

AI-Assisted Recovery Readiness for SaaS Workloads

A Framework for Anomaly Detection and Risk-Based Validation

SREEHARI PRASAD JINKA VENUGOPAL¹ (MAIL09HARI@GMAIL.COM)¹

¹ENTERPRISE PRODUCT MANAGEMENT, DATA PROTECTION, CYBER RESILIENCE, TRUSTWORTHY AI
GOVERNANCE PHOENIX, ARIZONA, USA.
ORCID: 0009-0002-5969-9357

Abstract

Software-as-a-Service (SaaS) platforms protect the availability of their own infrastructure, but customers remain responsible for the integrity and recoverability of the data and configuration they place in them. Mass deletion, ransomware-like encryption, malicious permission changes and silently failing backup jobs can all leave an organization unable to restore a workload within its recovery objectives, and many such conditions are discovered only when a restore is attempted. This paper proposes the AI-Assisted Recovery Readiness Framework (ARRF), which treats recovery readiness as a continuously measured property of each SaaS workload and not as a state assumed from the existence of a backup. ARRF combines three elements: an anomaly detection layer that learns from audit logs, key performance indicators and traces to flag incident precursors and degrading backup health; a readiness model that expresses each workload's state in plain categories; and a risk-based validation scheduler that decides which workloads receive restore tests and how often, within a fixed testing budget. A zero-trust decision and enforcement layer closes the loop by constraining risky activity while validation proceeds. We examine the design in two controlled simulations. In a benchmark of five detectors on synthetic SaaS telemetry (15,000 windows per run, 8% anomalous, ten runs), supervised gradient boosting and random forest classifiers reached F1-scores of 0.63 and 0.67, against 0.42 to 0.48 for isolation forest, one-class SVM and DBSCAN. In a 52-week scheduling simulation with 300 workloads and a weekly testing budget of 15%, risk-based validation reduced exposure on the most critical workloads by about 70% relative to round-robin testing, while keeping 92.5% of latent restore defects found within the year. These results are synthetic and show the logic of the design, not measurements from a production environment. The paper closes with governance guidance and a research agenda.

Keywords: Recovery Readiness, SaaS Resilience, Anomaly Detection, Risk-Based Validation, Zero Trust, Backup Verification, Machine Learning, Incident Response.

1. Introduction

Organizations now run a large share of their daily work in SaaS applications: collaboration suites, customer relationship management, human resources systems, ticketing platforms, code repositories and finance tools. The provider of such a service operates the infrastructure, replicates the data across facilities and publishes availability targets. What the provider does not do, in the usual division of responsibility, is protect the customer from the customer's own actions and from attackers who hold the customer's credentials. A user who deletes a library, an administrator who changes a sharing setting, a compromised service account that encrypts or overwrites content, and an integration that corrupts records are all, from the platform's perspective, legitimate operations that are faithfully replicated. Recovery from them is the customer's problem.

The usual answer is backup. Third-party and native backup services copy SaaS data and configuration to a separate store and offer point-in-time restore. Yet the existence of a backup job is a weak indicator of recovery readiness. Jobs may report success while skipping content because of changed permissions, throttling or newly added data sources. Retention in the

backup store may be shorter than the time needed to discover a problem. Restore procedures may never have been exercised for a given workload, so that their duration is unknown. Credentials needed for restore may be tied to the same identity provider that an attacker has compromised. The practical consequence is that an organization often learns the true state of its recovery capability during an incident, which is the worst time to learn it.

Two bodies of practice address parts of this problem. Contingency planning and incident response frameworks require organizations to define recovery objectives and to test their plans, but testing is typically scheduled by calendar, covers a small sample of systems and is expensive in staff time. Security analytics, increasingly based on machine learning, can detect unusual behavior in logs and telemetry, but its output is usually consumed as an alert for the security team and not as an input to resilience planning. The two rarely meet: the signal that something unusual is happening to a workload does not automatically change how closely its recoverability is checked.

This paper proposes to connect them. The central idea is that anomaly detection should serve two purposes at once, to raise alerts about incidents and to adjust confidence in the recovery readiness of the workloads concerned, and that restore validation should be allocated by risk instead of by rotation. The primary objective is to design and examine the AI-Assisted Recovery Readiness Framework (ARRF). The specific tasks are: (i) to define recovery readiness for SaaS workloads in measurable terms; (ii) to specify a telemetry and anomaly detection pipeline suited to heterogeneous SaaS environments; (iii) to define a readiness model and a risk-based scheduler for restore validation; (iv) to integrate both with a zero-trust decision and enforcement layer and with governance controls; and (v) to examine the resulting trade-offs in controlled simulations. The contribution is a coherent, testable blueprint, together with explicit statements about what the evidence does and does not show.

The remainder of the paper is organized as follows. Section II reviews the literature. Section III summarizes related work. Section IV presents the ARRF methodology. Section V details the experimental setup, Section VI reports results, Section VII discusses implications, Section VIII addresses limitations, and Section IX concludes.

