Automated Root Cause Analysis of Security Incidents Using Machine Learning

Rehana Devani

Academic background supplied by the author: Master's in Computing Science, The University of Texas at Arlington, Arlington, Texas, USA.

Article type: Method article with synthetic validation

Abstract

Security incident investigation requires distinguishing an initiating mechanism from downstream symptoms. An unusual event may be statistically predictive of an incident without being the mechanism that generated it. This study evaluates a transparent causal machine learning procedure that ranks standardised mechanism residuals in a supplied structural causal graph. A six-variable linear structural model represents abstract access, policy, execution, privilege, movement and impact signals. Across 30 independently seeded replications, each containing 2,000 normal training observations and 600 synthetic incidents, a four-unit innovation shift is injected into one of three candidate mechanisms. Five mechanisms are ranked, including two downstream distractors. Marginal anomaly scoring is compared with residual scoring under the correct graph and under deletion of one generating edge. Additional experiments vary independent sensor noise from zero to twice the innovation standard deviation. Without sensor noise, mean top-one accuracy is 98.41% for residual scoring under the supplied correct graph, 86.66% for marginal scoring and 94.26% after edge deletion. At a sensor-noise standard deviation of one, these values decline to 80.63%, 61.64% and 75.33%, respectively. A simulator-based repair check confirms that reversing the injected innovation removes most of the mean impact shift. These results validate the implementation under favourable, explicitly stated assumptions; they do not demonstrate causal discovery, effectiveness on operational incident records or readiness for autonomous containment. The contribution is a reproducible methodological study that separates mechanism localisation, graph dependence and counterfactual validation in security incident analysis.

Keywords: causal machine learning; root cause analysis; security incidents; structural causal models; anomaly attribution; synthetic validation

1 Introduction

Security operations generate observations at several levels, including authentication events, process execution, privilege changes and network activity. An incident can produce abnormalities across all of these levels. Selecting the most unusual observation is therefore an uncertain way to identify its origin. A downstream variable can amplify a modest upstream disturbance, while an initiating action may appear ordinary when considered alone. The operational question is not simply which measurement differs from normal behaviour, but which mechanism changed and how that change propagated.

Causal inference provides a language for separating observation, intervention and counterfactual reasoning [1,2]. These distinctions matter when investigators interpret model explanations as recommendations for remediation. Research on causal representation learning connects mechanisms with questions of generalisation [3], but an incident analyst still needs a defensible link between a model variable and a security-relevant process. In particular, statistical dependence does not establish that changing a correlated event would prevent an incident.

Existing causal outlier attribution explicitly addresses how an abnormal observation can arise through a causal system [4]. Graphical causal modelling software provides tools for fitting mechanisms and posing related queries [5]. The broader DoWhy workflow emphasises specification, identification, estimation and refutation as distinct stages [6]. These contributions motivate an empirical question with a deliberately narrower scope: when the graph is supplied, does modelling local mechanisms help distinguish an injected cause from its propagated symptoms?

This article answers that question through a controlled experiment rather than a claim of deployment performance. A small structural model provides exact ground truth, normal data for mechanism fitting and incident data with known perturbation locations. The experiment contrasts marginal scoring with mechanism residual scoring, then stresses the latter through measurement noise and graph misspecification. The graph is never learned from the evaluation incidents. Consequently, any observed advantage concerns conditional modelling with structural knowledge, rather than automatic discovery of causal direction.

The contribution consists of a fully specified benchmark, a transparent implementation and a set of diagnostic tests. It does not introduce a new causal identification theorem or claim that residual scoring itself is novel. Its practical value is to make explicit the assumptions often hidden when security analytics systems describe an output as a root cause. Four scientific figures and four tables present the generating structure, performance, errors and repair behaviour.

2 Related work and conceptual boundaries

2.1 Learning structures and learning mechanisms

A causal graph specifies which variables directly influence each mechanism. A mechanism model estimates how a variable depends on its parents. These are different learning problems. NOTEARS formulates directed acyclic graph learning as continuous optimisation [7], DAG-GNN extends graph learning through a neural generative formulation [8], and DYNOTEARS models contemporaneous and lagged dependencies in time series [9]. None is evaluated here, because introducing graph discovery would obscure whether errors arose from structure estimation or mechanism scoring.

