Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Improving Cloud Infrastructure Resilience Through Intelligent Threat Detection and Automated Risk Assessment

Rehana Devani

rehana.devani@gmail.com

Google (United States) https://orcid.org/0009-0003-8231-3448

Research Article

Keywords: Cloud security, cyber resilience, threat detection, risk assessment, incident response queues, service recovery

Posted Date: November 20th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11791539/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: The authors declare no competing interests.

Improving Cloud Infrastructure Resilience Through Intelligent Threat Detection and Automated Risk Assessment

Rehana Devani

Master's in Computing Science from The University of Texas at Arlington.

Abstract

An accurate cloud threat detector can still leave critical services exposed when a burst of alerts exceeds response capacity. This study examines whether automated risk assessment improves recovery by prioritising queued alerts according to estimated incident probability, service impact and response effort. A logistic detector trained on 60,000 synthetic cases was coupled to a non-preemptive response queue and evaluated across 48 Monte Carlo replicates of 3,600 candidates per workload. Three policies used the same case information and processing capacity: arrival order, incident probability and expected service impact per minute of work. Under nominal evidence and a burst with four responders, risk ranking produced 51.07 impact-weighted minutes per incident compared with 54.06 for probability ranking. With degraded telemetry, the corresponding means were 145.17 and 190.92. In the degraded scenario, the paired difference in weighted loss was −45.76 minutes (95% Monte Carlo interval −58.52 to −32.99), while the difference in the share restored within 60 minutes was −0.11 percentage points (−0.51 to 0.28). Under nominal evidence, adding eight minutes before alert release reduced 60-minute attainment for risk ranking from approximately 98.4% to 88.5%. The simulation identifies conditions under which prioritisation changes recovery outcomes and shows why those outcomes depend on evidence latency and the metric selected. The results describe modelled workloads and require confirmation with operational data.

Keywords. Cloud security, cyber resilience, threat detection, risk assessment, incident response queues, service recovery

1 Introduction

Cloud security operations frequently record the time an alert was raised and the time an analyst closed it. From a service owner's perspective, a different clock matters. It begins when a harmful event affects a workload and ends when a trusted level of functionality is restored. Detection, triage, containment and verification each occupy part of that interval. A detector can improve its ability to distinguish incidents from benign activity while the service still recovers slowly because evidence arrives late, responders are occupied or restoration depends on a separate team. The practical question is how much recovery performance changes when threat evidence is combined with service impact and expected response effort.

Cyber-resiliency engineering includes designing systems that can withstand and recover from adverse conditions [28]. Incident response and recovery guidance similarly places preparation, prioritisation, restoration and learning within the same operational cycle [6, 23, 25]. These sources motivate a distinction between predictive quality and response value. A high estimated incident probability can justify immediate attention, while a somewhat lower probability attached to a critical and quickly recoverable workload could represent a greater avoidable service loss. Recovery may also take longer when delayed containment allows changes to spread or increases the amount of state that must be checked.

This paper studies the alert-to-recovery interval through a reproducible discrete-event simulation. The experiment uses a fitted detector and compares scheduling rules under identical candidate streams and capacity. It addresses three questions. First, does risk ranking reduce service impact compared with arrival order and probability ranking? Second, how do evidence release delays and changes in detector inputs affect these comparisons? Third, how does responder capacity change waiting for benign cases? The outcome measures capture several operational objectives, so an apparent improvement on one measure can be read against the others. The simulation isolates mechanisms that are hard to compare in ordinary incident logs, where staffing and prioritisation often change together during a major incident.

2 Background and analytical scope

2.1 Resilience as a measurable recovery outcome

Service reliability practice uses explicit objectives to organise engineering decisions and to interpret latency distributions [7]. Cloud disaster recovery guidance distinguishes recovery time from permissible data loss [2]. In this study, recovery time is the elapsed time to the simulated restoration endpoint. The 60-minute threshold is a benchmark for the experiment. It does not stand for a contractual objective, a general standard or a recovery point objective. The study also uses an impact-weighted duration that is analogous to integrating a loss of performance over time [10]. Here the loss is modelled as constant for each unresolved incident and ends upon restoration. Such a rectangular approximation allows the model to compare incidents with different weights. It cannot represent partial availability, time-varying demand or duplicated impact across shared dependencies.

Tail behaviour matters when a workload depends on several services or when delays concentrate in a subset of cases. Research on the tail at scale describes how slow components can affect overall service behaviour [12]. The present model uses the 90th percentile of incident recovery time to show a different property from the weighted mean. A policy can improve the weighted mean by allocating capacity to certain critical incidents and still leave a slow subset of low-priority incidents. Similarly, a binary 60-minute target does not distinguish between a two-minute and a two-hour miss. Reporting the mean, the tail and the threshold together makes those consequences visible.

2.2 Threat evidence and its limitations

Provider audit sources capture different event classes. AWS CloudTrail, Google Cloud Audit Logs and Azure Activity Log illustrate variation in collection scope, administrative operations and access events [3, 14, 21]. Event time and arrival time can also differ in an out-of-order data pipeline [1]. Operational implementations should track both, along with the availability of expected sources. A queue can assign priority only after enough evidence arrives to create a case. Late evidence therefore has two effects. It may change the score and it may shorten the remaining time before a recovery objective is breached.

Reconstructing a threat from provenance can link events across processes and resources. SLEUTH supports audit-based scenario reconstruction [16]. KAIROS and MAGIC use whole-system provenance in threat detection or investigation, and ORTHRUS focuses on attribution quality [11, 17, 18]. Comparative work reports that simpler provenance approaches can perform well in benchmark settings, while research on audit datasets emphasises the role of background activity and data quality [8, 20]. These studies motivate using case-level signals from multiple sources. They do not determine how an incident queue should allocate its next available slot. The study's Gaussian

features stand in for evidence summaries. They do not reproduce the event structures or detection performance of those systems.

