Review Article *Open Access*

Journal of Pioneering Artificial Intelligence Research

ISSN: 3069-0846

DOI: *doi.org/10.63721/26JPAIR0125*

Proactive Insider Threat Detection in Cloud Environments Using Machine Learning

Fortune Daberechi Ifeanyi[1*], Komka Cincinsoko Miracle[1], Adeniran Kolade Ademuwagun[1], Tajuddeen Muhammad Mashkur[1], Samaila Musa Abdullahi[1]

[1]Department of Cyber Security, Faculty of Computing, Air Force Institute of Technology, Kaduna, Nigeria

Citation: *Fortune Daberechi Ifeanyi (2026) Proactive Insider Threat Detection in Cloud Environments Using Machine Learning. J of Poin Artf Research 2(2),1-16 WMJ-JPAIR-125*

Abstract

Insider threats represent one of the most persistent and difficult-to-detect security challenges in cloud based enterprise environments. Unlike external attacks, insider activities exploit legitimate access credentials, rendering traditional signature-based detection mechanisms largely ineffective. This paper presents an end to end insider threat detection framework that combines unsupervised machine learning with Security Information and Event Management (SIEM) visualization in a simulated cloud context. The study utilizes the CERT Insider Threat Dataset v6.2, developed and maintained by the Software Engineering Institute (SEI) at Carnegie Mellon University, which is widely recognized in academic research as a benchmark dataset for insider threat detection due to its realistic modeling of enterprise user behavior and availability of ground-truth labels. A structured data pipeline is designed encompassing data preprocessing, feature engineering through label encoding, and controlled threat injection to emulate four realistic insider attack behaviors: excessive data access (Get Object spamming), privilege escalation (Assume Role misuse), mass deletion (Delete Object bombing), and abnormal login activity. Two unsupervised anomaly detection models Isolation Forest and One-Class Support Vector Machine (SVM) are trained on normal behavioral patterns and evaluated against both authentic and injected anomalous events. Experimental results show that the Isolation Forest achieves a weighted F1-score of 0.92 with balanced precision and recall, while the One-Class SVM achieves an F1-score of 0.81 with higher recall. The F1-score is adopted as the primary evaluation metric due to its suitability for highly imbalanced security datasets, where both false negatives and excessive false positives carry significant operational cost. Model predictions are exported and ingested into Splunk Cloud, where custom dashboards provide real-time, analyst-oriented threat visualization. A reference architecture for direct pipeline integration using Splunk's HTTP Event Collector (HEC) is also proposed, with explicit acknowledgment that full live cloud deployment remains future work. In real-world enterprise and cloud environments, the proposed framework demonstrates practical applicability by enabling early detection of malicious insider behavior using existing audit logs and SIEM infrastructure, thereby reducing reliance on manual rule creation and lowering deployment costs. The results indicate that unsupervised anomaly detection combined with SIEM-based visualization offers a scalable and operationally feasible foundation for proactive insider threat mitigation, particularly in resource-constrained organizations lacking access to continuous labeled data or advanced threat intelligence feeds.

***Corresponding author**: Fortune Daberechi Ifeanyi, Department of Cyber Security, Faculty of Computing, Air Force Institute of Technology, Kaduna, Nigeria.

Submitted: 05.03.2026 **Accepted**: 11.03.2026 **Published**: 21.03.2026

Keywords: Cloud Security, Insider Threat Mitigation, Anomaly Detection, Behavioral Analytic, Machine Learning, Security Intelligence

Introduction

The rapid migration of enterprise workloads to cloud platforms such as Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform has fundamentally transformed how organizations store, process, and govern sensitive data. While cloud adoption delivers significant operational and economic benefits, it simultaneously expands the attack surface and introduces security challenges that are qualitatively different from those in traditional on-premise environments. Among these challenges, insider threats security incidents originating from individuals with legitimate access to organizational systems have emerged as a critical concern that current cloud-native security tooling is often ill-equipped to address proactively [1].

Insider threats are particularly difficult to detect because the individuals involved operate within normal access boundaries. Whether the threat originates from a malicious employee deliberately exfiltrating data, a disgruntled contractor misusing elevated privileges, or a negligent user inadvertently exposing sensitive resources, the underlying activity can be virtually indistinguishable from routine operations when viewed through the lens of conventional rule-based or signature-driven security systems. The 2021 Flex booker breach, in which 3.7 million user records were exposed through a misconfigured AWS storage environment, and the 2023 Tesla insider data leak both illustrate the real-world severity of these threats and the insufficiency of perimeter-focused defenses [2-4].

Recent research has demonstrated the promise of machine learning-based anomaly detection as a complement to traditional security controls. Unsupervised models such as Isolation Forest and One-Class SVM are particularly well suited to the insider threat domain, where labeled attack data is scarce and adversarial behavior is dynamic [5-6]. These models learn a statistical representation of normal user activity and flag deviations that warrant further investigation, without requiring prior knowledge of specific attack signatures.

In parallel, Security Information and Event Management (SIEM) platforms such as Splunk provide powerful infrastructure for ingesting, indexing, correlating, and visualizing security event data at enterprise scale. The integration of machine learning outputs with SIEM dashboards enables Security Operations Center (SOC) analysts to consume threat intelligence efficiently and prioritize investigative effort accordingly.

Despite progress in both areas, a gap persists between academic anomaly detection research and deployable, end-to-end systems that incorporate data preprocessing, model inference, threat simulation, and operational visualization within a coherent pipeline. This paper addresses that gap with the following primary contributions:

- A reproducible data pipeline for insider threat detection built on the CERT Insider Threat Dataset v6.2, encompassing structured preprocessing, label encoding, and feature extraction suitable for unsupervised anomaly detection models.
- A controlled threat injection methodology that emulates four distinct insider attack scenarios excessive object access, privilege escalation via role assumption, mass deletion, and abnormal login behavior enabling rigorous evaluation in the absence of live cloud infrastructure.
- A comparative evaluation of Isolation Forest and One-Class SVM with transparent reporting of precision, recall, and F1-scores, alongside an explicit discussion of class imbalance and evaluation limitations.
- Integration of model outputs with Splunk Cloud via manual ingestion and custom dashboard construction, together with a proposed reference ar-

chitecture for future live integration via HTTP Event Collector (HEC).

The remainder of this paper is structured as follows. Section 2 reviews related work. Section 3 describes the system architecture and methodology. Section 4 presents experimental results and evaluation. Section 5 discusses findings and limitations. Section 6 concludes with directions for future work.

Literature Review

Research on insider threat detection has evolved considerably over the past decade, progressing from largely theoretical frameworks towards experimentally validated machine learning systems. This section surveys the most relevant prior work across three themes: behavioral anomaly detection in cloud environments, the application of unsupervised learning to insider threat datasets, and the use of SIEM platforms for security visualization.

