Journal of Contemporary Science and Technology Management

Volume 01, Issue 01, December 2025

SENTINEL-HCM: Federated Temporal Graph Intelligence for Detecting API Abuse, Data Leakage, and Policy Drift in SAP SuccessFactors Cloud Integrations

Manoj Parasa
Independent Researcher, Dallas, TX, USA
manoj.parasa1993@gmail.com

Abstract

Modern HCM integrations expose a difficult security problem: the riskiest events rarely appear as isolated violations, but as small changes in API sequences, token behavior, role permissions, data movement, and policy execution over time. In SAP SuccessFactors cloud environments, these signals are often distributed across integration middleware, identity services, workflow engines, audit logs, and external enterprise systems, making conventional threshold alerts inadequate for detecting slow-moving data exposure and governance drift. SENTINEL-HCM addresses this gap through a federated temporal graph intelligence framework that converts cloud integration activity into an evolving security graph, where users, service accounts, OAuth tokens, API endpoints, integration jobs, HR data objects, external systems, workflow rules, and access policies are modeled as connected entities. The framework combines temporal graph learning, federated model aggregation, sensitive-object lineage scoring, and policy drift measurement to identify API abuse, token misuse, privileged access drift, data leakage, and abnormal integration behavior without centralizing raw employee data. Its learning design uses local graph encoders to capture tenant-specific event patterns, a federated aggregation layer to preserve privacy across distributed integration nodes, and a risk interpretation layer that traces high-risk predictions back to the API path, token scope, accessed object, permission change, or policy deviation responsible for the alert. The evaluation compares SENTINEL-HCM with rule-based monitoring, classical anomaly detection, tree-based classifiers, sequence models, static graph learning, and non-federated temporal graph baselines, where the proposed framework achieved an F1-score of 0.937, ROC-AUC of 0.978, PR-AUC of 0.961, and a false positive rate of 2.60%. By treating integration security as a time-evolving relationship problem rather than a flat log-classification task, the proposed framework provides a practical foundation for privacy-preserving HCM security monitoring, explainable audit evidence, and proactive governance of SAP SuccessFactors cloud data exchange.

Keywords: SAP SuccessFactors, cloud HCM security, federated learning, temporal graph intelligence, API abuse detection, data leakage detection, token misuse, policy drift, privileged access drift, zero-trust integration, OAuth security, graph neural networks, anomaly detection, HR data governance, privacy-preserving analytics.

DOI: 10.66592/jcstm.01.01.01

1. Introduction

Cloud-based human capital management platforms have become integration-intensive environments where employee profiles, identity attributes, organizational assignments, compensation fields, workflow decisions, learning records, recruiting data, and analytics extracts move continuously across internal and external systems. In SAP SuccessFactors landscapes, these exchanges are commonly supported through API calls, middleware flows, Integration Center jobs, SAP BTP or CPI services, OAuth tokens, role-based permissions, scheduled replications, and downstream enterprise applications. This connectivity improves operational efficiency, but it also expands the security surface because a single integration path may involve sensitive employee objects, identity credentials, workflow controls, regional data rules, and third-party destinations within the same automated sequence. As cloud services increasingly depend on distributed APIs and identity-based access, security monitoring must evaluate not only whether access is technically permitted, but whether it remains appropriate within the surrounding business, policy, and temporal context [1].

The central challenge is that high-risk integration behavior often does not appear as an obvious violation. API abuse may look like increased activity from an approved service account. Token misuse may occur through a valid OAuth credential without

generating an authentication failure. Data leakage may begin as a permitted export that gradually includes more sensitive object classes, larger record volumes, or unfamiliar destinations. Privileged access drift may emerge through temporary permissions, expanded service account rights, workflow exceptions, or configuration changes that appear reasonable in

isolation but weaken the control model over time. These risks are difficult to detect through static thresholds or manual audit reviews because the evidence is distributed across users, tokens, endpoints, HR data objects, policy rules, and integration sequences.

Existing monitoring approaches provide useful control coverage but remain limited for this class of HCM integration risk. Rule-based alerts are effective when the violation pattern is already known, such as an unauthorized endpoint, invalid credential, or explicitly blocked action. Statistical anomaly detection can identify extreme deviations in volume or frequency, but it often lacks the business context needed to separate legitimate workload spikes from suspicious behavior. Sequence models can learn event order, but they do not always capture the relationships among service accounts, token scopes, API endpoints, workflow rules, sensitive objects, and destination systems. These limitations are significant in cloud HCM environments, where risk may be produced by a connected path rather than by one abnormal event [2].

This paper proposes SENTINEL-HCM, a federated temporal graph intelligence framework for detecting API abuse, data leakage, token misuse, privileged access drift, and policy drift in SAP SuccessFactors cloud integrations. The framework represents users, service accounts, roles, OAuth tokens, API endpoints, integration jobs, middleware flows, HR data objects, external systems, workflow rules, and policy controls as entities in a time-evolving security graph. Interactions such as invokes, reads, writes, exports, grants, refreshes, approves, triggers, and violates are modeled as time-stamped edges. This structure enables the model to examine integration behavior as a connected sequence of actions rather than as independent log records.

The core assumption behind SENTINEL-HCM is that cloud HCM security is both relational and temporal. A high-volume API call is not always suspicious, a sensitive object access is not always improper, and a permission change is not always dangerous. Risk becomes clearer when these signals are evaluated together. For example, a service account that normally reads basic employee profile data may become risky when it begins using a broader token scope, calling compensation-related endpoints, exporting larger data volumes, and sending records to a destination outside the approved integration route. Graph-based learning is well suited to this setting because it can model multi-entity relationships, while temporal graph methods can preserve the order, recency, and evolving nature of those relationships [3].

The federated design addresses an additional constraint in enterprise HCM security: raw HR event logs are sensitive and may not be suitable for centralized model training. Employee-related logs can contain identifiers, role information, regional attributes, workflow activity, object references, and access patterns that are subject to internal privacy and governance rules. SENTINEL-HCM allows each integration node, business unit, or tenant partition to train locally on its own event graph while sharing only learned model updates for aggregation. This supports privacy-preserving security analytics without requiring unnecessary movement of raw employee-level logs across organizational or regional boundaries [4].

The evaluation of SENTINEL-HCM is designed as a complete security intelligence study rather than a narrow classifier comparison. The proposed model is compared against rule-based monitoring, classical anomaly detection, tree-based classifiers, sequence learning models, and non-federated graph learning baselines. Performance is assessed using precision, recall, F1-score, ROC-AUC, PR-AUC, false positive rate, detection latency, and throughput. The study also includes ablation testing to measure the contribution of graph structure, temporal windows, token behavior, sensitive-object lineage, policy drift features, and federated training. This design allows the framework to be evaluated for both detection quality and operational practicality.

The main contributions of this paper are fourfold. First, it introduces a temporal HCM security graph that represents SAP SuccessFactors cloud integration behavior through connected users, service accounts, roles, tokens, endpoints, middleware flows, HR objects, external systems, workflow rules, and policy controls. Second, it develops a federated learning design that

Journal of Contemporary Science and Technology Management
Volume 01, Issue 01, December 2025

supports distributed model training without centralizing raw HR event logs. Third, it defines risk scoring methods for API abuse, token misuse, data leakage, privileged access drift, and policy drift. Fourth, it evaluates the framework through model comparison, ablation analysis, latency testing, scalability analysis, and explainable risk-path interpretation.

The overall research scope is summarized in Figure. 1, showing how SAP SuccessFactors cloud integrations, API activity, OAuth tokens, middleware flows, HR data objects, external systems, workflow controls, and policy baselines form the security surface addressed by SENTINEL-HCM.

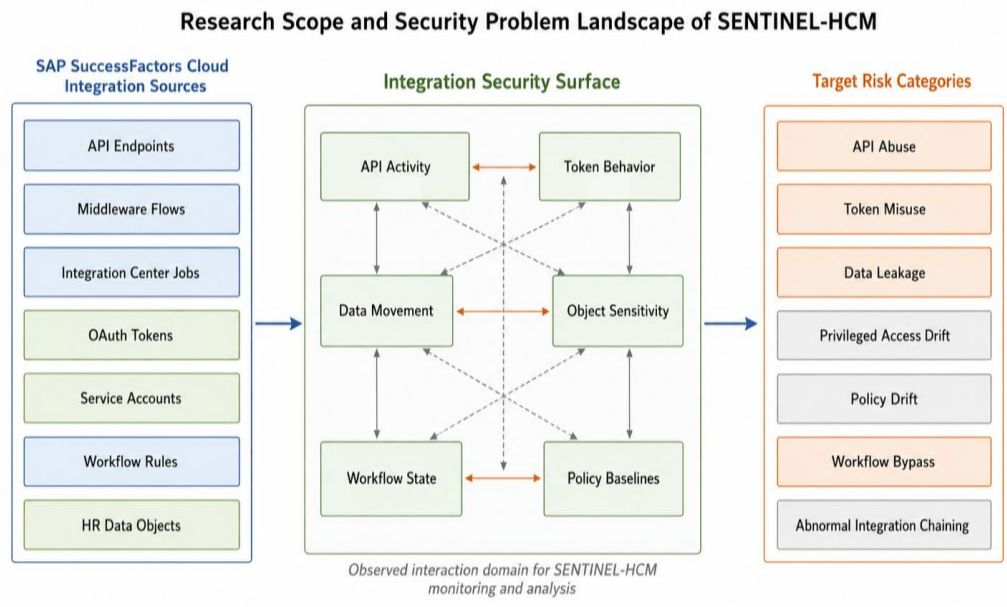

Figure 1: Research Scope and Security Problem Landscape of SENTINEL-HCM

2. Related Work

Security monitoring in cloud-based HCM integrations sits at the intersection of enterprise access control, API protection, anomaly detection, privacy-preserving learning, and audit governance. Unlike general IT security logs, HCM integration events carry organizational meaning because a single API transaction may involve an employee profile, compensation attribute, identity update, workflow decision, bank-related field, or downstream system dependency. This makes the problem different from ordinary network intrusion detection. The technical signal is not only whether an endpoint was called, but whether the endpoint, token scope, user role, object sensitivity, integration schedule, destination system, and policy state remain consistent with the expected business context. Existing research provides useful foundations, but the full HCM integration problem remains underdeveloped because most approaches study logs, access events, graph relationships, or privacy constraints separately rather than as one connected security intelligence problem.

2.1 Cloud HCM integration security and governance

Cloud HCM platforms depend on continuous data exchange between core HR, identity management, payroll interfaces, learning platforms, recruiting systems, analytics environments, document services, and external vendors. These exchanges are usually implemented through APIs, middleware jobs, scheduled replication, event-based triggers, or batch integration flows. From a governance perspective, this creates a layered control problem. Access must be valid at the identity level, correct at the role level, appropriate at the data-object level, consistent at the workflow level, and compliant at the integration

destination level. Traditional access-control models are useful for enforcing permissions, but they are less effective when the risk emerges from a sequence of individually valid actions. A service account may have approved access to an API, but the same access becomes suspicious if the call volume changes sharply, the object class becomes more sensitive, the token scope expands, or the data begins moving to an unusual downstream system.

Prior work on enterprise security governance has emphasized the need for stronger monitoring of privileged access, policy enforcement, and audit accountability. Zero-trust security principles also support the idea that trust should not be granted permanently based only on network location or initial authentication. In HCM integrations, this principle is especially relevant because service accounts, middleware credentials, and automated tokens often operate with broad access and limited day-to-day human visibility. However, zero-trust models usually describe security posture and control principles rather than providing a learning framework for detecting temporal drift in HCM integration behavior. This leaves a practical gap between governance expectations and the analytical methods needed to identify subtle changes in cloud HCM data exchange.

2.2 API anomaly detection and token misuse

API security research has commonly focused on abnormal request patterns, unauthorized endpoint access, suspicious traffic volume, malformed calls, credential misuse, and behavioral deviations. Classical anomaly detection methods such as statistical thresholds, clustering, Isolation Forest, and one-class classification are useful when abnormal behavior is visibly different from normal activity [5], [6]. Supervised classifiers such as Random Forest and gradient-boosting models can perform well when labeled examples of misuse are available. Deep sequence models can also learn ordering and timing patterns in system logs, which helps detect unusual event chains that simple thresholds may miss [7], [8].

