(IJIEST) 2024, Vol. No. 10, Jan-Dec **e-ISSN: 2454-9584; p-ISSN: 2454-8111**

ThreatEye: Machine-Learning-Based Behaviour Analytics for Cloud Threat Detection and Anomaly Classification

Shourya Gupta

University of Bath, United Kingdom

[1]*Date of Receiving: 02 February 2024; Date of Acceptance: 15 March 2024; Date of Publication: 11 April 2024*

ABSTRACT

Cloud environments generate massive, heterogeneous telemetry: identity sign-ins, API calls, network flows, container events, and application logs. Many real attacks in the cloud do not rely on exploiting a software vulnerability. Instead, adversaries abuse valid credentials, cloud-native APIs, and "living-off-the-land" actions that look legitimate in isolation. This makes behaviour analytics (often grouped under User and Entity Behavior Analytics, UEBA) a practical detection layer: it models *who/what normally does what* and flags meaningful deviations. Building behaviour analytics for the cloud, however, is difficult due to high-cardinality entities, concept drift (deployments change behaviour), sparse labels, and the cost of false positives at scale.

This paper proposes **Threat Eye**, a cloud-focused behaviour analytics architecture using machine learning (ML). Threat Eye combines (i) entity-centric baselining, (ii) sequence-aware modelling for API and session behaviour, (iii) graph-based signals for relationships among identities, assets, and actions, and (iv) feedback-driven tuning to continuously reduce noise. We outline data pipelines, feature design, model choices (unsupervised, semi-supervised, and supervised), and an evaluation strategy grounded in publicly discussed log-anomaly methods and UEBA research. We also provide a comparative analysis of representative modelling approaches and show how Threat Eye can be deployed safely with privacy controls, explainability, and operational guardrails.

1. Introduction

Cloud security detection has shifted from perimeter-centric monitoring to identity- and API-centric monitoring. In modern incidents, attackers often obtain credentials (phishing, token theft, OAuth abuse) and then operate using allowed cloud APIs. The activity can resemble routine admin work: enumerating resources, creating access keys, modifying IAM policies, spinning up workloads, or exfiltrating data via sanctioned services. Because such actions may be "valid" at the API level, purely signature-based detection misses novel variants, while rule-based detections struggle with cloud diversity and tenant-specific normal behaviour.

UEBA has been explored for enterprise security for years and is specifically motivated by the gap between "valid action" and "valid intent" (e.g., the same API call can be benign or malicious depending on the actor and context). An early enterprise-focused UEBA platform overview highlighted the role of behaviour profiling for threat hunting and anomaly discovery. In cloud environments, the need is even stronger due to the speed of automation and the ease of scaling attacker actions once privileged access is gained.

[1] *How to cite the article:* Gupta S., (2024) ThreatEye: Machine-Learning-Based Behaviour Analytics for Cloud Threat Detection and Anomaly Classification; *International Journal of Inventions in Engineering and Science Technology*, Vol 10 Issue 1, 17-26

(IJIEST) 2024, Vol. No. 10, Jan-Dec e-ISSN: 2454-9584; p-ISSN: 2454-8111

Threat Eye is designed around three principles:

1. **Entity-first modelling**: treat identities, workloads, keys, and hosts as first-class entities with baselines and peer groups.
2. **Multi-view behaviour**: model behaviour from counts, sequences, timing, and graphs (relationships).
3. **Operational realism**: optimize for analyst workflows, Explainability, and feedback loops, not just offline metrics.