2. LITERATURE REVIEW

A. Recovery readiness and the SaaS responsibility boundary

Recovery planning has a long history in information security and operations. The NIST contingency planning guide describes how organizations should identify critical functions, set recovery objectives, develop recovery strategies and test, train and exercise their plans [1]. The NIST Cybersecurity Framework treats recovery as one of its core functions alongside identification, protection, detection and response, and stresses that recovery plans must be executed and verified, not merely documented [2]. Incident handling guidance describes the lifecycle from preparation through detection, containment, eradication and recovery, and the need to learn from each event [3]. SaaS changes the boundary of responsibility. The provider keeps the service running and protects against infrastructure failure, but data loss caused by users, administrators, integrations and attackers sits on the customer side. Ransomware research shows that modern attacks combine credential theft, privilege escalation and bulk modification of data, which is exactly the type of event that platform replication does not undo [4]. Recovery readiness for a SaaS workload is therefore a property that the customer must establish and maintain. In this paper we define it as the demonstrated ability to restore a workload's data and configuration to a usable state within its recovery time and recovery point objectives, given the failure and attack conditions that are plausible for that workload. The word "demonstrated" is deliberate: readiness that has not been tested is an assumption.

B. Anomaly detection for operational telemetry

Anomaly detection is a mature field. Chandola and colleagues survey techniques and note that the right approach depends on the availability of labels, the nature of the anomalies and the structure of the data [5]. Ahmed and colleagues review network anomaly detection and its challenges, including class imbalance and evolving behavior [6]. Unsupervised methods are attractive because labelled incidents are rare. Isolation forest isolates points by random partitioning and treats those that are easy to isolate as anomalous [7]. The one-class support vector machine learns a boundary around

normal data [8]. Density-based clustering such as DBSCAN labels points in sparse regions as noise [9]. Supervised methods, such as random forests and gradient boosting, require labelled examples but often perform strongly when labels exist [10], [11].

The choice among them is not purely technical. Unsupervised methods can detect novel behavior but tend to produce many false alarms when normal behavior is heterogeneous, as it is across SaaS workloads with very different volumes and rhythms. Supervised methods are more precise on known patterns but may miss novel attacks and require labels from past incidents, exercises and red-team activity. Concept drift, in which normal behavior changes over time, affects both families and must be managed through retraining and monitoring [12]. Evaluation is also subtle: with rare anomalies, accuracy is a poor measure, since a detector that labels everything normal can score highly, and precision, recall and the precision-recall relationship are more informative [13].

C. Zero trust and automated response

Zero trust architecture, as described by NIST, removes implicit trust based on network location and requires that access to resources be decided per request on the basis of identity, device, context and policy, with continuous evaluation [14]. In a multi-cloud and SaaS environment this implies a policy decision component that consumes signals from identity providers and from detection systems, and an enforcement component that applies decisions such as step-up authentication, session termination, restriction of sensitive operations or micro-segmentation. Learning-based detection can supply risk signals to the policy engine, which can act on them in near real time, while orchestration platforms can automate the response and feed outcomes back to the detector. Several lines of research extend the detection layer with federated learning, which trains models across tenants or sites without centralizing raw data [15], and graph-based models, which exploit the relationships between users, resources and events [16].

D. Validation, resilience engineering and risk-based allocation

Site reliability engineering and chaos engineering argue that resilience claims should be tested in practice [17], [18]. Restore testing is the analogue for data protection: it exposes problems that monitoring of backup jobs cannot, such as missing content, incompatible configurations, expired credentials or unrealistically long restore times. The difficulty is cost. A full restore test of every workload every week is unaffordable, so organizations sample, and the sampling policy matters. A rotation guarantees coverage but treats every workload alike. A policy driven only by business criticality concentrates effort on important workloads but can starve the rest. Risk-based allocation attempts to direct effort toward workloads where a defect is both more likely and more costly, which requires reliable risk signals and safeguards against neglect. Machine learning systems also carry operational risks of their own, such as hidden dependencies and maintenance debt, which a deployment must plan for [19].

3. RELATED WORK

Prior work approaches the problem from several directions, and few contributions connect them.

In security analytics, a large literature applies machine learning to log and network data for intrusion and anomaly detection [5], [6]. These studies emphasize detection quality and rarely consider how detection should inform recovery planning. The typical output is an alert, and what happens to the organization's confidence in its recovery capability after the alert is outside scope.

In contingency planning and resilience, standards and guidance describe objectives, strategies and testing [1], [2], [3]. They are clear about what to do and less specific about how to prioritize testing across many workloads, particularly when the set of workloads is large and changes quickly, as it does in SaaS estates where teams create new sites, tenants, projects and integrations continuously.

In zero trust research, architectures define policy decision and enforcement points and the use of context signals [14], and research on learning-based detection proposes models that feed such systems [15], [16]. These works address access control and response, and they typically treat recovery as a downstream function.

In operations research and reliability engineering, risk-based inspection and testing strategies direct limited effort to items with higher failure likelihood and consequence. The principle is well established, but its application to restore validation for SaaS data, with signals drawn from

audit logs and backup telemetry, has received little attention.

The gap that ARRF addresses is therefore integration. No widely adopted approach (a) defines recovery readiness in measurable terms for SaaS workloads, (b) uses anomaly detection to update that measure, (c) allocates restore validation by risk under a fixed budget with safeguards against neglect, and (d) links the result to zero-trust enforcement and governance.