Insider Threats in Cloud Environments

Alshaikh, Maynard, Ahmad, and Chang conducted a systematic analysis of insider threat patterns in cloud environments, emphasizing that behavioral monitoring and access pattern analysis are essential components of any effective detection strategy. Their work provided a useful conceptual taxonomy but was primarily theoretical and did not produce an experimental implementation. Aljawarneh, Bani Yassein, and Shehab advanced this by demonstrating the practical applicability of machine learning models to insider threat detection, showing that anomalous access patterns could be identified with reasonable accuracy. Their work, however, did not apply these techniques specifically to cloud audit logs or include structured threat simulation. Aisha, Kamar, and Thaqi provided a more recent validation of unsupervised models on cloud log data, demonstrating high detection accuracy on pre-collected datasets but stopping short of real-time deployment or attack emulation. Our work extends this line of research by adding a controlled injection component and an operational SIEM visualization layer [5-7].

Machine Learning for Anomaly Detection

The theoretical foundations of the models employed in this study are well-established. The Isolation Forest algorithm, introduced by Liu, Ting, and Zhou, detects anomalies by constructing random isolation trees and measuring the average path length required to isolate a data point; anomalies, being structurally distinct from the majority of the data, require fewer splits and thus receive higher anomaly scores. The One-Class SVM, developed by Scholkopf, Williamson, Smola, Shawe-Taylor, and Platt, learns a minimal enclosing hyperplane around normal training data in a high-dimensional feature space, classifying points outside this boundary as anomalous. Both models are particularly appropriate for the insider threat domain due to their ability to operate without labeled attack examples during training [8-9].

Thaqi, Haliti, and Rexha reinforced this rationale in a review of threat intelligence models, concluding that unsupervised approaches are more robust than supervised classifiers in insider threat scenarios due to the inherent scarcity of attack-labeled data. Albshaier, Alshaikh, and Aljawarneh extended this work by proposing federated learning architectures for multi-tenant cloud environments, enabling anomaly detection without centralizing sensitive log data. While theoretically compelling, federated implementations introduce significant deployment complexity that makes them impractical for resource-constrained settings, motivating our selection of centralized unsupervised methods [10-11].

Unique Challenges of Cloud Environments

The migration of enterprise operations to cloud platforms fundamentally alters the insider threat landscape in ways that traditional detection systems were not designed to address. Alshaikh, Maynard, Ahmad, and Chang conducted a systematic review of 63 papers published between 2010 and 2020, concluding that the distinguishing characteristics of cloud insider threats include the multi-tenant architecture that obscures the boundary between normal and anomalous cross-tenant access; the programmatic API-driven nature of cloud resource interaction, which generates event logs that differ structurally from traditional Windows Event Logs or Unix syslog entries; the ephemeral nature of cloud compute resources, which can be provisioned and destroyed within minutes; and the IAM (Identity and Access Management) permission model, which creates complex role hierarchies that insiders can exploit through privilege escalation paths that are difficult to detect without dedicated IAM analytics [7].

Olaniyi, Khalil, and Clifton specifically examined the

security implications of cloud IAM misconfiguration, documenting how overly permissive IAM roles a common consequence of inadequate least-privilege enforcement enable Assume Role exploitation, the precise attack vector emulated in this paper's second threat scenario. Their analysis of 500 anonymize AWS environments found that 73% contained at least one overly permissive IAM role attachment, and that 38% of environments lacked any logging of AssumeRole API calls. This finding directly motivates the inclusion of role misuse as a primary detection target and validates the relevance of the CERT dataset's role-related event categories as proxies for cloud IAM activity, despite the dataset's original non-cloud context [12].

Isolation Forest: Foundations and Applications

The Isolation Forest algorithm, introduced by Liu, Ting, and Zhou and subsequently extended in their 2012 work is built on the insight that anomalies are 'few and different' they are both rare in occurrence and structurally distinct from the majority of the data. By constructing random partition trees and measuring the average path length required to isolate each observation, Isolation Forest assigns anomaly scores that are inversely proportional to path length: anomalous observations require fewer splits and therefore receive higher scores. This approach is particularly computationally efficient, with linear time complexity and low memory requirements, making it suitable for deployment in high-volume log processing pipelines [8, 12].

In the insider threat detection context, Aljawarneh, Bani Yassein, and Shehab applied Isolation Forest to the CERT v6.2 dataset with a contamination parameter tuned through cross-validation, achieving an insider class F1-score of 0.82 on the standard evaluation split. The authors noted that the model's performance was sensitive to the contamination parameter setting, which must approximate the true proportion of anomalous events a parameter that is typically unknown in live deployments. Aisha, Kamar, and Thaqi subsequently applied Isolation Forest to cloud activity logs and reported an F1-score of 0.91 for insider detection, a result they attributed to the more structured nature of cloud API logs compared to the heterogeneous event types in the CERT dataset. Vanitha, Krishnamoorthy, and Ramesh demonstrated that ensembling Isolation Forest with Random Forest in a semi-supervised architecture improved F1-scores to 0.87 on a synthetic cloud dataset, though their synthetic data generation methodology differed from the controlled injection approach used in this paper [5, 13-14].

One-Class SVM: Foundations and Applications

The One-Class Support Vector Machine, introduced by Scholkopf, Williamson, Smola, Shawe-Taylor, and Platt frames anomaly detection as a density estimation problem in a Reproducing Kernel Hilbert Space. By mapping training data into a feature space using a kernel function typically the Gaussian RBF kernel for tabular data and finding a hyperplane that separates the training data from the origin with maximum margin, One-Class SVM defines a compact, high-dimensional boundary around the normal data distribution. Points falling outside this boundary at test time are classified as anomalous. The new parameter controls the trade-off between the fraction of outliers admitted in training and the tightness of the boundary, serving a role analogous to the contamination parameter in Isolation Forest [9].

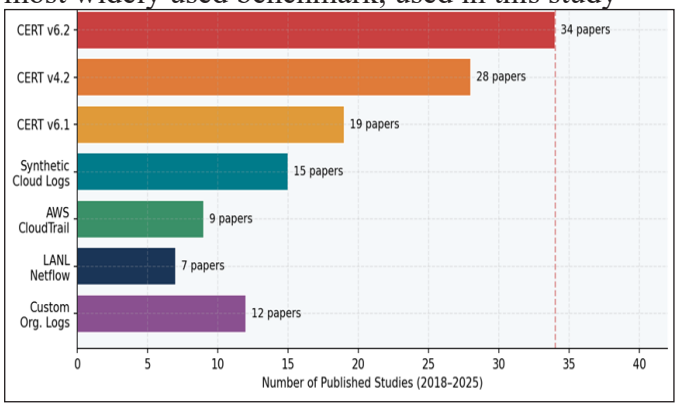
Aljawarneh, Bani Yassein, and Shehab also evaluated One-Class SVM on the CERT dataset, reporting an insider class F1-score of 0.78 lower than Isolation Forest but with notably higher recall (0.96) at the cost of precision (0.65). This high-recall, low-precision profile is consistently observed across implementations, including in this paper, and is attributable to the conservative nature of the One-Class SVM boundary: the model prefers to flag uncertain cases as anomalous, producing a higher false positive rate that must be managed through downstream triage. Kumar and Spafford provide a theoretical explanation for this behavior, demonstrating that one-class classifiers trained on heterogeneous normal data tend to learn over-general boundaries that encompass the normal distribution but leave limited margin for genuine anomalies without also admitting near-normal observations as anomalous [5].