Low prevalence complicates alert interpretation [5]. Security evaluation also faces differences between experimental and operational behaviour and risks from biased partitions [4, 26, 31]. The evaluation uses a single fitted detector and a separate stream generator for the test cohorts. A degraded evidence condition combines weaker malicious signals, a benign shift and more missing audit observations. These choices reflect documented concerns about model drift and uncertainty under distribution shift [13, 24]. They are scenarios for comparing policies. The numerical inputs are not estimates of any provider's telemetry quality.

2.3 Risk assessment as scheduling

NIST risk assessment guidance combines likelihood and consequence in the context of organisational decisions [19]. In a response queue, expected consequence alone leaves out the work required to investigate and contain a case. A short and consequential case may be a sensible use of the next available slot during congestion. The studied index divides expected consequence by an effort estimate. It is a heuristic for deciding what to start next. A global schedule could also use deadlines, shared dependencies, work already in progress and whether a specialist is required. Those complications would change the feasible assignments and are outside the experiment.

Risk priority is kept separate from authority to modify a resource. Zero-trust guidance supports explicit resource-scoped access decisions [27]. An operations implementation can use the risk score to route the case while requiring additional checks for changes with high potential disruption. ATT&CK cloud categories can help label cases and define scenario coverage [22], while continuous governance of model reliability fits the NIST AI risk management approach [32]. Table 1 relates the signal fields to the operational decisions they inform.

Table 1. Evidence and operational decisions

Input	Use in the studied method	Evidence required in deployment
Threat probability	Ranks likely incidents	Frozen model and observed input availability
Service impact	Weights probable consequence	Owner-validated service tier and dependency mapping
Response work	Normalises priority by expected effort	Estimated handling duration and responder skills
Recovery confirmation	Defines the end of incident impact	Service health and integrity checks

3 Methods

3.1 Synthetic case generation and threat scoring

The generator creates candidates independently within each cohort. Each candidate has a 3.5% probability of representing a true incident. Service impact is assigned independently as 0.5, 1 or 3 relative units with probabilities 0.35, 0.45 and 0.20. Every candidate has a work requirement drawn uniformly from the integers 3 through 11 minutes and a detection lag drawn from 1 through 8 minutes. A baseline restoration time is drawn from 8 through 19 minutes. The model assumes every incident eventually appears as a candidate and each candidate represents one distinct case. All benign candidates require work but accrue no incident impact. The effects of deferring a legitimate activity are represented by queue delay as a separate outcome.

The detector uses three standard normal features with positive-case mean shifts of 1.70, 1.10 and 0.95 in the nominal setting. The features represent abstract identity, behaviour and audit signals. The third feature is replaced by zero when missing, and the model also receives the availability indicator. Missingness is 6% in nominal cohorts. In the degraded cohort, the positive shifts are multiplied by 0.75, the mean of the first benign feature is increased by 0.45, and missingness rises to 24% for benign cases and 36% for true incidents. These adjustments jointly make evidence less informative. They vary together, so the contrast does not isolate any one mechanism. The classifier uses the features and the availability indicator only. It receives no service weight, effort, timing field or simulated ground-truth label at inference time.

One logistic regression model with the scikit-learn default regularisation and an iteration limit of 300 is fitted to 60,000 nominal candidates. It is held fixed for all evaluation workloads. A logistic output is interpreted as an estimated probability for the priority rule. Its calibration is not guaranteed under shift [15]. Detection is assessed using average precision alongside the nominal case prevalence and the Brier score for mean squared probability error [9, 29]. Neither metric depends on a threshold for action. Because every case enters the queue, average precision describes sorting potential rather than an operational recall rate.

3.2 Arrival streams and scheduling

Each of 48 replicates contains a nominal cohort and an independently generated degraded cohort of 3,600 cases each. A steady arrival stream is formed by cumulatively summing independent Poisson interarrival times with a mean of 2.2 minutes. In a separate burst stream, the mean is 0.65 minutes for cases 901 through 1,800 and 2.2 minutes elsewhere. The distribution can yield simultaneous arrivals. Both nominal and degraded evidence conditions use the same burst arrival times in each replicate, but they use different generated cases. For a given cohort and workload, all scheduling policies use exactly the same cases, times and scores.

An alert is released after its detection lag plus any added release delay. Each responder processes one case at a time. Every case is completed and the queue drains after the last arrival. The dispatcher chooses a ready case when a responder is free, and that case occupies the responder for the assigned work duration. Work proceeds without interruption. There is no analyst handover penalty or specialist skill constraint. A heap orders responders by the next time they are free and a second heap orders ready cases. Registered ties are resolved by release time and original case index. These choices prevent a later alert from taking a free slot before it becomes available.

Arrival order selects the earliest release. Probability ranking selects the highest fitted score. Risk ranking selects the largest value of $p \times I / W$, with p the estimated incident probability, I the relative service impact and W the assigned work in minutes. The scheduler is given the true generated work duration as its estimate. This grants the risk rule reliable effort information, an advantage that may disappear if effort predictions are poor. The study does not isolate gains due to the impact term from gains due to the effort term. Table 2 reports workload inputs and sensitivity ranges.