4. METHODOLOGY: THE DPVF FRAMEWORK

A. Architecture overview

ARRF has five layers. Layer 1 (telemetry) collects audit logs, key performance indicators and traces from the SaaS applications, the identity provider and the backup service. Layer 2 (detection) applies machine learning models to these data to flag anomalies. Layer 3 (readiness) combines detection output with backup health and workload attributes to assign each workload a readiness state. Layer 4 (validation) schedules restore tests by risk. Layer 5 (response and governance) applies zero-trust policy decisions, orchestrates responses and records outcomes that feed back to the earlier layers.

Figure 1 shows the detection pipeline that underlies Layers 1 and 2. Logs from multiple sources, key performance indicators and traces feed an AI model, whose output goes to an anomaly detector that produces threat detections and alerts. Figure 2 shows the broader architecture in which this layer sits: multi-cloud and SaaS data sources feed an AI-based anomaly detection layer, whose output goes to a zero-trust policy and decision engine that also receives context from federated identity providers, and decisions are carried out by an enforcement and orchestration layer built on SIEM and SOAR tooling, with a feedback path from enforcement outcomes back to the detection layer.

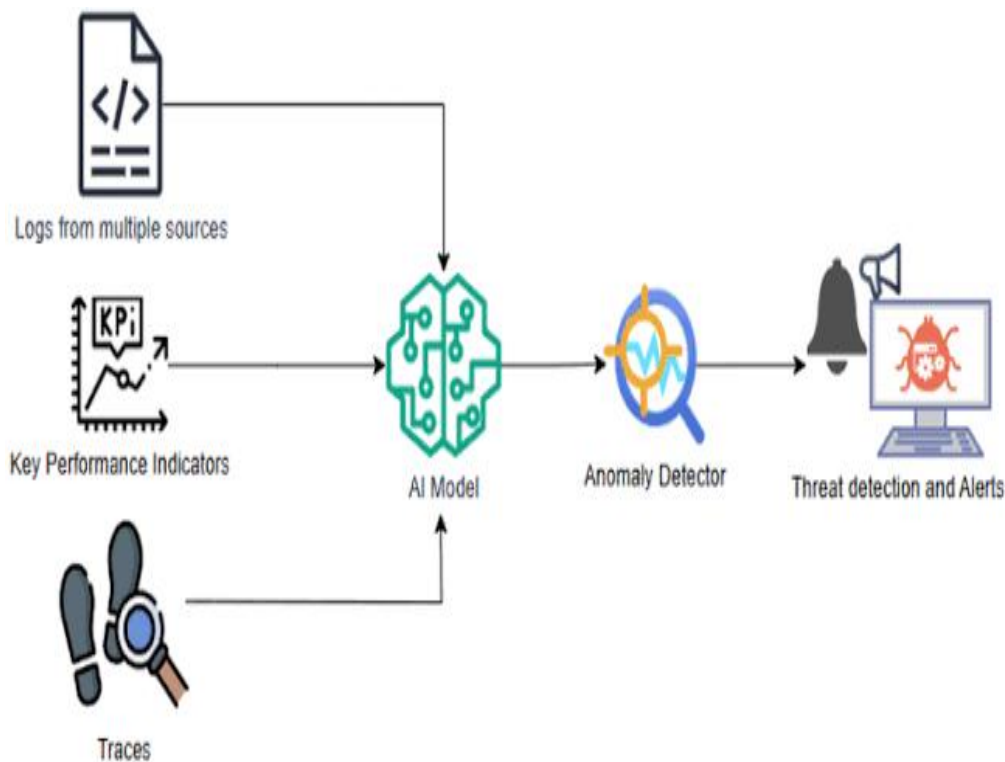


Figure 1. Detection pipeline: logs from multiple sources, key performance indicators and traces feed an AI model and an anomaly detector that produces threat detection and alerts.

