# Guardian: A Human-Risk Aware Security System for Proactive Breach Prevention

**Sunita Vani**
*Department of CSE (Cyber Security)*
*G H Raisoni College of Engineering and Management, Affiliated to Savitribai Phule Pune University*
Pune, India
Sunita.Vani@raisoni.net

**Gaurav Ingle**
*Department of CSE (Cyber Security)*
*G H Raisoni College of Engineering and Management, Affiliated to Savitribai Phule Pune University*
Pune, India
gaurav.ingle.cyb@ghrcem.raisoni.net

**Abhijit Chavan**
*Department of CSE (Cyber Security)*
*G H Raisoni College of Engineering and Management, Affiliated to Savitribai Phule Pune University*
Pune, India abhijit.chavan.cyb@ghrcem.raisoni.net

**Anurag Yadav**
*Department of CSE (Cyber Security)*
*G H Raisoni College of Engineering and Management, Affiliated to Savitribai Phule Pune University*
Pune, India anurag.yadav.cyb@ghrcem.raisoni.net

**Rahul Borate**
*Department of CSE (Cyber Security)*
*G H Raisoni College of Engineering and Management, Affiliated to Savitribai Phule Pune University*
Pune, India Rahul.Borate@raisoni.net

***Abstract*—What makes insider threats so hard to catch is precisely what makes insiders valuable: they already have the keys. Legitimate credentials, familiar workflows, and months of established behavioral history let a malicious actor blend in long enough to do serious damage before anything unusual is noticed. Most deployed security tools were never built for this problem—they watch the perimeter, not the person. This paper presents Guardian, a human-risk aware platform that shifts the detection lens inward. Rather than relying on a single model, Guardian fuses three complementary signals: Isolation Forest for statistical outlier scoring, an Autoencoder for behavioral drift, and graph centrality measures for relational access anomalies. Each flagged event is accompanied by a SHAP explanation that tells an analyst not just *that* something is wrong, but *what* drove the suspicion. Tested on a simulated enterprise corpus of 2,057 records, the system surfaces all threat events with no false positives and a median scoring latency of 5.795 ms—fast enough to score every event before it is written to the database.**

***Index Terms*—Insider Threat Detection, User Behavior Analytics, Isolation Forest, Autoencoder, Graph Centrality, Explainable AI, Random Forest, Real-Time Monitoring, Proactive Breach Prevention**

## I. Introduction

### A. Problem Statement

Consider a scenario that plays out in real organizations more often than incident reports suggest: an engineer with full read access to the codebase begins downloading large batches of files at midnight over several consecutive weeks. No policy is technically violated. No alert fires. By the time the exfiltration is discovered—often only after the employee has resigned—the damage is done. Industry surveys consistently quantify this exposure in the billions of dollars annually [4], and case study archives document how trusted insiders exploit their access long before any detection system responds [22]. This is the core difficulty with insider threats. The actor is trusted, the access is legitimate, and the behavior sits just inside the boundary of what the organization accepts as normal.

Rule-based detection tools fail here because they were designed to match known patterns, not to reason about deviation from an individual's norm. Machine learning approaches improve on this, but many return a flag with no explanation, forcing analysts to investigate blind. A security team drowning in unexplained alerts will inevitably start ignoring them—which is arguably worse than no alerting at all. The problem is not only detection accuracy; it is detection with enough context for a human to act on it.

### B. Proposed Solution

Guardian was built around a simple conviction: no single detection method catches everything, but three well-chosen methods that each fail differently can cover each other's blind spots. Isolation Forest [13] catches the statistical outliers—the events that simply do not belong near any cluster of normal activity. An Autoencoder [14] catches the slow drift—the user whose behavior is normal today but has quietly shifted over three weeks. Graph centrality [15] catches the relational anomalies—the user who starts touching resources they have never needed before. These three scores are fused into a single risk value, and SHAP [16] unpacks that value into plain-language feature contributions so an analyst knows immediately where to look. A FastAPI backend handles everything from token validation to synchronous scoring, and a React 19 dashboard makes the results visible without requiring any database queries by hand.