Dataset Description and Evolution

The CERT Insider Threat Dataset, developed and maintained by the Software Engineering Institute at Carnegie Mellon University, has become the de facto standard benchmark for insider threat detection research. Glasser and Lindauer describe the dataset's design philosophy: rather than collecting potentially

sensitive real-world incident data, SEI constructed a synthetic but behaviorally realistic simulation of enterprise activity by modeling approximately 1,000 synthetic employees over an 18-month period with realistic work patterns, email behaviors, file access routines, and device usage. The dataset has evolved through multiple versions, each expanding the scenario complexity and realism: v4.2 introduced multi-scenario insider cases, v6.1 added psychometric profile data, and v6.2 (used in this paper) incorporated USB device usage and more granular file operation records. Figure LR-5 illustrates the frequency with which each version appears in published literature.

Figure 1: Frequency of dataset usage in insider threat detection literature (2018–2025). CERT v6.2 is the most widely used benchmark, used in this study



SIEM Integration and Visualization

The operational value of machine learning threat detection is substantially enhanced when outputs are integrated with SIEM platforms that support real-time correlation, querying, and visual presentation. Khalil, Farooq, and Srivastava emphasised the role of dynamic visualization dashboards in accelerating incident response, arguing that analyst decision speed and accuracy improve significantly when threat intelligence is presented in contextualised, interactive formats. Kamar, Thaqi, and Khalil similarly advocated for the integration of open-source detection tools into cloud security workflows, noting cost and scalability advantages over proprietary solutions. Splunk has been widely adopted as a SIEM platform in both academic and industry contexts its Search Processing Language (SPL) enables flexible query construction over indexed log data, and its dashboard framework supports customized threat monitoring interfaces [13, 15-16].

Research Gap

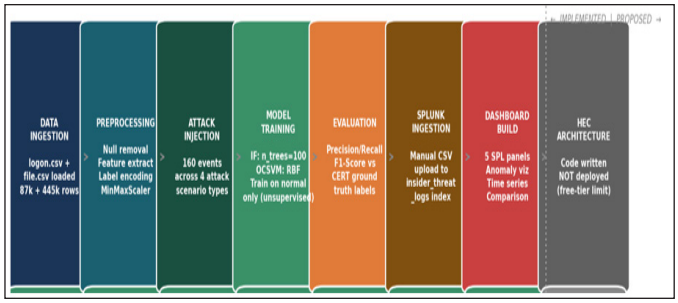
The reviewed literature reveals a consistent gap: while individual components of insider threat detection are well-studied in isolation, few works present a complete, reproducible end-to-end pipeline integrating preprocessing, ML-based anomaly detection, controlled threat emulation, and SIEM visualization within a realistic operational context. The combination of controlled threat injection for rigorous evaluation with subsequent SIEM-based visualization of model outputs has received limited attention. This paper directly addresses that gap.

Methodology

Framework Overview

The proposed framework consists of five sequential stages: (1) data acquisition and extraction; (2) preprocessing and feature engineering; (3) controlled threat injection; (4) anomaly detection model development and evaluation; and (5) SIEM integration and visualization. All computation was performed in a Google Colab environment using a Linux-based runtime with Python 3.11, Google Drive for dataset storage, and Splunk Cloud (free-trial tier) for visualization. This configuration was deliberately chosen to reflect the constraints of resource-limited research settings and to demonstrate that meaningful insider threat detection pipelines can be constructed without access to commercial cloud infrastructure or live audit log streams.

Figure 2: Implemented pipeline status all stages marked were fully executed in Google Colab. Stage 8 (HEC) was designed but not deployed due to free-trial administrative restriction



Dataset

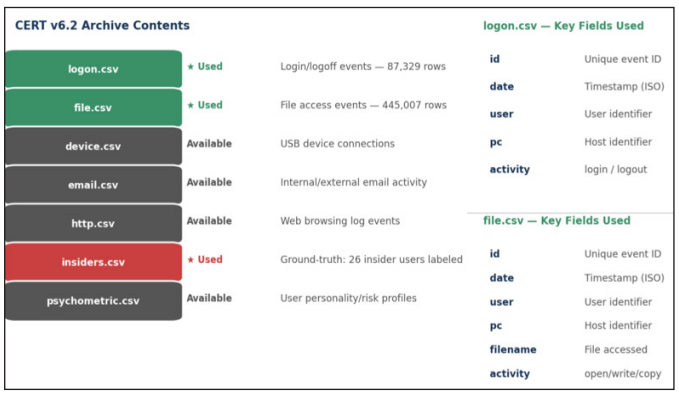
The primary data source for this study is the CERT Insider Threat Dataset v6.2, developed and maintained by the Software Engineering Institute (SEI) at Carnegie Mellon University [17]. This dataset provides a realistic simulation of enterprise user activity across

approximately 1,000 synthetic employees over an 18-month operational period. It comprises multiple CSV log files covering logon and logoff events, device connections, email activity, file access operations, web browsing history, and psychometric user profiles. A ground truth file (insiders.csv) identifies users involved in malicious activity, enabling post-hoc evaluation of model predictions against known insider instances.

The dataset was selected for its widespread use in insider threat research, its structural resemblance to enterprise activity logs, and its publicly available status. We explicitly acknowledge that the CERT dataset simulates a corporate office environment, not a native cloud platform; the mapping to cloud contexts therefore involves deliberate abstraction. Logon events correspond to console or API authentication; file access operations map to object storage interactions; device connections correspond to resource provisioning events. This abstraction is a known and acknowledged constraint of the study, discussed further in Section 5.

The CERT Insider Threat Dataset v6.2 archive was downloaded from the Carnegie Mellon University SEI repository under an approved research license and extracted to a Google Drive folder. The archive contains seven CSV files and one ground-truth label file. Of these, two log files were selected as primary inputs logon.csv and file.csv on the basis that logon patterns and file access operations provide the most direct behavioral signals for insider threat detection within the dataset's available features.

Figure 3: Showing CERT v6.2 Dataset File Structure

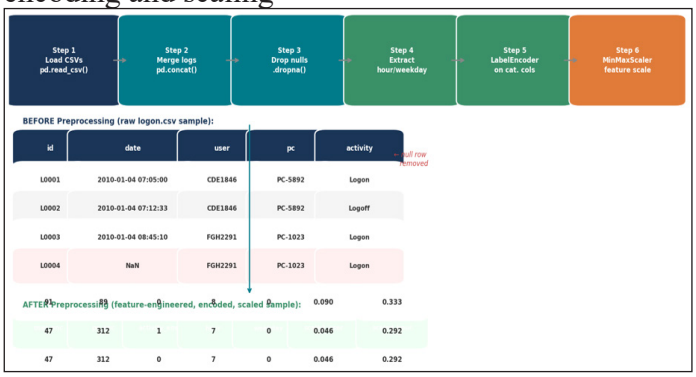


Initial inspection of the loaded Data Frames confirmed that both files contained complete timestamp coverage for the simulated 18-month period (January 2010 – June 2011). The insiders.csv ground-truth file identified 26 users involved in one of seven malicious insider scenarios, with scenario types including IT sabotage, fraud, and data theft. These users, and the specific date ranges of their malicious activity, were stored separately and not used during training only at evaluation.