Table 2. Workload and sensitivity design

Quantity	Primary settings	Sensitivity settings
Training and testing	60,000 training cases, 48 × 3,600 per test scenario	Fixed fitted detector
Positive prevalence	3.5% generating probability	Same across scenarios

Quantity	Primary settings	Sensitivity settings
Arrival intervals	Poisson mean 2.2 min, burst mean 0.65 min	Steady workload uses 2.2 min throughout
Responders and release delay	4 responders, zero added delay	2, 3, 4, 6, 8 responders and 0–16 min delay
Impact and amplification	0.5, 1 or 3 units, amplification 0.8	Amplification 0, 0.4, 0.8, 1.2

3.3 Restoration model and comparison measures

Let C be the elapsed time from case arrival to the end of response work. Incident recovery is T = C + B + ceil(gC), where B is the generated baseline restoration duration and g is a delay amplification factor. In the primary setting, g equals 0.8. It models the possibility that longer exposure increases the work required after containment. B and the added restoration interval do not hold a responder slot. This assumption corresponds to a separate restoration capacity sufficient to avoid a second queue. When g equals zero, waiting still contributes directly to elapsed recovery time but does not increase the post-response interval.

The primary endpoint is mean I × T across true incidents within each replicate. The other outcomes are the share of incidents with T ≤ 60 minutes, the within-replicate 90th percentile of T, the median queue wait for true incidents and the mean queue wait for benign cases. Queue wait starts at release and ends at the beginning of work. Recovery time starts at case arrival and includes evidence release and intervention. The study reports the mean of each replicate's metric. This gives each workload realisation equal weight and differs slightly from pooling incident observations across replicates.

The nominal burst uses four responders and zero additional delay. The capacity sweep varies staffing from two to eight responders. The delay sweep adds zero to sixteen minutes to all evidence releases while retaining four responders. A separate sensitivity analysis varies the amplification factor from zero to 1.2. Figures show pointwise 95% Monte Carlo intervals for replicate means. For paired comparisons, differences are computed within each common workload replicate before estimating the standard error. Intervals use a two-sided t multiplier of 2.012 on 47 degrees of freedom. These intervals describe simulation variation conditional on model and inputs. No adjustment was made across endpoints and scenarios.

4 Results

4.1 Detection and recovery under nominal evidence

Mean average precision was 0.579 under nominal evidence and 0.206 under degraded evidence. Mean Brier score rose from 0.022 to 0.034. The deterioration confirms that the stress condition made the ranking task harder under the chosen generator. Because model fitting is fixed, these results describe the response to a specific shift in the test streams. They do not identify how much accuracy is recoverable by recalibration or retraining.

In steady nominal operation with four responders, weighted incident minutes averaged 44.11 under arrival order and approximately 42.4 under each score-based policy. During the burst with the same number of responders, arrival order averaged 929.69 weighted minutes. Probability and risk ranking reduced this to 54.06 and 51.07 respectively. The difference between risk and probability was −3.00 weighted minutes, with a paired interval from −5.93 to −0.06. The corresponding 60-minute attainment was approximately 98.6% for probability ranking and 98.4% for risk ranking.

Thus the small weighted gain at four responders coexisted with a small reduction on the unweighted recovery target.

Figure 1 shows the capacity sensitivity. The weighted difference between risk and probability is most pronounced at two responders in the simulated nominal burst, at 170.64 versus 202.61. At eight responders the policies converge at roughly 41.5. The difference relative to arrival order is especially large because the burst creates a large backlog, so arrival order allocates many early slots to benign cases. Table 3 reports the three main scenario contrasts.

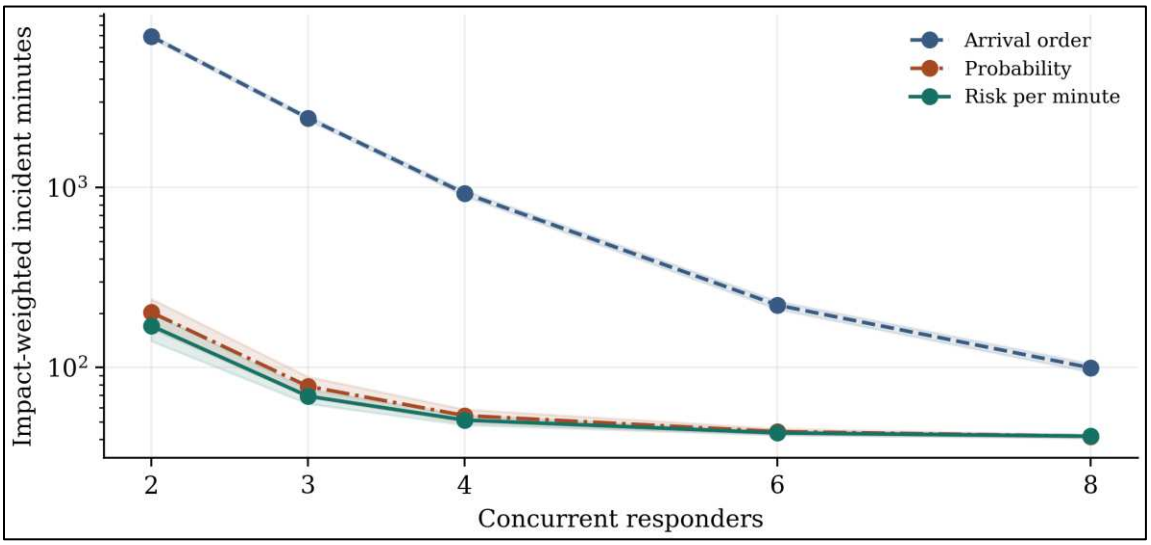


Figure 1. Impact-weighted incident minutes during the nominal burst by concurrent responder count. The vertical axis is logarithmic. Shading shows pointwise 95% Monte Carlo intervals.