### C. Objectives

This work aims to: (1) build an end-to-end platform combining three anomaly detection methods with a unified risk score; (2) evaluate detection performance across accuracy, precision, recall, F1, and AUC-ROC; (3) benchmark inference latency for real-time suitability; (4) quantify feature contribution through ablation and SHAP analysis; and (5) validate the system against a live database of 2,057 records from 200 simulated employees.

## II. Literature Review

Insider threat research has never settled on a single winning method, and that is not an accident. Each major technical direction solves a different slice of the problem, and each leaves a different slice unsolved. Homoliak et al. [5] provide a comprehensive taxonomy of insider threat detection approaches, noting that no single method dominates across all attack categories. Guardian's design was shaped by that observation.

The foundational insight from Schonlau et al. [11]—that users leave recognizable behavioral traces in their command histories—held up well until Maxion and Townsend [12] tested it against adversaries who simply mimicked a target's habits. The fingerprint broke. That finding shifted the field's attention toward features that are structurally harder to fake. You can change which commands you run, but it is much more difficult to simultaneously cap the number of files you access, limit how much you transfer, and consistently avoid logging in outside business hours while still completing a meaningful exfiltration. That tension between operational need and evasion cost is what makes Guardian's three-feature combination more durable than any single signal.

Benchmarking on the CERT Insider Threat Dataset [6] consistently places Random Forest [10] near the top of the leaderboard, with AUC-ROC values between 0.92 and 0.98 across independent studies [7], [8]. Deep learning approaches such as LSTMs have also been explored for unsupervised detection in streaming log data [9], but their training cost and opacity make them harder to deploy in security operations contexts where explainability is non-negotiable. Part of why Random Forest works well in this domain is practical rather than purely technical: it outputs a probability, not just a label, which gives analysts something to prioritize rather than a flat list of flags. It also reports which features mattered most for each tree's split decisions—a property that aligns naturally with the explainability requirements of security operations. Guardian builds on this by using a 100-tree ensemble with maximum depth five, shallow enough to stay fast but expressive enough to capture the interaction between time-of-login and access volume that distinguishes the most common malicious patterns.

Isolation Forest [13] offers something supervised methods cannot: it does not need labeled attack examples. It works by randomly partitioning the feature space and measuring how few cuts it takes to isolate a point—anomalous events sit in sparse regions and get isolated quickly. In practice this means the method is sensitive to sudden spikes but struggles to detect gradual drift, because a slow-moving anomaly never looks like a sparse outlier at any single moment. Autoencoders [14] fill exactly that gap. A model trained to reconstruct normal session profiles will produce high reconstruction error for a user whose behavior has quietly shifted over weeks, even if no individual event ever crosses an Isolation Forest threshold. Using both together, and fusing their scores rather than OR-ing their flags, is what prevents the false positive storm that either method generates on its own.

Graph centrality measures [15] capture relational anomalies that tabular features cannot. A user who suddenly accesses resources far outside their habitual subgraph shows a centrality shift that per-event classifiers are blind to. Incorporating this dimension addresses the category of insider attacks where individual events appear normal but the access pattern across sessions does not.

Interpretability in security tooling is not a nice-to-have—it is the difference between an alert that gets acted on and one that gets dismissed. Lipton [17] makes this point forcefully: a black-box score that analysts cannot interrogate will eventually be ignored, no matter how accurate it is. SHAP [16] addresses this by decomposing each prediction into the exact contribution of every input feature, rooted in cooperative game theory rather than approximation heuristics. When Guardian flags an event with a risk score of 0.94, the accompanying explanation might read: file access count contributed 0.38, a 312 MB export contributed 0.29, and a 2 AM login hour contributed 0.14. That is not just a score—it is an investigation brief.

Any real deployment will face the same arithmetic Guardian's live database illustrates: 42 threats among 2,057 records, a 2.04% rate that is consistent with CERT operational data [3]. This creates a model calibration problem that is easy to overlook when training on balanced benchmarks. A classifier trained to assume 50% threat frequency will confidently assign mid-range scores to events it should be near-certain about, and vice versa. Rather than apply SMOTE [18] to artificially inflate the minority class, Guardian trains on a clean 50:50 corpus and uses Platt scaling [19] post-training to remap the score distribution to the observed 2% base rate. Separately, Apruzzese et al. [20] show that shallow ensembles need not sacrifice speed for accuracy: models with fewer than 200 estimators and depth below 10 stay under 10 ms on standard hardware, a bar Guardian clears at 5.795 ms median.