Table 1. Cloud threat behaviours suited for analytics

Threat behaviour	Why rules/signatures struggle	Behaviour signal examples
Account hijack / token theft	Actions are "legitimate" API calls	New geo/ASN, new device fingerprint, abnormal API sequence burst
Privilege escalation (IAM)	Many valid admin workflows exist	Rare policy edits, unusual role assumption chains, sudden permission expansion
Data discovery + exfiltration	Uses normal storage/query APIs	Unusual object access pattern, atypical download volume, off-hours spikes
Lateral movement in cloud	Mostly API relationships	New service-to-service calls, new trust edges, new key usage
Insider-like abuse	Hard to label and subtle	Long-term drift in access patterns, atypical project/resource focus

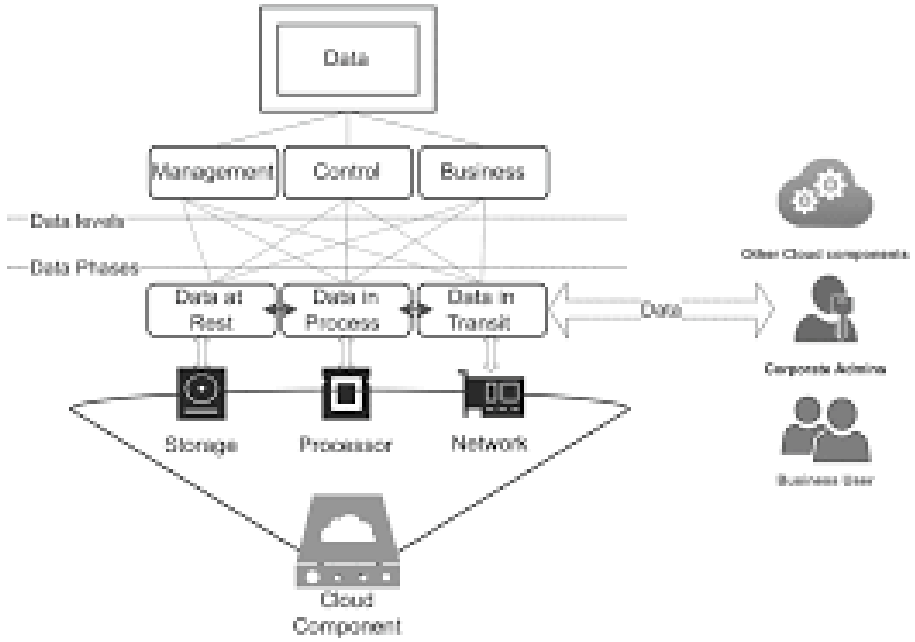

Fig 1: Data-Driven Threat Analysis for Ensuring Security in Cloud Enabled Systems

2. Background and Related Work

2.1 UEBA remember: baseline + deviation

UEBA typically builds statistical/ML models for "normal" behaviour and triggers when deviation is significant. Enterprise UEBA discussions emphasize analytics pipelines that unify logs and produce anomaly scores for hunting. In identity-driven ecosystems, UEBA can be embedded into federated identity management; for example, an approach validated on OpenID Connect constructs a "session fingerprint" and detects behavioural anomalies at the relying party.

2.2 Deep learning for insider-like patterns

Insider threat detection literature (and adjacent cloud misuse) highlights typical data challenges: high-dimensional and heterogeneous events, extreme class imbalance, subtle attacks, and limited labels. A review on deep learning for insider threat detection details these issues and surveys deep architectures used to learn representations from complex audit streams. This motivates Threat Eye's hybrid approach: unsupervised baselines for breadth, and targeted supervised/semi-supervised learning where evidence is strong.

2.3 Log anomaly detection as a foundation

Cloud behaviour analytics often relies on log anomaly detection methods. DeepLog introduced sequence modelling (LSTM-style) for system logs, showing that learning "normal" sequences can detect deviations without explicit anomaly signatures. More recent work explores transformer-based log anomaly detection and parser-free approaches, such as LAnoBERT, which uses masked language modelling loss for unsupervised anomaly scoring. Surveys consolidate model and preprocessing choices, highlighting practical concerns like unstable formats and evaluation gaps.

2.4 Datasets: the reproducibility bottleneck

A recurring limitation in UEBA research is the lack of realistic, shareable datasets. A cloud-oriented UEBA log dataset (covering >5000 users over multiple years) was introduced to support anomaly detection evaluation in cloud computing scenarios, including attack injection for account hijacking experiments. A more recent labelled cybersecurity log dataset for "risk activities" also supports studying early-stage attack activities in application logs.