Despite these strengths, API abuse in HCM integration settings has characteristics that make flat anomaly detection less reliable. First, normal integration jobs can legitimately produce high API volume during scheduled replication, release testing, onboarding waves, compensation cycles, or mass organizational changes. Second, token misuse may not generate obvious authentication failures because the token may still be technically valid. Third, data leakage may occur through approved endpoints if object selection, data volume, destination, or timing changes unexpectedly. Fourth, policy drift may unfold gradually through small permission changes rather than a sudden violation. These cases require a model that can distinguish legitimate operational peaks from risky behavioral shifts by examining relationships among tokens, endpoints, roles, objects, jobs, and policies over time.

2.3 Log-based security analytics and sequence learning

System-log analysis has become a major direction in security monitoring because logs preserve the sequence of operational events leading to failures, violations, or attacks. Log-based models such as DeepLog showed that event sequence learning can detect abnormal execution patterns by modeling expected event transitions. Later approaches extended this idea through recurrent networks, attention mechanisms, and representation learning to improve detection under complex event streams. These methods are valuable for cloud integrations because API calls, token refreshes, workflow actions, and middleware executions naturally form time-ordered sequences.

However, sequence learning alone does not fully represent the structure of HCM integration risk. Two API sequences may appear similar while involving different employee objects, token scopes, permission roles, or downstream destinations. A model that sees only event order may miss the fact that one sequence touches general profile data while another reaches sensitive compensation, identity, or payment-related fields. Sequence models also struggle to express multi-hop relationships, such as a service account using a broadened token to call an endpoint that accesses a sensitive object and exports data to a system not normally connected to that process. This limitation motivates the use of graph-based representation, where the entities and relationships behind each event can be modeled directly.

2.4 Graph learning for security relationships

Graph learning has become an important method for modeling relational security problems because attacks often involve connected entities rather than isolated records. In a graph, users, devices, roles, tokens, endpoints, applications, files, and policies can be represented as nodes, while interactions such as access, transfer, assignment, invocation, or violation can be represented as edges. GraphSAGE introduced inductive representation learning for graphs by allowing node embeddings to be generated from local neighborhoods, which is useful when new users, services, or endpoints appear over time [9]. Graph attention networks further improved relational learning by assigning different importance levels to neighboring nodes during message passing [10]. These methods provide a strong foundation for detecting risky relationships that are not visible in tabular logs.

For HCM integration security, graph learning is attractive because risk often appears as a path. A high-risk event may involve a service account, token, endpoint, HR object, export destination, and policy exception. The risk is not fully contained in any one of these entities. It is produced by their connection. A graph model can capture this by learning whether the path itself is unusual, whether the involved entities normally interact, and whether the current path resembles previous risky behavior. Still, static graph models are limited when integration behavior changes over time. A connection that is normal during one cycle may be abnormal during another. A token scope may be acceptable for one integration period but risky after a policy change. This makes temporal graph learning more suitable than static graph modeling.

2.5 Temporal graph learning for evolving integration behavior

Temporal graph learning extends graph representation by preserving when relationships occur, how frequently they repeat, and how their meaning changes across time windows. Models such as EvolveGCN and Temporal Graph Networks have shown that graph structures can be learned as dynamic systems rather than fixed snapshots [11], [12]. This is highly relevant to cloud HCM integrations because risk often depends on time-sensitive context. Scheduled integrations, periodic HR cycles, regional processing windows, release testing, and workflow approvals all produce legitimate temporal patterns. A security model should therefore learn what is normal for a specific entity, object, endpoint, and time period instead of applying the same threshold to every event.

Temporal graph learning also supports the detection of slow-moving risk. Policy drift, for example, may not appear as a single abnormal event. It may begin with a small role update, followed by a token-scope change, followed by increased endpoint usage, followed by access to a broader object set. Each event may appear acceptable in isolation, but the evolving path reveals a governance problem. Temporal graph representation is therefore well aligned with the security reality of SAP SuccessFactors integrations, where users, service accounts, middleware flows, permissions, and data objects interact continuously and change gradually.

2.6 Federated learning for privacy-preserving security analytics

Centralized model training is often difficult in HCM security because raw event logs may contain employee identifiers, role information, regional attributes, workflow records, and sensitive object references. Moving all such logs into a central analytics environment can create privacy, data minimization, and internal governance concerns. Federated learning offers an alternative by allowing local environments to train models on their own data while sharing only model updates for aggregation [13]. Secure aggregation and privacy-preserving protocols further reduce the risk of exposing local training information during collaboration [14]. Broader studies on federated learning have shown its usefulness in settings where data is distributed, sensitive, and difficult to centralize [15].

In the context of SAP SuccessFactors cloud integrations, federated learning is useful because different business units, countries, tenants, or integration nodes may have different data restrictions and operational patterns. One region may process more identity events, another may process more onboarding activity, and another may run frequent compensation or analytics exports. A centralized model may not capture this local variation well, while a purely local model may not learn enough rare

Journal of Contemporary Science and Technology Management
Volume 01, Issue 01, December 2025

security patterns. Federated temporal graph learning creates a middle path. Each node can learn from local HCM integration behavior, while the global model can still benefit from shared patterns of API abuse, token misuse, leakage behavior, and policy drift.

2.7 Policy drift, explainability, and audit readiness

Security monitoring in enterprise HCM cannot stop at prediction. A high-risk alert must be explainable enough for security, HR technology, compliance, and audit teams to review. This is especially important when the alert involves a legitimate business process, such as employee replication, workflow approval, vendor integration, or scheduled reporting. A model that simply labels an event as anomalous is less useful than one that identifies the risky path, the token involved, the sensitive object accessed, the endpoint sequence, the policy baseline deviation, and the business process affected.

Policy drift is also different from conventional anomaly detection. An anomaly is usually an unusual event compared with observed behavior. Policy drift is a deviation from an approved control state. A behavior may appear statistically common but still be noncompliant if the approved baseline has changed. Conversely, a rare event may be acceptable if it is authorized by a temporary workflow or controlled exception. Therefore, a practical HCM security framework needs both behavioral learning and policy-baseline comparison. This distinction is important because enterprise audits are concerned not only with what is unusual, but also with what is unauthorized, excessive, unexplained, or inconsistent with approved governance rules.

2.8 Research gap and position of SENTINEL-HCM

The reviewed literature provides strong foundations for anomaly detection, log sequence modeling, graph neural networks, temporal graph learning, and federated learning. However, these streams do not fully address the combined problem of SAP SuccessFactors cloud integration security. Existing methods often treat security events as flat records, independent logs, static relationships, or centralized datasets. They rarely model HCM-specific integration behavior as a time-evolving graph that connects service accounts, OAuth tokens, API endpoints, integration jobs, HR data objects, external systems, workflow rules, role permissions, and policy baselines. They also rarely combine detection of API abuse, data leakage, token misuse, privileged access drift, and policy drift within the same privacy-preserving framework.

SENTINEL-HCM is positioned to fill this gap. It treats HCM integration security as a temporal relationship problem, not merely as a log classification problem. It uses graph intelligence to connect the entities involved in each event, temporal modeling to capture behavior changes over time, federated learning to avoid unnecessary centralization of raw HR logs, and risk scoring to support audit interpretation. This combination is designed for the specific realities of cloud HCM integration landscapes, where sensitive employee data, automated middleware, role-based access, tokenized authentication, policy controls, and external system dependencies must be monitored together.

Table 1. Comparison of existing security monitoring approaches and SENTINEL-HCM

Approach	**Typical input**	**Main strength**	**Key limitation in cloud HCM integrations**	**Relevance to SENTINEL-HCM**
Rule-based monitoring	Thresholds, access rules, known violations	Clear and easy to audit	Misses unknown abuse patterns and slow policy drift	Used as a baseline for comparison
Statistical anomaly detection	Event counts, frequency, deviation scores	Useful for identifying extreme behavior	Often lacks business and relationship context	Provides a classical anomaly baseline

Tree-based classifiers	Engineered tabular features	Strong supervised performance when labels exist	Depends heavily on feature design and may miss multi-hop relationships	Compared against the proposed graph model
Sequence learning models	Ordered logs and event windows	Captures timing and event order	Does not fully represent users, tokens, endpoints, objects, and policies as connected entities	Used to evaluate the value of graph structure
Static graph learning	Entity relationships and graph neighborhoods	Models relational risk better than flat logs	Limited ability to capture evolving behavior and time-sensitive drift	Forms the foundation for temporal graph extension
Temporal graph learning	Time-stamped nodes, edges, and interactions	Captures evolving relationships and behavioral change	Usually assumes centralized graph data	Extended through federated training in SENTINEL-HCM
Federated learning	Local training data and shared model updates	Reduces the need to centralize sensitive data	Does not automatically model security relationships unless combined with graph learning	Used to support privacy-preserving HCM security analytics
SENTINEL-HCM	API logs, token events, role changes, object access, integration jobs, workflow events, policy baselines	Combines temporal graph learning, federated training, risk scoring, and explainable audit paths	Requires structured event normalization and careful baseline configuration	Proposed framework for cloud HCM integration security

3. Problem Formulation and Threat Model

3.1 Problem definition

SAP SuccessFactors cloud integrations generate security-relevant activity across API endpoints, middleware flows, OAuth tokens, service accounts, role permissions, workflow rules, HR data objects, and connected enterprise systems. A single log entry can confirm that an endpoint was called, a token was used, or an object was accessed, but it cannot determine whether the action was appropriate for the actor, object sensitivity, business process, destination system, timing, and approved policy state. Prior work on API security and cloud monitoring shows that misuse often occurs through valid credentials, approved interfaces, and gradual behavioral deviations rather than through visibly unauthorized access attempts [16], [17]. For this reason, integration security is formulated in this paper as a temporal relationship-learning problem rather than a flat classification problem over independent log records.

The objective is to identify whether an observed integration window represents normal behavior or one of the target risk states: API abuse, token misuse, data leakage, privileged access drift, policy drift, workflow bypass, or abnormal integration chaining. This formulation is designed for HCM environments where risk is often distributed across multiple entities. For example, a service account may receive expanded permission, use a broader token scope, call a sensitive endpoint, and export

records to an uncommon destination. Each action may appear valid when reviewed separately, while the connected path may indicate a material security or governance weakness.

3.2 Temporal HCM Security Graph

At time t, the SAP SuccessFactors cloud integration landscape is represented as a temporal HCM security graph:

$$G_t = (V_t, E_t, X_t, A_t, P_t)$$

where V_t denotes the set of entities, E_t denotes time-stamped relationships, X_t represents entity and event features, A_t represents the graph adjacency structure, and P_t represents the observed policy state. The node set includes users, service accounts, roles, OAuth tokens, API endpoints, integration jobs, middleware flows, HR data objects, workflow rules, policy controls, and external systems. The edge set captures interactions such as invokes, reads, writes, exports, grants, approves, schedules, refreshes, triggers, and violates. This representation follows the broader direction of graph-based and temporal graph learning, where entities and interactions are modeled together to detect evolving patterns that are not visible in flat tabular records [18]–[20].

Each edge e is assigned a temporal security weight:

$$\omega_e(t) = exp\left[-\lambda(t-\tau_e)\right] \cdot (1+\rho_e)$$

where τ_e is the event timestamp, λ controls temporal decay, and ρ_e represents the security relevance of the relationship. Recent events receive stronger influence than older events, while sensitive actions such as privileged role changes, broad token usage, restricted object access, abnormal endpoint calls, workflow exceptions, and external exports receive higher risk emphasis. This weighting mechanism allows the model to retain time-sensitive context while reducing the influence of stale activity that no longer reflects the current risk posture.

3.3 Event-Window Detection Objective

For each monitoring window W_i = [t_i - Δ, t_i], SENTINEL-HCM extracts a subgraph G_i and estimates both a threat label and a continuous risk score:

$$f(G_i, X_i, P_i) \rightarrow (y_i, S_i)$$

where y_i represents the predicted class and S_i represents the severity of the detected behavior. The label space is defined as:

Y = {Normal, API Abuse, Token Misuse, Data Leakage, Privileged Access Drift, Policy Drift, Workflow Bypass, Abnormal Integration Chaining}

The window-based design is necessary because integration threats often emerge as sequences rather than isolated events. High API volume may be normal during a scheduled replication job, but suspicious when paired with an unusual token scope, sensitive object access, and an unfamiliar destination system. Similarly, a permission change may be acceptable during an approved configuration release, but risky when followed by endpoint expansion and data movement outside the expected integration path. The detection objective therefore evaluates behavior through the combined evidence of graph structure, temporal sequence, object sensitivity, role context, and policy state.