Table 3. Mean recovery outcomes for four responders

Workload and evidence	Policy	Weighted min	Within 60 min (%)	Recovery p90 (min)
Steady, nominal	Arrival order	44.11	99.51	46.91
Steady, nominal	Probability	42.40	99.95	44.51
Steady, nominal	Risk per minute	42.38	99.97	44.54
Burst, nominal	Arrival order	929.69	26.22	1594.81
Burst, nominal	Probability	54.06	98.64	46.49
Burst, nominal	Risk per minute	51.07	98.37	46.48
Burst, degraded	Arrival order	952.21	25.15	1594.01
Burst, degraded	Probability	190.92	91.30	88.52
Burst, degraded	Risk per minute	145.17	91.19	67.53

4.2 Evidence release and degraded signals

Adding eight minutes to evidence release reduces the share restored within 60 minutes under risk ranking from approximately 98.4% to 88.5%. At twelve added minutes, probability and risk ranking each restore about 62.8% within the objective. A uniform release shift largely preserves which cases compete at the same time, and the structure of the response queue changes little. Recovery worsens because the added delay directly increases elapsed exposure and its modelled post-response

amplification. Therefore the curve in Figure 2 is a timing sensitivity. Variable or incident-dependent release delays could change the queue order as well.

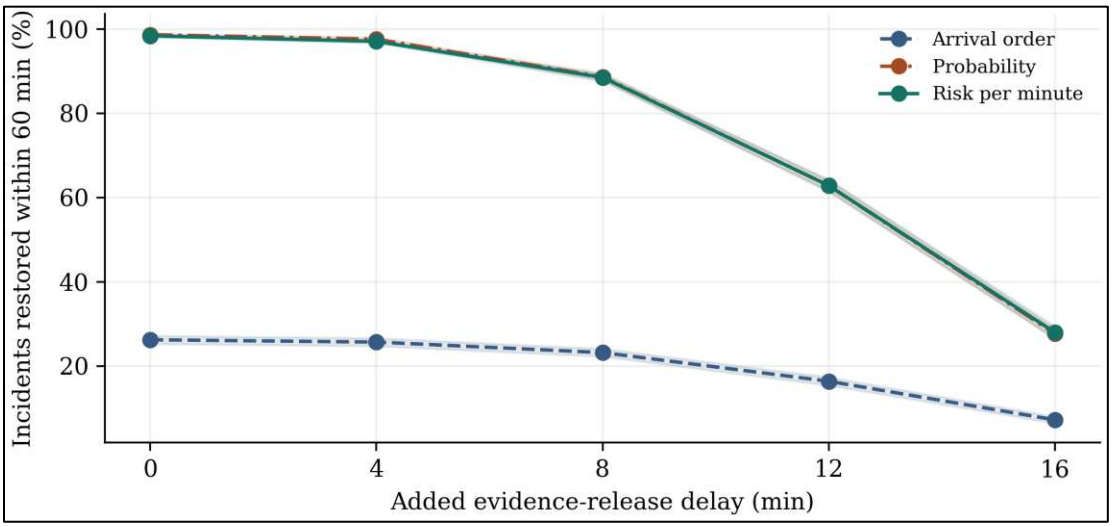


Figure 2. Share of incidents restored within 60 minutes as a function of added evidence-release delay. Four responders process a nominal burst. Shading shows pointwise 95% Monte Carlo intervals.

In the degraded burst, weighted minutes are 190.92 under probability ranking and 145.17 under risk ranking. The paired difference is −45.76 (95% interval −58.52 to −32.99). The difference in 60-minute attainment is approximately −0.11 percentage points, with an interval from −0.51 to +0.28. Figure 3 shows that the mean within-replicate 90th percentile increases under degraded evidence for both score-based methods. The mean is 88.52 minutes for probability ranking and 67.53 minutes for risk ranking. The paired difference of −20.99 minutes has an interval from −53.40 to +11.43. The tail estimate varies substantially across the simulated workloads. The weighted endpoint provides clearer evidence of an advantage at this setting because it places additional emphasis on high-impact incidents.

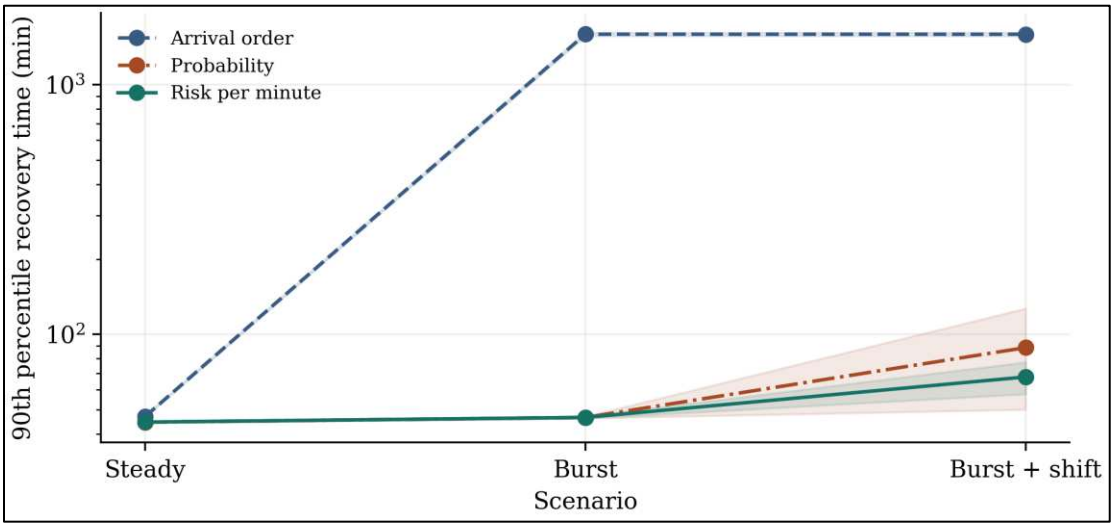


Figure 3. Mean within-replicate 90th percentile of incident recovery time under steady, burst and degraded burst conditions. The vertical axis is logarithmic. Shading shows pointwise 95% Monte Carlo intervals.

