ LOF corroboration) [11]
- D: k-Means + k-NN (hybrid streaming) [13]
- E: Evolutionary ensemble UBA [15]
- F: Rule-based self-healing (Drools) [3]
- G: Suricata + ML alert module (SUESSA) [4]
- H: Honeypot telemetry + enrichment [2]
- I: Observability-driven anomaly detection [8].

TABLE I. PLACEMENT

	Data & Features	Where It Runs / Pipeline
A	NetFlow: bytes, packets, ports, duration, protocol	Batch/near-real-time model service; scores and metadata indexed to ELK for dashboards
B	Network flow sequences / temporal patterns	Sidecar or parallel analytics service to ELK ingestion; sequence modelling results visualised in Kibana (alerts come from alerting rules/connectors)
C	Postfix mail log vectors (program ratios over short windows)	Logstash – Elasticsearch features; PCA/LOF job produces anomaly scores back into ELK
D	Streaming NetFlow features	Kafka – Spark ML – ELK, clustering/classification upstream; ELK stores alerts/contexts
E	Multi-source user/app behaviour logs	ELK on Kubernetes with distributed evolutionary learners; semi-supervised risk scores per user/app
F	Cloud/service events and health logs	ELK analytics – Drools action engine for remediation/ticketing; feedback into ELK
G	IDS alerts + IoT/sensor metadata	Suricata inline; ELK for analytics; ML module thresholds and routes alerts to responders
H	Cowrie/Dionaea/Glastopf logs; VirusTotal results	Docker sensors – Beats/Logstash – ELK; enrichment at ingest for analyst context
I	Logs, traces, metrics (SLO/SLA)	OpenTelemetry/Prometheus/Grafana alongside ELK; correlation feeds SOC triage

If tenants have labelled NetFlow samples, it is best to favour XGBoost with ELK for explainable triage [14]. For unlabelled logs (e.g., mail), PCA/LOF plugs in with minimal ops change [11]. For bursting traffic or strict real-time SLAs, it is best to move compute left with Kafka/Spark, then persist to ELK [13]. In IoT/OT or inline prevention scenarios, to keep Suricata close to the wire and use ELK for cross-tenant analytics [4]. For user behaviour analytics at scale, deployment of evolutionary ensembles on Kubernetes with tenant-aware indices [15]. Where automation matters (SLA/SRE), the connection of ELK detections to Drools for self-healing [3]. In cloud-native apps, it is recommended to combine logs, traces, and metrics with ELK to localise faults quickly across tenants [8].

Different tenants bring different constraints. Where labels are scarce, unsupervised/statistical approaches (e.g., PCA with LOF) deliver quick wins and transparent thresholds [11]. Where labels exist (historical incidents, red-team traffic), XGBoost offers strong accuracy with feature importances that analysts can defend in tenant-facing reviews [6], [14]. Deep models capture temporal effects but require tuning and careful management of imbalances [14]. For behavioural baselining across diverse users/apps, evolutionary ensembles proved robust under missing/unbalanced data but raise governance complexity in multitenant settings [15]. Finally, rule engines convert detections into self-healing actions that reduce MTTR, an operational outcome repeatedly emphasised in multi-cloud scenarios [3], [4].

Table II. summarises why to pick each method and what to watch for in multitenant operations (drift, imbalance, governance).

TABLE II. TRADE-OFFS

	Strengths	Limitations
A	High accuracy on labelled traffic; feature importances aid analyst explainability in Kibana	Needs quality labels; retraining/drift monitoring; may be biased towards the majority classes
B	Captures temporal/sequence effects; competitive on mixed streams	Sensitive to class imbalance, tuning/runtime cost, and more complicated to explain to tenants
C	Label-efficient, transparent thresholds; quick to operationalise in ELK	Windowing/feature scaling matters; seasonal patterns can trigger false positives
D	Online clustering + fast classification; keeps indexing stable during bursts	Choosing k, handling concept drift, requires streaming ops maturity
E	Robust to missing/unbalanced data; semi-supervised risk scoring at scale	Model governance/updates are complex; needs strong RBAC/index isolation
F	Deterministic, auditable actions; reduces MTTR and analyst toil	Rule upkeep; limited generalisation beyond encoded scenarios
G	Inline prevention + analytics; elastic architecture for IoT/smart-city workloads	Tuning across heterogeneous devices; ops overhead for high-volume telemetry
H	Rich attacker intel; low resource footprint; improves rule/model updates	Bias toward opportunistic scans; not preventive by itself
I	Full-stack context (logs/metrics/traces) speeds root cause; tenant scoping is natural	Integration/cardinality management; trace sampling can hide edge cases

If explainability for tenant-facing alerts is required, XGBoost or PCA should be preferred, where feature attributions/thresholds are visible in ELK [11], [14]. When facing sparse labels, it is recommended to leverage PCA/LOF or streaming clustering and add human-in-the-loop labelling over time [11], [13]. For automated remediation (SRE/SLA) to pair any detector with Drools rules to standardise actions and evidence capture [3]. For multi-tenant microservices, it is the combination of observability tooling with ELK, so each tenant gets scoped, actionable context without leakage [8] and for IoT/OT tenants, Suricata at the edge and consolidate analytics in ELK to balance inline protection with cross-tenant insights [4].

