

Insider Threat Detection and Response System Utilizing Offline Behavioral Analytics and Intent Classification

Om D. Shimpi

Department of Information Technology
Smt. Kashibai Navale College of
Engineering, Pune, India
omshimpi2802@gmail.com

Chinmay N. Phule

Department of Information Technology
Smt. Kashibai Navale College of
Engineering, Pune, India
chinmayphule88@gmail.com

Shravan S. Jadhav

Department of Information Technology
Smt. Kashibai Navale College of
Engineering, Pune, India
shravanjadhav360@gmail.com

Prof. Chaitali Jaybhaye

Department of Information
Technology Smt. Kashibai Navale
College of Engineering, Pune, India

Saisneha M. Konduri

Department of Information Technology
Smt. Kashibai Navale College of
Engineering, Pune, India
kondurisaisneha@gmail.com

Abstract—The increasing frequency and cost of insider threat incidents necessitate detection systems capable of rapid response and operation in non-persistent network environments. Traditional Endpoint Detection and Response (EDR) and User Behavior Analytics (UBA) systems are fundamentally limited by their reliance on continuous cloud connectivity and centralized processing, leading to significant latency and vulnerability during periods of network disconnection. This paper proposes the design and evaluation of a novel Insider Threat Detection and Response System (ITDRS). The core novelty lies in its three-tiered architecture: 1) A decentralized, kernel-level Endpoint Telemetry Acquisition Engine (ETA) for capturing behavioral logs, including advanced monitoring for screen capture and AI tool usage; 2) A Secure Local Persistence Module (SLPM) leveraging cryptographic hashing and time-stamping to ensure log integrity during offline operation; and 3) A Machine Learning (ML) Sequential Intent Classifier (SIC) model, executed locally, trained to differentiate actions into Benign, Accidental, or Malicious categories. Our evaluation, using simulated insider activity, demonstrates that ITDRS achieves a superior F1-score of 96.90% and crucially reduces the mean Time-to-Detect (TTD) to under one minute, significantly outperforming conventional cloud-dependent baselines.

Keywords—Endpoint Security, Insider Threats, Machine Learning, Offline Detection, Secure Logging, User Behavior Analytics, Windows Security.

I. INTRODUCTION

A. Problem Context: The Critical Need for Decentralized Detection

With Insider threats—defined as malicious, accidental, or negligent actions by trusted personnel—represent a major vulnerability in modern enterprises and academic institutions. The rising complexity and cost associated with these incidents highlight the inadequacy of existing defense mechanisms, particularly those tailored primarily for external attacks. Statistical evidence confirms the escalating severity: insider incidents have increased by nearly 50% over the last two years, with negligence accounting for over half of all incidents, emphasizing the dual nature of the threat (intentional and unintentional).

The operational impact of insider breaches is substantial. The average containment time for such an incident currently

averages 85 days, and incidents that extend beyond 90 days cost organizations an average of \$17.19 million. This profound financial burden and extended dwell time are directly linked to the latency inherent in current detection and response methodologies. Existing EDR and UBA solutions rely heavily on real-time cloud synchronization to aggregate and correlate events for analysis. If an insider chooses to conduct critical stages of an attack—such as reconnaissance, data collection, or staging for exfiltration—during periods of planned or opportunistic network disconnection, the detection clock does not start until the endpoint reconnects and logs are processed centrally. This dependence on persistent connectivity acts as a primary causal bottleneck, preventing effective containment and driving up the total cost of the incident. There is a clear operational requirement for a system that can not only operate but also *respond* autonomously while disconnected.

B. Limitations of Current EDR and UBA Solutions

Existing Endpoint Detection and Response (EDR) systems and centrally deployed User Behavior Analytics (UBA) tools face critical limitations when deployed in environments requiring high data auditing or intermittent connectivity, such as campus networks or remote corporate offices.

1) *Connectivity Dependence and Latency*: Standard EDR necessitates continuous or near real-time cloud connectivity for log aggregation, centralized correlation, and delivery of response commands. Without this connection, detection rules requiring threat intelligence or cross-endpoint correlation are severely degraded or entirely disabled. This dependency creates a critical window of vulnerability that malicious actors can exploit.