Temporal observations require particular care. Runge and colleagues developed conditional independence methods for nonlinear time series [10], and PCMCI+ extends this framework to contemporaneous relationships [11]. A timestamp order alone does not resolve aggregation, clock error or hidden common causes. The present simulator uses independent incident snapshots with an explicit structural order. It therefore makes no claim about recovering attack timing or handling feedback in a live environment.

Identifiability also depends on distributional restrictions. LiNGAM uses linearity, non-Gaussian disturbances and causal sufficiency [12]. Additive-noise approaches can identify structure under other conditions [13]. Greedy equivalence search concerns structure identification within graphical constraints [14], while invariant prediction uses stability across environments [15]. The simulator instead uses Gaussian innovations and a supplied graph. Gaussian linear observational data would not, by themselves, justify all of the directed edges used in the experiment.

2.2 Explanations and security investigation

Predictive feature attribution and causal mechanism attribution answer different questions. SHAP provides a general framework for explaining model predictions [16], whereas the distribution used when excluding features has causal implications [17]. A feature can receive substantial predictive credit because it measures a consequence of the incident. Interpretable models can also make the relationship between an explanation and

its assumptions easier to inspect [18]. Here, each candidate receives a score from a fitted local equation rather than an explanation of a separate incident classifier.

Security evaluation introduces additional constraints. Sommer and Paxson describe difficulties in transferring machine learning to operational intrusion detection [19]. Arp and colleagues identify methodological pitfalls involving data, evaluation and interpretation [20]. Accordingly, this study keeps test incident labels out of fitting, uses independent simulation seeds and avoids interpreting a favourable synthetic result as evidence of enterprise effectiveness.

Provenance-based systems establish important capabilities for investigation. SLEUTH reconstructs attack scenarios from audit data [21], ATLAS learns sequence patterns to recover attack stories [22], KAIROS models anomalous provenance evolution [23], and MAGIC uses masked graph representation learning for threat detection [24]. Their outputs and threat models differ from the numerical localisation task studied here. A provenance dependency records a possible information or execution relationship; it is not automatically an identified statistical intervention model. No performance comparison with these systems is claimed. Table 1 separates the relevant inferential tasks.

Table 1. Distinct questions in automated incident analysis

Task	Output	Evidence required
Anomaly detection	Unusual observation	Normal reference and detection evaluation
Attack reconstruction	Linked events or sequences	Auditable provenance relationships
Mechanism localisation	Ranked changed equations	Specified graph and mechanism assumptions
Causal discovery	Graph or equivalence class	Identifiability conditions and structural validation
Intervention evaluation	Predicted outcome under action	Identified intervention model and held-out tests

3 Methods

3.1 Study design and simulation scope

This is a computational methods study using exclusively generated numerical data. The six variables have security-inspired names, but do not represent measured probabilities, event counts or an organisation's actual systems. They are continuous, dimensionless signals. The names establish an interpretable dependency ordering without asserting a realistic attack ontology. Access and policy are exogenous mechanisms; execution, privilege and movement are intermediate mechanisms; impact is the final outcome.

Let X be a six-dimensional column vector and let U contain mutually independent standard normal innovations. The generating equations are $X = WX + U$, or equivalently $X = (I - W)^{-1}U$. The matrix W is strictly lower triangular. Its seven nonzero entries are $W_{31} = 0.9$, $W_{32} = 0.6$, $W_{43} = 1.1$, $W_{52} = 0.7$, $W_{54} = 0.8$, $W_{64} = 0.8$ and $W_{65} = 0.9$, using one-based indexing. All other entries are zero. Figure 1 compares ordinary correlation with these direct generating coefficients.