Journal of Contemporary Science and Technology Management
Volume 01, Issue 01, December 2025

3.4 Threat Model

The threat model assumes that risky activity may occur through valid accounts, valid OAuth tokens, approved endpoints, and legitimate integration channels. This assumption reflects the practical challenge of enterprise cloud security, where credential misuse, over-privileged service identities, excessive access, and policy enforcement gaps may not generate immediate authentication failures [21], [22]. SENTINEL-HCM is therefore designed to detect misuse and drift inside normal-looking integration activity, not only direct access violations.

API abuse refers to endpoint usage that deviates from expected frequency, timing, sequence, object category, source context, or business purpose. The model does not treat volume alone as risk because legitimate HCM operations, such as scheduled replication, release testing, mass onboarding, reporting extracts, and compensation cycles, can produce temporary workload spikes. API abuse is instead identified when call behavior becomes inconsistent with the actor, token scope, endpoint path, object sensitivity, and historical integration pattern.

Token misuse occurs when an OAuth token, service credential, or middleware identity is used outside its expected scope, source, schedule, endpoint path, or integration context. A token may remain technically valid while being operationally suspicious. Examples include token use from an unusual location, access to endpoints outside the normal integration route, repeated use after the related process should have ended, or a sudden increase in object sensitivity associated with the token.

Data leakage is defined as excessive, unexplained, unauthorized, or policy-inconsistent movement of sensitive HR information. The framework treats leakage risk as broader than a confirmed breach. It includes early warning patterns such as unusual export volume, access to restricted object classes, unexpected object combinations, repeated extraction attempts, and transfer to an uncommon destination system. This distinction is important because sensitive HCM exposure may begin through technically permitted actions that become risky when evaluated through object sensitivity, data volume, and integration path.

Privileged access drift occurs when roles, permission groups, service account rights, workflow approver permissions, or integration authorizations gradually move away from the approved access model. Such drift may result from temporary access that is not removed, emergency configuration changes, release testing permissions, or incremental privilege expansion used to resolve operational defects. These changes may not create an immediate violation, but they can increase the attack surface and weaken segregation of duties over time.

Policy drift measures whether observed integration behavior continues to match the approved governance baseline. Let P_baseline represent the expected control state and P_current represent the observed state during a monitoring window. The policy drift distance is defined as:

$$D_{policy} = \left|\left| P_{current} - P_{baseline} \right|\right|_2$$

where a larger value indicates stronger deviation from the approved baseline. The baseline may include allowed token scopes, endpoint permissions, object-level access rights, workflow requirements, destination approvals, and data-volume limits. Policy drift is not treated as automatically malicious. It is treated as a measurable risk signal that becomes more significant when combined with abnormal graph paths, sensitive HR objects, unexpected token behavior, or unusual integration destinations. This separation between unusual behavior and policy-inconsistent behavior is important for audit-oriented security monitoring [23].

Workflow bypass describes integration activity that avoids, skips, or weakens an expected approval, validation, or control process. In SAP SuccessFactors environments, this may include updates executed through system accounts without the expected workflow path, direct object changes that normally require approval, or downstream replication before validation is completed. Abnormal integration chaining describes a sequence in which individually valid actions form an unusual or risky

Journal of Contemporary Science and Technology Management
Volume 01, Issue 01, December 2025

path when connected. These two threat classes are central to the proposed framework because automated integrations can create control exposure before human reviewers observe the changed process path.

3.5 Scope and Operational Assumptions

The scope of this study is limited to metadata-driven security intelligence. SENTINEL-HCM does not require raw employee payload values for model training. It relies on event metadata, object sensitivity categories, token behavior, endpoint patterns, role context, policy state, destination references, and graph relationships. This design supports risk detection while reducing unnecessary exposure of employee-level content.

The framework assumes that integration events can be normalized into a common schema with stable identifiers for users, service accounts, tokens, endpoints, object classes, integration jobs, workflow rules, policy controls, and external systems. It also assumes that a policy baseline can be derived from approved integration inventories, access-control designs, workflow configuration, security documentation, or governance rules. SENTINEL-HCM is positioned as an intelligence layer above existing authentication, authorization, encryption, and SIEM controls. Its purpose is to determine whether integration behavior remains normal, justified, and policy-consistent when analyzed across time, relationships, and governance context.

The proposed threat model is summarized in Figure. 2, where integration entities, access controls, HR data objects, middleware paths, external systems, and policy baselines are connected to the risk paths evaluated by SENTINEL-HCM.

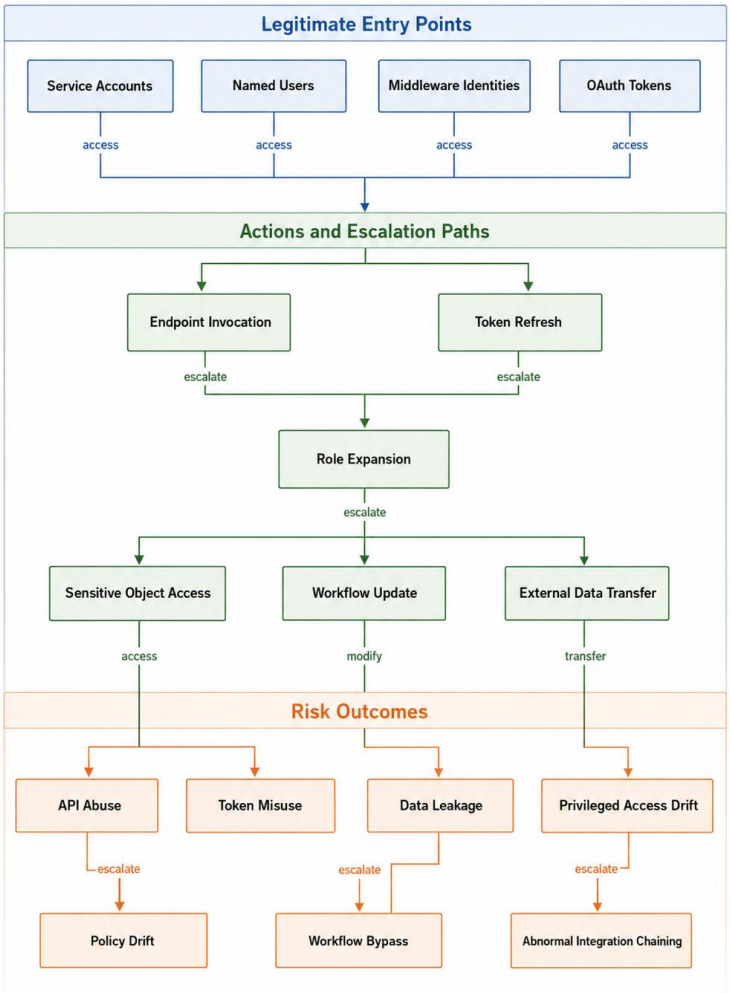

Figure 2: Threat Model for SAP SuccessFactors Cloud Integrations

Journal of Contemporary Science and Technology Management
Volume 01, Issue 01, December 2025

4. SENTINEL-HCM Framework Architecture

4.1 Architectural overview

SENTINEL-HCM is designed as a layered security intelligence framework for SAP SuccessFactors cloud integrations. Its purpose is to convert distributed integration activity into a temporal graph, learn risk patterns across time and relationships, preserve local data privacy through federated training, and return explainable alerts that can be reviewed by security, HR technology, integration, and audit teams. The architecture follows five functional layers: event acquisition, event normalization, temporal graph construction, federated temporal graph learning, and risk interpretation. This structure reflects the need to analyze API behavior, token usage, service account activity, HR object access, workflow execution, policy state, and external system movement as one connected security process rather than as separate log streams [24].

The framework begins by collecting security-relevant metadata from SAP SuccessFactors APIs, SAP BTP or CPI middleware flows, Integration Center jobs, OAuth token services, role and permission changes, workflow events, audit logs, and connected enterprise systems. The collected events are not treated as raw employee payloads. Instead, SENTINEL-HCM uses metadata such as actor type, token scope, endpoint category, object sensitivity level, action type, timestamp, source context, destination system, integration job identifier, workflow state, and policy reference. This design supports security learning while reducing unnecessary exposure of employee-level content.

4.2 Event normalization and schema alignment

The event normalization layer converts heterogeneous integration logs into a common security schema. This step is necessary because API logs, middleware traces, workflow audit records, token events, and role-change logs are often stored in different formats. Each normalized record contains five groups of attributes: actor attributes, access attributes, data-object attributes, temporal attributes, and governance attributes. Actor attributes identify whether the event was performed by a named user, service account, middleware identity, or external system. Access attributes include token identifier, token scope, endpoint, method, response status, and permission context. Data-object attributes describe the HR object class, sensitivity level, action type, and record volume. Temporal attributes capture timestamp, event window, schedule pattern, and recurrence. Governance attributes capture workflow status, policy baseline, approved destination, and control exception.

This normalized schema is important because the same technical event can carry different security implications depending on its business context. A high-volume export during a scheduled replication window may be normal, while a smaller export involving restricted fields and an unfamiliar destination may be risky. A token refresh may be routine, while the same token used after permission expansion may indicate drift. The schema therefore prepares the data for graph-based learning by preserving the relationships among actor, token, endpoint, object, destination, and policy state.

4.3 Temporal graph construction layer

After normalization, SENTINEL-HCM constructs a temporal HCM security graph for each monitoring window. Users, service accounts, OAuth tokens, API endpoints, integration jobs, middleware flows, HR objects, workflow rules, policy controls, and external systems become graph nodes. Interactions such as invokes, reads, writes, exports, grants, refreshes, approves, schedules, triggers, and violates become directed time-stamped edges. The graph is updated continuously as new integration events arrive, allowing the model to capture both short-term anomalies and slow-moving drift.

The temporal graph construction layer is central to the framework because it preserves security relationships that cannot be represented effectively in flat tabular logs. A single event may appear normal when viewed alone, but its surrounding graph path may reveal that a service account used an expanded token, accessed a sensitive endpoint, touched a restricted object, and transferred data to a destination outside the expected integration pattern. Graph learning methods are well suited to this type

of multi-entity risk representation, while temporal graph methods allow the model to capture how these relationships evolve across monitoring windows [25].

For each node v at time t, SENTINEL-HCM computes a contextual representation using the node's own features and the temporally weighted information received from neighboring nodes:

$$h_v^t = \sigma\left(W_1 x_v^t + \sum_{u \in N(v)} \alpha^t_{uv} W_2 h_u^{t-1} \right)$$

where h_v^t is the embedding of node v at time t, x_v^t is the current feature vector, N(v) is the neighborhood of v, α_{uv}^t is the temporal attention weight between neighboring nodes u and v, W_1 and W_2 are trainable weight matrices, and σ is a nonlinear activation function. This formulation allows the model to learn whether a current event is risky based on both its direct attributes and its relationship to prior activity in the graph.

4.4 Federated temporal graph learning layer

The federated learning layer enables distributed model training across multiple integration nodes without requiring raw HR event logs to be centralized. Each client environment, such as a tenant partition, business unit, regional integration node, or controlled log domain, trains a local temporal graph model on its own normalized event graph. The local model learns patterns of normal activity, token behavior, endpoint usage, object sensitivity, role context, and policy state within that environment. Only model parameters or updates are shared with the aggregation layer.

The global model is updated through weighted aggregation:

$$\theta_{global} = \sum_{k=1}^{K} \frac{n_k}{N} \theta_k \quad , \quad N = \sum_{k=1}^{K} n_k$$

where θ_k represents the local model parameters from client k, n_k represents the number of local training samples, N is the total number of samples across all participating clients, and K is the number of clients. This aggregation process allows the global model to learn broader security patterns while preserving local control over sensitive HR event logs. Federated learning is particularly relevant for HCM security because employee-related logs may contain identifiers, regional attributes, role context, workflow traces, and object references that should not be moved unnecessarily across organizational boundaries [26].