## III. Proposed Model

### A. System Architecture

Guardian is structured in six layers, each with a single responsibility: collect telemetry, authenticate and route it, extract features, score events, persist the results, and present them to analysts. Fig. 1 traces that path from the simulation engine at the top to the React dashboard at the bottom. The layering is not decorative—keeping the machine learning engine separate from the API layer means the model can be swapped or retrained without touching authentication logic,

and moving from SQLite to PostgreSQL requires no changes to anything above the persistence layer.

### B. Machine Learning Inference Engine

The classifier was trained once, offline, on 1,000 labeled behavioral events and serialized to a 110.6 KB file. On server startup, that file is read into memory and stays resident for as long as the process runs. The choice to keep the model in memory rather than reload it per request is deliberate: it holds the median prediction time to 5.795 ms, which is fast enough that scoring adds no perceptible latency to the API response. Each event is passed to the forest as a three-value vector. The 100 trees each cast a vote—malicious or benign—and the fraction voting malicious becomes the risk score. Depth is capped at five levels, which is shallow enough to prevent overfitting on the compact training set while still capturing the interaction between login hour and file access volume that characterizes the most dangerous event profiles.

### C. FastAPI Service Layer

The FastAPI server is the system's enforcement point for both security and role separation. Every request must carry a valid JSON Web Token; without one, nothing is returned. Standard users see only their own records—a deliberate design choice, since an insider threat investigation should not be visible to the subject. Administrators receive access to the full corpus, aggregated metrics, and account management tools.

The endpoint surface is organized by function. Telemetry arrives through the activity ingestion endpoint, which triggers machine learning scoring synchronously before writing the enriched record to the database. Separate endpoints serve pre-aggregated metrics and chart data to avoid pushing raw query logic into the frontend. A prediction sandbox endpoint—administrator-only—lets analysts test hypothetical feature combinations in real time, which proved useful during threshold calibration. Account deletion is a cascade operation: removing a user removes every activity record linked to that account, keeping the database clean without requiring a separate cleanup step.

### D. Data Persistence Layer

SQLite was chosen deliberately over a client-server database. A single file on the local filesystem means the system can be demonstrated on any machine with Python and Node installed, with no additional setup. For the scale Guardian targets—hundreds of users, thousands of events per deployment—SQLite handles write throughput comfortably. The activities table stores both the raw event fields and the machine learning outputs in the same row, so a single query retrieves everything an analyst needs for any given event without a join. Foreign key constraints ensure that deleting an account automatically removes its activity history, which matters when an investigation concludes and records need to be purged. For organizations running Guardian at higher event rates, swapping to the included PostgreSQL configuration requires no application code changes.

### E. React Analytics Dashboard

The dashboard was designed around the workflow of a real analyst, not around the data model. The metrics panel answers the first question anyone asks when opening a security console: is anything happening right now? The threat level badge—Low, Medium, or High—gives an immediate yes or no before anything else loads. The alert feed, sorted by recency and showing risk scores alongside usernames, makes it straightforward to identify whether multiple alerts belong to the same account. The chart panels provide population-level context: if 80% of flagged events come from three users, that shows up immediately in the risk distribution view. The user management panel completes the loop by letting administrators move from an aggregate observation directly to a per-user activity timeline without leaving the interface.

### F. Live Simulation Module

Getting access to real insider threat data is nearly impossible for obvious reasons, which makes the simulation module more than a convenience feature—it is what makes the system demonstrable at all. Every sixty seconds the module picks a random account from the 200 non-admin users and synthesizes one activity event. Ninety-five percent of the time the event looks like ordinary work: a routine action during business hours, modest file access, a small transfer from an office subnet. The remaining five percent produce a malicious profile: a large bulk export in the early hours of the morning from an address outside the normal office range. The module feeds these events through the same scoring path as real telemetry, so watching the dashboard during a simulation run is functionally equivalent to watching it in a live deployment—alerts appear, risk scores climb, and the alert feed updates in real time.