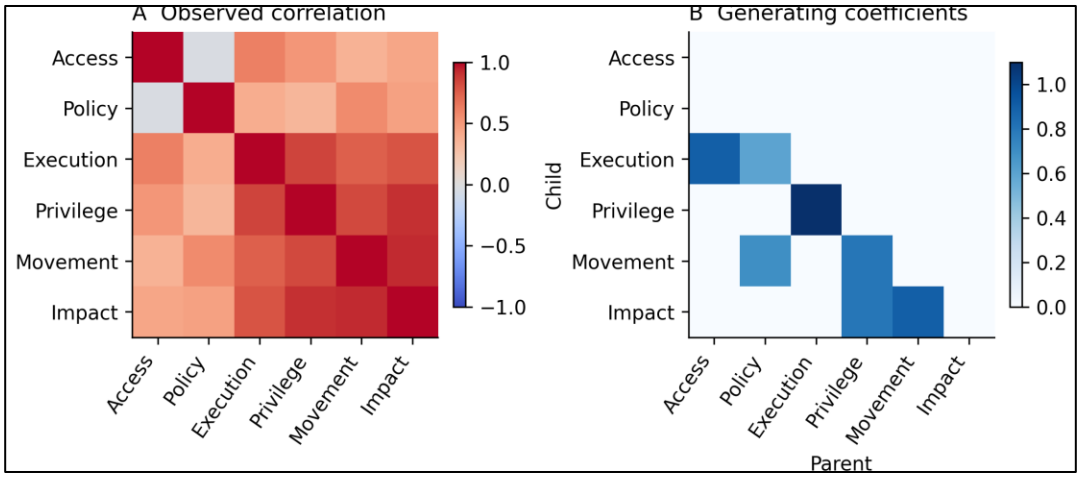


Figure 1. Normal-data correlation and true direct coefficients. Panel A uses 2,000 noiseless training observations from seed 0. Panel B shows W, with children on rows and parents on columns. Both panels concern synthetic dimensionless signals; correlation is not an inferred causal graph.

For each seed from zero to 29, the generator produces 2,000 normal training observations and 600 incident observations. The injected mechanism cycles through access, policy and execution, yielding 200 incidents per cause in every replication. Each incident adds four to the chosen innovation. All other innovations remain sampled from their normal distribution. This is a soft change to an exogenous disturbance, not an observation selected because it exceeds a detector threshold. Incident detection is outside the experiment's scope.

The design has five localisation candidates: access, policy, execution, privilege and movement. Impact is excluded because it is the designated outcome. Privilege and movement are never injected and act as downstream distractors. Candidate causes are balanced, but the procedure is not told that only the first three mechanisms can be injected. Restricting candidates to the three injected mechanisms would make the task easier and would conceal false attribution to propagated symptoms. Table 2 records the fixed experimental choices.

Table 2. Synthetic experimental design

Component	Specification
Replications	30 independent seeds, numbered 0–29
Training and evaluation	2,000 normal observations and 600 incidents per seed
Generating model	Six-node linear acyclic model; seven nonzero coefficients
Innovations	Independent standard normal variables
Injected change	Add 4 to access, policy or execution innovation
Cause balance	200 incidents per injected mechanism in each seed
Ranked candidates	Access, policy, execution, privilege and movement
Measurement noise SD	0, 0.25, 0.5, 1 and 2; independent in training and test
Structural stress	Remove access → execution; refit remaining mechanisms
Uncertainty	Mean and Student t interval across 30 seed-level metrics

3.2 Training and mechanism scoring

Normal training data alone determine means, variances and regression coefficients. For each non-root candidate j, ordinary least squares estimates its conditional mean from the supplied parent set. The training residual standard deviation supplies the denominator of the mechanism score. For an incident observation x, the score is $S_j(x) = |x_j - \hat{f}_j(x\text{parents})| / \hat{\sigma}_j$. For graph roots, the same expression reduces to an absolute marginal z score. No incident labels are used to choose coefficients, thresholds or parent sets.

A large residual indicates that the variable differs from what its local parent values predict. In a correctly specified additive structural model, propagated abnormalities can be explained by changed parent values, whereas an altered innovation remains locally surprising. This reasoning depends on the parent set and noise assumptions. The score is not a posterior probability that the variable caused the incident, and equal residual scores do not establish equal intervention effects.

The marginal baseline uses the absolute z score of each candidate with respect to normal training data. This baseline receives the same candidate set and observations but no graph. The correctly structured residual procedure receives the seven-edge generating graph. The misspecified procedure receives a graph in which access to execution is deleted; coefficients for all remaining parent sets are re-estimated from the normal training data. This deletion creates a purposeful structural error, not random sampling uncertainty in graph learning.