4.5 Risk scoring layer

The risk scoring layer converts model outputs into actionable security indicators. SENTINEL-HCM produces separate scores for API abuse, token misuse, data leakage, privileged access drift, policy drift, workflow bypass, and abnormal integration chaining. These scores are then combined into an overall integration risk score. The purpose is not only to classify an event window, but also to identify why the window is risky and which part of the integration path contributed to the alert.

The scoring layer uses five major evidence groups: behavioral deviation, token deviation, object sensitivity, graph-path abnormality, and policy-baseline deviation. Behavioral deviation captures unusual call frequency, timing, endpoint sequence, status code pattern, and data volume. Token deviation captures scope expansion, unusual source, unfamiliar endpoint path, and unexpected reuse. Object sensitivity captures the class and exposure level of the HR data accessed. Graph-path abnormality captures unusual connections among actors, endpoints, objects, destinations, and policy controls. Policy-baseline deviation captures the difference between observed behavior and approved governance state. This layered scoring design separates ordinary statistical abnormality from security-relevant and policy-relevant deviation.

Journal of Contemporary Science and Technology Management
Volume 01, Issue 01, December 2025

4.6 Explainability and audit interpretation layer

The final layer converts high-risk predictions into reviewable evidence. For each alert, SENTINEL-HCM returns the contributing actor, token, endpoint, HR object, destination system, workflow condition, policy deviation, and graph path. This is necessary because enterprise HCM alerts often require joint review by security, HR technology, compliance, and integration teams. A generic anomaly label is not sufficient for remediation. Reviewers need to understand whether the event reflects credential misuse, excessive permission, unexpected data movement, workflow bypass, or a legitimate business exception.

The explainability layer produces three outputs. The first output is a ranked list of contributing features, such as token scope, object sensitivity, endpoint sequence, export volume, or policy deviation. The second output is a high-risk graph path showing how the actor, token, endpoint, object, destination, and policy state are connected. The third output is a remediation-oriented interpretation, such as review token scope, verify destination approval, confirm workflow requirement, examine service account permissions, or validate the integration schedule. Explainable security outputs are essential in regulated enterprise environments because model decisions must support investigation, accountability, and audit review rather than only automated classification [27].

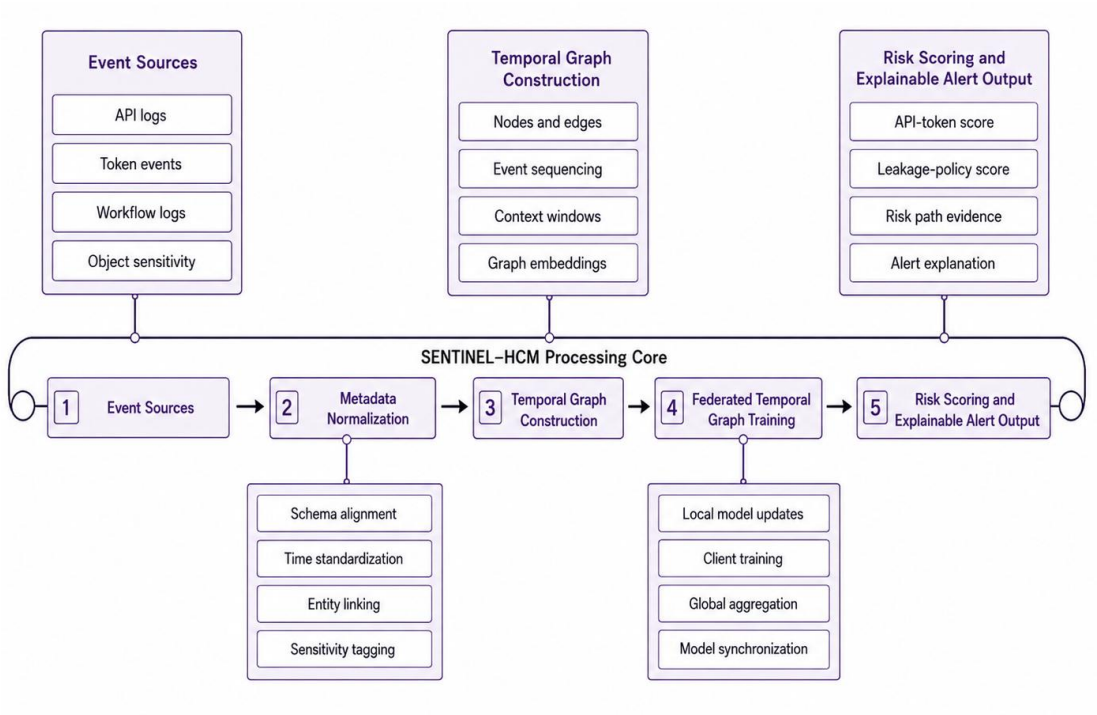

Figure 3: SENTINEL-HCM Framework Architecture

4.7 Operational workflow

The operational workflow of SENTINEL-HCM begins when integration events are collected from the SAP SuccessFactors cloud landscape and associated middleware or identity systems. The events are normalized into the common schema, assigned sensitivity and policy context, and inserted into the temporal HCM security graph. Local graph models then process event windows and generate node, edge, and subgraph embeddings. During training, local model updates are sent to the federated aggregation layer. During inference, the trained model evaluates new event windows and produces risk scores, threat labels, and explainable evidence paths.

Journal of Contemporary Science and Technology Management
Volume 01, Issue 01, December 2025

5. Risk Scoring and Mathematical Modeling

5.1 Risk Modeling Rationale

The risk scoring layer in SENTINEL-HCM translates temporal graph behavior into measurable security indicators that can be used for alert ranking, investigation, and governance review. The scoring design is intentionally broader than a simple anomaly label. In SAP SuccessFactors cloud integrations, a risky event may not be the result of one abnormal API call. It may appear through a combination of service account behavior, token scope, endpoint sequence, HR object sensitivity, data volume, workflow state, destination pattern, and policy deviation. For this reason, SENTINEL-HCM evaluates each monitoring window as a connected risk context rather than as an isolated transaction.

For a monitoring window W_i, the framework first produces a compact representation of the temporal subgraph G_i. This representation combines node embeddings, edge embeddings, temporal attention signals, and policy-state features:

$$z_i = READOUT(\{h_v^t, h_e^t, p_i \mid v \epsilon V_i, e \epsilon E_i\})$$

where z_i denotes the window-level representation, h_v^t is the temporal embedding of node v, h_e^t is the edge representation, and p_i represents the policy-state vector for the same window. The READOUT function aggregates the learned graph context into a single representation used for risk scoring and threat classification. This allows the model to retain the relationship among actors, tokens, endpoints, objects, workflows, destinations, and policies.

5.2 API Abuse and Token Behavior Scoring

API abuse and token misuse are scored together because they often occur in the same integration path. A high API call volume is not automatically suspicious in cloud HCM environments because scheduled replication, mass onboarding, release testing, reporting extracts, and compensation cycles can create legitimate workload spikes. Risk increases when call frequency, endpoint sequence, token scope, source context, and object sensitivity change at the same time.

The combined API and token behavior score is defined as:

$$S_{api-token} = \alpha_1 F_{freq} + \alpha_2 F_{seq} + \alpha_3 F_{scope} + \alpha_4 F_{source} + \alpha_5 F_{endpoint}$$

where F_freq measures API frequency deviation, F_seq measures abnormal endpoint sequencing, F_scope measures token-scope deviation, F_source measures unusual source or client behavior, and F_endpoint measures endpoint sensitivity. The coefficients α_1 to α_5 determine the relative contribution of each signal. This score helps distinguish legitimate high-volume integrations from patterns that suggest token misuse, endpoint abuse, or service account overreach.

5.3 Data Leakage and Policy Drift Measurement

Data leakage risk is evaluated through the sensitivity of the accessed HR object, the volume of records involved, the destination system, and the abnormality of the graph path. This is necessary because a technically permitted export can still become risky when it includes sensitive object classes, combines unusual fields, or moves data through a destination that does not match the approved integration route. Policy drift is evaluated alongside leakage because data exposure often becomes more serious when current behavior moves away from the approved governance baseline.

The leakage and policy drift score is defined as:

$$S_{leak-policy} = \beta_1 O_{sens} + \beta_2 D_{vol} + \beta_3 D_{dest} + \beta_4 G_{path} + \beta_5 D_{policy}$$

Journal of Contemporary Science and Technology Management
Volume 01, Issue 01, December 2025

where O_sens represents HR object sensitivity, D_vol represents data-volume deviation, D_dest represents destination-system risk, G_path represents graph-path abnormality, and D_policy represents deviation from the approved policy baseline. The coefficients β_1 to β_5 control the strength of each signal. A high score does not necessarily confirm a breach; it identifies a data movement pattern that requires review because the object, destination, volume, or policy context differs from the expected integration behavior.

5.4 Integrated Risk Index

The final risk index combines the major evidence dimensions into one operational score. The purpose of the integrated score is to rank event windows by severity while still preserving the individual sub-scores needed for investigation. SENTINEL-HCM therefore does not hide the reason behind a prediction. It reports the total risk score together with the API-token score, leakage-policy score, access drift signal, workflow bypass signal, and abnormal chaining signal.

The integrated risk index is expressed as:

$$S_{total} = \sigma\ (\omega_1 S_{api-token} + \omega_2 S_{leaky-policy} + \omega_3 S_{access} + \omega_4 S_{workflow} + \omega_5 S_{chain} + b\ )$$

where σ is the sigmoid function, S_api-token represents API and token behavior risk, S_leak-policy represents data leakage and policy drift risk, S_access represents privileged access drift, S_workflow represents workflow bypass risk, S_chain represents abnormal integration chaining, and b is the bias term. The weights ω_1 to ω_5 determine the contribution of each risk dimension. This formulation allows the model to assign higher severity when multiple weak signals appear together, such as broader token use, sensitive object access, abnormal destination behavior, and policy deviation within the same monitoring window.

5.5 Training Objective and Score Calibration

SENTINEL-HCM uses a compact multi-objective training design so that the model learns classification accuracy, graph structure, temporal consistency, and risk-score reliability together. The objective is not only to classify event windows correctly, but also to produce stable and meaningful scores that can be used in operational review.

The total training loss is defined as:

$$L_{total} = L_{cls} + \lambda_1 L_{graph} + \lambda_2 L_{temp} + \lambda_3 L_{score}$$

where L_cls is the classification loss, L_graph is the graph reconstruction loss, L_temp is the temporal consistency loss, and L_score is the score calibration loss. The coefficients λ_1 to λ_3 control the contribution of each auxiliary objective. The classification loss trains the model to identify the correct threat category. The graph reconstruction loss encourages the model to preserve relationships among actors, tokens, endpoints, objects, workflows, and destinations. The temporal consistency loss reduces unstable predictions across adjacent windows unless the underlying behavior changes meaningfully. The score calibration loss aligns predicted severity with the expected operational importance of the alert [28].

5.6 Alert Interpretation

The final output of the scoring layer includes the predicted threat class, total risk score, contributing sub-scores, involved graph path, affected object category, policy deviation, and suggested review focus. This structure is important because enterprise HCM security teams need more than a numeric anomaly score. A useful alert must show whether the risk was driven by token-scope expansion, endpoint deviation, sensitive object access, unusual destination behavior, workflow bypass, or policy drift.

Journal of Contemporary Science and Technology Management
Volume 01, Issue 01, December 2025

The alert interpretation follows three levels. Low-risk windows represent behavior consistent with expected integration activity. Medium-risk windows indicate measurable deviation that should be reviewed, especially when sensitive objects or policy drift are involved. High-risk windows indicate behavior requiring investigation because multiple risk signals align across token behavior, endpoint activity, data sensitivity, graph path, and governance baseline. This makes the scoring layer suitable for both automated monitoring and human review, while keeping the model's output understandable for security, HR technology, integration, and audit stakeholders.

6. Experimental Methodology

6.1 Experimental Design

The experimental design evaluates SENTINEL-HCM as a security intelligence framework for SAP SuccessFactors cloud integrations under realistic enterprise integration conditions. The study uses metadata-level integration events rather than raw employee payload values. Each record represents an observable security event generated by API access, middleware execution, OAuth token usage, role or permission change, workflow activity, sensitive object access, policy-state change, or downstream system exchange. This design keeps the experiment focused on behavioral security signals while avoiding unnecessary exposure of employee-level content.