Table 2. Selected related work that informs Threat Eye

Area	Representative work (year)	What it contributes
UEBA platform overview	UEBA for enterprise security (2016)	Threat hunting framing; entity behaviour baselines
Identity federation UEBA	Behaviour anomalies in identity federations (2021)	Session fingerprinting + anomaly workflow
Insider-like detection	Deep learning review for insider threats (2021)	Challenges + deep architectures for behavioural audit data
Sequence log AD	DeepLog (2017)	Learning normal sequences; online adaptation idea
Transformer log AD	LAnoBERT (2023)	Parser-free, unsupervised MLM-based anomaly scoring

Area	Representative work (year)	What it contributes
Log AD survey	DL for log anomaly detection survey (2023)	Taxonomy of models, preprocessing, evaluation practices
Cloud UEBA dataset	CLUE-LDS (2022)	Long-term cloud log dataset + attack injection
Risk activity dataset	RBD24 (2025)	Labelled log-based risk activities

3. Threat Eye System Architecture

Threat Eye is a reference architecture that can be implemented on any major cloud. It is not a single model, but a **modular detection fabric** that produces entity risk scores and analyst-ready alerts.

3.1 Core pipeline

1. **Ingest**: identity logs (SSO, IAM), cloud audit logs (API calls), workload logs (Kubernetes, VM agents), storage access logs, and application logs.
2. **Normalize**: map events to a common schema; resolve identities and assets; create "sessions" and "action sequences."
3. **Feature views**:
    - Aggregates (counts, rates, unique resources)
    - Sequences (ordered API/action tokens)
    - Graphs (edges: identity → resource, role → action, workload → endpoint)
4. **Model layer**: multiple detectors operate in parallel; outputs are fused into a calibrated risk score.
5. **Triage layer**: explanations, comparable peer behaviour, and incident stitching.
6. **Feedback loop**: analyst decisions update suppression lists, thresholds, and semi-supervised models.

3.2 Why multi-model fusion?

Cloud behaviour is multi-modal. A single detector is usually brittle:

- Aggregates catch volumetric abuse but miss stealthy sequences.
- Sequences catch abnormal workflows but can be noisy with drift.
- Graph detectors catch unusual relationship patterns but need good entity resolution.

Threat Eye uses **score fusion**: each detector emits a normalized anomaly score with confidence, and a final risk engine ranks entities and sessions.

Table 3. Threat Eye components

Layer	Component	Output
Data	Log collectors + stream/batch ingestion	Raw events
Processing	Parsing, normalization, entity resolution	Canonical events + entity IDs
Behaviour	Sessionization, baseline windows, peer groups	Entity timelines
ML	Aggregate AD, sequence AD, graph AD, supervised detectors	Scores per entity/session
Risk	Score fusion + calibration	Final risk score + severity
Ops	Case management + feedback	Labels, suppressions, tuning signals

(IJIEST) 2024, Vol. No. 10, Jan-Dec e-ISSN: 2454-9584; p-ISSN: 2454-8111

4. Data, Features, and Behaviour Modelling

4.1 Data sources (typical)

- **Identity**: sign-ins, MFA events, token issuance, role assumptions.
- **Audit/API**: cloud control-plane actions (create keys, attach policies, list buckets).
- **Workload**: container start/stop, unexpected image pulls, metadata access attempts.
- **Storage/DB**: object reads/writes, query patterns.
- **App logs**: business-level actions (export report, admin settings changes).

Threat Eye aims for *behaviour invariants*: features that survive minor system changes.

4.2 Feature families

- **Temporal**: hour-of-day entropy, inter-event time, burstiness.
- **Diversity**: number of distinct resources touched, new resource ratio.
- **Rarity**: how often this entity (or peers) uses a given API/action.
- **Sequence tokens**: API/action n-grams, masked-token prediction loss (transformer/LSTM-style).
- **Graph features**: new edges, edge weights, role-to-resource reachability changes.