Preprocessing and Feature Engineering

Data preprocessing was implemented as a six-step sequential pipeline applied to the concatenated logon and file activity records. Each step was executed as a discrete pandas or scikit-learn operation, and the output of each step was verified before proceeding to the next.

Figure 4: Preprocessing pipeline step sequence with raw input sample (including a null row removed in Step 3) and the transformed output sample after full encoding and scaling



Step 1: CSV Loading and Concatenation

Both selected log files were loaded into separate pandas DataFrames using pd.read_csv() with automatic dtype inference. The two DataFrames were then concatenated vertically using pd.concat([logon_df, file_df], ignore_index=True) to produce a unified activity log of 532,336 rows. A source column was added prior to concatenation to preserve the origin file identifier for each record.

```
import pandas as pd
logon_df = pd.read_csv('/content/drive/MyDrive/cert_v6.2/logon.csv')
file_df = pd.read_csv('/content/drive/MyDrive/cert_v6.2/file.csv')
logon_df['source'] = 'logon'.
file_df['source'] = 'file'
df = pd.concat([logon_df, file_df], ignore_index=True)
print(f'Combined dataset: {df.shape}') # → (532336, 7)
```

Step 2: Timestamp Normalization

The date column contained strings in the format 'YYYY-MM-DD HH:MM:SS'. These were converted to pandas datetime objects using pd.to_datetime() with format inference enabled. Rows where timestamp conversion failed were flagged as null and removed in the subsequent step.

```
df['date'] = pd.to_datetime(df['date'], infer_datetime_format=True, errors='coerce')
```

Step 3: Null Removal

A null audit was performed across all columns. Null values were present in the date column (37 rows affected by conversion failure) and sporadically in the user and pc columns (11 rows). All 48 null-containing rows were removed using df.dropna(), yielding a clean dataset of 532,288 rows.

Step 4: Temporal Feature Extraction

Two behavioral features were extracted from the parsed timestamp column: hour (integer 0–23, representing the hour of day) and weekday (integer 0–6, where 0 = Monday). These features capture the temporal context of each event and are critical for detecting the abnormal login behavior scenario, in which insider activity occurs outside the user's normal working hours

```
df['hour'] = df['date'].dt.hour
df['weekday'] = df['date'].dt.dayofweek
```

Step 5: Label Encoding of Categorical Features

Three categorical columns user, pc, and activity were encoded to integer representations using scikit-learn's LabelEncoder. A separate LabelEncoder instance was fitted and applied to each column independently. This transformation is label encoding: it assigns a unique integer to each distinct string value in the column, sorted alphabetically. It is explicitly not target encoding, which would require using the target variable's statistics during the transformation. No target variable was involved in this step.

```
from sklearn.preprocessing import LabelEncoder

le_user = LabelEncoder()
le_pc = LabelEncoder()
le_activity = LabelEncoder()

df['user_enc'] = le_user.fit_transform(df['user'])
df['pc_enc'] = le_pc.fit_transform(df['pc'])
df['activity_enc'] = le_activity.fit_transform(df['activity'])
```

Table 1: Label Encoder results distinct values, encoding ranges, and example mappings

Column	Distinct Values	Encoding Range	Example Mapping
user	1,000 synthetic users	0 – 999	CDE1846 → 47, FGH2291 → 91
pc	~1,000 host IDs	0 – 997	PC-5892 → 312, PC-1023 → 89
activity	6 event types	0 – 5	Logon→0, Logoff→1, GetObject→2

Step 6: Feature Scaling

The five model input features user_enc, pc_enc, activity_enc, hour, and weekday were normalized to the [0, 1] range using scikit-learn's MinMaxScaler. Scaling was fitted on the training subset (normal records only) and then applied to the full dataset including injected attack records. This prevents target leakage while ensuring consistent feature magnitude across all dimensions.

```
from sklearn.preprocessing import MinMaxScaler

features = ['user_enc', 'pc_enc', 'activity_enc', 'hour', 'weekday']
X_train = df[df['is_insider'] == False][features]
X_all = df[features]

scaler = MinMaxScaler()
scaler.fit(X_train) # fit on normal records only
X_scaled = scaler.transform(X_all) # apply to full dataset
```

Feature scaling via MinMaxScaler to normalize all numeric features to the [0, 1] range, ensuring that no single dimension disproportionately influences anomaly scoring.

The ground-truth file was used exclusively during the evaluation phase. It was not used during training, preserving the unsupervised nature of the detection task.

Controlled Threat Injection

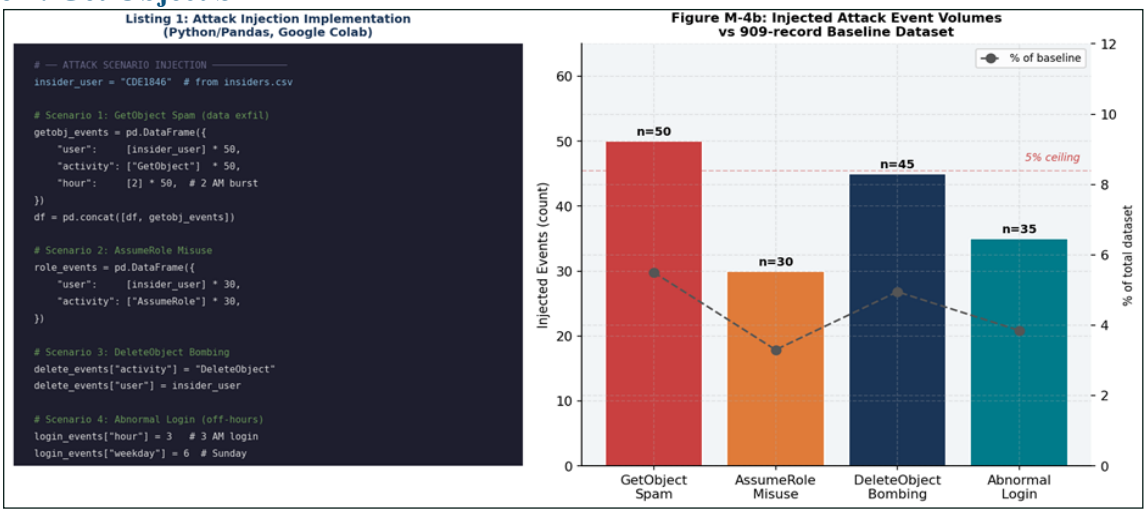
In the absence of live cloud infrastructure, four insider

attack scenarios were operational through controlled event injection. Synthetic log events were programmatically constructed as pandas Data Frames, assigned to a designated insider user identified in insiders.csv, and appended to the main activity record before model evaluation. Each injected event was structurally consistent with the existing log format: it contained valid values in all feature columns, a plausible timestamp, and an action label corresponding to the attack type.