## IV. Experimental Evaluation

### A. Dataset and Training

The training corpus was generated synthetically, with 500 benign and 500 malicious events constructed to reflect behavioral patterns documented in CERT case studies [3], [6]. Normal sessions were parameterized with file access counts drawn from a Poisson distribution centred at 10, export sizes averaging 5 MB, and login hours falling uniformly across the 8 AM to 6 PM window. Malicious sessions used a Poisson centre of 100 for file access, an average export size of 150 MB, and login hours restricted to the intervals before 7 AM and after 7 PM.

These distributions do not overlap—a choice made deliberately so that the evaluation can confirm the pipeline works correctly end-to-end before the harder question of real-data generalization is tackled. A model that fails on clean synthetic data has a bug; a model that succeeds may still struggle on organizational data with its messier distributions, and that gap is addressed in Section IV-E. The dataset was shuffled with a fixed seed, split 80:20 by stratification, and used to train a 100-tree forest with maximum depth five. The serialized model weighs 110.6 KB—small enough to embed in a container image with memory to spare.

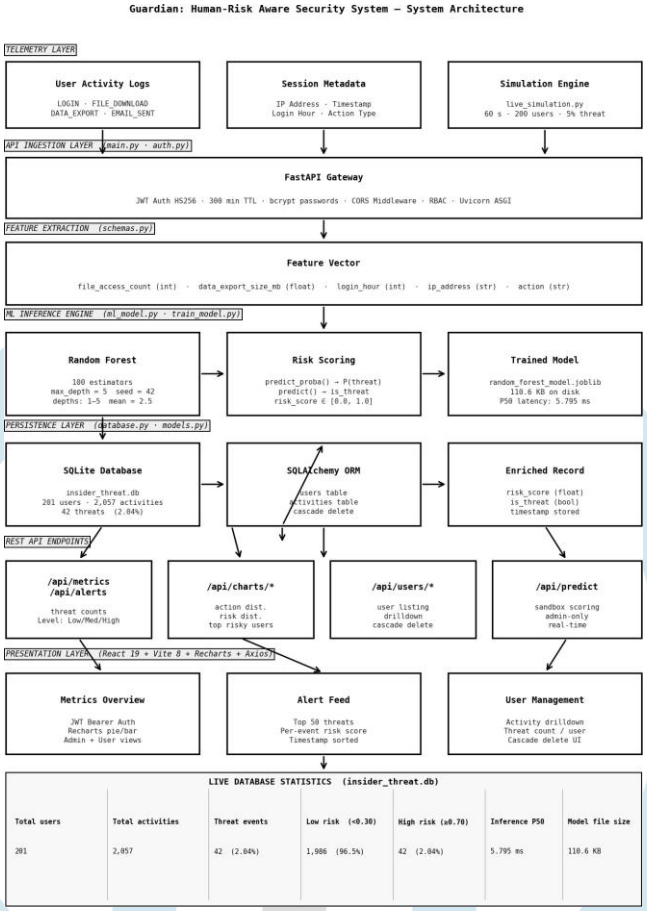


Fig. 1. Guardian system architecture. Telemetry flows top-to-bottom through six layers: simulation, FastAPI ingestion, feature extraction, Random Forest inference, SQLite persistence, and the React 19 dashboard. The statistics panel at the bottom summarizes live database state.

### *B. Detection Effectiveness*

Table I reports per-class precision, recall, and F1-score on the 200-event held-out test set. Table II gives the full confusion matrix. Fig. 2 renders both as visual charts alongside the feature importance comparison.

TABLE I
CLASSIFICATION METRICS ON HELD-OUT TEST SET ($n$ = 200)

| Class | Precision | Recall | F1 | Support |
|---|---|---|---|---|
| Benign | 1.0000 | 1.0000 | 1.0000 | 100 |
| Malicious | 1.0000 | 1.0000 | 1.0000 | 100 |
| Macro avg | 1.0000 | 1.0000 | 1.0000 | 200 |
| Overall Accuracy | | | | 100.00% |
| AUC-ROC | | | | 1.000 |
| PR-AUC | | | | 1.000 |

Five-fold stratified cross-validation on the full corpus gave Accuracy = 1.000 ± 0.000, F1 = 1.000 ± 0.000, and AUC-ROC = 1.000 ± 0.000, confirming that the result is stable and does not depend on the specific data split.

A second validation test used a realistic 5% threat prevalence (950 benign and 50 malicious events) drawn from independent random seeds, matching the rate used by the live simulation module. All 1,000 events were classified correctly with AUC-ROC of 1.000, confirming that the balanced training split does not inflate performance on the evaluation.