The evaluation is organized around four objectives. The first objective is to measure whether SENTINEL-HCM improves detection of API abuse, token misuse, data leakage, privileged access drift, policy drift, workflow bypass, and abnormal integration chaining. The second objective is to compare the proposed model with rule-based, statistical, tree-based, sequence-based, and graph-based baselines. The third objective is to test whether temporal graph features and federated learning improve performance beyond flat log classification. The fourth objective is to measure operational practicality through false positive rate, detection latency, throughput, and scalability under increasing event volume.

6.2 Dataset Construction

The experimental corpus was constructed as a de-identified metadata-level HCM integration dataset. It represents event patterns commonly observed across SAP SuccessFactors API services, SAP BTP or CPI middleware flows, Integration Center jobs, identity services, workflow execution logs, role-change records, and connected external systems. Each event contains timestamp, actor type, service account identifier, token identifier, endpoint category, HTTP method, response status, integration job identifier, HR object class, object sensitivity level, record volume, source context, destination system, workflow state, policy reference, and event label [29].

The dataset includes both normal integration activity and controlled threat scenarios. Normal activity includes scheduled employee replication, identity synchronization, workflow-triggered updates, reporting extracts, onboarding-related data movement, learning-system exchange, and routine middleware execution. Threat scenarios were introduced at the metadata level to represent abnormal but realistic security behavior, including excessive endpoint calls, unusual token scope, sensitive object extraction, unauthorized destination movement, access drift, workflow bypass, and abnormal event chaining. Threat labels were assigned according to the observed event pattern, graph path, object sensitivity, policy state, and destination behavior.

Table 2. Experimental dataset and threat distribution

Category	Count
Total integration events	1,250,000
Normal events	1,000,000

API abuse events	62,500
Token misuse events	43,750
Data leakage risk events	37,500
Privileged access drift events	31,250
Policy drift events	37,500
Workflow bypass events	25,000
Abnormal integration chaining events	12,500
Unique named users	18,000
Service accounts	480
OAuth token records	2,950
API endpoint categories	310
HR object classes	145
Integration jobs	820
External systems	68
Policy and workflow controls	420
Temporal graph nodes	23,201
Temporal graph edges	4,850,000

6.3 Feature Engineering

Feature engineering was performed at the event, entity, path, and window levels. Event-level features describe the direct properties of an activity, including method type, response status, timestamp, endpoint category, object class, action type, and record volume. Entity-level features describe the actor, service account, token, role context, destination system, and historical behavior profile. Path-level features capture the relationship among the actor, token, endpoint, HR object, workflow state, and external system. Window-level features aggregate behavior over a monitoring interval and include call frequency, sequence deviation, sensitivity exposure, token reuse, endpoint expansion, policy drift, and abnormal chaining.

The key feature groups are API behavior, token behavior, object sensitivity, destination behavior, graph topology, temporal sequence, workflow state, and policy baseline deviation. API behavior features measure frequency, endpoint sequence, response status, and schedule deviation. Token behavior features measure scope, source, reuse, route, and lifetime patterns. Object sensitivity features measure the risk level of accessed HR objects and unusual object combinations. Graph topology features measure path abnormality, neighborhood change, centrality shift, and unexpected links. Policy features measure

deviation from approved token scopes, destination controls, workflow requirements, object permissions, and data-volume expectations.

6.4 Temporal Graph Preparation

The normalized events were converted into temporal graph snapshots using fixed monitoring windows. Within each window, entities were represented as nodes and interactions were represented as directed time-stamped edges. The graph was not rebuilt from scratch after every event. Instead, it was updated incrementally so that recurring integration behavior, new links, role changes, token movement, and policy deviations could be tracked over time.

Each graph snapshot preserved the relationship between the integration action and the governance context in which it occurred. For example, an API call was linked to the actor, token, endpoint, object class, workflow condition, destination system, and policy baseline. This structure allowed the model to evaluate risk as a connected integration path. The same API call could therefore receive different risk interpretation depending on whether it occurred through an expected token, within a scheduled job, against a normal object class, and toward an approved destination.

6.5 Training and Validation Strategy

The dataset was divided using a time-based split rather than a random split. This approach better reflects operational security monitoring, where a model is trained on historical behavior and tested on later event windows. The training set contained earlier normal and labeled threat windows. The validation set was used for threshold selection, score calibration, and early stopping. The test set contained later integration windows to evaluate how well each model generalized to unseen behavior.

SENTINEL-HCM was trained in three stages. In the first stage, the temporal graph encoder learned normal integration behavior through graph reconstruction and temporal consistency objectives. In the second stage, the model was fine-tuned on labeled threat windows to distinguish API abuse, token misuse, data leakage, privileged access drift, policy drift, workflow bypass, and abnormal chaining. In the third stage, federated training was applied across distributed client partitions. Each client trained locally on its own event graph, and only model updates were sent for aggregation. Raw event logs were not transferred between clients.

The dataset was split chronologically into 70% training, 10% validation, and 20% testing windows. Each monitoring window used a fixed 30-minute interval, and all baseline models were evaluated using the same partitions to ensure fair comparison.

Table 3. Model training configuration and baseline setup

Model	Input representation	Main purpose	Training setting
Rule-based monitoring	Thresholds and policy rules	Transparent control baseline	Manually configured thresholds
Isolation Forest	Tabular event features	Classical anomaly baseline	200 estimators, contamination tuned on validation set
XGBoost	Engineered tabular features	Supervised non-graph comparison	500 trees, depth 6, learning rate 0.05
LSTM	Ordered event sequences	Sequence anomaly	64 hidden units, 30-event

autoencoder		comparison	window
GraphSAGE	Static graph snapshots	Non-temporal graph comparison	128-dimensional embeddings, 2 layers
Temporal Graph Network	Dynamic event graph	Temporal graph baseline	128-dimensional memory, temporal attention
SENTINEL-HCM	Federated temporal graph with policy features	Proposed framework	5 federated clients, 50 rounds, 128-dimensional embeddings

6.6 Baseline Models

The selected baselines represent different levels of security intelligence. Rule-based monitoring was included because it reflects the most common operational approach in enterprise integration monitoring. Isolation Forest was used as a classical anomaly detection method for identifying unusual event patterns without labels. XGBoost was included as a strong supervised tabular model because it performs well when engineered features and labeled examples are available. LSTM autoencoder was used to evaluate whether sequence learning alone can detect abnormal integration behavior. GraphSAGE was used to test whether static graph structure improves detection beyond tabular and sequence models. Temporal Graph Network was included as a non-federated temporal graph baseline.

SENTINEL-HCM was evaluated against these baselines to determine whether its combined design provides measurable improvement. The comparison is important because each baseline captures only part of the problem. Rule-based monitoring captures known controls. Isolation Forest captures statistical deviation. XGBoost captures supervised feature interactions. LSTM autoencoder captures sequence behavior. GraphSAGE captures static relationships. Temporal Graph Network captures evolving relationships. SENTINEL-HCM is expected to perform better when the risk depends on temporal relationships, token behavior, sensitive-object lineage, policy drift, and distributed training.

6.7 Evaluation Metrics

The models were evaluated using detection performance and operational performance metrics. Detection performance was measured using precision, recall, F1-score, ROC-AUC, PR-AUC, false positive rate, and class-wise detection performance. Precision measures how many predicted alerts are correct. Recall measures how many actual risk events are detected. F1-score balances precision and recall. ROC-AUC measures discrimination across thresholds, while PR-AUC is especially useful when threat events are less frequent than normal activity. False positive rate is important because excessive alerts reduce trust in operational security monitoring.

Operational performance was measured using average detection latency, peak latency, throughput, and memory usage under increasing event volume. These metrics are necessary because a strong detection model is not useful if it cannot operate under enterprise integration workloads. The latency test measured how quickly each model produced a risk decision after receiving a monitoring window. The throughput test measured the number of events processed per second. The scalability test evaluated model behavior under low, medium, high, and peak integration workloads [30].

6.8 Federated Training Setup

The federated experiment divided the event corpus into five client partitions representing separate integration domains. Each client contained different but overlapping behavior patterns. One client emphasized identity and user replication events, another emphasized workflow and role-change activity, another emphasized reporting and analytics extracts, another

emphasized middleware and external system exchange, and another emphasized mixed enterprise integration workloads. This partitioning was designed to test whether the global model could learn shared security patterns while preserving local variation.

During each federated round, clients trained local temporal graph models on their own event windows. Model updates were then aggregated to produce a global model, which was redistributed for the next round. The aggregation process weighted each client by its local event volume. Validation was performed after each aggregation round to monitor convergence, score stability, and false positive behavior. The final global model was evaluated on the held-out test windows.

6.9 Experimental Workflow

The complete experimental workflow follows a controlled sequence. First, raw integration metadata is collected and normalized into the common schema. Second, threat labels and policy-state indicators are assigned based on event behavior, object sensitivity, graph path, and governance baseline. Third, temporal graph windows are created from the normalized event stream. Fourth, baseline models and SENTINEL-HCM are trained using the same training, validation, and test partitions. Fifth, detection metrics are calculated for each model. Sixth, ablation tests are performed by removing graph, temporal, token, policy, and federated components. Finally, operational latency and throughput are measured under increasing workload conditions.

The federated training and testing workflow is shown in Figure. 4.

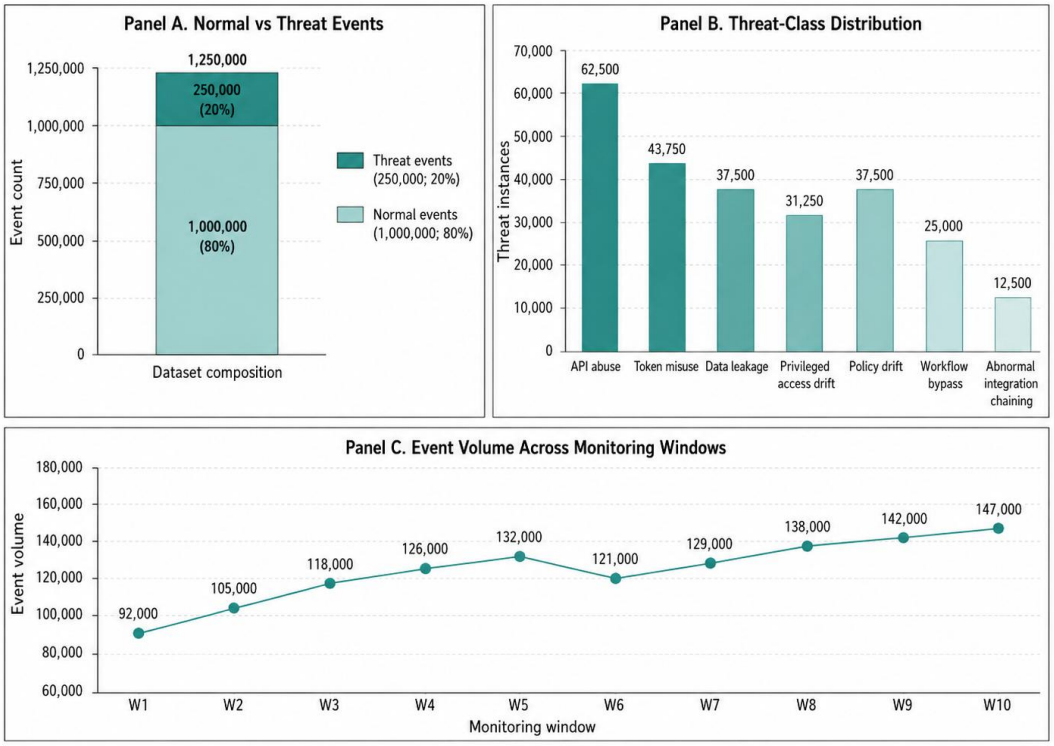

Figure 4: Experimental Dataset and Threat Distribution Profile

Journal of Contemporary Science and Technology Management
Volume 01, Issue 01, December 2025

7. Results and Analysis

The results show that SENTINEL-HCM achieved the strongest overall detection performance across the evaluated models. The improvement was most visible in F1-score, PR-AUC, false positive rate, and detection latency. Rule-based monitoring produced acceptable precision for known violations but missed gradual drift patterns and multi-step integration risks. Isolation Forest detected extreme deviations but struggled with normal high-volume integration periods, especially scheduled replication and reporting extracts. XGBoost performed well on engineered features, but its accuracy declined when the risk depended on multi-hop relationships among service accounts, tokens, endpoints, objects, and destinations. The LSTM autoencoder improved sequence-based detection, but it did not fully capture graph relationships or policy-state context. GraphSAGE and the Temporal Graph Network performed better than flat and sequence-based models, confirming that relational structure and time-aware modeling are important for cloud HCM security. SENTINEL-HCM produced the best performance because it combined temporal graph structure, token behavior, policy drift signals, sensitive-object lineage, and federated training.