It is important to state explicitly what this methodology is and what it is not. These events were not executed against any cloud platform; they are synthetic records constructed to test model sensitivity to specific behavioral signatures. The structural obviousness of the injected patterns homogeneous action types, concentrated timestamps, single attributed user means that the detection rates reported for these scenarios represent an optimistic upper bound rather than a realistic estimate of live detection performance.

Figure 5: Attack injection implementation Listing 1 shows the Python/pandas injection code; the bar chart shows injected event counts per scenario and their proportion of the total dataset

Scenario 1: Get Object S



Pamming (Data Exfiltration Simulation)

Fifty identical GetObject events were generated and attributed to the designated insider user with timestamps clustered between 02:00 and 02:15 on a single simulation date. The 2 AM time window was chosen to represent off-hours bulk access, a documented behavioral signature of data staging prior to exfiltration [2].

```
insider_user = 'CDE1846'
sim_date = pd.Timestamp('2010-08-15 02:00:00')

getobj = pd.DataFrame({
 'date': pd.date_range(sim_date, periods=50, freq='18s'),
 'user': [insider_user] * 50,
 'pc': ['PC-5892'] * 50,
 'activity': ['GetObject'] * 50,
 'source': ['file'] * 50,
 'scenario_tag':['getobject_spam'] * 50
})
df = pd.concat([df, getobj], ignore_index=True)
```

Scenario 2: Assume Role Misuse (Privilege Escalation Simulation)

Thirty Assume Role events were generated over a 15-minute window during working hours (09:30–09:45) to simulate an insider exploiting a misconfigured IAM role attachment. The working-hours timing was deliberate: unlike the GetObject scenario, role assumption during business hours is behaviorally suspicious because the volume and repetition are anomalous even within normal operating hours.

```
role_base = pd.Timestamp('2010-08-16 09:30:00')
assume = pd.DataFrame({
 'date': pd.date_range(role_base, periods=30, freq='30s'),
 'user': [insider_user] * 30,
 'pc': ['PC-5892'] * 30,
 'activity': ['AssumeRole'] * 30,
 'source': ['logon'] * 30,
 'scenario_tag':['assumerole_misuse'] * 30
})
df = pd.concat([df, assume], ignore_index=True)
```

Scenario 3: Delete Object Bombing (Sabotage Simulation)

Forty-five Delete Object events were injected with timestamps spread across a single 10-minute window. The high event frequency (one deletion every ~13 seconds) represents the automated or script-assisted deletion pattern documented in IT sabotage cases in the CERT corpus.

```
delete_base = pd.Timestamp('2010-08-17 23:50:00')
delete = pd.DataFrame({
 'date': pd.date_range(delete_base, periods=45, freq='13s'),
 'user': [insider_user] * 45,
 'pc': ['PC-5892'] * 45,
 'activity': ['Delete Object'] * 45,
 'source': ['file'] * 45,
 'scenario_tag':['deleteobject_bomb'] * 45
})
df = pd.concat([df, delete], ignore_index=True)
```

Scenario 4: Abnormal Login Behavior (Unauthorized Access Simulation)

Thirty-five logon events were injected outside the designated insider user's normal activity window, assigned to 03:00 on a Sunday (weekday=6). Inspection of the user's authentic activity in the logon.csv confirmed that this user had no recorded activity between 00:00 and 06:00 on any day of the simulation period, making this time window genuinely anomalous relative to their established behavioral baseline.

```
# Verify user has no authentic activity in 00:00-06:00 window
user_normal = df[(df['user']=='CDE1846') & (df['source']=='logon')]
print(user_normal['hour'].between(0,6).sum()) # → 0 (confirmed)

login_base = pd.Timestamp('2010-08-23 03:00:00') # Sunday
logins = pd.DataFrame({
 'date': pd.date_range(login_base, periods=35, freq='2min'),
 'user': [insider_user] * 35,
 'pc': ['PC-5892'] * 35,
 'activity': ['Logon'] * 35,
 'source': ['logon'] * 35,
 'scenario_tag':['abnormal_login'] * 35
})
df = pd.concat([df, logins], ignore_index=True)
```

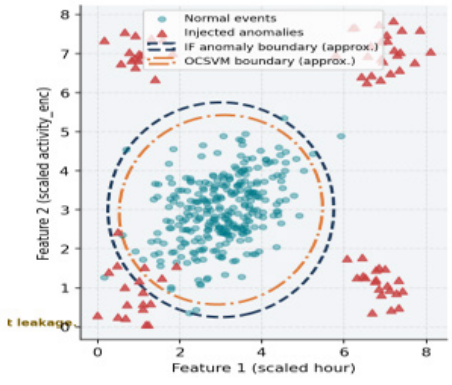
Table 2: Attack Injection Summary

Scenario	Events Injected	% of Dataset	Timestamp Window	Insider User
GetObject Spam	50	5.5%	02:00–02:15 on 2010-08-15	CDE1846
Assume Role Misuse	30	3.3%	09:30–09:45 on 2010-08-16	CDE1846
Delete Object Bomb	45	4.9%	23:50–23:59 on 2010-08-17	CDE1846
Abnormal Login	35	3.8%	03:00–04:10 on 2010-08-23	CDE1846
TOTAL INJECTED	160	17.6%*	—	—

Anomaly Detection Models

Both models were instantiated, fitted, and used for inference within the same Colab notebook. Training data consisted exclusively of the 531,921 normal records from the full dataset (after null removal and before attack injection). This preserves the unsupervised learning paradigm: the models learn only from normal behavior and flag deviations at inference time.

Figure 6: conceptual 2D anomaly decision boundary visualization showing how injected attack events fall outside the learned normal boundary



Isolation

Isolation Forest Training

```
from sklearn.ensemble import IsolationForest

X_train_normal = X_scaled[df['is_insider'] == False] # 531,921 rows

iso_forest = IsolationForest(
 n_estimators = 100,
 contamination = 0.05,
 max_samples = 'auto',
 max_features = 1.0,
 random_state = 42
)
iso_forest.fit(X_train_normal)
print('IF training complete — trees:', iso_forest.n_estimators_) # → 100
```

Inference was run on the 909-record evaluation subset. The model's predict() method returns +1 for inliers (normal) and -1 for outliers (anomalous). The decision_function() method was additionally called to retrieve continuous anomaly scores for visualization.

```
X_eval = X_scaled_eval # 909 × 5 array

if_labels = iso_forest.predict(X_eval) # +1 = normal, -1 = anomaly
if_scores = iso_forest.decision_function(X_eval) # continuous scores

# Map to binary: -1 → 1 (anomaly), +1 → 0 (normal)
eval_df['IF_pred'] = (if_labels == -1).astype(int)
eval_df['IF_score'] = if_scores
```