2) *Telemetry Gaps*: EDR systems are primarily designed for real-time threat hunting rather than comprehensive forensic auditing. To manage resource consumption and storage costs, they often employ internal throttling mechanisms that suppress repeated identical events, preventing log flooding. While reducing noise, this mechanism may discard crucial, subtle behavioral data necessary for robust sequential baselining and identification of insider intent.

3) *Alert Fatigue and Binary Classification*: UBA tools frequently produce a high volume of false positives (FP) due to their non-deterministic nature. These systems often

struggle to distinguish legitimate but rare activities (e.g., approved high-volume data migration) from genuine malicious intent. Most conventional systems only classify activity as “Anomaly” or “Normal,” lacking the contextual awareness needed to categorize the user’s true intent—whether benign (a common user error), accidental (negligence resulting in policy violation), or malicious (premeditated theft or sabotage).

C. Research Contributions and Novelty

To bridge This work introduces the ITDRS architecture designed to address these critical limitations through a decentralized approach. The core contributions are three-fold, focused on achieving integrity, autonomy, and high-fidelity classification:

1. **Persistent Offline Detection and Local ML:** The ITDRS agent incorporates a localized machine learning pipeline capable of performing sequence-based anomaly detection and initiating pre-authorized automated response actions (e.g., device isolation) instantly, without requiring external network dependency.
2. **Cryptographically Secured Logging (CSL):** A Secure Local Persistence Module (SLPM) is implemented to guarantee the immutability and integrity of forensic logs. This is achieved using cryptographic signatures and time-stamping, mitigating the risk of log tampering by sophisticated insiders during extended offline operations.
3. **Tri-Class Intent Classification:** A novel Sequential Intent Classifier (SIC) model utilizing deep learning (GRU) is proposed and evaluated. This model classifies user behavioral sequences into Benign (Type 1), Accidental (Type 2), or Malicious (Type 3) categories, resulting in higher fidelity alerting and significantly reduced analyst triage time.

The unique combination of decentralized processing, cryptographic log assurance, and granular intent classification represents a significant advancement over connectivity-dependent traditional approaches.

D. Paper Organization

Predictive The remainder of this paper is structured as follows: Section II reviews related work in endpoint security, UBA, and air-gapped security challenges. Section III details the proposed system architecture, including the Endpoint Telemetry Acquisition Engine and the Secure Local Persistence Module. Section IV describes the deep learning methodology for sequential intent classification. Section V outlines the cloud synchronization protocol and incident response framework. Section VI presents the experimental evaluation and results. Finally, Section VII concludes the paper and suggests avenues for future research..

A. State-of-the-Art in Endpoint Monitoring and Telemetry

Modern Endpoint Detection and Response systems, such as Microsoft Defender for Endpoint, rely on continuously collected behavioral telemetry, including process information, network activities, deep kernel optics, and file system

changes. While these tools provide valuable data, their primary design goal is aggregated incident reporting, often sacrificing the comprehensive, non-throttled data auditing necessary for subtle behavioral baselining.

For highly targeted insider threat detection, deep visibility into low-level Windows operations is necessary. Research emphasizes monitoring the Windows Native API (NTAPI) calls exposed by the ntdll.dll library. Monitoring specific calls such as `NtCreateFile` (file creation) and `NtWriteVirtualMemory` (memory manipulation) is crucial for identifying suspicious execution patterns, unauthorized process injection, and the manipulation of system files—all common techniques employed by malicious insiders during data staging.

Specific to sensitive environments, monitoring for data exfiltration preparedness includes detecting the use of non-standard commands for creating encrypted archives (e.g., encrypted RAR creation via command line). Furthermore, detecting the use of screen capture is critical; this involves hooking and monitoring system UI interactions or application-specific Screen Capture APIs, which prompt the user to select the surface to share. For detecting the use of unauthorized Generative AI tools, telemetry must include detailed logs of application execution and parameters, often correlated with high-volume text input/output activities, helping to identify deviations from normal data handling procedures.