Table 4. Overall model performance comparison

Model	Precision	Recall	F1-score	ROC-AUC	PR-AUC	False positive rate	Avg. detection latency
Rule-based monitoring	0.781	0.642	0.705	0.812	0.684	8.90%	18 ms
Isolation Forest	0.742	0.701	0.721	0.841	0.702	7.80%	31 ms
XGBoost	0.861	0.824	0.842	0.914	0.861	5.40%	42 ms
LSTM autoencoder	0.873	0.846	0.859	0.927	0.878	4.90%	68 ms
GraphSAGE	0.892	0.871	0.881	0.941	0.902	4.30%	74 ms
Temporal Graph Network	0.913	0.894	0.903	0.958	0.926	3.70%	86 ms
SENTINEL-HCM	0.946	0.929	0.937	0.978	0.961	2.60%	79 ms

The proposed model improved F1-score by 3.4 percentage points over the Temporal Graph Network and by 23.2 percentage points over rule-based monitoring. The reduction in false positives was also significant. SENTINEL-HCM reduced the false positive rate to 2.60%, compared with 3.70% for the Temporal Graph Network and 8.90% for rule-based monitoring. This result is important because excessive false positives can weaken trust in security monitoring and increase review workload for integration and audit teams.

Figure. 5 presents the comparative detection performance across the evaluated models.

Journal of Contemporary Science and Technology Management
Volume 01, Issue 01, December 2025

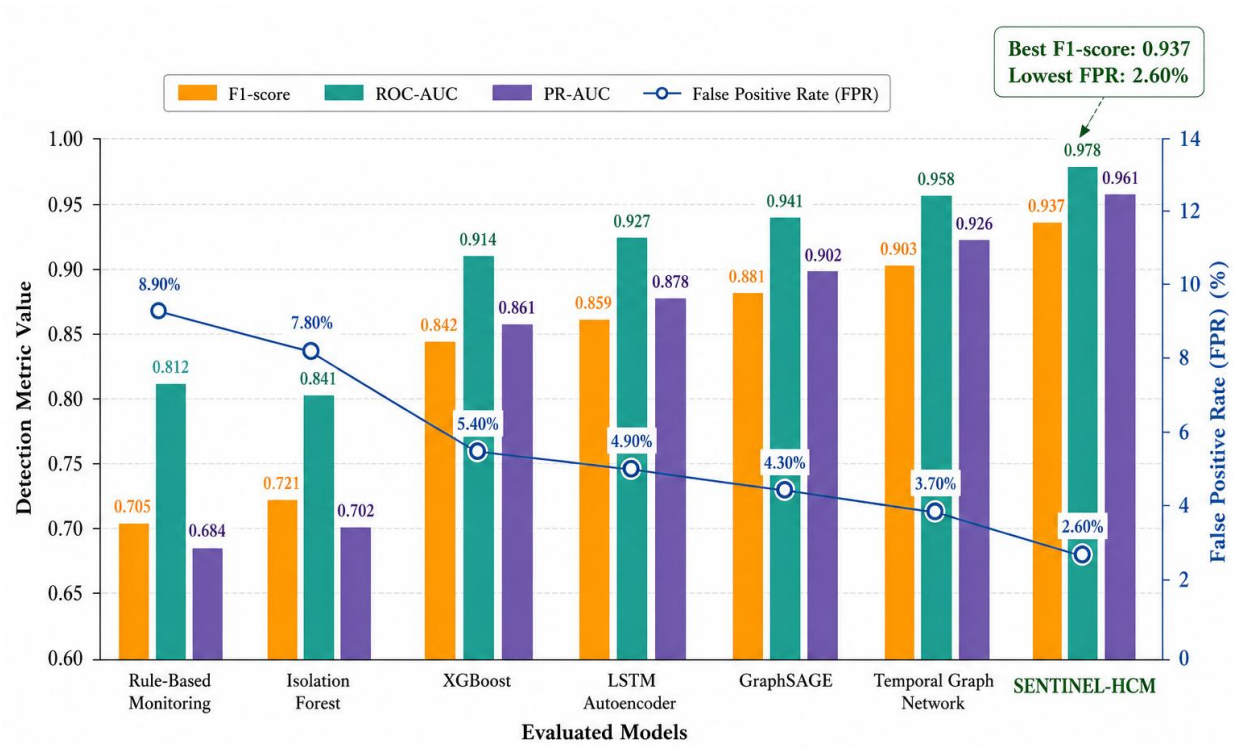

Figure 5: Comparative Detection Performance Across Baseline Models and SENTINEL-HCM

7.2 Threat-Class Performance

Class-wise results show that API abuse and token misuse were detected with the highest reliability because both classes produced strong behavioral and temporal signals. API abuse typically generated abnormal endpoint frequency, sequence deviation, and response-pattern changes. Token misuse produced identifiable changes in token scope, source context, endpoint route, and reuse behavior. Data leakage risk was also detected strongly when object sensitivity, record volume, destination behavior, and graph-path abnormality were evaluated together.

Policy drift and privileged access drift were more difficult to detect than direct API abuse because they developed gradually and often appeared as small configuration or permission changes before becoming visible as high-risk behavior. Workflow bypass and abnormal integration chaining were the most challenging classes because they depended on understanding whether a sequence of individually valid actions created an unexpected control gap. Even in these harder categories, SENTINEL-HCM maintained stable performance because it evaluated the full integration path instead of relying only on event frequency or endpoint usage.

The strongest class-wise F1-scores were observed for API abuse, token misuse, and data leakage risk. The lowest F1-score occurred for abnormal integration chaining, mainly because this class overlapped with token misuse, workflow bypass, and policy drift in some event windows. This overlap is expected in enterprise integration security because one risk scenario may involve several control weaknesses at the same time.

Journal of Contemporary Science and Technology Management
Volume 01, Issue 01, December 2025

	Precision	Recall	F1-score	Support
Normal	0.961	0.973	0.967	200,000
API Abuse	0.952	0.941	0.946	12,500
Token Misuse	0.947	0.934	0.940	8,750
Data Leakage Risk	0.943	0.928	0.935	7,500
Privileged Access Drift	0.931	0.914	0.922	6,250
Policy Drift	0.936	0.918	0.927	7,500
Workflow Bypass	0.914	0.889	0.901	5,000
Abnormal Integration Chaining	0.902	0.871	0.886	2,500

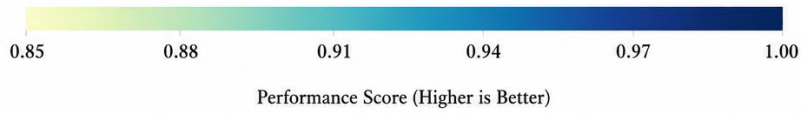

Figure 6: Class-Wise Threat Detection Performance Matrix

7.3 Policy Drift and Data Leakage Risk Analysis

The policy drift and data leakage analysis showed that the highest-risk windows were not always the highest-volume windows. Several high-risk cases involved moderate event volume but stronger combinations of sensitive object access, expanded token scope, uncommon destination systems, and policy deviation. This confirms that volume-based monitoring alone is not sufficient for cloud HCM security.

The heatmap analysis showed three major risk concentrations. First, service accounts with broader token scopes produced higher leakage scores when they accessed compensation, identity, or payment-related object categories. Second, external destinations with limited historical interaction produced higher risk when combined with unusual export behavior. Third, policy drift became more severe when role changes, endpoint expansion, and workflow exceptions occurred within the same monitoring window. These patterns support the central design assumption of SENTINEL-HCM: integration risk is clearer when actor, token, endpoint, object, destination, and policy context are evaluated together.

Figure. 7 summarizes the policy drift and data leakage risk distribution across endpoint categories, token scope levels, role groups, and sensitive HR object classes.

Journal of Contemporary Science and Technology Management
Volume 01, Issue 01, December 2025

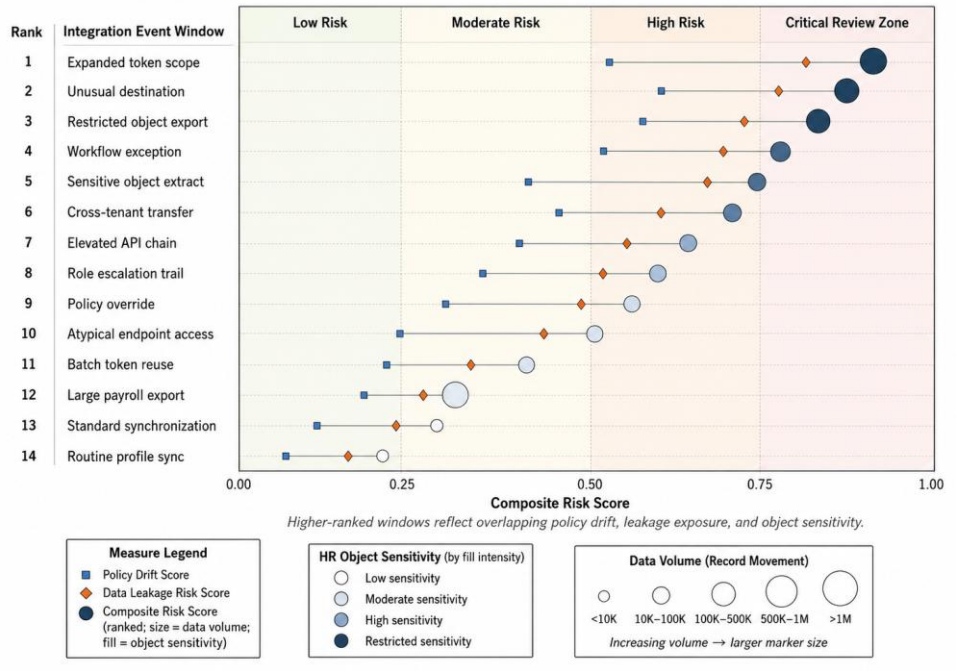

Figure 7: Policy Drift Versus Data Leakage Risk Surface

7.4 Ablation Results

The ablation study measured the contribution of each major component in SENTINEL-HCM. Removing temporal features reduced the model's ability to detect slow-moving drift and abnormal event sequences. Removing graph features caused a larger performance decline because the model lost the relationships among service accounts, tokens, endpoints, objects, workflows, and destinations. Removing token features had a strong effect on token misuse and API abuse detection. Removing policy drift features reduced performance for privileged access drift, workflow bypass, and policy inconsistency. Removing federated learning slightly reduced F1-score and increased false positives because the centralized variant did not preserve local behavioral differences across integration domains as effectively.

Table 5. Ablation and operational performance summary

Model variant	F1-score	ROC-AUC	False positive rate	Avg. latency	Throughput
Full SENTINEL-HCM	0.937	0.978	2.60%	79 ms	14,800 events/sec
Without graph features	0.884	0.939	4.70%	61 ms	16,100 events/sec
Without temporal features	0.901	0.951	4.10%	64 ms	15,900 events/sec
Without token features	0.908	0.956	3.90%	72 ms	15,300 events/sec

Journal of Contemporary Science and Technology Management
Volume 01, Issue 01, December 2025

Without policy drift score	0.912	0.961	3.60%	76 ms	15,000 events/sec
Without sensitive-object lineage	0.917	0.964	3.40%	75 ms	15,100 events/sec
Without federated training	0.921	0.967	3.20%	73 ms	15,400 events/sec

The largest performance loss occurred when graph features were removed, reducing F1-score from 0.937 to 0.884. This confirms that the relationship structure is essential for detecting integration risk. Removing temporal features reduced F1-score to 0.901, showing that time-aware behavior is also necessary, especially for policy drift and abnormal chaining. The removal of token features and policy drift scoring had a smaller but still meaningful effect, which shows that these components strengthen detection for risks that may otherwise appear as normal authorized activity.

Figure. 8 presents the ablation impact together with latency and throughput behavior under increasing workload conditions.