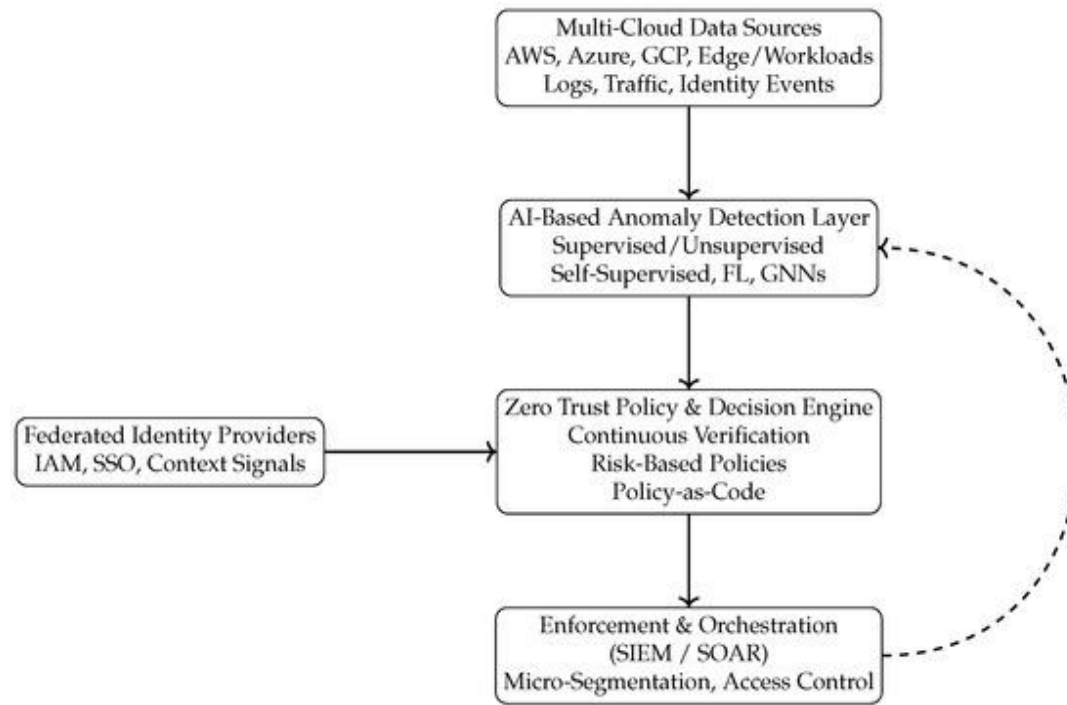


Figure 2. Zero-trust architecture for multi-cloud and SaaS environments: data sources, AI-based anomaly detection, policy and decision engine with federated identity context, and enforcement and orchestration with a feedback loop.

B. Telemetry and feature design

Three kinds of signal are used. Audit logs record who did what to which object: deletions, uploads, downloads, permission and sharing changes, sign-in events and administrative actions. Key performance indicators summarize service behavior and backup health: API error rates, latency, backup job success and failure, and the age of the most recent successful backup. Traces link operations across services and reveal unusual call patterns, such as an integration touching many objects in a short time. Table 1 lists the signals and the features derived from them.

TABLE 1. Telemetry Signals and Derived Features (per workload and hour)

Signal source	Features	What they indicate
Audit logs	Delete count, upload count, rename rate, download volume	Mass deletion, bulk modification, exfiltration
Audit logs	Permission changes, external shares	Privilege abuse, oversharing
Identity events	Failed sign-ins, new locations or addresses	Credential attack, account takeover
Traces	API call volume	Automated or integration-driven activity
Key performance indicators	Error rate, latency	Service degradation, throttling
Backup service	Age of last good backup, job failures	Degrading recovery readiness

Features are computed over fixed windows, here one hour, per workload. Because workloads differ greatly in volume and rhythm, features are transformed to dampen heavy tails and compared against the workload's own history as well as against the population. Without this step a busy workload would appear anomalous simply because it is busy.

C. Detection layer

The detection layer can use supervised and unsupervised methods, and ARRF recommends combining them. Supervised models, trained on labelled windows from past incidents, tabletop exercises, red-team activity and synthetic attack injection, provide high precision on known patterns. Unsupervised models, trained on normal behavior, provide coverage of novel behavior at the cost of more false alarms and are best used to flag windows for review or to raise the validation priority of a workload, not to trigger disruptive enforcement on their own. Federated training and graph-based models can be added where data cannot be pooled or where relationships among entities matter [15], [16].

Several operational safeguards apply. Models are retrained on a schedule and when drift indicators cross a threshold [12]. Evaluation uses precision, recall and F1 and not accuracy, because anomalies are rare [13]. Every alert carries an explanation of the features that drove it, so that analysts can judge it. Analyst feedback on alerts becomes new labels.

D. Readiness model

Each workload is assigned a readiness state, expressed in plain categories so that business owners and auditors can understand them. Table 2 defines the states and what they imply.

TABLE 2. Recovery Readiness States

State	Meaning	Typical evidence	Consequence
Verified	A restore test succeeded recently and meets objectives	Recent successful restore, healthy backups, no anomalies	Normal validation cadence
Assumed	Backups run but no recent restore test	Healthy job reports, test overdue	Schedule validation
At risk	Anomalies or backup degradation detected	Anomaly signals, failed or stale backups	Raise validation priority; notify owner
Compromised	Evidence of active incident affecting the workload	Confirmed alert, mass change	Contain, restore from clean point, validate after
Failed	A restore test failed or missed objectives	Failed restore, missing content	Remediate, retest, escalate

Transitions between states are driven by evidence. A successful restore test moves a workload to verified, and the state decays to assumed as the test ages. An anomaly or backup degradation moves a workload to at risk. A confirmed incident moves it to compromised, and a failed test to failed. The decay rule is important: readiness is perishable, since both the workload and the environment change.

E. Risk-based validation scheduling

Restore validation is performed within a fixed budget, for example a fixed fraction of workloads per week. The scheduler ranks workloads and tests the highest-ranked within the budget. The ranking combines four ingredients, each of which is described in words and set by the organization.

The first is business criticality, drawn from the business impact assessment and reflected in the recovery objectives. The second is backup health, from job success, backup age and coverage checks. The third is change velocity, the rate at which the workload's content, configuration and integrations change, since fast-changing workloads are more likely to drift away from what the backup captures. The fourth is the anomaly signal from the detection layer, which indicates whether recent behavior has been unusual. A workload with high criticality, weak backup health, rapid change and recent anomalies is tested first.

