



Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

Hackathon 2026 – I.R. Journal SecureLogAI

UWTSD HACKATHON 2026 · PROJECT REPORT, PRESENTED

SecureLog AI

ML-Powered Log Anomaly Detection

A theoretical and reflective account of how and why the project was researched, planned, and built

TEAM

Andy, Frontend & API Lead

Piero, Machine Learning & Data Lead

Two-person team · six-week hybrid Agile + Gantt delivery

504

LOG EVENTS
ANALYSED

26

ANOMALIES
SURFACED

\$0

INFRASTRUCTURE
COST

Replacing brittle, rule-based alerting with adaptive, unsupervised learning

Individual Reflective Journal
*SecureLog AI: ML-Powered Log
Anomaly Detector*

Team: Piero & Andy (2 members)

Student name: Pier Paolo Penne

Student ID number: 2336786

Module: Emerging Trends

Module code: ACCB6019

Lecturer: Sadeque Shaikh

Table of Contents

Introduction.....	2
Role and Contribution	3
Teamwork Experience and Evaluation	3
Technical and Sustainability Reflection	3
Design Limitation and Risk Factors	4
Ethical, Legal and Professional Considerations.....	4
Future Work and Improvements	4
Personal Reflection and Conclusion	4
References	6

Introduction

Anomaly detection in system and application logs is a mature, well-studied field, so before writing a line of code I already knew that unsupervised methods such as Isolation Forest are the conventional default, precisely because labelled anomaly data is rarely available in real deployments [1]. What I had not fully appreciated was how much of that maturity rests on a messier problem underneath it: recent survey work on log parsing confirms that failure signals remain inconsistently formatted and resistant to simple rule-based filtering even today [2], and that finding shaped almost every early decision I made. This journal reflects on my role turning raw log text into a trustworthy anomaly score, covering parsing, feature engineering, and the Isolation Forest model itself, an approach recent work continues to apply to real-time network security detection [3], and what building that layer taught me about how much of the machine learning part of a project is really about handling messy input first.

Role and Contribution

My primary contribution was `parser.py`, `features.py`, and `model.py`. Because real logs rarely arrive in one clean format, the parser needed to detect and normalise several shapes: generic timestamped lines, syslog, Apache/Nginx combined format, and CSV exports, a choice motivated directly by recent survey evidence that inconsistent formatting remains one of the main obstacles to automated log analysis [2], not an edge case to design around later. `features.py` builds a numeric matrix combining temporal, severity, and volume signals with a TF-IDF representation of the message text [4], reduced through the latent-semantic tradition behind singular-value-based dimensionality reduction [5]. `model.py` wraps this matrix in an Isolation Forest [3], implemented through scikit-learn's standard estimator interface [6]. I also worked with Andy on the results rendering, including the anomaly-score scatter plot built with Matplotlib [7], one of the clearest artefacts we produced for showing the pipeline worked end to end on our 504-row test file.

Teamwork Experience and Evaluation

Splitting the codebase early into parser, feature-engineering, model, and interface layers let Andy and me work largely independently, which suited how differently we approach a problem. One decision I am glad we made jointly was comparing Isolation Forest [3] against Local Outlier Factor, whose density-based scores a recent comparative study of unsupervised log-anomaly algorithms confirms are computed relative to a fixed fitted set [8], in a shared notebook before committing to Isolation Forest for the app. That fixed-set behaviour fits poorly with a web service scoring whatever gets uploaded next, and testing the assumption together, rather than assuming it, saved us a late rewrite. I would rate our teamwork 4 out of 5. Our two halves, the modelling core and the FastAPI interface, only really proved themselves once integrated in week four, later than either of us would have liked, though communication stayed honest and neither of us missed a milestone.

Technical and Sustainability Reflection

The technical decision I am most confident in was keeping feature engineering interpretable rather than reaching for a pretrained embedding model. TF-IDF weighting [4] combined with a small dimensionality-reduction step [5] meant I could always trace a flagged anomaly back to the vocabulary driving its score, something a dense embedding would have hidden from a user trying to trust the output. Running entirely on scikit-learn's CPU-based estimators [6] kept the stack free to run, with no GPU and no recurring cloud cost, a genuine sustainability advantage over cloud-hosted alternatives, though it scales comfortably only to demo-sized files rather than production volumes. What I underestimated was how sensitive Isolation Forest is to its contamination prior: without labelled data to tune against, I chose five percent

largely by eyeballing results against our synthetic dataset, and I would rather admit that than dress it up as more rigorous than it was.

Design Limitation and Risk Factors

The clearest limitation specific to my part of the system is parser coverage. It reliably handles the formats we tested, but a genuinely novel format would silently fall back to keeping the raw line as an unparsed message rather than failing loudly, an intentional trade-off that still leaves a real gap. A more fundamental limitation, shared with Andy's interface work, is that our anomaly score has no ground-truth validation beyond a synthetic dataset with known injected anomalies. The literature identifies this as a recurring challenge for unsupervised methods [1], and a textbook instance of how ML systems accumulate hidden technical debt around unvalidated assumptions [9], precisely the kind of undocumented risk the NIST AI risk-management framework now asks teams to surface rather than leave implicit [10]. There is a related contamination-drift risk: our five percent prior was tuned against synthetic data, and a real log source with a different baseline anomaly rate could cause systematic over- or under-flagging over time, a specific case of concept drift in deployed models [11].