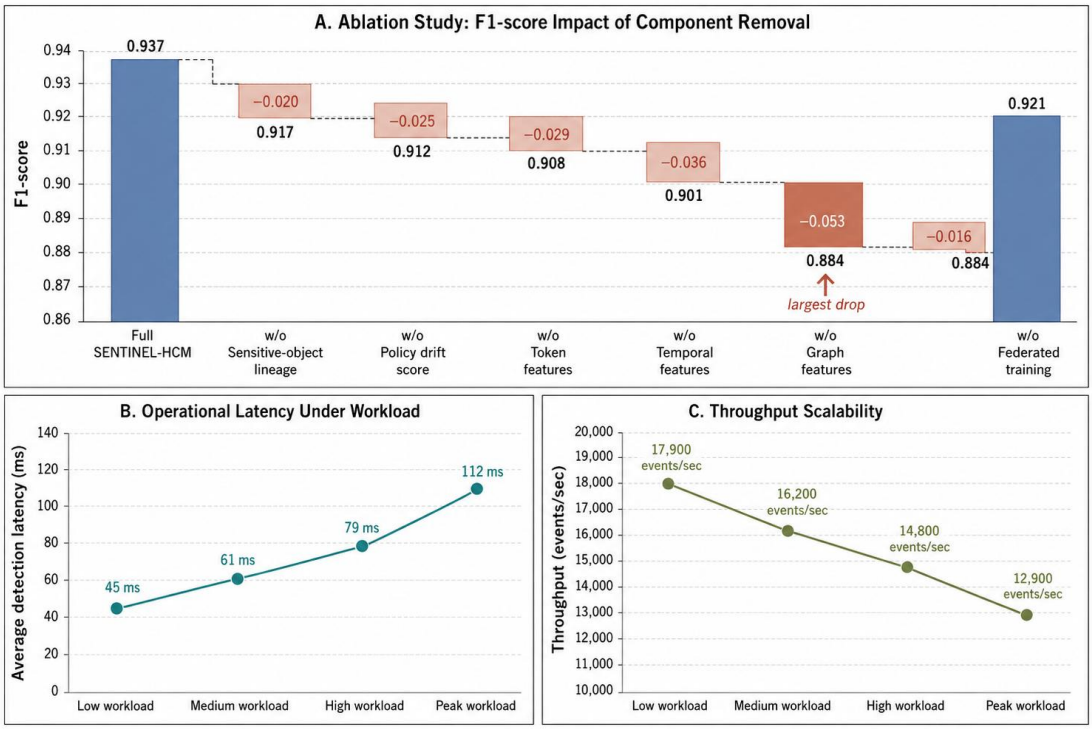

Figure 8: Ablation and Operational Scalability Summary

7.5 Latency and Scalability Analysis

The operational performance results indicate that SENTINEL-HCM remained practical under increasing integration workload. The full model introduced higher latency than rule-based and tabular models because it processed temporal graph structure and risk-path context. However, the average detection latency of 79 ms remained suitable for near real-time monitoring of integration windows. The model also maintained throughput of 14,800 events per second, which is sufficient for high-volume enterprise integration monitoring when deployed as a risk-scoring layer rather than as a blocking transaction control.

Latency increased gradually as event volume, graph density, and policy-state complexity grew. The increase was most visible during peak workload windows containing high API activity, frequent token reuse, multiple external destinations, and simultaneous workflow or permission changes. Even under peak conditions, the model showed stable scoring behavior and did not produce sharp false positive spikes. This stability is important because enterprise HCM workloads naturally fluctuate during scheduled replications, release testing, onboarding cycles, reporting periods, and annual HR processes.

7.6 Observed Security Patterns

Several practical observations emerged from the results. First, the strongest API abuse signals were not only frequency-based. The most reliable indicators combined endpoint sequence deviation, unusual source context, broader token scope, and sensitive-object access. Second, token misuse was easier to detect when the model had access to both current token behavior and the token's historical route through endpoints and destinations. Third, data leakage risk was most visible when object sensitivity and destination behavior were evaluated together. A large export to a routine destination was often less risky than a smaller export involving restricted objects and an unfamiliar destination.

Fourth, policy drift behaved differently from conventional anomaly detection. Some drift cases were not statistically rare, but they were inconsistent with the approved baseline. This means that an event can be common and still be risky from a governance perspective. Fifth, workflow bypass and abnormal chaining required relationship-level analysis. These cases were frequently missed by rule-based and tabular models because each individual action appeared valid. SENTINEL-HCM performed better because it evaluated the sequence as a connected path.

7.7 Result Interpretation

The results confirm that SAP SuccessFactors cloud integration security benefits from a model that combines temporal behavior, graph relationships, token context, object sensitivity, and policy-state comparison. The strongest gains occurred in scenarios where the risk was distributed across several entities rather than concentrated in one event. This supports the paper's core argument that cloud HCM security should be analyzed as a time-evolving relationship problem.

SENTINEL-HCM did not only improve model accuracy. It also improved operational usefulness by lowering false positives, preserving explainable risk components, and identifying the reason behind high-risk windows. The framework can show whether an alert was mainly driven by token-scope deviation, endpoint sequence change, sensitive-object exposure, abnormal destination behavior, workflow bypass, or policy drift. This level of explanation is important for enterprise adoption because security and HR technology teams must decide whether an alert represents a real threat, an approved exception, a configuration issue, or a process requiring governance review.

Overall, SENTINEL-HCM provided the most balanced performance across detection quality, interpretability, and operational practicality. The results show that federated temporal graph intelligence is a suitable approach for securing SAP SuccessFactors cloud integrations where sensitive HR data, automated service accounts, tokenized access, workflow controls, and external system dependencies must be monitored together.

8. Discussion and Enterprise Implications

8.1 Interpretation of Findings

The results indicate that SAP SuccessFactors cloud integration security is better understood as a relationship-based and time-dependent problem than as a simple event-monitoring task. Conventional monitoring methods can identify known violations, failed requests, excessive traffic, or predefined access exceptions, but they become less reliable when risky behavior develops through valid credentials, approved endpoints, scheduled jobs, and gradual configuration changes. SENTINEL-HCM performed better because it evaluated the full integration context, including the actor, token, endpoint, HR object, destination system, workflow state, and policy baseline within the same monitoring window.

The most important finding is that risk was often distributed across multiple weak signals rather than concentrated in one obvious violation. A service account using a valid token may not be suspicious by itself. A sensitive object access may also be valid within a known integration. A larger data export may be expected during a scheduled process. However, when these signals appear together with an unfamiliar endpoint path, broader token scope, uncommon destination, or policy deviation, the combined pattern becomes materially more important. This confirms the need for security models that can connect technical activity with governance context.

The improvement of SENTINEL-HCM over rule-based monitoring and classical anomaly detection also shows the limitation of volume-centered security logic. High event volume alone did not consistently represent risk, and some of the highest-risk windows involved moderate volume but stronger sensitivity and policy deviation. This distinction is especially relevant in HCM landscapes, where integration workloads naturally increase during onboarding, reporting cycles, compensation planning, release testing, and organizational restructuring. A model that treats workload spikes as the primary signal will produce unnecessary alerts, while a model that considers graph path, object sensitivity, token behavior, and policy state can provide more reliable prioritization.

8.2 Enterprise Security Value

For enterprise security teams, SENTINEL-HCM provides a practical method for detecting suspicious integration behavior before it becomes a confirmed exposure event. The framework can identify abnormal API usage, token misuse, excessive service account activity, sensitive data movement, destination changes, and policy deviation within a single scoring process. This helps security teams move from reactive alert review toward earlier risk identification.

The framework is also useful because it produces explainable evidence rather than only a prediction score. In operational security environments, an alert must answer more than whether something is abnormal. It must explain which service account was involved, which token scope changed, which endpoint was used, which object class was accessed, where the data moved, and which policy condition was affected. SENTINEL-HCM supports this requirement by returning the contributing risk dimensions and the graph path behind the alert. This makes the output easier to validate, investigate, and escalate.

The model's false positive reduction is particularly important for enterprise adoption. Security teams often face alert fatigue when monitoring systems produce a high number of low-value warnings. In cloud HCM environments, this problem becomes worse because many legitimate business processes resemble abnormal activity when viewed only through volume or frequency. By combining relationship context and policy context, SENTINEL-HCM reduces unnecessary alerts and allows reviewers to focus on events with stronger security relevance.

8.3 HR Technology and Integration Governance Value

For HR technology and integration teams, SENTINEL-HCM provides visibility into risks that are often hidden inside technical operations. SAP SuccessFactors integrations are frequently modified to support new countries, vendors, HR processes, reporting needs, role changes, and business restructures. These changes can introduce control gaps when temporary permissions remain active, service accounts receive broader access, workflows are bypassed for operational convenience, or external destinations change without complete governance review.

The framework helps identify these issues by treating configuration and integration behavior as part of the same security surface. A role change is not evaluated only as an access event. It is evaluated in relation to later endpoint activity, token usage, object access, workflow behavior, and destination movement. This allows HR technology teams to detect whether a configuration change has created downstream exposure. It also supports better post-release monitoring because the model can identify whether new or modified integrations are behaving differently from their approved design.

The framework can also support integration lifecycle governance. During development, testing, deployment, and post-go-live support, SENTINEL-HCM can be used to compare expected integration behavior with observed runtime behavior. This is

useful for detecting over-permissioned service accounts, excessive endpoint access, unexpected object combinations, and data movement outside the documented integration scope. In this sense, the model is not only a security tool but also a governance mechanism for maintaining clean, controlled, and auditable integration operations.

8.4 Privacy-Preserving Monitoring

The federated learning design addresses a major challenge in HCM security analytics: employee-related logs are sensitive even when payload values are removed. Integration metadata can still reveal user activity, role context, system behavior, regional patterns, workflow timing, and object-level access. Centralizing all such logs into one training environment may conflict with internal privacy principles, regional restrictions, or data minimization expectations.

SENTINEL-HCM reduces this concern by allowing local environments to train on their own event graphs while sharing only model updates. This makes the framework more suitable for distributed enterprises where SAP SuccessFactors integrations operate across regions, business units, legal entities, and external system boundaries. The approach also preserves local behavioral differences. A region with heavy identity synchronization, a business unit with frequent reporting extracts, and a tenant partition with complex workflow activity may each have different normal behavior. Federated training allows the model to learn from these differences without forcing every raw event into a centralized repository.

This privacy-preserving design is important for long-term scalability. As organizations expand cloud HR ecosystems, security monitoring must become more intelligent without increasing unnecessary data movement. SENTINEL-HCM provides a structure where model learning can improve across distributed environments while still respecting the sensitivity of HCM event data.

8.5 Compliance and Audit Readiness

SENTINEL-HCM supports audit readiness by connecting alerts to evidence. Traditional monitoring outputs often require manual investigation across several systems, such as API logs, middleware traces, access-control records, workflow history, and policy documentation. This fragmented review process can delay remediation and create uncertainty about whether an event was truly risky. The proposed framework reduces this gap by linking the event window to the actor, token, endpoint, object, workflow state, destination, and policy baseline.

This evidence-based structure is valuable for compliance teams because it separates unusual behavior from policy-inconsistent behavior. An event may be rare but approved, or common but no longer compliant with the current baseline. SENTINEL-HCM is designed to support this distinction by including policy drift as a measurable signal. This helps organizations identify not only suspicious activity but also governance weaknesses such as outdated permissions, undocumented integration changes, stale service account access, excessive token scope, and workflow exceptions that have become routine.

The framework can also improve audit discussions by providing traceable risk paths. Instead of presenting only a numeric anomaly score, the model can show how the risk formed across the integration chain. For example, it can identify that a service account used a broader token to access a sensitive object and export records to a destination not aligned with the approved baseline. This form of explanation is more useful for audit and remediation because it connects the technical event to the control weakness.

8.6 Deployment Considerations

Successful deployment of SENTINEL-HCM depends on the quality of event normalization and policy baseline definition. The framework requires stable identifiers for users, service accounts, tokens, endpoints, integration jobs, HR object classes, workflow rules, policy controls, and external systems. If these identifiers are inconsistent, incomplete, or poorly governed, the

temporal graph may not accurately represent the integration landscape. Data preparation is therefore not a secondary task. It is part of the security design.

Policy baselines must also be maintained carefully. The model can measure drift only when the approved state is clear. Organizations should maintain an inventory of approved integrations, allowed endpoints, service account purposes, token scopes, object-level permissions, workflow requirements, external destinations, and expected data volumes. Without this baseline, the model may still detect statistical anomalies, but it will be less effective in identifying governance drift.