Two safeguards prevent the scheduler from neglecting parts of the estate. An ageing

guarantee ensures that no workload goes untested longer than a maximum interval, regardless of its risk score. A minimum coverage rule reserves a small share of the budget for random selection, which provides an unbiased sample for estimating the true defect rate and for detecting errors in the risk model itself. Tests are scoped to the workload’s objectives: a test should restore representative content and configuration, check integrity and measure restore duration against the recovery time objective.

F. F. Zero-trust response and orchestration

The policy and decision engine in Figure 2 uses risk signals from the detection layer and context from federated identity providers to make decisions that vary with the readiness state. For a workload that is at risk, the engine can require stronger authentication for destructive operations, rate-limit bulk deletion, restrict external sharing, or require approval for permission changes. For a compromised workload, it can suspend sessions and revoke tokens. Policies are expressed as code, version-controlled and tested, so that changes are reviewable. Enforcement and orchestration tools execute decisions, open tickets, notify owners and trigger additional validation. The feedback path then reports whether the response was effective and whether the alert was true, which improves the models.

Two cautions apply. Automated enforcement can itself cause disruption, for example by blocking legitimate bulk operations during a migration, so enforcement based on unsupervised signals should be limited to reversible, low-impact actions. And the recovery path must not depend on the same compromised identities as the production path: restore credentials, backup administration and approval chains need separate identity and privilege boundaries.

G. Governance and operating model

ARRF assigns roles. Workload owners confirm criticality, objectives and exemptions. A resilience owner operates backups, validation and the scheduler. A security operations owner operates detection and response. A risk and audit owner reviews evidence and reports to management. A readiness board meets regularly to review readiness states, failed tests, model performance and exceptions.

We recommend tracking a small set of indicators: the share of workloads in each readiness state, the age of the last successful restore per class of workload, the time between a defect arising and being found, restore duration against objectives, detector precision and recall, alert volume per analyst, and the number of policy exceptions. These indicators turn readiness from an assertion into evidence that can be audited..

H. Asset valuation

Asset valuation determines what an incident would cost. DPVF values data assets along four lines. Direct recovery cost includes the labor and technology needed to restore data and systems. Business interruption covers revenue lost and costs incurred while processes are unavailable. Liability and regulatory exposure covers fines, notification, legal costs and contractual penalties. Intangible loss covers reputation and customer attrition, which are uncertain but are often the largest component in severe cases. Each asset also receives an importance level reflecting its role in critical operations. Valuation is expressed as ranges, not point values, and the ranges are refined as incident data accumulate. Table 1 lists the cost and value categories used in the framework.

5. EXPERIMENTAL SETUP

The experiments are limited in purpose. Production SaaS environments cannot safely be used to compare design choices, labelled incident data are scarce and sensitive, and the true state of recovery readiness is unobserved until a restore is attempted. We therefore used two synthetic experiments, implemented in Python with fixed random seeds and standard machine learning libraries. The first examines detection quality. The second examines the effect of validation scheduling on how long restore defects remain undiscovered.

A. Detection benchmark

Synthetic telemetry. Each run simulates 60 workloads over 250 hours, giving 15,000 workload-hour windows. Each window has 13 features corresponding to Table 1. Normal behavior is generated from workload-specific baselines, with workloads differing in scale by

large factors, a daily rhythm and random variation. Eight percent of windows are anomalous. Each anomalous window belongs to one of five types: mass deletion, bulk download and external sharing consistent with exfiltration, abnormal permission changes, ransomware-like bulk renaming with elevated errors, and backup degradation with ageing backups and job failures. The intensity of each anomaly varies, so that some are subtle and overlap with normal variation.

Detectors. Five detectors are compared: gradient boosting classifier (GBC), random forest (RF), isolation forest (IF), one-class support vector machine (OC-SVM) and DBSCAN. The supervised models are trained on 70% of windows with labels and tested on the remaining 30%. Isolation forest is trained without labels and given the approximate anomaly fraction as its contamination setting. The one-class SVM is trained on normal windows only. DBSCAN is applied to the test windows, with its neighborhood radius set from the distribution of nearest-neighbor distances, and points labelled as noise are treated as anomalies. The unsupervised methods therefore receive more favorable settings than they would in practice, where the anomaly fraction is unknown.

Metrics. We report precision, recall, F1 (the balance of the two) and accuracy, averaged over ten independent runs with different seeds. Because eight percent of windows are anomalous, a detector that labels everything normal would obtain an accuracy of about 92%, and accuracy should be read with that baseline in mind.

B. Validation scheduling simulation

Synthetic estate. Each replication simulates 300 workloads over 52 weeks. Workloads have a business criticality of low, medium or high (50%, 30% and 20% of the estate), a change velocity drawn from a heavy-tailed distribution, and a backup health level. A latent restore defect, such as content that the backup silently does not capture, can arise in any workload in any week, with a probability that rises with change velocity and with poor backup health. A defect persists until a restore test of that workload finds it, and a test detects an existing defect with a probability of 95%. The expected number of defects is about 93 per replication.