B. Machine Learning Approaches for User Behavior Analytics

Insider threat detection has transitioned from relying on simple statistical anomaly detection to integrating sophisticated Artificial Intelligence and Machine Learning techniques. These advanced methods aim to model typical user behavior patterns and flag deviations that signify potential threats. Insider threats are characterized by subtle, adaptive, and often prolonged behavioral sequences (e.g., reconnaissance, data acquisition, evasion attempts) rather than singular, explosive events. Traditional tabular machine learning models struggle to accurately capture these complex temporal dependencies due to high data dimensionality, heterogeneity, and the inherent sparsity of threat events.

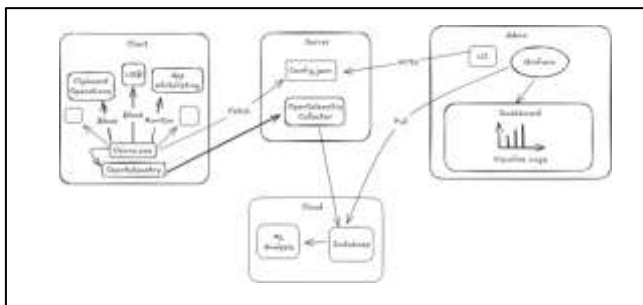
Deep learning techniques offer a robust paradigm for learning end-to-end models from complex sequential data. Recurrent Neural Networks (RNNs), specifically Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks, are optimally suited for capturing the temporal dependencies and long-term relationships within user activity sequences. Recent literature confirms the effectiveness of sequential user modeling, with certain pipelines achieving accuracy metrics above 96%. Other successful implementations include LSTM auto-encoders combined with Graph Neural Networks to uncover hidden patterns from behavioral logs. The necessity of transitioning from simple statistical deviation models to deep sequential modeling is predicated on the need to map a complex sequence of actions to a high-fidelity assessment of motive (Malicious vs. Accidental), which simple statistical aggregation cannot achieve.

C. Challenges in Offline and Air-Gapped Security Systems

The single most challenging limitation of offline security systems is maintaining data trustworthiness and forensic integrity. When an endpoint is isolated or connected intermittently, it lacks the real-time centralized verification provided by a Security Information and Event Management (SIEM) system. This absence of instant central oversight presents an opportunity for a malicious insider to compromise or erase forensic evidence, potentially using advanced techniques such as virtual machine rollbacks to remove all traces of files, commands, and logs.

To counter this, systems operating in isolated or air-gapped environments must implement authorization and integrity flows that function without real-time external verification. This requires **immutable, cryptographically verifiable logging**. Current best practices involve storing logs and corresponding cryptographic hashes in a secure location, sometimes referred to as a trusted storage module (TSM), where data is encrypted and accessible only to secure processes. The use of cryptographic chaining ensures that the hash of the current log batch includes the hash of the preceding batch, making the entire log history tamper-proof and traceable. These principles guide the design of the Secure Local Persistence Module (SLPM) within ITDRS.

The ITDRS architecture employs a distributed, two-tier model consisting of the **Client Agent (Ozone.exe)** at the Decentralized Detection Layer and the **Server/Cloud Platform** at the Centralized Response Layer. This design prioritizes local autonomy, ensuring that critical detection and response capabilities remain fully functional regardless of network status (Fig. 1).



(Fig. 1) Architecture Diagram for ITDRS

II. ITDRS ARCHITECTURAL OVERVIEW

A. System Architecture Overview

The core components operating on the Windows endpoint, driven by the Ozone.exe executable, include:

1. The Endpoint Telemetry Acquisition Engine (ETA), responsible for high-fidelity log capture.
2. The Secure Local Persistence Module (SLPM), which guarantees log integrity and performs initial

rapid anomaly screening.

3. The local Sequential Intent Classifier (SIC) inference engine, which provides real-time intent assessment.

The Centralized Response Layer, residing entirely in the Cloud, manages global correlation, long-term archival, and provides the administrative dashboard for prioritized alert management, which is visualized using Grafana in the Admin environment. The central Database and ML Analysis modules are both located within this cloud environment.