Ethical, Legal and Professional Considerations

Log files routinely contain personal or identifying information, IP addresses, usernames, hostnames, so even a prototype like this raises data-protection questions well before any real deployment. Ethical guidance on database and log handling stresses data minimisation, access control, and clear retention policies as baseline obligations wherever personal data may appear [12], principles that line up with the “Privacy as the Default Setting” foundation of Privacy by Design [13]. Everything in our system runs locally and nothing leaves the user's machine, which limits immediate risk, but a real deployment on live UK organisational logs would need to satisfy the UK's Data (Use and Access) Act 2025 and UK GDPR directly [14], [15], plus explicit redaction of identifying fields, an obligation I had not fully appreciated applied to the modelling side of the pipeline and not only the upload-handling side Andy worked on.

Future Work and Improvements

Building on the concept-drift literature, my priority next step would be periodic retraining on a rolling window rather than a single static fit, so the contamination prior can adapt as a log source's baseline behaviour changes [11]. I would also want to explore recent end-to-end AutoML pipelines for unsupervised log anomaly detection, which automate exactly the kind of manual threshold-tuning I currently do by eye, and could turn the contamination prior into something the system learns rather than something I guess [16]. Finally, extending the feature set with network-level signals, drawing on established approaches for detecting SSH brute-force attacks from aggregated flow data [17], would strengthen detection of the coordinated login-based pattern our prototype already flags qualitatively from log text alone.

Personal Reflection and Conclusion

This project left me with a sharper sense that keeping things simple and keeping them responsible are not in tension. TF-IDF and Isolation Forest are about as simple as a genuinely useful anomaly detector gets, and it was that simplicity, not despite it, that let me reason clearly about their limitations instead of trusting a black box I could not interrogate. Revisiting the anomaly-detection survey literature [1] while writing this journal clarified where our specific gaps, no ground truth, a static contamination prior, sit within a longer-running set of open problems in the field. Handing my model's output to Andy's FastAPI layer [18] and watching it become something a person could read and act on was my first real experience of building the part of a system that has to be trustworthy, not just accurate.

References

- [1] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys*, vol. 41, no. 3, Article 15, pp. 1-58, 2009.
- [2] T. Zhang, H. Qiu, G. Castellano, M. Rifai, C. S. Chen, and F. Pianese, "System log parsing: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 8, pp. 8596-8614, 2023.
- [3] P. Zhang et al., "Real-time malicious traffic detection with online isolation forest over SD-WAN," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 2076-2090, 2023.
- [4] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information Processing & Management*, vol. 24, no. 5, pp. 513-523, 1988.
- [5] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American Society for Information Science*, vol. 41, no. 6, pp. 391-407, 1990.
- [6] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825-2830, 2011.
- [7] J. D. Hunter, "Matplotlib: A 2D graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90-95, 2007.
- [8] A. C. E. de Moura et al., "Anomaly detection in logs: A comparative analysis of unsupervised algorithms," in *Proc. 13th Symp. Languages, Applications and Technologies (SLATE), OASlcs*, vol. 120, 2024, pp. 12:1-12:14.
- [9] D. Sculley et al., "Hidden technical debt in machine learning systems," in *Proc. 28th Int. Conf. Neural Information Processing Systems (NeurIPS)*, 2015, pp. 2503-2511.
- [10] NIST, "Artificial intelligence risk management framework (AI RMF 1.0)," *NIST AI 100-1*, 2023.
- [11] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Computing Surveys*, vol. 46, no. 4, Article 44, pp. 1-37, 2014.
- [12] E. Pina, J. Ramos, H. Jorge, P. Vaz, J. Silva, C. Wanzeller, M. Abbasi, and P. Martins, "Data privacy and ethical considerations in database management," *Journal of Cybersecurity and Privacy*, vol. 4, no. 3, pp. 494-517, 2024.
- [13] A. Cavoukian, "Privacy by design: The 7 foundational principles," *Information and Privacy Commissioner of Ontario, Canada*, 2009.
- [14] UK Parliament, *Data (Use and Access) Act 2025*, c. 18. London, U.K.: HM Stationery Office, 2025.
- [15] European Parliament and Council of the European Union, *General Data Protection Regulation (GDPR), Regulation (EU) 2016/679*, 2016.
- [16] S. Zhang et al., "End-to-end AutoML for unsupervised log anomaly detection," in *Proc. 39th IEEE/ACM Int. Conf. Automated Softw. Eng. (ASE)*, 2024, pp. 1680-1692.

- [17] M. M. Najafabadi, T. M. Khoshgoftaar, C. Calvert, and C. Kemp, "Detection of SSH brute force attacks using aggregated netflow data," in Proc. 2015 IEEE 14th Int. Conf. Machine Learning and Applications (ICMLA), Miami, FL, USA, 2015, pp. 283-288.
- [18] S. Ramirez, FastAPI, 2018. [Online]. Available: <https://fastapi.tiangolo.com>