Budget and strategies. In every week the organization can test 15% of workloads (45 of 300). Three strategies are compared. Round-robin tests workloads in rotation, so that every workload is tested about every seven weeks. Criticality-weighted ranks workloads by criticality, with a preference that grows as time since the last test increases. Risk-based ranks workloads by a combination of criticality, backup health, change velocity and an anomaly signal, with an ageing guarantee of 13 weeks. The risk-based scheduler sees only noisy proxies of the true change velocity and backup health, and its anomaly signal is only imperfectly related to the presence of a defect. The results average 300 replications.

Metrics. We report the share of defects found within the year, the share found within four weeks of arising, the number of weeks that defects remained open per defect (counting defects still unresolved at year-end), the total number of weeks during which a critical workload carried an unresolved defect, and the share of defects still unresolved at year-end.

All assumptions, including the anomaly intensities, the defect hazard and the detection probability of a test, are inputs of the simulations and not measurements.

6. RESULTS

A. Detection quality

Table 3 and Figure 3 report the benchmark. The two supervised models performed best on F1. Random forest reached an F1-score of 0.67 and gradient boosting 0.63, with precision of 0.92 and 0.88 and recall of 0.53 and 0.50. The unsupervised methods had lower F1: 0.44 for isolation forest, 0.48 for the one-class SVM and 0.42 for DBSCAN. Their profiles differ. The one-class SVM had the highest recall of all methods (0.63) but low precision (0.38), which means that most of its alerts were false. DBSCAN had very high precision (0.95) but the lowest recall (0.27): the points it labelled as noise were nearly always anomalous, but it found only about a quarter of the anomalies. Isolation forest was weak on both counts, with precision and recall of about 0.44.

TABLE 3. Simulated Detection Performance (mean of 10 runs; 4,500 test windows per run; 8% anomalous)

Detector	Precision	Recall	F1-score	Accuracy
GBC (supervised)	0.877	0.495	0.632	0.955
RF (supervised)	0.919	0.530	0.672	0.960
IF (unsupervised)	0.439	0.440	0.439	0.913
OC-SVM (trained on normal data)	0.384	0.633	0.478	0.892
DBSCAN (unsupervised)	0.946	0.273	0.423	0.942

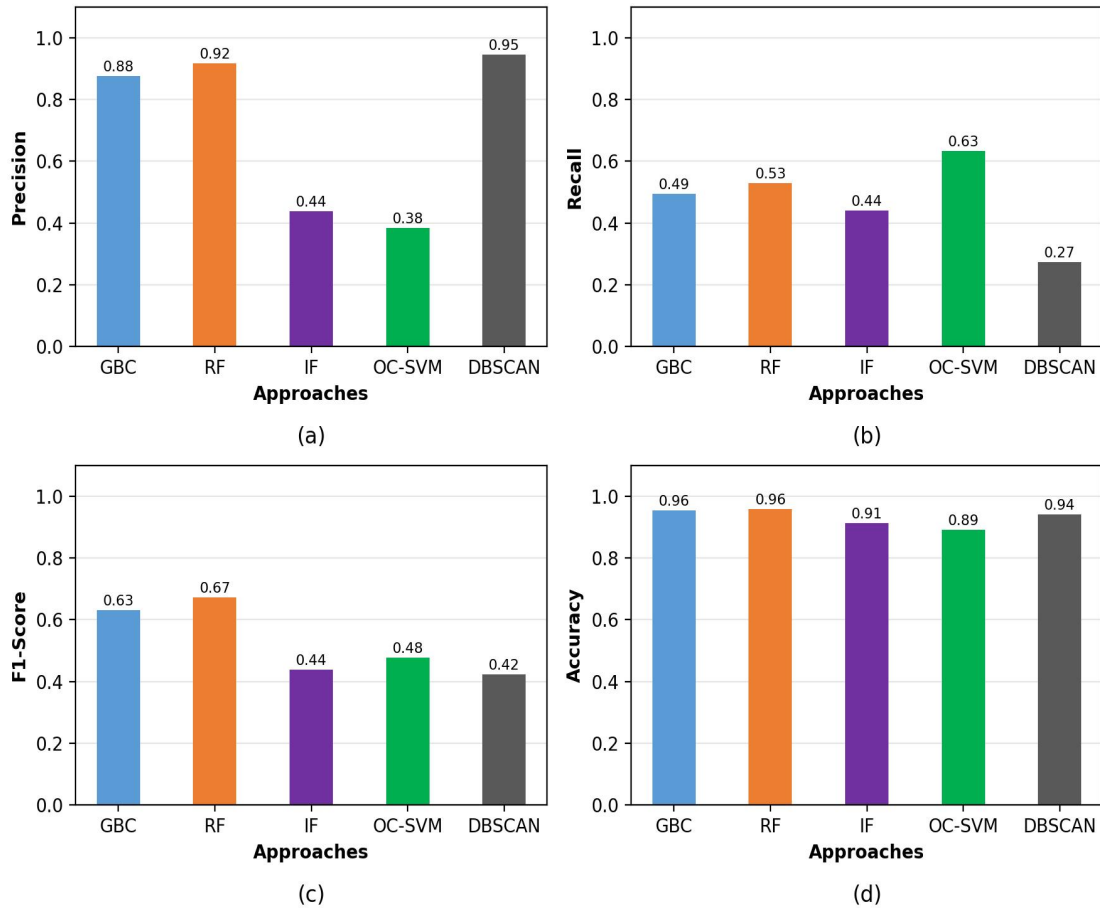


Figure 3. Precision, recall, F1-score and accuracy of the five detectors on synthetic SaaS telemetry: (a) precision, (b) recall, (c) F1-score, (d) accuracy.