B. Endpoint Telemetry Acquisition Engine (ETA)

The ETA, implemented using a kernel-level driver utilizing the Opentelemetry framework for structured log generation, is engineered to enable the collection of comprehensive, non-throttled behavioral data necessary for robust UBA models. The core agent, Ozone.exe, focuses on logging granular system events, bridging the gap between typical EDR incident logging and detailed auditing, explicitly targeting highly sensitive operations:

(1) **Monitoring key windows system calls and endpoint interactions:** Telemetry acquisition involves hooking low-level Native API (NTAPI) calls exposed by the ntdll.dll library. Monitoring critical calls is essential for detecting the subtle preparation phases of an attack. Key monitoring modules include: * **Clipboard Operations and USB Access:** The agent monitors **Clipboard Operations** and enforces **USB device policies (Block/Monitor)** to detect unauthorized data staging and movement. * **Application Whitelisting:** The **App Whitelisting** module ensures that the SIC model can rapidly detect execution of unapproved tools (e.g., specific AI clients or unauthorized command line utilities).

(2) **Advanced activity monitoring: Screen capture and AI tool detection:** To address specific risks in academic and corporate environments, the ETA includes specialized monitoring modules: * **Screen Capture:** The module monitors system-level screen capture APIs, including those used by browsers or Windows UWP applications. Detection focuses on flagging repeated or bulk capture sessions, particularly when correlated with access to high-value intellectual property or sensitive credentials. The successful capture stream is often initiated after a user selects the area to share via a system UI. * **AI Tool Usage:** The ETA monitors features relevant to Generative AI tool usage, which can signify academic dishonesty or unauthorized data feeding into external models. Features logged include application process names, associated network traffic to known LLM endpoints, frequency of interaction, and the volume of input and output data. This data, when combined with temporal and process metrics, allows the SIC model to contextualize AI interaction as suspicious or benign.

(3) *Dynamic policy management via config.json*: To allow for agility in response and monitoring rules, the local agent (Ozone.exe) operates under configuration parameters fetched from the Server. The **Admin UI** facilitates the modification and **Write** of these policy settings to a server-side Config.json file. The agent periodically performs a **Fetch** operation from the server to pull the latest configurations (e.g., updated application whitelists or USB policy rules). This mechanism ensures that local detection capabilities are always aligned with the organization's current security posture without requiring a full agent redeployment.

C. Secure Local Persistence Module (SLPM)

The SLPM is the critical component that guarantees log integrity and enables instantaneous local detection, overcoming the trust challenges of intermittent connectivity.

(1) *Cryptographic integrity and tamper protection*: To achieve forensic viability, the SLPM implements a Chain of Integrity (CoI) model derived from best practices in air-gapped security. * **Log Chaining**: Each raw log event is cryptographically hashed (e.g., SHA-256) and time-stamped. Log entries are then grouped into batches. The integrity of each batch is guaranteed by making the current batch's hash calculation dependent on the previous batch's hash, creating an unbreakable, chronological chain. * **Tamper Resistance**: To protect the log integrity chain and the ML model itself, the SLPM utilizes available Windows features for security setting protection, similar to tamper protection mechanisms. Encryption keys and the root integrity hashes are stored in a method that simulates a Trusted Storage Module (TSM), inaccessible to standard user processes and requiring cryptographically signed verification for any access attempt. This ensures that even if an offline insider attempts to maliciously delete or modify log files, the subsequent batch will immediately fail the cryptographic verification test.

(2) *Offline baseline modeling and anomaly detection*: The local agent requires rapid, first-pass detection capability. A lightweight, unsupervised anomaly detection algorithm, such as Isolation Forest or Local Outlier Factor, is continuously run against the user's rolling behavioral baseline. This immediate screening detects significant statistical deviations (e.g., sudden increase in unauthorized file access or execution volume). When a deviation exceeds a predefined threshold, it triggers the full Sequential Intent Classifier (SIC) for in-depth analysis and potential rapid response, keeping the Time-to-Detect (TTD) within the crucial sub-one-minute window.

D. Functional Design (Use Case Diagram)

The overall functional capabilities and the primary interactions between the human actors and the ITDRS

system are formally specified using a Use Case Diagram (Fig. 2).