Recommendations for using the tables:

- Start with data reality: if a tenant offers labelled NetFlow, Table I. points to XGBoost collocated with ELK or upstream scoring; Table II. clarifies accuracy/explainability and the need for drift monitoring.
- If telemetry is unlabelled (e.g., mail/system logs), Table I. routes you to PCA/LOF inside the ELK analytics loop; Table II. warns about windowing/seasonality.
- For strict latency or bursting traffic, Table I. directs detection to Kafka/Spark; Table II. highlights concept-drift handling and streaming ops maturity.
- In IoT/OT or inline contexts, Table I. keeps Suricata on the wire with ELK analytics; Table II. flags tuning across heterogeneous devices.
- Where automation is a goal, pair any detector with Drools/playbooks (Table I.) and consult Table II. to anticipate rule upkeep and audit needs.
- For tenant transparency, prefer methods with auditable thresholds/attributions (PCA/LOF, tree ensembles) and surface them in per-tenant Kibana spaces.

VIII. Conclusion

This review of 13 case studies shows that a multitenant SOC offered as an open cloud service is most effective when it couples an ELK-centred telemetry plane with fit-for-data ML elements, validated and governed through attack simulation/observability loops, and operationalised via automation. Architecturally, the recurring backbone consists of ELK on containers/Kubernetes, with optional Kafka and Spark for stream-time analytics. This setup supports per-tenant indices/RBAC, scales horizontally, and absorbs bursts without degrading other tenants.

Methodologically, four families of approaches recur: (i) tree ensembles (e.g., XGBoost) for labelled NetFlow, offering accuracy and analyst-friendly feature importances; (ii) deep models for sequence-aware traffic analysis where data and tuning capacity exist; (iii) unsupervised/statistical methods (e.g., PCA with LOF) for unlabelled log streams, delivering fast, transparent anomaly surfacing; and (iv) hybrid streaming (k-Means + k-NN) to catch zero-day behaviours without overwhelming the indexer. For user and application behaviour, evolutionary ensembles provide robustness under missing/unbalanced data at a multitenant scale. Crucially, several studies convert detections into deterministic actions (self-healing with Drools) and assurance evidence (ATT&CK-mapped simulations; observability scoring), tightening MTTR and trust with each tenant. In IoT/OT and smart-city contexts, Suricata + ELK + ML balances inline prevention with cross-tenant analytics; containerised honeypots plus enrichment supply low-cost attacker intelligence to refine rules and models. ELK also complements traditional SIEM by adding big-data scale and analyst-oriented visualisation.

A multitenant SOC delivered as an open cloud service should be anchored on ELK (Elasticsearch, Logstash, Kibana, Beats), deployed on containers/Kubernetes for horizontal scaling, per-tenant indices, and role-based access, ensuring isolation and transparent dashboards for each tenant. Stream processing should be placed in front of ELK when tenants require strict latency or face bursty traffic, keeping indexing stable while running online detection upstream. In IoT/OT contexts, Suricata inline should be kept at the edge for prevention/NSM and funnel enriched events to ELK for cross-tenant analytics and reporting. Containerised honeypots can be used to harvest attacker behaviours at low cost and feed enrichment back into rules and models. The loop can then be closed with attack simulation and observability scoring to verify coverage against ATT&CK and demonstrate posture improvements to each tenant.

Methods should be selected according to data realities and explainability requirements: XGBoost fits settings with labelled NetFlow and a need for defensible feature importances; PCA/LOF or streaming clustering (k-Means + k-NN) suits unlabelled or rapidly evolving logs; DNN/RNN are best reserved for sequence-heavy traffic where tuning capacity exists; and evolutionary ensembles are appropriate for tenant-specific user/application behaviour under missing or imbalanced data. Detections should be integrated with rule engines (e.g., Drools) to standardise remediation and reduce MTTR. At the same time, drift monitoring, retraining schedules, and audit trails should be instrumented so tenants can review thresholds, attributions, and actions directly in ELK/Kibana.

Findings reflect 13 WoS-sourced publications filtered around ELK/OpenSearch, which may bias coverage toward ELK-family stacks and under-represent other open pipelines; only two OpenSearch-related items surfaced, limiting parity analysis. The corpus mixes prototypes, deployments, and surveys with non-uniform datasets and metrics, constraining direct meta-comparison of algorithms. Several results derive from single organisations, honeypots, or specific IoT/OT testbeds, so external validity across sectors and threat mixes is cautious. Some studies lack released datasets/code or consistent evaluation protocols, which reduces reproducibility and hinders exact benchmarking.