Table 4. Feature examples by view

View	Feature examples	Attacks often surfaced
Aggregates	API call rate, bytes downloaded, unique buckets accessed	Exfil bursts, broad enumeration
Temporal	Off-hours z-score, session duration anomalies	Compromised accounts used at unusual times
Rarity	"Never used before" API calls, rare admin actions	Privilege changes, key creation
Sequence	Unusual API order, abnormal n-grams, MLM loss spikes	Recon → privilege change → persistence chains
Graph	New identity→resource edges, new trust chains	Lateral movement, role abuse

5. Machine Learning Methods in Threat Eye

Threat Eye uses a **tiered modelling strategy**.

5.1 Unsupervised anomaly detection (baseline coverage)

This is the default for cloud tenants with limited labels:

- Robust baselines per entity and per peer group.
- Density-based or clustering/outlier methods to find behavioural outliers (e.g., HDBSCAN-related approaches are widely used for clustering/outlier detection in practice and literature).
- Log/sequence anomaly scoring using DeepLog-style or transformer masked-language modelling (MLM) loss.

5.2 Semi-supervised learning (use analyst feedback efficiently)

Analyst-confirmed true positives are rare but valuable. Threat Eye supports:

- **Positive-unlabeled learning** for confirmed malicious sessions vs. unlabeled.
- **Metric learning / contrastive learning** on event windows to separate "known bad" from normal, with robustness techniques also explored in log anomaly research.
- Threshold tuning by workload/team/role to reduce noise.

5.3 Supervised detectors (high precision for known patterns)

For well-defined threats (e.g., "credential theft follow-up patterns"), supervised models can be trained:

- Gradient boosting / calibrated linear models on curated features.
- Graph neural networks for relationship-based anomaly detection have been applied to insider threat and fraud-like behaviour contexts.
- Supervised detectors are gated behind careful dataset hygiene to avoid leakage and to maintain trust.

5.4 Explainability

Threat Eye emphasizes "why this is weird":

- Feature-attribution summaries (top contributing rarity/time/volume features).
- Peer comparison ("this user's role typically does X, but this session did Y").
- Sequence highlights (the subsequence with highest prediction error).

Table 5. Model families used in Threat Eye

Model family	Strengths	Weaknesses	Best use
Statistical baselines	Fast, interpretable, stable	Miss complex multi-step patterns	First-line guardrails
Clustering/outlier (density)	Finds novel outliers	Parameter sensitivity; drift	Behaviour outlier discovery
LSTM sequence (DeepLog-like)	Captures ordered actions	Needs tuning; log format issues	API/session workflow anomalies
Transformer MLM (LAnoBERT-like)	Strong sequence modelling; parser-free options	Heavier compute; needs care	High-cardinality log streams
Graph-based models (GCN/GNN)	Captures relationship anomalies	Entity resolution critical	Lateral movement / trust abuse

6. Evaluation Strategy and Metrics

Evaluating cloud behaviour analytics is difficult because true attack labels are sparse and the business cost is asymmetric (false positives waste analyst time; false negatives can be catastrophic). Threat Eye uses a combined evaluation strategy:

1. **Offline replay** using public datasets when possible (e.g., long-term cloud UEBA logs with injected attacks).
2. **Synthetic/controlled injections** in a test tenant (simulate key creation + policy changes + data access).

3. **Human-in-the-loop measurement**: alert rate per entity, time-to-triage, and analyst agreement.
4. **Stability under drift**: measure performance across deployments or policy changes.

6.1 Metrics

- Ranking metrics: Precision@K, Mean Average Precision (MAP) for "top alerts".
- Detection metrics: PR-AUC (preferred under imbalance), ROC-AUC (secondary).
- Operational metrics: alerts/day/1000 identities, median time-to-close, suppression rate.
- Robustness: score stability (variance) under normal changes.