4.3 Benign cases and delay amplification

With four responders in the nominal burst, mean benign wait is approximately 416.74 minutes under arrival order, 430.87 under probability and 401.17 under risk. Risk minus probability is −29.70 minutes (95% interval −30.41 to −28.99). Ranking by work as well as expected impact can clear some short jobs earlier, reducing mean wait despite allocating early slots to many true incidents. This does not mean all benign cases benefit. The model reports their mean wait, which can conceal long deferrals for particular tenants or case classes. At two responders, risk ranking still leaves mean benign queue wait above 3,100 minutes. Figure 4 shows why improved weighted recovery cannot substitute for adequate capacity.

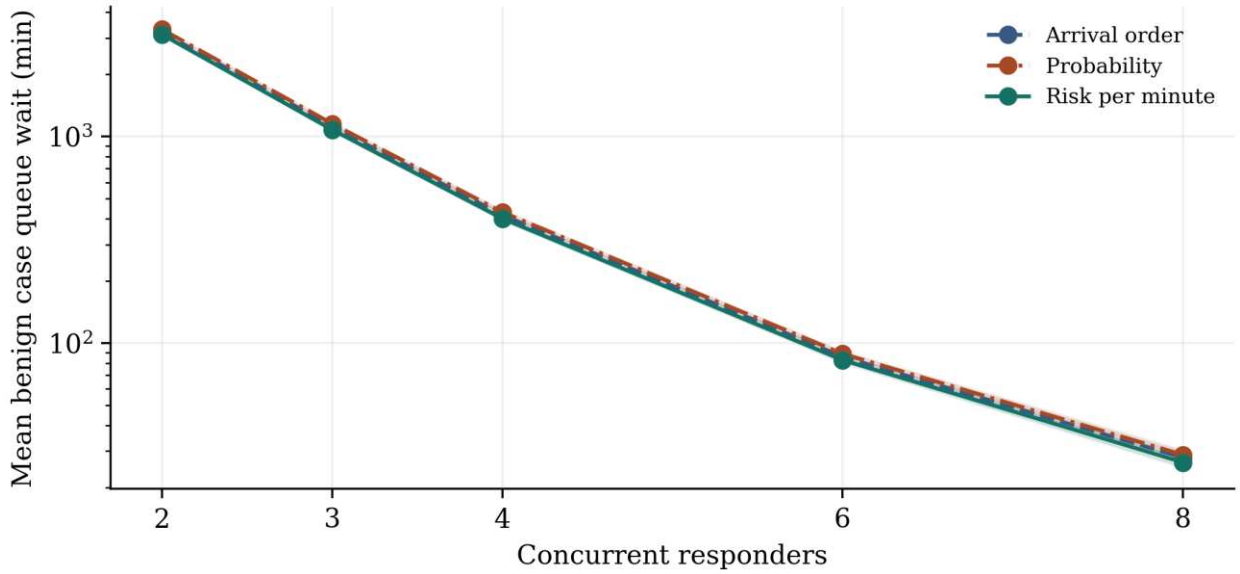


Figure 4. Mean benign queue wait by concurrent responder count in the nominal burst. The vertical axis is logarithmic. Shading shows pointwise 95% Monte Carlo intervals.

The amplification sensitivity changes the weighted level while leaving the queue assignments fixed. At zero amplification, probability and risk ranking yield 36.98 and 35.32 weighted minutes. At 1.2 amplification, the means are 62.37 and 58.71. This pattern follows from applying a larger post-response penalty to cases with long elapsed time. The result is therefore contingent on the recovery equation. Table 4 reports the principal sensitivity comparisons and their interpretation.

Table 4. Selected sensitivity results

Contrast at nominal burst	Probability ranking	Risk per minute
Weighted minutes with 2 responders	202.61	170.64
Weighted minutes with 8 responders	41.47	41.49
Within 60 minutes with 12 min added release delay	62.8%	62.8%
Weighted minutes with no amplification	36.98	35.32
Weighted minutes with 1.2 amplification	62.37	58.71

5 Discussion

5.1 What the policy comparison establishes

The experiment shows a mechanism through which threat ranking can improve service outcomes without adding responders. Under a burst, useful evidence reserves early work for likely incidents. Incorporating impact and effort can then change which likely incidents receive those early slots. In the degraded scenario, the risk rule retains a weighted advantage because service impact and work provide additional ordering information independent of the degraded classifier features. The same advantage need not occur when criticality data are stale or when complex cases are systematically underestimated. The model supplies perfect work estimates and a correctly recorded impact tier for every candidate.

Differences between metrics are operationally important. A response leader interested in limiting aggregate service loss could prefer the weighted outcome. A service owner tracking a customer-facing 60-minute objective might see little improvement. A manager concerned about unresolved work must consider the delay for benign cases because delayed closure can increase repeat alerts, duplicate investigation or customer disruption. Mean wait also leaves out dispersion. A later evaluation should report queue age by service class and investigate whether short high-scoring cases systematically push older work beyond an acceptable review interval.