Accuracy is high for every detector, between 0.89 and 0.96, but it is dominated by the large normal class: the all-normal baseline is about 0.92, and isolation forest and the one-class SVM scored below it. This illustrates why accuracy should not be used to compare detectors on rare events. The most important observation is the recall of the best models, about 0.5 to 0.53. Roughly half of the anomalies, mostly the low-intensity ones, were not detected even by the supervised models. In a deployment, this argues against treating the detector as the only line of defense, and for the independent validation layer that the framework proposes.

B. Validation scheduling

Table 4 reports the scheduling simulation. Round-robin testing found 95.4% of the defects within the year and left 4.6% unresolved, with a mean of 3.1 open weeks per defect. It treated all workloads alike, however, and critical workloads carried unresolved defects for 59.1 workload-

weeks per year in total. Criticality-weighted testing reduced this critical exposure to 8.6 workload-weeks, the lowest of the three, but at a large cost: it found only 45.4% of all defects, left 54.6% unresolved at year-end and averaged 16.7 open weeks per defect, because the low-criticality workloads were rarely tested.

TABLE 4. Simulated Validation Scheduling (300 workloads, 52 weeks, 15% weekly budget, 300 replications)

Strategy	Defects found within the year (%)	Found within 4 weeks (%)	Open weeks per defect	Critical-workload open weeks per year	Unresolved at year-end (%)
Round-robin	95.4	69.3	3.10	59.1	4.6
Criticality-weighted	45.4	28.3	16.70	8.6	54.6
Risk-based (ARRF)	92.5	60.9	3.87	17.7	7.5

The risk-based strategy occupied a middle position that most organizations would find attractive. It found 92.5% of defects within the year, close to the 95.4% of round-robin, and averaged 3.9 open weeks per defect. It cut critical-workload exposure to 17.7 workload-weeks, about 70% below round-robin, but did not reach the 8.6 of the criticality-only strategy. In other words, the strategy gave up about three percentage points of overall coverage and about 0.8 open weeks per defect to reduce the exposure of the most important workloads by roughly seventy percent. It did not dominate round-robin on every measure: round-robin was slightly better on total coverage and on the share of defects found within four weeks.

C. Summary of findings

Three findings stand out. First, supervised detectors outperformed unsupervised detectors on F1, but even the best missed about half of the anomalies, so detection cannot substitute for validation. Second, the unsupervised detectors had very different error profiles, which supports the proposal to use them for prioritization and review and not for disruptive enforcement. Third, no scheduling strategy was best on every measure. Rotation maximizes coverage, criticality weighting minimizes exposure of the most important workloads but abandons the rest, and risk-based scheduling balances the two, provided that the ageing guarantee and the random coverage reserve are in place.

7. DISCUSSION

The simulations support the central argument of the framework: recovery readiness is something to be measured, and detection and validation are more useful together than apart. The detection benchmark shows that anomaly detection on heterogeneous SaaS telemetry is hard, and that its output is best treated as evidence about risk rather than as a verdict. A recall of about one half means that a quiet attack could pass without alert. Restore validation addresses a different question, namely whether the workload can be recovered if something has gone wrong, and it does so independently of whether anything was detected.

The scheduling results show why allocation matters. The common practice of testing a few critical systems carefully and the rest rarely corresponds to the criticality-weighted strategy, which produced the lowest exposure on critical workloads and the worst overall coverage, with more than half of the defects still unresolved at the end of the year. Rotation avoids this but exposes critical workloads for longer. A risk-based approach, using several signals and safeguards, achieves most of the benefit of both. The scheduler's value depends on the quality of its signals: in the simulation it saw only noisy proxies, and its advantage over rotation on critical workloads would shrink if those proxies were poor and grow if they were better.

For practitioners, four recommendations follow. Define readiness in terms that business owners understand, and report the share of workloads in each state, including the large number that are only assumed to be recoverable. Use anomaly detection both to alert and to reprioritize validation, but limit automated enforcement based on unsupervised signals to reversible actions. Protect the validation budget with an ageing guarantee and a random reserve, so that the

scheduler cannot silently neglect part of the estate. And separate the recovery path from the production path in terms of identity, privilege and credentials, because a recovery plan that depends on the compromised environment may not work when it is most needed.

The framework also has implications for compliance and assurance. Regulators and auditors increasingly ask not only whether backups exist but whether recovery has been demonstrated. A record of restore tests, their results and the reasoning behind their scheduling provides evidence that can be audited. Where regulation concerns operational resilience of critical services, the readiness states and indicators in ARRF offer a direct way to report on it.

Finally, the framework is deliberately conservative about autonomy. Detection and scheduling models inform decisions, and disruptive actions, exemptions and changes to objectives remain human decisions with logged rationale. A model that quietly lowered the validation frequency for a workload because its recent behavior looked normal would be a failure by the framework's own standards, which is why the ageing guarantee is a rule and not a model output.