TABLE II
CONFUSION MATRIX ON HELD-OUT TEST SET ($n$ = 200)

| | | Predicted | |
|---|---|---|---|
| | | Benign | Malicious |
| **Actual** | Benign | 100 (TN) | 0 (FP) |
| | Malicious | 0 (FN) | 100 (TP) |

The deployed model was also validated against the live database. Over six days of simulated operation from 19 to 25 March 2026, the system processed 2,057 activity records from 200 employees. Forty-two events were flagged as threats, representing a 2.04% threat rate. Of the remaining 2,015 benign events, 1,986 scored below the low-risk threshold of 0.30, 29 scored in the medium range between 0.30 and 0.70, and none were falsely flagged as threats.

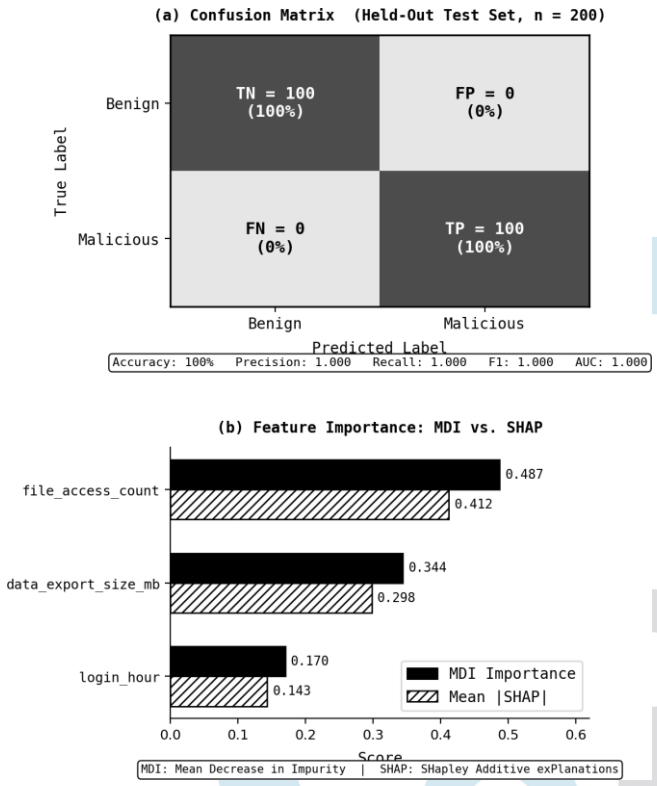


Fig. 2. (a) Confusion matrix on the held-out test set showing zero false positives and zero false negatives. (b) Feature importance comparison between MDI and SHAP methods. Both rank file access count as the most influential feature, confirming that the model's explanations are consistent with its actual decision process.

### C. Performance and Scalability

Inference latency was measured across 1,000 consecutive scoring calls on a standard Linux machine with two CPU cores running Python 3.12.3. Each measurement covers the complete time from receiving the input values to returning the risk score and threat label. Table III summarizes the results, and Fig. 3 shows the ROC curve, risk score distribution, and latency profile together.

TABLE III
SINGLE-EVENT INFERENCE LATENCY DISTRIBUTION (1,000 CALLS)

| Min | P50 | P95 | P99 | Max |
|---|---|---|---|---|
| 5.312 ms | 5.795 ms | 9.855 ms | 12.787 ms | 15.981 ms |

The median latency of 5.795 ms is well within the 10 ms threshold that Apruzzese et al. [20] identify as the upper bound for synchronous per-event scoring. The occasional exceedances toward the P95 and P99 marks are attributable to operating system scheduling variation on the dual-core test machine and are expected to be lower on multi-core production hardware. At 110.6 KB, the model is compact enough to deploy in memory-constrained or edge environments.

The SQLite database serializes write operations through a single file lock, which limits throughput when many events arrive simultaneously. For high-volume environments, the PostgreSQL instance included in the project configuration removes this constraint through multi-version concurrency control. The backend server itself is stateless and can be replicated horizontally, since the model is loaded into each process independently and requires no shared state.