The model's steady and burst contrast can also help distinguish two separate interventions. When the queue is rarely occupied, changing ranking has few opportunities to improve outcomes. Under congestion, ranking can materially change which cases wait. Increased response staffing helps both score-based and arrival-order policies, and near the high end of the simulated staffing range the incremental value of a more elaborate priority index becomes small. The numerical staffing levels cannot be transferred directly to a production team, where shift schedules, specialist routing, concurrent incidents and provider limitations determine effective capacity.

5.2 Operational validation

A deployment study could begin with a retrospective replay of approved case metadata. The replay should preserve event creation, evidence availability, queue admission, first response, containment completion and confirmed restoration. Feature construction must use only information available when the case becomes eligible for response. Otherwise a retrospective score might benefit from signals recorded after an analyst started work. A frozen time-separated evaluation, with tenant-level grouping, would address some of the evaluation problems documented in security machine learning [4, 26].

Service-impact estimates should be connected to versioned service inventories and confirmed by owners. The monetary or user-facing interpretation of the weights depends on workload demand and service dependencies. If two affected workloads share one dependency, summing both weights could count the same loss twice. A replay can add an impact-only comparator, a probability-over-effort comparator and a policy with ageing or deadlines. These additions would separate the contribution of impact, effort and time already spent waiting. Estimation noise should be introduced explicitly because historical time to closure is affected by the policy that originally selected the case.

An observation-only deployment can retain alerts and compute alternative schedules without changing work assignments. It can measure score drift, missing evidence and differences in recommendations over time. Long-lived production systems face dependencies among data sources,

features and downstream decisions [30]. A versioned case record can therefore store input availability, model output, business impact, effort estimate and the rule that produced the recommendation. Where a recommendation leads to a change in cloud state, authorisation and recovery verification remain separate. Confirming that a restriction was executed does not establish that the affected service has returned to trusted operation.

6 Limitations

The generated stream treats cases as independent and gives every incident a candidate. Actual attacks can be missed entirely, can create several alerts and can correlate across accounts. The simulation therefore excludes unrecovered incidents that were never detected. It also omits interactions between cases in the containment process, adverse consequences from response actions and a limited restoration team. The severe burst and unlimited queue drainage make extremely long waits possible. In practice, incident teams could drop duplicate reports, reassign staff, transfer cases or invoke a separate major-incident process. These actions would change the counterfactual comparison with arrival order.

The model also assumes positive-class probabilities from a single fitted detector and uses only one bundled degraded-evidence scenario. Its work estimates equal realised durations, task impacts are independent of true incident status and recovery grows linearly with exposure apart from rounding. Production data could violate all of these assumptions. A model that underestimates labour for critical cases, for example, would schedule those cases too early compared with its own intended rule. Conversely, a recovery team with scarce specialists could create a second queue that dominates any gains at the first stage. The plots quantify simulation behaviour and provide no measured estimates of production recovery, security incident severity or cloud infrastructure availability.

7 Conclusion

Cloud resilience depends on what happens between detection and verified restoration. In this simulation, prioritising queued cases by incident probability materially improves recovery over arrival order during a burst. Adding service impact and response effort reduces weighted incident minutes further, especially under degraded evidence. Its advantage on the binary recovery target is small and uncertain under degraded evidence, and severe overload still produces long benign waits. The results support measuring detection latency, capacity, recovery and service impact together when validating risk-based incident scheduling.

Declarations

Ethics approval and consent to participate. This computational study used synthetic data only and involved no human participants, personal data or live cloud systems. Consent for publication is not applicable to the generated data.

Data and code availability. Appendix A provides the seeded simulation code that generates the study cohorts and metric arrays. No external dataset is required. Figure intervals and table statistics use the aggregation specified in Methods.

Competing interests, funding and author contribution. These statements require confirmation from Rehana Devani before public submission.

Generative AI disclosure. OpenAI ChatGPT assisted with literature discovery, study design, simulation and analysis code, visualisation and manuscript drafting. The named author must independently review, verify and approve the study and manuscript before submission.

References

[1] Akidau, T., et al. (2015). The Dataflow model: A practical approach to balancing correctness, latency, and cost in massive-scale, unbounded, out-of-order data processing. Proceedings of the VLDB Endowment, 8(12), 1792–1803. https://doi.org/10.14778/2824032.2824076

[2] Amazon Web Services. (n.d.). Disaster recovery of workloads on AWS. https://docs.aws.amazon.com/whitepapers/latest/disaster-recovery-workloads-on-aws/introduction.html

[3] Amazon Web Services. (n.d.). Understanding CloudTrail events. AWS CloudTrail User Guide. https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudtrail-events.html

[4] Arp, D., et al. (2022). Dos and don'ts of machine learning in computer security. USENIX Security Symposium. https://www.usenix.org/conference/usenixsecurity22/presentation/arp

[5] Axelsson, S. (2000). The base-rate fallacy and the difficulty of intrusion detection. ACM Transactions on Information and System Security, 3(3), 186–205. https://doi.org/10.1145/357830.357849

[6] Bartock, M., Cichonski, J., Souppaya, M., Smith, M., Witte, G., & Scarfone, K. (2016). Guide for cybersecurity event recovery. NIST SP 800-184. https://doi.org/10.6028/NIST.SP.800-184

[7] Beyer, B., Jones, C., Petoff, J., & Murphy, N. R. (Eds.). (2016). Site reliability engineering. O'Reilly Media. https://sre.google/sre-book/table-of-contents/

[8] Bilot, T., et al. (2025). Sometimes simpler is better: A comprehensive analysis of state-of-the-art provenance-based intrusion detection systems. USENIX Security Symposium. https://www.usenix.org/conference/usenixsecurity25/presentation/bilot