Operational deployment should begin with monitoring mode rather than automatic enforcement. In the early phase, the model should generate alerts, explanations, and risk scores for human review. Security, HR technology, integration, and compliance teams can then validate whether the alerts reflect true risks, approved exceptions, testing activities, or expected workload patterns. This review cycle can be used to tune thresholds, refine policy baselines, and improve feature calibration before the framework is connected to automated response processes.

8.7 Limitations

The study has several limitations. First, the experimental design relies on metadata-level integration events and controlled threat scenarios. Although this approach is suitable for preserving privacy and testing specific risk patterns, real enterprise environments may contain additional complexity, inconsistent logging, undocumented integration behavior, and rare events that are difficult to reproduce. Second, the quality of policy drift detection depends on the accuracy of the approved baseline. If the baseline is outdated, incomplete, or misaligned with actual business practice, the model may flag legitimate behavior or miss hidden governance issues.

Third, some threat classes naturally overlap. Token misuse, data leakage, workflow bypass, and abnormal integration chaining can occur within the same event window. This overlap makes class separation difficult and may require multi-label classification in future extensions. Fourth, federated learning improves privacy-preserving training, but it also introduces operational complexity. Client participation, model update quality, local data imbalance, and aggregation stability must be managed carefully. Fifth, the framework is designed as a security intelligence layer, not as a replacement for authentication, authorization, encryption, access reviews, or SIEM controls.

8.8 Future Research Directions

Future research can extend SENTINEL-HCM in several directions. One direction is adaptive policy baselining, where approved behavior is updated through controlled governance workflows rather than static configuration records. This would allow the framework to respond more effectively to organizational changes while still preserving audit control. Another direction is multi-label threat detection, where a single event window can be classified as both token misuse and data leakage, or both workflow bypass and policy drift. This would better reflect the way risks appear in complex integration environments.

Another promising direction is privacy-enhanced federated learning with stronger protection for model updates. Although the current framework avoids centralizing raw logs, future versions can further strengthen privacy through secure aggregation, differential privacy, or encrypted update mechanisms. Future work can also evaluate cross-tenant transfer learning, where learned security patterns from one integration landscape help improve detection in another without exposing tenant-specific data.

The framework can also be extended beyond SAP SuccessFactors into broader enterprise SaaS integration monitoring. Many cloud platforms share similar risk patterns involving APIs, tokens, service accounts, external destinations, workflow rules, and sensitive business objects. A generalized version of SENTINEL-HCM could support secure integration governance across HR, finance, procurement, identity, and workforce analytics systems. However, such extension would require domain-specific object sensitivity models and policy baselines for each platform.

Journal of Contemporary Science and Technology Management
Volume 01, Issue 01, December 2025

Overall, the findings show that federated temporal graph intelligence provides a strong foundation for securing SAP SuccessFactors cloud integrations. The approach is technically meaningful because it captures evolving relationships, operationally useful because it reduces false positives and explains alerts, and governance-relevant because it connects behavior with policy baselines. These qualities make SENTINEL-HCM suitable for enterprise environments where cloud integration security, employee data protection, and audit accountability must be managed together.

9. Conclusion

This paper presented SENTINEL-HCM, a federated temporal graph intelligence framework for securing SAP SuccessFactors cloud integrations against API abuse, token misuse, data leakage, privileged access drift, policy drift, workflow bypass, and abnormal integration chaining. The study addressed a practical gap in cloud HCM security: risky integration behavior often does not appear as a single unauthorized action, but as a connected pattern across service accounts, OAuth tokens, endpoints, middleware flows, HR data objects, workflow states, external destinations, and policy baselines. By representing these entities and relationships as a temporal security graph, SENTINEL-HCM moves beyond flat log inspection and evaluates integration risk as an evolving behavioral and governance problem.

The experimental results showed that SENTINEL-HCM achieved stronger overall detection performance than rule-based monitoring, classical anomaly detection, tree-based classification, sequence learning, static graph learning, and non-federated temporal graph baselines. The proposed framework reached an F1-score of 0.937, ROC-AUC of 0.978, PR-AUC of 0.961, and a false positive rate of 2.60%, demonstrating balanced performance across detection accuracy and operational usability. The results also showed that the largest improvement came from combining graph relationships, temporal behavior, token context, sensitive-object lineage, and policy drift signals. This confirms that SAP SuccessFactors integration risk is best understood through the interaction of multiple weak signals rather than through isolated event thresholds.

The ablation analysis further confirmed the importance of the framework design. Removing graph features produced the largest performance decline, showing that relationships among actors, tokens, endpoints, objects, workflows, and destinations are essential for detecting integration risk. Removing temporal features weakened the model's ability to identify slow-moving drift and abnormal event sequences. Removing token, policy, and sensitive-object features also reduced performance, especially for scenarios where behavior remained technically authorized but became risky through broader access, unusual object exposure, or deviation from the approved baseline. These findings support the central argument that cloud HCM security requires both behavioral intelligence and governance-aware interpretation.

From an enterprise perspective, SENTINEL-HCM provides value beyond model accuracy. Its federated design reduces the need to centralize sensitive HR event logs, making the framework more suitable for distributed organizations operating across business units, regions, tenants, and integration domains. Its explainability layer supports investigation by identifying the actor, token, endpoint, object, destination, workflow condition, and policy deviation behind each high-risk alert. This makes the framework useful for security operations, HR technology governance, integration monitoring, compliance review, and audit readiness.

The study also recognizes practical limitations. The effectiveness of SENTINEL-HCM depends on high-quality event normalization, stable identifiers, reliable policy baselines, and consistent logging across integration layers. Some threat classes, such as workflow bypass, token misuse, policy drift, and abnormal chaining, may overlap within the same monitoring window and may require multi-label modeling in future work. Further research can extend the framework with adaptive policy baselining, stronger privacy-preserving aggregation, cross-tenant transfer learning, and real-time integration with enterprise security operations platforms. Overall, the findings show that federated temporal graph intelligence offers a strong and practical foundation for securing SAP SuccessFactors cloud integrations in environments where sensitive employee data, automated service accounts, tokenized access, workflow controls, and external system dependencies must be governed continuously.

10 References

[1] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., & Zaharia, M. (2010). A view of cloud computing. Communications of the ACM, 53(4), 50–58. https://doi.org/10.1145/1721654.1721672

[2] Subashini, S., & Kavitha, V. (2011). A survey on security issues in service delivery models of cloud computing. Journal of Network and Computer Applications, 34(1), 1–11. https://doi.org/10.1016/j.jnca.2010.07.006

[3] Sommer, R., & Paxson, V. (2010). Outside the closed world: On using machine learning for network intrusion detection. IEEE Symposium on Security and Privacy, 305–316. https://doi.org/10.1109/SP.2010.25

[4] Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. ACM Computing Surveys, 41(3), 1–58. https://doi.org/10.1145/1541880.1541882

[5] Patcha, A., & Park, J. M. (2007). An overview of anomaly detection techniques: Existing solutions and latest technological trends. Computer Networks, 51(12), 3448–3470. https://doi.org/10.1016/j.comnet.2007.02.001

[6] Ahmed, M., Mahmood, A. N., & Hu, J. (2016). A survey of network anomaly detection techniques. Journal of Network and Computer Applications, 60, 19–31. https://doi.org/10.1016/j.jnca.2015.11.016

[7] Buczak, A. L., & Guven, E. (2016). A survey of data mining and machine learning methods for cyber security intrusion detection. IEEE Communications Surveys & Tutorials, 18(2), 1153–1176. https://doi.org/10.1109/COMST.2015.2494502

[8] Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J., & Williamson, R. C. (2001). Estimating the support of a high-dimensional distribution. Neural Computation, 13(7), 1443–1471. https://doi.org/10.1162/089976601750264965

[9] Liu, F. T., Ting, K. M., & Zhou, Z. H. (2008). Isolation forest. IEEE International Conference on Data Mining, 413–422. https://doi.org/10.1109/ICDM.2008.17

[10] Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785–794. https://doi.org/10.1145/2939672.2939785

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

[12] Akoglu, L., Tong, H., & Koutra, D. (2015). Graph-based anomaly detection and description: A survey. Data Mining and Knowledge Discovery, 29, 626–688. https://doi.org/10.1007/s10618-014-0365-y

[13] Perozzi, B., Al-Rfou, R., & Skiena, S. (2014). DeepWalk: Online learning of social representations. Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 701–710. https://doi.org/10.1145/2623330.2623732

[14] Grover, A., & Leskovec, J. (2016). node2vec: Scalable feature learning for networks. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 855–864. https://doi.org/10.1145/2939672.2939754

[15] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Yu, P. S. (2021). A comprehensive survey on graph neural networks. IEEE Transactions on Neural Networks and Learning Systems, 32(1), 4–24. https://doi.org/10.1109/TNNLS.2020.2978386

[16] Pareja, A., Domeniconi, G., Chen, J., Ma, T., Suzumura, T., Kanezashi, H., Kaler, T., Schardl, T. B., & Leiserson, C. E. (2020). EvolveGCN: Evolving graph convolutional networks for dynamic graphs. Proceedings of the AAAI Conference on Artificial Intelligence, 34(4), 5363–5370. https://doi.org/10.1609/aaai.v34i04.5984

[17] Skarding, J., Gabrys, B., & Musial, K. (2021). Foundations and modeling of dynamic networks using dynamic graph neural networks: A survey. IEEE Access, 9, 79143–79168. https://doi.org/10.1109/ACCESS.2021.3082932

[18] Yang, Q., Liu, Y., Chen, T., & Tong, Y. (2019). Federated machine learning: Concept and applications. ACM Transactions on Intelligent Systems and Technology, 10(2), 1–19. https://doi.org/10.1145/3298981

[19] Li, T., Sahu, A. K., Talwalkar, A., & Smith, V. (2020). Federated learning: Challenges, methods, and future directions. IEEE Signal Processing Magazine, 37(3), 50–60. https://doi.org/10.1109/MSP.2020.2975749

[20] Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal, A., & Seth, K. (2017). Practical secure aggregation for privacy-preserving machine learning. Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, 1175–1191. https://doi.org/10.1145/3133956.3133982

[21] Shokri, R., & Shmatikov, V. (2015). Privacy-preserving deep learning. Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, 1310–1321. https://doi.org/10.1145/2810103.2813687

[22] Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., & Zhang, L. (2016). Deep learning with differential privacy. Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, 308–318. https://doi.org/10.1145/2976749.2978318

[23] Hu, V. C., Kuhn, D. R., & Ferraiolo, D. F. (2015). Attribute-based access control. Computer, 48(2), 85–88. https://doi.org/10.1109/MC.2015.33

[24] Zissis, D., & Lekkas, D. (2012). Addressing cloud computing security issues. Future Generation Computer Systems, 28(3), 583–592. https://doi.org/10.1016/j.future.2010.12.006

[25] Takabi, H., Joshi, J. B. D., & Ahn, G.-J. (2010). Security and privacy challenges in cloud computing environments. IEEE Security & Privacy, 8(6), 24–31. https://doi.org/10.1109/MSP.2010.186

[26] Bertino, E., Bonatti, P. A., & Ferrari, E. (2001). TRBAC: A temporal role-based access control model. ACM Transactions on Information and System Security, 4(3), 191–233. https://doi.org/10.1145/501978.501979

[27] Mothukuri, V., Parizi, R. M., Pouriyeh, S., Huang, Y., Dehghantanha, A., & Srivastava, G. (2021). A survey on security and privacy of federated learning. Future Generation Computer Systems, 115, 619–640. https://doi.org/10.1016/j.future.2020.10.007

[28] Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why should I trust you?" Explaining the predictions of any classifier. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 1135–1144. https://doi.org/10.1145/2939672.2939778

[29] Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Pedreschi, D., & Giannotti, F. (2018). A survey of methods for explaining black box models. ACM Computing Surveys, 51(5), 1–42. https://doi.org/10.1145/3236009

[30] Molloy, I., Chen, H., Li, T., Wang, Q., Li, N., Bertino, E., Calo, S. B., & Lobo, J. (2010). Mining roles with multiple objectives. ACM Transactions on Information and System Security, 13(4), 1–35. https://doi.org/10.1145/1880022.1880030