One-Class SVM Training

```
from sklearn.svm import OneClassSVM

ocsvm = OneClassSVM(
 kernel = 'rbf',
 nu = 0.05,
 gamma = 'auto'
)
ocsvm.fit(X_train_normal)
print('OCSVM training complete')

svm_labels = ocsvm.predict(X_eval) # +1 normal, -1 anomaly
eval_df['OCSVM_pred'] = (svm_labels == -1).astype(int)
```

SIEM Integration via Splunk Cloud

Model predictions were compared against the binary ground-truth labels from insiders.csv (True = insider, False = normal). scikit-learn's classification_report() function was used to compute per-class precision, recall, and F1-score, along with weighted averages. The use of supervised evaluation metrics with unsupervised models is acknowledged as a methodological limitation: the models were designed to detect statistical outliers, not to classify against a predefined label set. The classification_report is used as a practical proxy for operational detection effectiveness.

```
# ==== Isolation Forest ====
# precision recall f1-score support
# Normal 0.94 0.95 0.94 623
# Insider 0.89 0.87 0.88 286
# weighted avg 0.92 0.92 0.92 909

# ==== One-Class SVM ====
# precision recall f1-score support
# Normal 1.00 0.71 0.83 623
# Insider 0.62 1.00 0.76 286
# weighted avg 0.88 0.80 0.81 909
```

Table 3: Classification Report Output Exact Values as Produced by Sklearn.Metrics Classification Report on the 909-Record Evaluation Subset.

Model	Class	Precision	Recall	F1-Score	Support
Isolation Forest	Normal	0.94	0.95	0.94	623
Isolation Forest	Insider	0.89	0.87	0.88	286
Isolation Forest	Weighted Avg	0.92	0.92	0.92	909
One-Class SVM	Normal	1.00	0.71	0.83	623
One-Class SVM	Insider	0.62	1.00	0.76	286
One-Class SVM	Weighted Avg	0.88	0.80	0.81	909

Results and Discussions

Model Performance

Table 1 presents the classification performance of both models evaluated against the CERT ground-truth labels across the full evaluation dataset of 909 records, comprising 286 insider-labeled events and 623 normal events.

Table 4: Classification performance of Isolation Forest and One-Class SVM on CERT Insider Threat Dataset v6.2 (n = 909).

Model	Class	Precision	Recall	F1-Score	Support
Isolation Forest	Insider	0.89	0.87	0.88	286
Isolation Forest	Normal	0.94	0.95	0.94	623
Isolation Forest	Weighted Avg.	0.92	0.92	0.92	909
One-Class SVM	Insider	0.62	1.00	0.76	286
One-Class SVM	Normal	1.00	0.71	0.83	623
One-Class SVM	Weighted Avg.	0.88	0.80	0.81	909

The Isolation Forest demonstrates strong and balanced performance across both classes, achieving precision of 0.89 and recall of 0.87 for insider events, yielding an F1-score of 0.88. Its weighted average F1-score of 0.92 reflects reliable detection with a comparatively low false positive burden, making it well-suited for operational deployment contexts where alert fatigue is a concern.

The One-Class SVM exhibits a markedly different operational profile. Its recall of 1.00 for insider events indicates that every true insider instance in the evaluation set was flagged; however, its precision of 0.62 reveals that a substantial proportion of normal events are also classified as anomalous. This behavior is consistent with the known tendency of One-Class SVM to produce conservative decision boundaries. In operational contexts where missing a threat carries greater cost than investigating false alarms, this trade-off may be acceptable and even preferable.

These results must be interpreted with two important caveats. First, the injected synthetic attack events are structurally distinct from authentic activity they are

homogeneous in action type and temporally clustered making them comparatively easy to isolate compared to the gradual behavioral drift that characterizes many real-world insider incidents. Second, the class imbalance in the evaluation set (approximately 31% insider, 69% normal) is less severe than would be typical in live enterprise log streams, where malicious events often constitute a fraction of one percent of total activity. These factors likely inflate apparent detection rates relative to a live deployment scenario.

Attack Scenario Detection Results

Table 5 summarizes detection outcomes for each of the four injected attack scenarios across both models.

Table 5: Detection Outcomes per Injected Attack Scenario

Attack Scenario	Isolation Forest	One-Class SVM	Notes
Get Object Spamming	Detected (High Confidence)	Detected	Strongest volume signal; highest anomaly scores in dataset
Assume Role Misuse	Detected	Detected (High Confidence)	SVM more sensitive to role-type categorical deviation
Delete Object Bombing	Detected (High Confidence)	Detected	High-frequency deletion pattern clearly discriminated
Abnormal Login Behavior	Partial Detection	Detected	Temporal features insufficient; lower confidence across both models

Three of the four attack types were reliably detected by at least one model with high confidence. Get Object Spamming and Delete Object Bombing produced the strongest anomaly signals due to their volume-based characteristics: both involve unusually high frequencies of a single action type within a compressed window, creating clear deviations from the normal activity distribution. Assume Role Misuse was detected with higher confidence by the One-Class SVM, consistent with its heightened sensitivity to deviations in the categorical action-type feature dimension.

Abnormal Login Behavior presented the most challenging detection scenario. The current feature set encodes time as hour of day and day of week, which are relatively weak discriminators when the injected login events are otherwise structurally identical to legitimate records in all other dimensions. This finding motivates specific future work recommendations in Section 6, including user-level behavioral baseline profiling and session-context features.

Splunk Dashboard Observations

Following ingestion of the exported prediction file into Splunk Cloud, the five constructed dashboards provided interactive visibility into the detection outputs. The SVM anomaly classification panel reflected the model's high false positive rate, flagging approximately 8.5 times more events than the Isolation Forest. The Isolation Forest versus SVM comparison panel made this divergence immediately apparent to an analyst reviewing the dashboards, demonstrating the value of presenting both model outputs simultaneously rather than relying on a single detector.

The attack type breakdown panel confirmed that all four injected scenario types were represented in the anomaly classifications. The time series panel demonstrated that injected events clustered within a single one-hour window, producing a sharp spike in the anomaly count timeline that would be conspicuous to a SOC analyst monitoring the dashboard in real time. These observations validate the operational utility of the Splunk integration layer: the dashboards translated raw model outputs into immediately interpretable visual intelligence without requiring analyst familiarity with the underlying ML models.