In conclusion, SOC architects and platform engineers should prioritise standardised, privacy-preserving multitenant benchmarks (flows, logs, traces) with ATT&CK-mapped ground truth so detectors and pipeline placements can be compared fairly. Data science/MLOps teams should build drift-aware, explainable MLOps for the SOC, continuous drift checks, tenant-visible attributions/thresholds, and full audit

trails, integrated with ELK spaces and case management, which will improve transparency and trust. SREs and data-platform owners should conduct a systematic evaluation of Kafka/Spark vs. ELK-side placement under bursty loads and tight SLAs, including resource governance and cost-to-detect per tenant. IoT/OT security teams, advancing multitenancy with device-class–aware tuning, lightweight edge models, and strict isolation in cross-tenant analytics is essential. Research groups and vendors should focus on extending behaviour analytics with evolutionary, federated, or semi-supervised learning to handle missing labels and privacy constraints better. Security leadership and governance, formalise automation safety (rollback, blast-radius limits, canary remediation) and measure MTTR versus false-action trade-offs should be a priority. Finally, procurement and platform decision-makers should run like-for-like OpenSearch vs. ELK evaluations for ML-assisted anomaly detection in multitenant SOCs, and observability/platform teams, deepen observability–security fusion (logs/metrics/traces with ELK analytics) that will accelerate tenant-scoped root-cause analysis.

ACKNOWLEDGMENT

Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I05-03-V02-00010.

REFERENCES

[1] R. Andrade and J. Torres, “Enhancing intelligence SOC with big data tools,” Nov. 2018.

[2] A. Kyriakou and N. Sklavos, “Container-Based Honeypot Deployment for the Analysis of Malicious Activity,” *IEEE Xplore*, Oct. 01, 2018.

[3] H. Mfula and J. K. Nurminen, “Self-Healing Cloud Services in Private Multi-Clouds,” *2018 International Conference on High Performance Computing & Simulation (HPCS)*, Jul. 2018.

[4] B. P. Gautam and S. Norio, “SUESSA: Sustainable & Ultra-Elastic Stack Security Architecture for Securing IoT Networks of Future Smart Cities,” *2020 Eighth International Symposium on Computing and Networking Workshops (CANDARW)*, Nov. 2020, pp. 387–390.

[5] Abigail, M. Xie, Ryan, C. Sterner, T. Choi, and N. Dong, “SDGen: A Scalable, Reproducible and Flexible Approach to Generate Real World Cyber Security Datasets,” *Communications in computer and information science*, Jan. 2022, pp. 102–115.

[6] M. D. Poat, J. Lauret, and D. Fedele, “Flexible visualization of a 3rd party Intrusion Prevention (Security) tool: A use case with the ELK stack,” *Journal of physics. Conference series*, vol. 2438, no. 1, Feb. 2023, pp. 1–5.

[7] R. Currie and W. Yuan, “Building a Flexible and Resource-Light Monitoring Platform for a WLCG-Tier2,” *EPJ Web of Conferences*, vol. 295, 2024.

[8] Ummay Faseeha, H. J. Syed, F. Samad, Sehar Zehra, and H. Ahmed, “Observability in Microservices: An In-Depth Exploration of Frameworks, Challenges, and Deployment Paradigms,” *IEEE Access*, Jan. 2025, pp. 1–1.

[9] M. Mičiak *et al.*, “Effective Education System for Athletes Utilising Big Data and AI Technology,” *Data*, vol. 10, no. 7, Jun. 2025, pp. 102–102.

[10] Nikola Štaffenová and Alžbeta Kucharčíková, “Digitalization in the Human Capital Management,” vol. 11, no. 7, Jul. 2023, pp. 337–337.

[11] C.-P. Tran and D.-K. Tran, “Anomaly Detection in POSTFIX mail log using Principal Component Analysis,” *2018 10th International Conference on Knowledge and Systems Engineering (KSE)*, Nov. 2018, pp. 107–112.

[12] J. Halvorsen, J. Waite, and A. Hahn, “Evaluating the Observability of Network Security Monitoring Strategies With TOMATO,” *IEEE Access*, vol. 7, 2019, pp. 108304–108315.

[13] J. Jose, Talha Ahmed Khan, W. Akbar, Muhammad Afaq, and W.-C. Song, “An ML Based Anomaly Detection System in real-time data streams,” *2021 International Conference on Computational Science and Computational Intelligence (CSCI)*, Dec. 2021.

[14] C.-T. Yang, Y.-W. Chan, J.-C. L. Liu, Endah Kristiani, and C.-H. Lai, “Cyberattacks Detection and Analysis in a Network Log System Using XGBoost with ELK Stack,” Aug. 23, 2021.

[15] G. Folino, C. Otranto Godano, and F. S. Pisani, “An ensemble-based framework for user behaviour anomaly detection and classification for cybersecurity,” *The Journal of Supercomputing*, Jan. 2023.