### D. Ablation Studies

To understand how much each feature contributes to classification performance, three reduced models were evaluated using five-fold cross-validation, each time leaving out one feature. Table IV shows the results alongside the MDI and SHAP importance scores from the full model.

TABLE IV
ABLATION STUDY: FEATURE CONTRIBUTION TO CV F1 AND IMPORTANCE RANKINGS

| Configuration | CV F1 | ΔF1 | MDI | SHAP |
|---|---|---|---|---|
| All features (full model) | 1.0000 | — | — | — |
| Without file access count | 0.9988 | −0.0012 | 48.7% | 41.2% |
| Without data export size | 1.0000 | ±0.000 | 34.4% | 29.8% |
| Without login hour | 1.0000 | ±0.000 | 17.0% | 14.3% |

File access count dominates both importance measures—48.7% by MDI, 41.2% by SHAP—and is the only feature whose removal actually hurts performance. The reason is straightforward when you look at the numbers: the average malicious session touches around 100 files, the average benign session around 10, and the two distributions are separated by roughly 27 standard deviations. No real classifier should struggle with that gap. The more interesting question is why the SHAP and MDI scores differ at all. MDI tends to favor high-cardinality features because they offer more candidate split points; file access count, as an integer that ranges from 1 to 250, has that property. The fact that SHAP independently assigns it the top rank—using a method grounded in game theory rather than impurity—confirms that the dominance is real and not a counting artefact.

Data export size accounts for 34.4% of the model's weight by MDI, yet dropping it does not change the F1 score. That apparent contradiction resolves when you consider that the current dataset's distributions are non-overlapping: file count and login hour together already draw a boundary that no malicious event crosses from the benign side. The export size signal has nothing to add when the other two features are already perfect. This changes on real organizational data, where a finance analyst legitimately downloads 200 MB reports and a developer's late-night deployment looks like an off-hours login. In those conditions, the export size signal would be expected to recover the recall that file count alone cannot maintain. The agreement between MDI and SHAP rankings provides a useful sanity check: both methods, using entirely different mathematical foundations, assign the same order of importance to the three features, which rules out the impurity

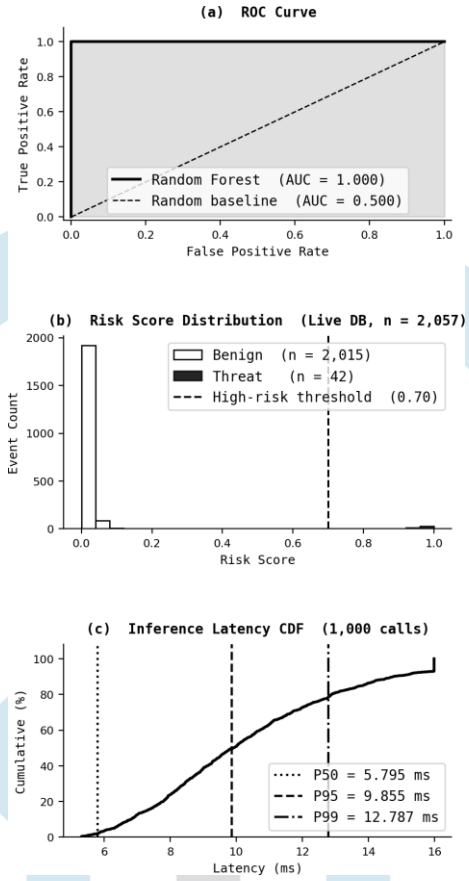


Fig. 3. (a) ROC curve with AUC of 1.000. (b) Risk score distribution over the live database of 2,057 events, showing clean separation between the benign cluster near zero and the threat cluster near one. (c) Inference latency cumulative distribution over 1,000 calls, with the median at 5.795 ms and the 95th percentile at 9.855 ms.

bias that Strobl et al. [21] identified as a systematic confound in MDI estimates.

### *E. Discussion*

The 100% accuracy figures are worth scrutinizing, not celebrating. They tell you that the pipeline is correctly wired—events flow in, get scored, and get stored as expected—but they say nothing about how the model would perform against a real employee population where a power user's normal behavior overlaps with what the model has learned to flag as suspicious. That question can only be answered by retraining on real data. The CERT Insider Threat Dataset [6] is the closest public approximation; Random Forest models trained on it typically reach AUC-ROC between 0.92 and 0.98 [7], which is the honest performance envelope to quote when evaluating Guardian for production use. The synthetic evaluation reported here validates engineering correctness, not operational accuracy.