Candidates are sorted by descending score. Stable sorting resolves exact ties by variable order, although ties are uncommon with continuous generated data. The experiment implements all procedures directly rather than invoking a causal attribution library. DoWhy and DoWhy-GCM inform the separation of modelling and validation, but their attribution algorithms are not benchmarked [5,6].

3.3 Sensor noise and evaluation measures

Sensor noise is added independently to every measured variable in both training and incident data. Its standard deviation is zero, 0.25, 0.5, one or two. The latent incident samples remain shared across noise levels within a seed, whereas measurement noise is newly drawn. At nonzero noise, even the correct latent graph does not guarantee correctly specified conditional models for the measured variables. Measurement error can bias fitted relationships and induce dependence between the measured residuals.

The primary outcome is top-one localisation accuracy, the fraction of incidents whose highest-ranked mechanism matches the injected location. Secondary outcomes are top-three accuracy and mean reciprocal rank. A correct location at rank r contributes 1/r to reciprocal rank. Each seed produces one value per metric and condition. Reported means are averages across 30 seeds. Confidence intervals use a two-sided Student t interval over these 30 values, with 29 degrees of freedom. They describe Monte Carlo uncertainty under the chosen generator, not uncertainty about real security environments.

Every noise condition contains 18,000 simulated incident evaluations per procedure across the 30 replications. These are not 18,000 independent enterprise incidents. Observations within a replication share fitted training parameters. Seed-level intervals respect that shared fitting stage. Table 3 reports representative conditions, while Figure 2 shows all five noise levels. No post hoc search for a favourable graph, shift size or noise range was performed.

3.4 Simulator repair check

A second diagnostic asks whether removing an assumed disturbance reduces simulated impact. Let $A = (I - W)^{-1}$. If the injected innovation is k, restoring its pre-shift value changes impact by $4A_{6k}$. The true repair subtracts this quantity from the incident's latent impact. The selected repair instead subtracts $4A_{6p}$, where p is the mechanism selected by the correctly structured residual scorer without sensor noise. Both calculations use the true generating matrix and the known injected shift magnitude.

This is an oracle-assisted consistency check. It verifies propagation and localisation within the simulator, but does not evaluate a learned counterfactual estimator. When the selected mechanism is wrong, subtracting four from its innovation is a hypothetical intervention rather than recovery of its actual disturbance. Consequently, selected repair must not be interpreted as an implementable security response. The construction preserves the remaining innovations, allowing the simulator to compare alternative worlds for the same numerical incident.

3.5 Implementation and reproducibility

The experiment uses NumPy for sampling and array operations [25], SciPy for Student t quantiles [26], Matplotlib for figures [27], and scikit-learn for linear regression [28]. The execution environment used NumPy 2.3.5, SciPy 1.17.0, Matplotlib 3.10.8 and scikit-learn 1.8.0. The numerical core is included in Appendix A so that the simulation can be reproduced without access to a private dataset. The supplied graphs and parameter values are part of the study design, rather than estimates about an actual organisation.

4 Results

4.1 Separation of causes from propagated symptoms

The correlation matrix in Figure 1 contains substantial associations between variables without direct connecting edges. This is expected because upstream innovations propagate along several paths. The panel therefore illustrates why an observed dependence matrix cannot simply be treated as the causal coefficient matrix. The generating graph is known by construction; visual inspection of correlation is not used to select or orient edges.

Without sensor noise, residual scoring with the correct supplied graph achieves 98.41% mean top-one accuracy, compared with 86.66% for the marginal baseline. Their respective 95% intervals are 98.17–98.65% and 86.02–87.29%. Deleting the access-to-execution edge lowers residual accuracy to 94.26%, with an interval of 93.90–94.63%. These differences demonstrate that conditional expectations can help isolate a shifted innovation in this generator, while also showing that part of the improvement depends on structural information supplied in advance.

Table 3. Synthetic localisation performance. Top-one entries show mean ± 95% interval half-width in percentage points; top-three entries are percentages.