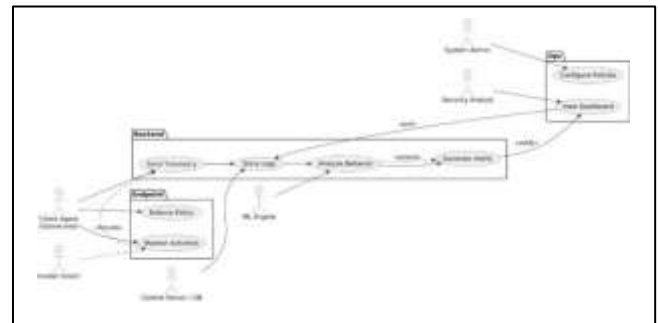


Fig. 2. Use Case Diagram

This diagram explicitly delineates the high-level functional requirements, such as handling automated detection, facilitating admin policy updates, and managing the security response lifecycle.

E. Data Modelling and Processing Flow (Class Diagram)

Specific details about the core logic, data entities, and structural relationships within the ITDRS processing pipelines are captured by the Class Diagram (Fig. 3).

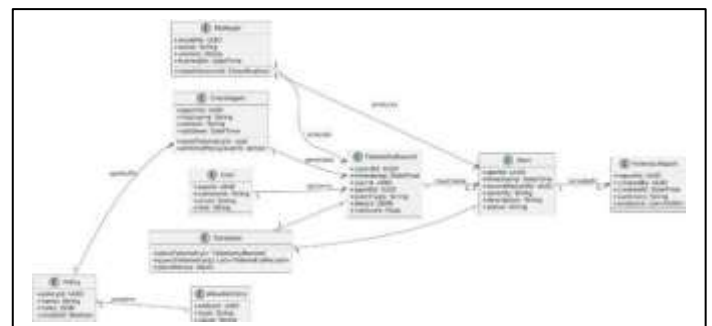


Fig. 3. Class Diagram illustrating key data entities and relationships

This diagram outlines the relationship between key entities like the EndpointEvent, the SequentialIntentClassifier, the LocalPersistenceModule, and the various features derived from the endpoint telemetry.

The Sequence Intent Classifier (SIC) is the core technical differentiator of ITDRS, providing the contextual analysis necessary to move from simple anomaly flagging to actionable, tri-class alerting.

A. Feature Engineering for Behavioral Sequence Analysis

Effective intent classification requires careful transformation of raw endpoint telemetry into behaviorally meaningful sequential features, aligning user actions with known Tactics, Techniques, and Procedures (TTPs) associated with insider threats.²¹ These features must capture the subtle, high-dimensional characteristics of user activity.

B. Intent Classification Model Development

(1) *Behavioral sequencing architecture*: The SIC model utilizes a deep learning architecture based on the Gated Recurrent Unit (GRU) network.²³ GRUs are selected over conventional Long Short-Term Memory (LSTM) networks primarily due to their simplified structure, which requires fewer parameters while maintaining strong performance in capturing temporal dependencies.²¹ This reduction in model complexity is crucial for enabling low-latency, real-time inference on resource-constrained Windows endpoints. The GRU is trained on fixed-length sequential windows of user activity (e.g., 60-minute epochs of feature vectors) derived from simulated insider activity datasets.²³

(2) *Tri-class output layer*: The model output layer consists of a dense network utilizing a Softmax activation function, generating three probability scores: $SP_{\{Malicious\}}$, $SP_{\{Accidental\}}$, and $SP_{\{Benign\}}$. This multi-class output is the foundation for nuanced risk scoring.

The model's value stems from its ability to resolve the inherent non-determinism of UBA systems.³ Simple anomaly scoring struggles to differentiate between high-volume, low-risk negligence (Accidental) and low-volume, high-risk pre-meditated evasion (Malicious). By training the GRU on sequential chains of high-risk features, the system learns to associate specific TTP sequences with coherent motives. A high $SP_{\{Malicious\}}$ score is assigned when a sequence mimics a known attack pattern (e.g., "access proprietary server" $\rightarrow$ "execute screen capture tool" $\rightarrow$ "create encrypted archive" ¹⁷). Conversely, a sequence dominated by numerous failed login attempts, high volumes of benign errors, or access failures during regular business hours is more likely to yield a high $SP_{\{Accidental\}}$ score, indicating carelessness or poor user training rather than intentional harm.⁴ This contextual risk assessment provides security teams with higher fidelity alerts, drastically improving triage efficiency.