[9] Brier, G. W. (1950). Verification of forecasts expressed in terms of probability. Monthly Weather Review, 78(1), 1–3. https://doi.org/10.1175/1520-0493(1950)078%3C0001:VOFEIT%3E2.0.CO;2

[10] Bruneau, M., et al. (2003). A framework to quantitatively assess and enhance the seismic resilience of communities. Earthquake Spectra, 19(4), 733–752. https://doi.org/10.1193/1.1623497

[11] Cheng, Z., et al. (2024). KAIROS: Practical intrusion detection and investigation using whole-system provenance. IEEE Symposium on Security and Privacy. https://doi.org/10.1109/SP54263.2024.00005

[12] Dean, J., & Barroso, L. A. (2013). The tail at scale. Communications of the ACM, 56(2), 74–80. https://doi.org/10.1145/2408776.2408794

[13] Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. ACM Computing Surveys, 46(4), Article 44. https://doi.org/10.1145/2523813

[14] Google Cloud. (n.d.). Cloud Audit Logs overview. Cloud Logging documentation. https://docs.cloud.google.com/logging/docs/audit

[15] Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017). On calibration of modern neural networks. Proceedings of Machine Learning Research, 70, 1321–1330. https://proceedings.mlr.press/v70/guo17a.html

[16] Hossain, M. N., et al. (2017). SLEUTH: Real-time attack scenario reconstruction from COTS audit data. USENIX Security Symposium. https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/hossain

[17] Jia, Z., Xiong, Y., Nan, Y., Zhang, Y., Zhao, J., & Wen, M. (2024). MAGIC: Detecting advanced persistent threats via masked graph representation learning. USENIX Security Symposium. https://www.usenix.org/conference/usenixsecurity24/presentation/jia-zian

[18] Jiang, B., et al. (2025). ORTHRUS: Achieving high quality of attribution in provenance-based intrusion detection systems. USENIX Security Symposium. https://www.usenix.org/conference/usenixsecurity25/presentation/jiang-baoxiang

[19] Joint Task Force Transformation Initiative. (2012). Guide for conducting risk assessments. NIST SP 800-30 Rev. 1. https://doi.org/10.6028/NIST.SP.800-30r1

[20] Liu, J., Inam, M. A., Goyal, A., Riddle, A., Westfall, K., & Bates, A. (2025). What we talk about when we talk about logs: Understanding the effects of dataset quality on endpoint threat detection research. IEEE Symposium on Security and Privacy. https://doi.org/10.1109/SP61157.2025.00112

[21] Microsoft. (n.d.). Activity Log in Azure Monitor. Microsoft Learn. https://learn.microsoft.com/en-us/azure/azure-monitor/fundamentals/activity-log

[22] MITRE. (n.d.). Enterprise ATT&CK cloud matrix. https://attack.mitre.org/matrices/enterprise/cloud/

[23] Nelson, A., Rekhi, S., Souppaya, M., & Scarfone, K. (2025). Incident response recommendations and considerations for cybersecurity risk management. NIST SP 800-61 Rev. 3. https://doi.org/10.6028/NIST.SP.800-61r3

[24] Ovadia, Y., et al. (2019). Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift. Advances in Neural Information Processing Systems, 32. https://proceedings.neurips.cc/paper/2019/hash/8558cb408c1d76621371888657d2eb1d-Abstract.html

[25] Pascoe, C., Quinn, S., & Scarfone, K. (2024). The NIST Cybersecurity Framework 2.0. NIST CSWP 29. https://doi.org/10.6028/NIST.CSWP.29

[26] Pendlebury, F., Pierazzi, F., Jordaney, R., Kinder, J., & Cavallaro, L. (2019). TESSERACT: Eliminating experimental bias in malware classification across space and time. USENIX Security Symposium. https://www.usenix.org/conference/usenixsecurity19/presentation/pendlebury

[27] Rose, S., Borchert, O., Mitchell, S., & Connelly, S. (2020). Zero trust architecture. NIST SP 800-207. https://doi.org/10.6028/NIST.SP.800-207

[28] Ross, R., Pillitteri, V., Graubart, R., Bodeau, D., & McQuaid, R. (2021). Developing cyber-resilient systems. NIST SP 800-160 Vol. 2 Rev. 1. https://doi.org/10.6028/NIST.SP.800-160v2r1

[29] Saito, T., & Rehmsmeier, M. (2015). The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. PLOS ONE, 10(3), e0118432. https://doi.org/10.1371/journal.pone.0118432

[30] Sculley, D., et al. (2015). Hidden technical debt in machine learning systems. Advances in Neural Information Processing Systems, 28. https://research.google/pubs/hidden-technical-debt-in-machine-learning-systems/

[31] 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

[32] Tabassi, E. (2023). Artificial intelligence risk management framework (AI RMF 1.0). NIST AI 100-1. https://doi.org/10.6028/NIST.AI.100-1

Appendix A Reproduction code

Run the first listing as study.py with Python, NumPy and scikit-learn. Save the second as plots.py and run it from the same directory with Matplotlib after generating results.npz.