Noise SD	Procedure	Top one %	Top three %	MRR
0	Marginal	86.66 ± 0.63	98.74	0.9251
0	Known graph	98.41 ± 0.24	99.94	0.9915
0	Missing edge	94.26 ± 0.36	99.50	0.9685
1	Marginal	61.64 ± 0.72	93.04	0.7727
1	Known graph	80.63 ± 0.53	97.50	0.8896
1	Missing edge	75.33 ± 0.59	97.07	0.8604

Top-three accuracy is high even when top-one performance is weaker. In the noiseless condition, the corresponding values are 99.94% for correct-graph residual scoring, 98.74% for marginal scoring and 99.50% for the missing-edge procedure. Mean reciprocal ranks are 0.9915, 0.9251 and 0.9685. Thus, the experiment distinguishes improved first-choice localisation from the easier task of including the true mechanism in a short candidate list.

4.2 Degradation under measurement and structural error

Figure 2 shows a decline in top-one accuracy as sensor noise increases. At a noise standard deviation of one, correct-graph residual scoring achieves 80.63%, compared with 61.64% for marginal scoring and 75.33% with the missing edge. The correct-graph advantage persists at this condition, but its absolute accuracy is materially below the noiseless result. An impressive result from clean structural data therefore cannot substitute for characterising the measurement process.

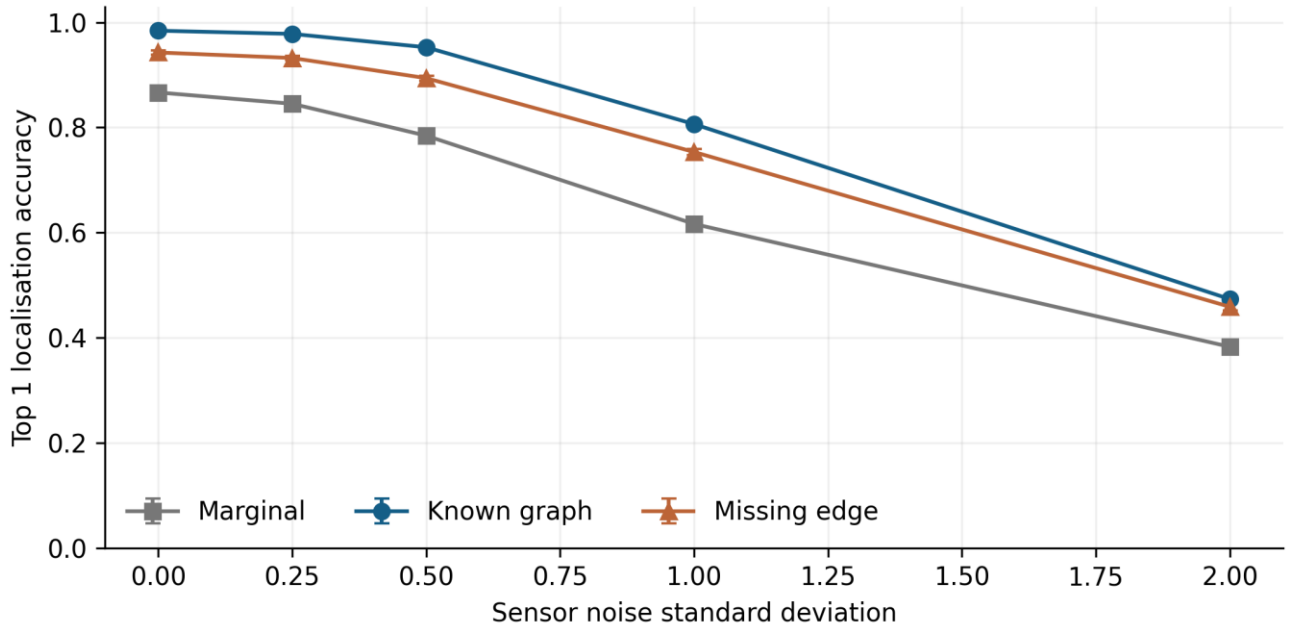


Figure 2. Localisation accuracy under sensor noise. Points are means over 30 seeds; bars are 95% Student t intervals over seed-level accuracies. Known graph denotes supplied latent structure, with coefficients learned from measured training data. All observations are synthetic.

The missing-edge condition is informative because the method still learns coefficients rather than merely applying the generator's equations. Execution is predicted without one of its true parents. An unusual access value can therefore leave an unexplained execution residual, creating a competing candidate downstream from the injected mechanism. Refitting the remaining parents cannot generally restore the missing structural relationship. This is a concrete example of how a local explanation can inherit a modelling error.