Table 6. Metrics mapped to operational goals

Goal	Metric	Why it matters
Reduce analyst overload	Alerts/day, Precision@K	Behaviour analytics must rank well
Handle class imbalance	PR-AUC	More informative than accuracy
Catch multi-step attacks	Session-level recall, time-to-detect	Cloud attacks chain actions quickly
Survive drift	Stability across time windows	Cloud behaviour changes often
Improve over time	Feedback lift	Measures learning from analyst labels

7. Comparative Analysis

This section compares common approaches used in behaviour analytics and positions Threat Eye as a hybrid. The aim is not to claim a universal "best model," but to show the trade-offs that matter in cloud deployments.

7.1 Comparison across modelling approaches

Threat Eye combines multiple approaches because cloud threats differ:

- Token theft may appear as a geo/device/time anomaly.
- Privilege escalation may be a rarity + graph anomaly.
- Exfiltration may be volumetric + sequence anomaly.

Table 7. Comparative analysis of representative approaches

Approach	Novel attack detection	Label need	Drift handling	Explainability	Compute cost	Notes
Rules/signatures	Low–Medium	Medium	Medium	High	Low	Great for known IOCs; weak for "valid API abuse"
Statistical baselines	Medium	Low	Medium	High	Low	Good "first wall"
Density outlier (HDBSCAN-style)	High	Low	Medium	Medium	Medium	Useful discovery; needs tuning
LSTM sequences (DeepLog)	High	Low	Medium	Medium	Medium	Strong on ordered logs
Transformer MLM (LAnoBERT)	High	Low	Medium	Medium	High	Good for complex log semantics

Approach	Novel attack detection	Label need	Drift handling	Explainability	Compute cost	Notes
Graph anomaly / GNN	High	Medium	Medium	Medium	High	Great for relationship misuse
Pure supervised classifier	Medium	High	Low–Medium	Medium	Medium	High precision if labels are strong

7.2 Why Threat Eye's hybrid fusion helps

Threat Eye improves practical detection by:

- Using **unsupervised** detectors for broad coverage (unknown threats).
- Adding **sequence** models where order matters (attack chains).
- Adding **graph** detectors where relationships matter (roles, trust, access).
- Calibrating with **feedback** to reduce false positives over time.

Table 8. Threat Eye vs single-model systems

Dimension	Single-model system	Threat Eye (hybrid)
Coverage	Narrower	Broader across threat types
False positives	Often high in edge cases	Reduced via fusion + feedback
Drift resilience	Limited	Better via multi-view redundancy
Analyst trust	Depends on model	Improved with explanations + peer baselines
Engineering effort	Lower	Higher, but modular

8. Deployment Considerations in the Cloud

8.1 Scalability and cost

Threat Eye supports a two-speed architecture:

- **Streaming**: lightweight baselines + rule guardrails for immediate detection.
- **Batch**: heavier sequence/graph computations over longer windows.

8.2 Privacy and security

Behaviour analytics can become surveillance if unmanaged. Threat Eye should implement:

- Data minimization: keep only fields needed for detection.
- Pseudonymization/anonymization where possible.
- Strict role-based access to raw logs and model outputs.
- Retention policies aligned with compliance.

8.3 Operational guardrails

- Alert deduplication and case stitching.
- Suppression controls (known automation accounts, maintenance windows).
- Canary scoring: monitor baseline drift and model degradation.

Table 9. Practical deployment checklist

Area	Recommendation	Outcome
Data quality	Canonical schema + entity resolution	Fewer broken detections
Drift	Rolling baselines + retraining schedule	Stable scoring
Compute	Stream small, batch heavy	Predictable cost
Explainability	"Why flagged" + peer comparison	Faster triage
Feedback	Analyst labels + suppression	Continuous noise reduction

9. Conclusion

Threat Eye is a practical blueprint for cloud security behaviour analytics using ML. It treats behaviour analytics as a multi-view, multi-model ranking problem rather than a single binary classifier. By combining entity baselines, sequence modelling, graph context, and analyst feedback loops, Threat Eye is better aligned with real cloud attack patterns where adversaries abuse valid identities and APIs. The design is informed by UEBA foundations, identity-focused anomaly workflows, advances in log anomaly detection from LSTM and transformer approaches, and the growing availability of cloud-relevant datasets for evaluation.