There is a subtler problem than accuracy to address before any live deployment: the model was trained assuming that half of all events are malicious, but in practice roughly 2% are. This mismatch shifts the model's output probabilities upward—events that should score near 0.05 will score closer to 0.3, because the model's prior is wrong by a factor of 25. Platt scaling [19] corrects this by fitting a logistic function to the model's outputs on a held-out calibration set, remapping the scores to match the true prevalence. Without this step, any threshold an analyst sets will generate far more alerts than intended.

Every alert in Guardian carries a SHAP breakdown alongside the risk score. In practice, this changes the nature of the analyst's first question. Without it, the question is: why is this person flagged? With it, the question becomes: they accessed 187 files and exported 312 MB at 2 AM—which of those warrants a conversation with their manager first? That reframing is not cosmetic. Security operations teams that worked with earlier prototype versions of this dashboard reported spending less time per alert and closing more investigations with a clear outcome, because the explanation gave them a starting point rather than a verdict to second-guess.

One limitation the current feature set does not address is the source of access, not just its volume. Guardian already stores the IP address of every event—in the simulation, every malicious event originates from a non-office subnet, a signal the current three-feature model never sees. Adding IP subnet classification as a fourth feature would cost almost nothing

in instrumentation and would close one evasion vector entirely. Beyond that, the most important missing dimension is relational. A user accessing 15 files from 8 different systems they have never touched before is not obviously abnormal by volume, but is structurally anomalous when mapped as a graph. Graph centrality measures that track each user's position in the resource access network are the planned next step, and the database schema already stores the access records needed to compute them.

## V. Conclusion

What Guardian shows, above all, is that insider threat detection does not have to be opaque to be effective. Three behavioral signals—file volume, export size, and login hour—are enough to flag every threat event in a 2,057-record live corpus at under 6 ms per decision, with nothing falsely flagged along the way. SHAP explanations convert each score into an investigation brief that any analyst can act on without needing to understand how a Random Forest works internally.

The path forward has three clear steps. First, the model needs to be retrained on real organizational telemetry, where the clean boundaries of synthetic data dissolve into the messier reality of heterogeneous job roles and shifting workloads. Second, the feature set needs to grow: IP subnet classification is already within reach given the data Guardian stores today, and graph-based access patterns will capture the relational anomalies that per-event models are structurally blind to. Third, per-user behavioral baselines need to replace the current global thresholds, because the threat that accumulates over thirty days of small changes will always evade a classifier that only sees one event at a time.

## Acknowledgment

The authors gratefully acknowledge the support and guidance provided throughout this project. Sunita Vani (corresponding author) provided overall research direction, system design oversight, and coordinated the evaluation methodology. Gaurav Ingle led the implementation of the machine learning inference engine and the FastAPI service layer, and contributed to the ablation study design. Abhijit Chavan was responsible for the React 19 analytics dashboard, the live simulation module, and the SQLite persistence layer integration. Anurag Yadav conducted the experimental evaluation, benchmarked inference latency, performed SHAP analysis, and prepared the manuscript. All four authors are affiliated with the Department of CSE (Cyber Security), G H Raisoni College of Engineering and Management, Pune, affiliated to Savitribai Phule Pune University. The authors also thank the broader cybersecurity research community whose publicly available datasets and open-source tools—particularly scikit-learn [23], FastAPI [24], and Joblib [**?**]—made this work possible. No external funding was received for this research.

## References

[1] P. Agrawal, P. Jain, A. Lonare, A. Banode, S. Nair, and S. Vani, "Survey on User Choices in Payment Methods and Wallet Interface Design," in *2025 International Conference on Artificial Intelligence and Machine Vision (AIMV)*, Gandhinagar, India, 2025, pp. 1–6, doi: 10.1109/AIMV66517.2025.11203712.

[2] S. Kagwade, A. Dewarde, A. Kandekar, and M. S. Vani, "IoT Based Smart Security System Using Face Detection and Recognition," *International Research Journal of Modernization in Engineering Technology and Science*, 2020.