The experiment does not include latent confounding, adversarial log modification, nonlinear mechanisms or simultaneous causes. Measurement error is the only stochastic stress added beyond the baseline generator. These exclusions limit the interpretation of robustness: the observed ordering of methods applies to the tested conditions, not to every way an incident model can fail.

4.3 Error localisation and repair behaviour

Figure 3 pools 18,000 noiseless incidents for the correct-graph residual procedure into a row-normalised confusion matrix. Columns include the two distractor mechanisms, so an incorrect downstream selection remains visible. A small fraction of incidents are mislocalised even with the generating graph supplied because unrelated Gaussian innovations can produce larger residuals than the shifted mechanism. The four-unit intervention makes localisation favourable but does not make the shifted residual deterministically largest.

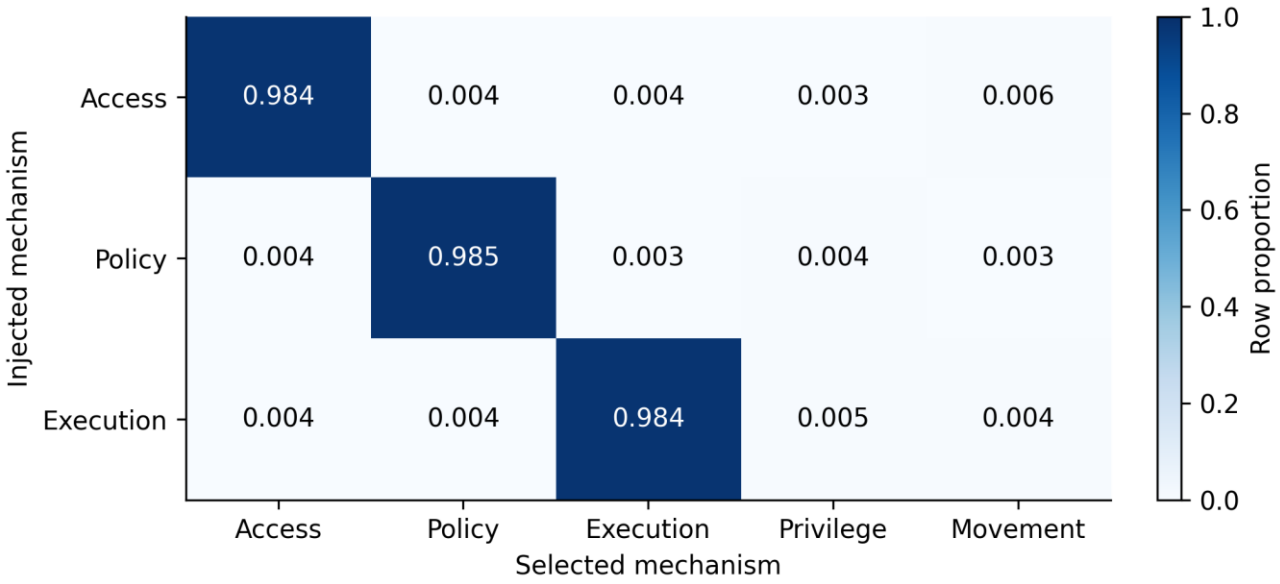


Figure 3. Mechanism confusion for the correctly structured residual procedure without sensor noise. Rows are injected causes and columns are selected candidates. Entries are row proportions pooled over 18,000 synthetic incidents. Privilege and movement are downstream distractors.

Figure 4 compares latent impact before and after simulator repair. Mean incident impact is 6.487 synthetic units. Reversing the true injected disturbance reduces the mean to 0.074; reversing the selected disturbance reduces it to 0.088. The near-zero repaired means are compatible with the zero-mean baseline innovations. They should not be reported as percentages of real attacks prevented or as measured reductions in organisational loss.