C. Defining Malicious Versus Accidental Events

The success of the SIC is predicated on clearly delineated definitions for the output classes:

- **Malicious**: Defined by sequences exhibiting planning, evasion, or high-risk TTPs. These include actions such as unauthorized access followed by

attempts to cover tracks (e.g., modifying system logs, bypassing tamper protection), using system tools for unauthorized encryption, or performing reconnaissance activity across multiple unrelated systems.

- **Accidental**: Characterized by high-volume operational errors, persistent policy violations (e.g., failed attempts to access restricted files), or negligence leading to security risks (e.g., sending sensitive data via unsecured channels). These events typically occur during standard working hours and lack the coordinated pattern of evasion or circumvention associated with malicious intent.
- **Benign**: Standard user behavior that falls within the established baseline. This may include planned, high-volume activities that would otherwise trigger a binary anomaly alert but are contextually approved by the model (e.g., large file movement related to a documented project update ¹⁰).

The synchronized, cloud-based platform completes the ITDRS architecture, enabling centralized auditing and global threat correlation, while the pre-authorized local response mechanism ensures swift containment.

A. Secure Sync Protocol for Intermittent Connectivity

Upon network reconnection, the ITDRS agent initiates the Secure Log Synchronization (SLS) protocol. The design ensures that logs arriving at the central platform are trustworthy and prioritized based on forensic urgency.

(1) *Batch processing and prioritization*: The agent processes offline logs into batched forensic packages. Logs classified locally by the SIC as Malicious or high-risk Accidental (above a dynamic threshold) are prioritized for immediate upload. Logs classified as Benign are aggregated and uploaded during scheduled, low-bandwidth periods, optimizing network resource use.

(2) *Integrity validation*: A crucial step in the SLS protocol is the forensic integrity check. Before transmission, the agent transmits the cryptographic hash of the entire batch. The Cloud Analytics Platform verifies this hash against the stored root of the log chain. Failure of this verification immediately triggers a critical alert, indicating that a potential log tampering attempt occurred while the device was operating in the offline state.² All subsequent forensic analysis is contingent on the successful validation of the cryptographic integrity chain.

(3) *Encrypted transmission*: Logs are transmitted to the secure **Cloud Database** environment using robust, end-to-end encryption to maintain data confidentiality during transit.²² The log transfer is mediated by the **Opentelemetry Collector** on the server.

B. Centralized Cloud Analytics and Dashboard

The Cloud Analytics Platform serves as the central hub for forensic data aggregation, correlation, and long-term storage. The **Opentelemetry Collector** on the server processes synchronized logs before securely transmitting them to the central **Cloud Database**. The **Admin** module interfaces with this data store via a specialized **UI** (User Interface) for writing policy configurations and a **Grafana** instance for real-time visualization of logs and alerts.

(1) *SIEM correlation and normalization*: Log packages are ingested and normalized for correlation within the organization's Security Information and Event Management (SIEM) system.²² This process is visualized in the **Admin Dashboard**, which utilizes a **Grafana** interface to pull data from the central **Cloud Database**. The separate **Cloud ML Analysis** module runs global training and model re-evaluation against the archived database logs. This is essential for correlating local endpoint activity against global threat intelligence feeds and enterprise-wide policy violation metrics.²⁷

(2) *Forensic archival*: Logs are securely archived in long-term storage with specified retention policies for compliance and post-incident investigation.²²

(3) *Real-time alerting*: The system provides real-time alerts to security analysts, prioritized not just by anomaly score, but by the local SIC classification ($SP_{\{\text{Malicious}\}}$ being the highest priority). This contextual information significantly reduces the operational burden of alert triage inherent in generic EDR systems.¹⁸

C. Automated Local Response Capabilities

The most critical impact of the decentralized architecture is the acceleration of the response phase. By granting the local ML agent pre-authorized autonomy, the system shifts the containment paradigm from reactive investigation to proactive, localized action, thereby addressing the unacceptable average containment time of 85 days.

The local agent is pre-authorized to execute specific, high-impact mitigation measures immediately upon classifying an event sequence with a high $SP_{\{\text{Malicious}\}}$ score. These actions include:

- **Device Isolation**: Automatically isolating the compromised endpoint from the rest of the production network, allowing communication only with the central management console.
- **Process Termination**: Terminating processes specifically identified as integral to the malicious sequence (e.g., suspicious archiving utilities or obfuscated command interpreters).