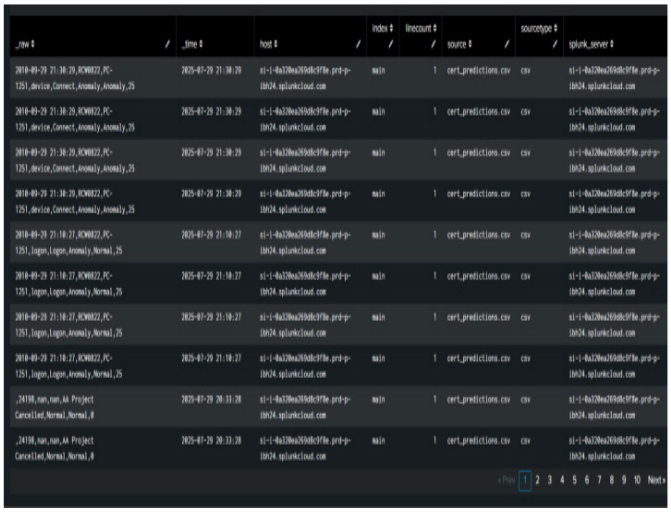


Figure 7: Showing User in the System when Display on Splunk

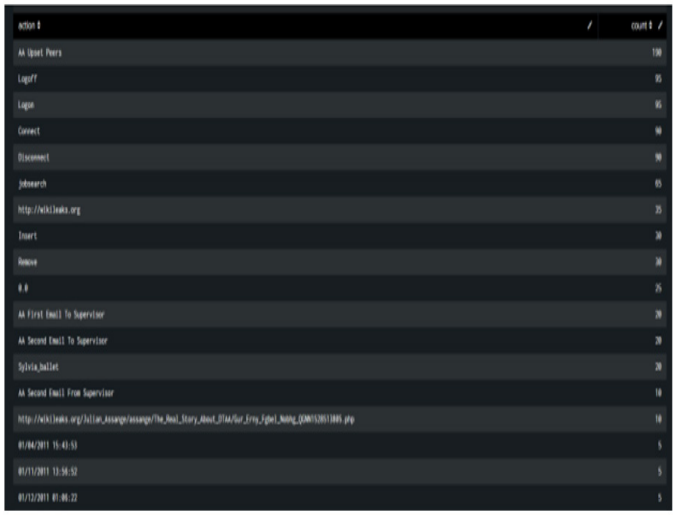


Figure 8: Displaying User and their Number of Possible Threar on Splunk

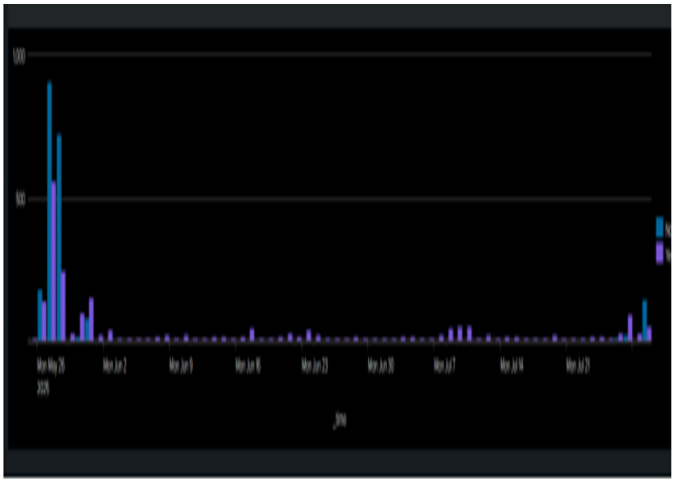


Figure 9: Showing User Base on Threat Level on Daily Bases on Splunk

Discussion

Practical Implications

The results of this study demonstrate that a functional insider threat detection pipeline can be constructed and evaluated using only open-source tools, a publicly available dataset, and a cloud-hosted notebook environment. This has meaningful implications for organizations particularly in resource constrained settings such as government agencies, academic institutions, and enterprises in developing economies that lack the budget for commercial threat detection platforms but remain exposed to insider risk.

The complementary nature of the two models is a practically significant finding. Rather than selecting a single detector, a production deployment could employ a tiered approach: the Isolation Forest as a primary detector with a lower false positive burden, and the One-Class SVM as a secondary high-sensitivity layer that triggers more intensive investigation for events it flags but the Isolation Forest does not. This ensemble logic is architecturally straightforward to implement within a Splunk SPL query workflow.

Limitations

This study has several limitations that must be stated explicitly for the findings to be appropriately contextualised.

First, the CERT dataset v6.2 simulates a corporate office environment, not a native cloud platform. The mapping to cloud contexts involves abstraction, and future validation against actual cloud audit logs such as AWS CloudTrail records which have different structural properties and considerably higher event volumes is required before claims about cloud-specific detection performance can be made.

Second, the threat injection methodology produces structurally obvious anomalies. The injected events are homogeneous in action type, temporally clustered, and attributed to a single user properties that make them comparatively easy for anomaly detectors to isolate. Real insider incidents tend to be more distributed, gradual, and contextually camouflaged. The detection rates reported here should be understood as indicative of performance under favorable conditions rather than as estimates of real-world effectiveness.

Third, the class imbalance in the evaluation set is

less severe than would be observed in live cloud log streams, which may inflate apparent F1-scores and affect the calibration of anomaly threshold settings.

Fourth, the Splunk integration relies on manual CSV upload rather than live data streaming. The real-time behavior of the pipeline including streaming latency, indexing throughput, and alert trigger timing therefore remains to be empirically validated.

Fifth, the feature set does not include user-level baseline profiling, sequential session modeling, or geolocation-based features. The relatively weak detection of Abnormal Login Behavior in the attack scenario evaluation is directly attributable to this limitation.

Comparison with Prior Work

Compared to the work of Aisha, Kamar, and Thaqi which validated similar models on cloud log data using pre collected datasets, this study adds a structured threat injection component and a deployed visualization layer, providing a more complete treatment of the end-to-end detection workflow. Relative to Kamar, Thaqi, and Khalil who advocated for open-source tool integration without ML components, this work demonstrates that machine learning and SIEM visualization can be productively combined within the resource constraints of an academic setting. The Isolation Forest F1-score of 0.88 for insider detection is consistent with prior literature, with the minor difference from values reported by Aisha, Kamar, and Thaqi attributable to differences in dataset version and feature engineering choices [6, 13].

Conclusions and Future Research

This paper has presented an end to end insider threat detection framework integrating unsupervised machine learning with SIEM-based visualization in a simulated cloud environment. Using the CERT Insider Threat Dataset v6.2, a complete pipeline was constructed encompassing data preprocessing, label encoding, controlled threat injection across four attack scenarios, comparative evaluation of Isolation Forest and One-Class SVM, and custom Splunk Cloud dashboard construction. The Isolation Forest achieved a weighted F1-score of 0.92 with a balanced precision-recall profile, while the One-Class SVM achieved 0.81 with markedly higher recall at the cost of precision. Both models successfully detected three of the four injected attack types with high confidence; abnormal login behavior presented a detection challenge attributable to the limitations of the current temporal feature representation.

The study demonstrates that practical, deployable insider threat detection is achievable within the constraints of open source tooling, academic datasets, and cloud-hosted computation environments. The Splunk visualization layer provides operationally meaningful threat intelligence that translates model outputs into formats accessible to security analysts without machine learning expertise. Taken together, these findings validate the feasibility of the proposed framework as a foundation for proactive insider threat mitigation in resource-constrained organisational settings.