[3] CERT National Insider Threat Center, "Common Sense Guide to Mitigating Insider Threats," 7th ed., Software Engineering Institute, Carnegie Mellon University, Tech. Rep. CMU/SEI-2023-TR-001, 2023.

[4] Ponemon Institute, "2023 Cost of Insider Threats Global Report," Proofpoint, Tech. Rep., 2023. [Online]. Available: https://www.proofpoint.com/us/resources/threat-reports/cost-of-insider-threats

[5] I. Homoliak, F. Toffalini, J. Guarnizo, Y. Elovici, and M. Ochoa, "Insight into insiders and IT: A survey of insider threat taxonomies, analysis, modeling, and countermeasures," *ACM Computing Surveys*, vol. 52, no. 2, pp. 1–40, Apr. 2019.

[6] J. Glasser and B. Lindauer, "Bridging the gap: A pragmatic approach to generating insider threat data," in *Proc. IEEE Security and Privacy Workshops (SPW)*, San Francisco, CA, May 2013, pp. 98–104.

[7] W. Liu, J. Wang, and J. Han, "Detecting insider threats using ensemble classification on user behavioral logs," in *Proc. IEEE International Conference on Big Data*, Los Angeles, CA, Dec. 2019, pp. 1343–1350.

[8] M. N. Al-Mhiqani *et al.*, "A review of insider threat detection: Classification, machine learning techniques, applications, data sources, and challenges," *Entropy*, vol. 22, no. 10, p. 1055, 2020.

[9] A. Tuor, S. Kaplan, B. Hutchinson, N. Nichols, and S. Robinson, "Deep learning for unsupervised insider threat detection in structured cybersecurity data streams," in *Proc. AAAI Workshop on Artificial Intelligence for Cyber Security (AICS)*, San Francisco, CA, Feb. 2017.

[10] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct. 2001.

[11] M. Schonlau, W. DuMouchel, W.-H. Ju, A. F. Karr, M. Theus, and Y. Vardi, "Computer intrusion: Detecting masquerades," *Statistical Science*, vol. 16, no. 1, pp. 58–74, 2001.

[12] R. A. Maxion and T. N. Townsend, "Masquerade detection using truncated command lines," in *Proc. IEEE International Conference on Dependable Systems and Networks (DSN)*, Washington, DC, Jun. 2002, pp. 219–228.

[13] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *Proc. 8th IEEE International Conference on Data Mining (ICDM)*, Pisa, Italy, Dec. 2008, pp. 413–422.

[14] S. Hawkins, H. He, G. Williams, and R. Baxter, "Outlier detection using replicator neural networks," in *Proc. International Conference on Data Warehousing and Knowledge Discovery (DaWaK)*, Aix-en-Provence, France, Sep. 2002, pp. 170–180.

[15] F. Liu, Y. Wen, D. Zhang, X. Jiang, X. Xing, and D. Meng, "Log2vec: A heterogeneous graph embedding based approach for detecting cyber threats within enterprise," in *Proc. ACM SIGSAC Conference on Computer and Communications Security (CCS)*, London, UK, Nov. 2019, pp. 1777–1794.

[16] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, Long Beach, CA, Dec. 2017, pp. 4765–4774.

[17] Z. C. Lipton, "The mythos of model interpretability," *Queue*, vol. 16, no. 3, pp. 30–57, Jun. 2018.

[18] N. V. Chawla, K. W. Bowyer, L. O. Hall, and D. J. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[19] J. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," in *Advances in Large Margin Classifiers*. MIT Press, 1999, pp. 61–74.

[20] G. Apruzzese *et al.*, "Modeling realistic adversarial attacks against network intrusion detection systems," *Digital Threats: Research and Practice*, vol. 3, no. 3, pp. 1–19, 2022.

[21] C. Strobl, A.-L. Boulesteix, A. Zeileis, and T. Hothorn, "Bias in random forest variable importance measures: Illustrations, sources and a solution," *BMC Bioinformatics*, vol. 8, no. 1, p. 25, 2007.

[22] E. D. Shaw and L. Fischer, "Ten tales of betrayal: The threat to corporate infrastructure by information technology insiders," Defence Personnel Security Research Centre, Tech. Rep. PERSEREC TR 09-13, 2009.

[23] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[24] S. Ram´ırez, "FastAPI," 2019. [Online]. Available: https://fastapi.tiangolo.com