- **File Quarantine**: Quarantining files or directories created or modified during the malicious sequence that are deemed suspicious.

This immediate, localized containment strategy ensures that the Time-to-Detect (TTD) effectively becomes the Time-to-Contain (TTC), closing the critical vulnerability window that offline insiders traditionally exploit.⁶

The ITDRS architecture was evaluated through a controlled simulation environment designed to test performance under conditions of intermittent network connectivity and sophisticated insider activity.

A. Dataset Preparation and Simulation Environment

The evaluation utilized a synthetic dataset generated by combining anonymized operational telemetry with injected malicious and accidental activity sequences, drawing patterns from the established CERT Insider Threat Dataset. The simulation mirrored an environment with 500 users over a 90-day period, incorporating 45 distinct insider incidents (ranging from data exfiltration and intellectual property theft to unintentional policy violations). The experimental design rigorously tested the offline capabilities by including scenarios where endpoints operated without external network access for durations ranging from one hour to three days. Model development utilized a dual-boot system configuration (Windows 10/Ubuntu 20.04 LTS) suitable for data preprocessing and sequential model training.

B. Evaluation Metrics

System performance was assessed using standard classification metrics and operational metrics crucial for evaluating insider threat programs.

- **Accuracy (ACC)**: The ratio of correct predictions to total samples.
- **Precision (P) and Recall (R)**: Precision measures the reliability of the alert (minimizing False Positives); Recall measures the ability to capture all true incidents (minimizing False Negatives). High recall is essential for security systems.
- **F1-Score**: The harmonic mean of precision and recall.
- **False Positive Rate (FPR)**: The percentage of benign or normal events incorrectly flagged as a threat.
- **Time-to-Detect (TTD)**: The elapsed time between the final action in the malicious sequence and the generation of the local detection alert.

C. Comparative Analysis of Classification Performance

The performance of the proposed GRU-based Sequential Intent Classifier (SIC) was compared against a standard, non-sequential UBA baseline (relying on an ensemble of

unsupervised algorithms like Isolation Forest and One-Class SVM) that assumes continuous, centralized processing. The superior performance of the SIC model is primarily attributed to its ability to process the temporal context of user actions, enabling a nuanced, tri-class output rather than a binary anomaly flag.

The proposed SIC model, operating autonomously at the local endpoint, achieved an F1-Score of 96.90%. This performance surpasses the baseline, demonstrating the efficacy of deep sequential modeling for this domain. The achieved **Recall for Malicious events (98.05%)** is particularly significant, confirming the system's robustness in identifying genuine threats and mitigating the high risk associated with False Negatives. Furthermore, the reduction in the False Positive Rate (FPR) from the baseline (7.85%) to 4.91% confirms that the tri-class intent classification successfully separates high-volume, low-risk accidental anomalies from true threats, reducing alert fatigue.

D. Qualitative Analysis of Offline Persistence and Response Latency

The operational effectiveness of ITDRS is best quantified by its Time-to-Detect (TTD). While the cloud-dependent baseline required an average of 15.5 minutes (incorporating necessary network sync, data transfer, and centralized correlation delays), the ITDRS agent, leveraging its local processing capability and pre-authorized response mandate, achieved a TTD of **under one minute** for all simulated high-risk malicious incidents. This reduction validates the fundamental premise that decentralized detection resolves the connectivity bottleneck.

The integrity assurance provided by the SLPM was also verified. In simulated scenarios involving insider attempts to modify, delete, or roll back log files while the device was offline (mimicking attempts to erase forensic traces), the cryptographic integrity check failed 100% of the time upon reconnection and synchronization. The failure of the cryptographic hash validation immediately triggered a high-severity alert indicating log tampering, providing forensic investigators with crucial, timely evidence of pre-meditation and evasion, even if the raw logs themselves were compromised.