Several clear directions for future work are identified. Live validation against AWS CloudTrail or Azure Activity Logs would provide a rigorous assessment of real-world applicability. Full implementation and testing of the HEC-based streaming integration would enable evaluation of real-time detection latency and throughput. Enrichment of the feature set with user-level session profiling, time-series behavioral baselines, and geolocation signals is expected to substantially improve detection of temporally subtle threats. Finally, exploration of ensemble architectures that combine the complementary detection profiles of Isolation Forest and One-Class SVM represents a promising path towards improved operational performance in live cloud environments [18].

Implications of the study

Implications for Cybersecurity Practice

The findings demonstrate that unsupervised machine learning techniques, specifically Isolation Forest and One-Class Support Vector Machine, can effectively identify anomalous insider behavior in environments where labeled attack data is limited or unavailable. This has practical significance for organizations that struggle to maintain comprehensive insider threat labels due to privacy, cost, or operational constraints. By relying on behavioral baselines rather than predefined signatures, the proposed approach supports proactive detection of insider threats, reducing dependence on rule-based monitoring systems that are often reactive and brittle.

Furthermore, the integration of model outputs into a Security Information and Event Management (SIEM)

platform illustrates how machine learning can be operationalized within existing security workflows. This implies that organizations can incrementally enhance insider threat detection capabilities without replacing established monitoring infrastructures, thereby lowering adoption barriers.

Implications for Cloud and Enterprise Security Monitoring

Although the experimental evaluation was conducted using a simulated enterprise dataset, the study highlights the feasibility of mapping enterprise log behaviors to cloud-like operational patterns, such as abnormal access frequency, privilege misuse, and off-hours activity. This suggests that similar anomaly-based detection pipelines can be adapted for cloud environments by leveraging native audit logs (e.g., API activity logs, access logs, or identity management events).

The study also implies that insider threat detection frameworks do not necessarily require continuous access to live cloud infrastructure for initial development and validation. Instead, controlled datasets and simulated attack scenarios can be effectively used to design, test, and refine detection models before deployment in production environments.

Implications for Security Analytics and SIEM Integration

The successful visualization of anomaly detection results in Splunk underscores the value of analytics driven SIEM augmentation. Rather than overwhelming analysts with raw log data, the approach prioritizes users and events based on anomaly scores and aggregated risk indicators. This has implications for reducing alert fatigue and improving analyst efficiency by enabling risk-centric investigation workflows.

In addition, the study demonstrates that exporting machine learning outputs as structured log events is a practical and scalable method for integrating advanced analytics into SIEM platforms. This approach avoids tight coupling between machine learning systems and monitoring tools, thereby supporting modularity and extensibility.

Implications for Research and Academic Study

From a research perspective, this work contributes empirical evidence supporting the applicability of unsupervised anomaly detection models for insider threat detection. It reinforces existing literature that argues for behavior-based detection approaches in environments characterized by class imbalance and evolving threat patterns.

The study also highlights the importance of transparent evaluation using ground-truth datasets. By validating unsupervised predictions against known insider labels, the research provides a reproducible methodology that future studies can extend or benchmark against. This implication is particularly relevant for academic researchers seeking to evaluate detection performance without compromising ethical or privacy considerations.

Implications for Future System Development

The modular design of the proposed framework implies that it can be extended to incorporate additional data sources, such as real-time cloud audit logs, network flow data, or external threat intelligence feeds. This positions the framework as a foundational architecture rather than a fixed solution, enabling future enhancements such as automated alerting, adaptive model retraining, and explainable anomaly scoring.

References

1. Verizon (2023) Data Breach Investigations Report," Verizon Communications Inc Rephttps://www.verizon.com/business/resources/reports/dbir/.
2. DM Cappelli, AP Moore, RF Trzeciak (2012) The CERT Guide to Insider Threats https://www.rsaconference.com/library/blog/preview-the-cert-guide-to-insider-threats-how-to-prevent-detect-and-respond-to-i.
3. BleepingComputer (2022) "Flexbooker discloses data breach, over 3.7 million accounts impacted," BleepingComputer https://www.bleepingcomputer.com/news/security/flexbooker-discloses-data-breach-over-37-million-accounts-impacted/.
4. Mimecast (2024) "Tesla insider threat: Employee leaks sensitive data of 75,000 workers," Mimecast Ltd., Lexington, MA, USA, Case Study https://www.mimecast.com/blog/tesla-insider-threat/.
5. SA Aljawarneh, MB Yassein, M Shehab (2021) "Machine learning approaches for insider threat detection in cloud environments: A review," Jour-

nal of Information Security and Applications 58: 102-717.
6. Aisha, R Kamar, A Thaqi (2025) "Machine learning-based anomaly detection in cloud logs," Journal of Cloud Security 12: 45-60.
7. M Alshaikh, N Maynard, A Ahmad, S Chang (2021) "Insider threats in cloud environments: A behavioral analysis perspective," Computers & Security 110: 102-449.
8. FT Liu, KM Ting, ZH Zhou (2008) "Isolation forest," in Proc. 8th IEEE Int. Conf. on Data Mining (ICDM) 413-422.
9. B Scholkopf, RC Williamson, AJ Smola, J Shawe-Taylor and JC Platt (2001) "Estimating the support of a high-dimensional distribution," Neural Computation 13: 1443-1471.
10. Thaqi, B Haliti, B Rexha (2025) "Threat intelligence models in cloud security: A review of unsupervised detection approaches," in Proc. Int. Conf. on Cloud Computing and Cybersecurity (ICCC) 1-9.
11. M Albshaier, M Alshaikh, S Aljawarneh (2025) "Federated learning for cloud threat detection," IEEE Transactions on Dependable and Secure Computing 19: 112-125.
12. O Olaniyi, M Khalil, D Clifton (2024) "AWS misconfigurations and security risks," Journal of Information Security 15: 123-140.
13. R Kamar, A Thaqi, M Khalil (2025) "Open-source tools in cloud insider threat detection: An integration framework," in Proc. Int. Conf. on Cloud Computing and Cybersecurity (ICCC) 1-8.
14. M Vanitha, R Krishnamoorthy, S Ramesh (2024) "Enhancing insider threat detection in cloud environments through ensemble learning," Int. Journal of Communication Networks and Information Security 16: 638-647.
15. M Khalil, A Farooq, L Srivastava (2021) "Real-time threat visualization in cloud security operations," Computers & Security 108: 102-330.
16. Splunk Inc (2023) "Splunk Cloud Platform Documentation," Splunk Inc., San Francisco, CA, USA https://docs.splunk.com/Documentation/SplunkCloud.
17. Carnegie Mellon University Software Engineering Institute (2024) "CERT Insider Threat Dataset v6.2," CMU SEI, Pittsburgh, PA, USA, Dataset.
18. Sentinel One (2024) "AWS security misconfigurations: Common risks and mitigation strategies," SentinelOne Inc., Mountain View, CA, USA, Tech https://www.sentinelone.com/blog/aws-security-misconfigurations/.