8. LIMITATIONS AND FUTURE WORK

The principal limitation is that both experiments are synthetic. The telemetry generator encodes assumptions about what normal and anomalous behavior look like, and real SaaS environments contain more types of activity, more irregular rhythms, and attackers who adapt to avoid detection. The absolute performance figures should not be expected to hold in practice, and the relative ordering of the detectors may differ with real data. The scheduling simulation assumes a particular relation between change velocity, backup health and defects, a fixed test detection probability and an independent arrival of defects, whereas real defects cluster and are correlated across workloads that share configuration or integrations. The accuracy of the risk signals was assumed and not varied in a sensitivity analysis.

A second limitation concerns the benchmark design. The unsupervised detectors were given favorable settings, such as knowledge of the anomaly fraction, and the supervised models were trained and tested on windows from the same workloads and generator, which flatters them relative to a deployment in which new workloads and new attack types appear. Temporal splits, cross-workload evaluation and evaluation on novel attack types would give a harsher and more realistic picture.

Third, ARRF as presented does not model the cost of restore tests or of false alarms, and it treats the effect of enforcement on user productivity qualitatively. A deployment would need to quantify these costs.

Future work should therefore include: (i) evaluation on real telemetry from production SaaS environments with prospective measurement of detection, restore success and exposure; (ii) sensitivity analyses over the quality of risk signals, defect correlation and test effectiveness; (iii) temporal and cross-workload evaluation of detectors, including adversarial and novel attacks; (iv) study of federated and graph-based detection for multi-tenant and multi-cloud estates; and (v) cost-aware formulations of the validation scheduler that trade test effort against expected loss.

9. CONCLUSION

This paper presented ARRF, a framework that treats recovery readiness for SaaS workloads as a continuously measured property. It combines an anomaly detection layer built on logs, key performance indicators and traces; a readiness model with plain states that decay without fresh evidence; a risk-based scheduler that allocates restore validation within a fixed budget with safeguards against neglect; and a zero-trust decision and enforcement layer with governance and feedback. In controlled simulations, supervised detectors outperformed unsupervised ones but still missed about half of the anomalies, and risk-based validation reduced exposure on the most critical workloads by about 70% relative to rotation while finding 92.5% of latent restore defects within the year.

The simulations are evidence of coherence and not of field effectiveness. The next step is empirical: pilot the framework in a production SaaS estate, measure detection, restore success and exposure against current practice, and refine the risk signals. If those results are consistent with the logic shown here, organizations can replace the assumption that backups imply recoverability with evidence that recovery has been demonstrated, and can direct limited validation effort to where it matters most.

References

1. M. Swanson, P. Bowen, A. W. Phillips, D. Gallup, and D. Lynes, Contingency Planning Guide for Federal Information Systems, NIST Special Publication 800-34, Revision 1, National Institute of Standards and Technology, 2010.
2. National Institute of Standards and Technology, The NIST Cybersecurity Framework (CSF) 2.0, NIST CSWP 29, 2024.
3. P. Cichonski, T. Millar, T. Grance, and K. Scarfone, Computer Security Incident Handling Guide, NIST Special Publication 800-61, Revision 2, National Institute of Standards and Technology, 2012.
4. A. Kharraz, W. Robertson, D. Balzarotti, L. Bilge, and E. Kirda, "Cutting the Gordian knot: A look under the hood of ransomware attacks," in Proc. Int. Conf. Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA), 2015, pp. 3–24.
5. V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," ACM Computing Surveys, vol. 41, no. 3, art. 15, 2009.
6. M. Ahmed, A. N. Mahmood, and J. Hu, "A survey of network anomaly detection techniques," Journal of Network and Computer Applications, vol. 60, pp. 19–31, 2016.
7. F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in Proc. 8th IEEE Int. Conf. Data Mining (ICDM), 2008, pp. 413–422.
8. B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," Neural Computation, vol. 13, no. 7, pp. 1443–1471, 2001.
9. M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in Proc. 2nd Int. Conf. Knowledge Discovery and Data Mining (KDD), 1996, pp. 226–231.
10. L. Breiman, "Random forests," Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.
11. J. H. Friedman, "Greedy function approximation: A gradient boosting machine," Annals of Statistics, vol. 29, no. 5, pp. 1189–1232, 2001.
12. J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," ACM Computing Surveys, vol. 46, no. 4, art. 44, 2014.
13. T. Saito and M. Rehmsmeier, "The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets," PLoS ONE, vol. 10, no. 3, e0118432, 2015.
14. S. Rose, O. Borchert, S. Mitchell, and S. Connelly, Zero Trust Architecture, NIST Special Publication 800-207, National Institute of Standards and Technology, 2020.
15. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in Proc. 20th Int. Conf. Artificial Intelligence and Statistics (AISTATS), 2017, pp. 1273–1282.
16. T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in Proc. Int. Conf. Learning Representations (ICLR), 2017.
17. B. Beyer, C. Jones, J. Petoff, and N. R. Murphy, Eds., Site Reliability Engineering: How Google Runs Production Systems. Sebastopol, CA, USA: O'Reilly Media, 2016.
18. A. Basiri et al., "Chaos engineering," IEEE Software, vol. 33, no. 3, pp. 35–41, 2016.
19. D. Sculley et al., "Hidden technical debt in machine learning systems," in Advances in Neural Information Processing Systems 28, 2015.