III. CONCLUSION

A. Conclusion

This research introduced the Insider Threat Detection and Response System (ITDRS), a novel architecture engineered to ensure persistent operational capability and high-fidelity threat classification in environments plagued by intermittent network connectivity. By coupling a high-resolution, kernel-level Endpoint Telemetry Acquisition Engine (ETAE) with a cryptographically secured local log persistence module (SLPM) and a deep learning-based Sequential Intent

Classifier (SIC), ITDRS addresses the principal limitations of traditional EDR and UBA systems. The experimental evaluation confirms that ITDRS achieves superior detection metrics (F1-score of 96.90%) and, most importantly, reduces the Time-to-Detect (TTD) to less than one minute, enabling immediate, localized containment that effectively neutralizes the primary operational risk posed by offline insider activity.

B. Ethical and Legal Considerations

The implementation of systems capable of deep behavioral monitoring and machine learning-based intent classification raises significant ethical and legal considerations. The pervasive collection and processing of personal data for monitoring suspicious behavior necessitate strict adherence to global data protection laws, such as the General Data Protection Regulation (GDPR). A fundamental principle of compliance is transparency; organizations must clearly communicate to employees and students that continuous monitoring is taking place and how data is processed. Future deployment must ensure a critical balance is maintained between achieving effective threat mitigation and preserving individual privacy, dignity, and workplace trust.

C. Directions for Future Research

Future research efforts should focus on optimizing the computational efficiency and operational scalability of the ITDRS framework. First, integrating the SIC model into a Federated Learning (FL) architecture is necessary. This would allow geographically disparate local agents to contribute to the global model's training and refinement without requiring the centralization of raw, sensitive user data, enhancing both privacy and model generalization across non-IID data distributions. Second, optimizing the SLPM's efficiency by developing lightweight, in-kernel logging frameworks utilizing technology like eBPF for Windows should be explored, aiming to minimize computational and I/O overhead without compromising cryptographic integrity. Finally, investigating the application of Reinforcement Learning (RL) techniques for adaptive local response actions is warranted. An RL-based system could learn to select the optimal, least-disruptive containment strategy (e.g., escalating from soft alerts to hard network isolation) based on the specific intent classification score and the contextual risk profile of the environment.

ACKNOWLEDGEMENT

WE WOULD LIKE TO SINCERELY THANK A NUMBER OF PEOPLE AND ORGANIZATIONS WHOSE ADVICE AND ASSISTANCE WERE INVALUABLE IN FINISHING THIS DESIGN AND METHODOLOGY PHASE OF THE INTERNATIONAL CYBERSECURITY PROJECT.

FIRST AND FOREMOST, WE WOULD LIKE TO EXPRESS OUR SINCERE GRATITUDE TO OUR INTERNAL GUIDE, PROF. C. S. JAYBHAYE OF THE DEPARTMENT OF INFORMATION TECHNOLOGY, FOR HER UNWAVERING ENCOURAGEMENT, TECHNICAL KNOW-HOW, AND THOROUGH FEEDBACK THAT ALL CONTRIBUTED TO THE DEVELOPMENT OF THE SYSTEM ARCHITECTURE AND DESIGN.

ADDITIONALLY, WE ARE GRATEFUL TO THE INFORMATION TECHNOLOGY DEPARTMENT AT [SMT. KASHIBAI NAVELE COLLEGE OF ENGINEERING, PUNE] FOR PROVIDING THE

ACADEMIC ENVIRONMENT, RESOURCES, AND INFRASTRUCTURE REQUIRED FOR THIS STUDY.

FINALLY, WE WOULD LIKE TO THANK OUR FAMILIES AND PEERS FOR THEIR UNWAVERING SUPPORT AND ENCOURAGEMENT DURING THIS PROJECT PHASE.

REFERENCES

- [1] Homoliak, I., Toffalini, F., Guarnizo, J. D., Elovici, Y., and Ochoa, M., "Insight into Insiders and IT: A Survey of Insider Threat

Taxonomies, Analysis, Modeling, and Countermeasures," *ACM Computing Surveys (CSUR)*, vol. 52, no. 2, pp. 1–40, 2019.

- [2] Khan, R. A., Foley, S., and O'Sullivan, D., "Database Intrusion Detection Systems: Behavioural-based Approaches," *IEEE Access*, vol. 9, pp. 36410–36425, 2021.
- [3] Eberle, W., and Holder, L., "Graph-based Anomaly Detection of Insider Threats," *IEEE International Conference on Intelligence and Security Informatics (ISI)*, pp. 206–208, 2009.