study.py

```
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import average_precision_score, brier_score_loss

SEED = 261114
REPS = 48
N = 3600
LOADS = [2, 3, 4, 6, 8]
DELAYS = [0, 4, 8, 12, 16]
FORMS = ["FIFO", "Probability", "Risk"]

def sample(rng, n, shift=0):
    y = rng.random(n) < .035
    impact = rng.choice([.5, 1., 3.], n, p=[.35, .45, .20])
    service = rng.integers(3, 12, n)
    detection = rng.integers(1, 9, n)
    base_restore = rng.integers(8, 20, n)
    x = rng.normal(size=(n, 3)) + y[:, None] * [1.7, 1.1, .95] * (1-.25*shift)
    x[:,0] += (~y) * .45 * shift
    observed = rng.random(n) >= .06 + shift * (.18 + .12*y)
    x[:, 2] *= observed
    return y, impact, service, detection, base_restore, np.column_stack([x, observed])

def evaluate(data, arrivals, p, responders, policy, delay, rate=.8):
    y, impact, work, detection, base_restore, _ = data
    release = arrivals + detection + delay
    order = np.argsort(release, kind="stable")
    ranked = [release, -p, -(p * impact / work)][policy]
    import heapq
    waiting = []
    workers = [(0, i) for i in range(responders)]
    starts = np.empty(len(y), dtype=int)
    cursor = 0
    while cursor < len(y) or waiting:
        free, worker = heapq.heappop(workers)
        if not waiting and cursor < len(y):
            free = max(free, release[order[cursor]])
        while cursor < len(y) and release[order[cursor]] <= free:
            i = order[cursor]
            heapq.heappush(waiting, (ranked[i], release[i], i))
            cursor += 1
        _, _, i = heapq.heappop(waiting)
        starts[i] = free
        heapq.heappush(workers, (free + work[i], worker))
    completion = starts + work - arrivals
    recovery = completion[y] + base_restore[y] + np.ceil(rate * completion[y])
    loss = impact[y] * recovery
    return np.array([loss.mean(), np.mean(recovery <= 60),
                     np.quantile(recovery, .9), np.median(starts[y]-release[y]),
                     np.mean(starts[~y]-release[~y])])

def run():
    rng = np.random.default_rng(SEED)
    train = sample(rng, 60000)
    clf = LogisticRegression(max_iter=300).fit(train[-1],train[0])
    capacities, lags, shift_rows, rates, metrics = [], [], [], [], []
    for _ in range(REPS):
        normal = sample(rng, N)
        stress = sample(rng, N, 1)
        arrival = np.cumsum(rng.poisson(2.2, N))
        # Burst lasts for 900 candidates, with subsequent input returning to normal.
        interval = rng.poisson(2.2, N)
        interval[900:1800] = rng.poisson(.65, 900)
```

```
        burst = np.cumsum(interval)
        p = clf.predict_proba(normal[-1])[:,1]
        q = clf.predict_proba(stress[-1])[:,1]
        def panel(d, a, s, workers=4, lag=0, rate=.8):
            return [evaluate(d,a,s,workers,k,lag,rate) for k in range(3)]
        capacities.append([panel(normal,burst,p,c) for c in LOADS])
        lags.append([panel(normal,burst,p,lag=d) for d in DELAYS])
        shift_rows.append([panel(normal,arrival,p),panel(normal,burst,p),
                           panel(stress,burst,q)])
        rates.append([panel(normal,burst,p,rate=r) for r in [0,.4,.8,1.2]])
        metrics.append([[average_precision_score(d[0],s),brier_score_loss(d[0],s)]
                        for d,s in [(normal,p),(stress,q)]])
    return dict(capacity=np.array(capacities),delay=np.array(lags),
                shift=np.array(shift_rows),rate=np.array(rates),prediction=np.array(metrics))

if __name__ == "__main__":
    result=run()
    np.savez("results.npz",**result)
    for k,v in result.items():
        print(k,np.round(v.mean(0),3))
    for j in [0,1,2,4]:
        d=result["shift"][:,1,2,j]-result["shift"][:,1,1,j]
        h=2.012*d.std(ddof=1)/np.sqrt(REPS)
        print("paired",j,d.mean(),d.mean()-h,d.mean()+h)
```

plots.py

```
import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt
import numpy as np
from study import REPS, LOADS, DELAYS
results = np.load("results.npz")
plt.rcParams.update({"font.family":"serif","font.size":10,
                     "axes.spines.top":False,"axes.spines.right":False})
labels = ["Arrival order","Probability","Risk per minute"]
colors = ["#385a83","#a64a26","#177266"]
def plot(number, x, y, xlabel, ylabel, log=False, tags=None):
    fig, ax = plt.subplots(figsize=(6.55,3.05),layout="constrained")
    for j,(label,color,style) in enumerate(zip(labels,colors,["--","-.", "-"])):
        values=y[:,:,j]
        m=values.mean(axis=0)
        h=2.012*values.std(axis=0,ddof=1)/np.sqrt(REPS)
        ax.plot(x,m,style,marker="o",color=color,label=label)
        ax.fill_between(x,m-h,m+h,color=color,alpha=.12)
    if log:
        ax.set_yscale("log")
    ax.set(xlabel=xlabel,ylabel=ylabel,xticks=x)
    if tags:
        ax.set_xticklabels(tags)
    ax.grid(alpha=.20)
    ax.legend(frameon=False,fontsize=8)
    fig.savefig(f"fig{number}.png",dpi=320)
    plt.close(fig)
plot(1,LOADS,results["capacity"][:,:,:,0],"Concurrent responders",
     "Impact-weighted incident minutes",True)
plot(2,DELAYS,100*results["delay"][:,:,:,1],"Added evidence-release delay (min)",
     "Incidents restored within 60 min (%)")
plot(3,range(3),results["shift"][:,:,:,2],"Scenario",
     "90th percentile recovery time (min)",True,["Steady","Burst","Burst + shift"])
plot(4,LOADS,results["capacity"][:,:,:,4],"Concurrent responders",
     "Mean benign case queue wait (min)",True)
```