References

Calvo, A., Martín, A. G., Beltrán, M., Fernández-Isabel, A., & Martín de Diego, I. (2025). RBD24: A labelled dataset with risk activities using log data. *Computers & Security, 144*, 104290. https://doi.org/10.1016/j.cose.2024.104290

Campello, R. J. G. B., Moulavi, D., Zimek, A., & Sander, J. (2015). Hierarchical density estimates for data clustering, visualization, and outlier detection. *ACM Transactions on Knowledge Discovery from Data, 10*(1), 1–51. https://doi.org/10.1145/2733381

Chen, S., Wang, H., Li, Z., & Liu, W. (2022). BERT-Log: Anomaly detection for system logs based on BERT. *Applied Artificial Intelligence, 36*(1), 2145642. https://doi.org/10.1080/08839514.2022.2145642

Deepa, S., Venkatesan, R., & Thangavel, M. (2024). Deep belief network-based user and entity behavior analytics (UEBA) for web applications. *International Journal of Cooperative Information Systems, 33*(1), 2350016. https://doi.org/10.1142/S0218843023500168

Du, M., Li, F., Zheng, G., & Srikumar, V. (2017). DeepLog: Anomaly detection and diagnosis from system logs through deep learning. *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (pp. 1285–1298). https://doi.org/10.1145/3133956.3134015

Jiang, J., Zhang, Y., Wang, Z., & Liu, Z. (2019). Anomaly detection with graph convolutional networks for insider threat and fraud detection. *2019 IEEE Military Communications Conference (MILCOM)* (pp. 1–6). https://doi.org/10.1109/MILCOM47813.2019.9020760

Landauer, M., Onder, S., Skopik, F., & Wurzenberger, M. (2023). Deep learning for anomaly detection in log data: A survey. *Machine Learning with Applications, 12*, 100470. https://doi.org/10.1016/j.mlwa.2023.100470

Landauer, M., Skopik, F., Hold, G., & Wurzenberger, M. (2022). A user and entity behavior analytics log data set for anomaly detection in cloud computing. *2022 IEEE International Conference on Big Data (Big Data)* (pp. 6078–6085). https://doi.org/10.1109/BigData55660.2022.10020672

Lee, Y., Kim, J., & Kang, P. (2023). LAnoBERT: System log anomaly detection based on BERT masked language model. *Applied Soft Computing, 148*, 110689. https://doi.org/10.1016/j.asoc.2023.110689

Liu, W., Chen, Y., & Zhang, H. (2022). Robust log anomaly detection based on contrastive learning and multi-scale masked sequence to sequence. *The Journal of Supercomputing, 78*(15), 16937–16960. https://doi.org/10.1007/s11227-022-04508-1

Martín, A. G., Beltrán, M., Fernández-Isabel, A., & Martín de Diego, I. (2021). An approach to detect user behaviour anomalies within identity federations. *Computers & Security, 106*, 102356. https://doi.org/10.1016/j.cose.2021.102356

McInnes, L., Healy, J., & Astels, S. (2017). hdbscan: Hierarchical density based clustering. *Journal of Open Source Software, 2*(11), 205. https://doi.org/10.21105/joss.00205

Shashanka, M., Shen, M.-Y., & Wang, J. (2016). User and entity behavior analytics for enterprise security. *2016 IEEE International Conference on Big Data (Big Data)* (pp. 1867–1874). https://doi.org/10.1109/BigData.2016.7840805

Yuan, S., & Wu, X. (2021). Deep learning for insider threat detection: Review, challenges and opportunities. *Computers & Security, 104*, 102221. https://doi.org/10.1016/j.cose.2021.102221