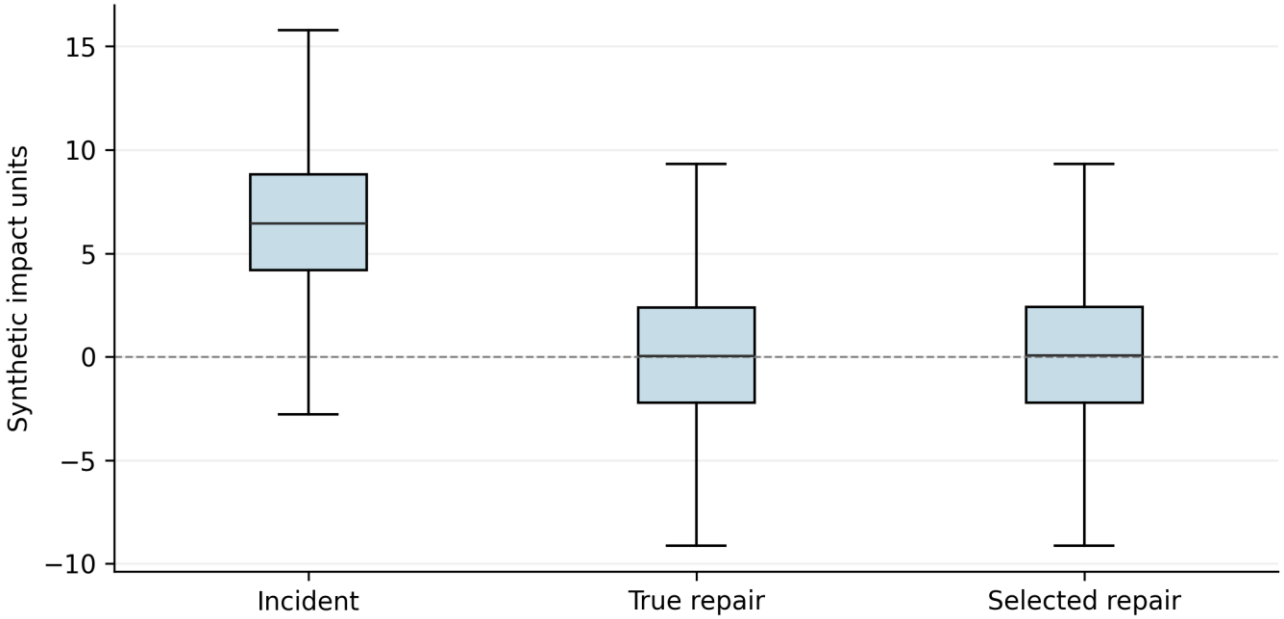


Figure 4. Simulator repair distributions for 18,000 noiseless synthetic incidents. Boxes show quartiles and medians; whiskers extend to the most extreme points within 1.5 interquartile ranges and outliers are hidden. True and selected repairs use the true propagation matrix and known four-unit shift. They are oracle-assisted checks, not learned remediation outcomes.

The boxplots retain substantial spread after repair because normal innovations still contribute to impact. Successful removal of the injected shift does not set every variable to zero. This distinction matters in interpreting counterfactual checks: an intervention can remove the specific abnormal contribution while

leaving ordinary variation intact. The residual between selected and true repair also reflects occasional localisation errors and differences in propagation strength among candidates.

5 Discussion

5.1 What the validation establishes

The findings support a bounded conclusion. Given a correct parent structure, normal training data and a single sufficiently large innovation shift, fitted local residuals can identify the changed mechanism more accurately than marginal deviations. The design separates a direct local innovation from an effect that is unusual because its parents changed. This is consistent with the broader motivation of causal anomaly attribution [4], but does not establish equivalence to more elaborate attribution methods.

The framework also clarifies what must be defended before a root-cause claim is operationally credible. A causal analysis needs a well-defined target, explicit structural assumptions and a measurement model. Surveys of causal learning emphasise the variety of tasks and assumptions involved [29,30]. In security investigation, these assumptions must connect with evidence such as process lineage, identity relationships and controlled tests. A fitted coefficient or model explanation cannot establish that connection on its own.

The repair check offers a useful distinction between explaining a mechanism and estimating an intervention. The localisation procedure learns regressions, but the repair calculation uses the simulator's true coefficients. Reporting both as one learned capability would overstate the study. Future work should replace the oracle calculation with independently fitted counterfactual models and compare their estimates against held-out interventions.

5.2 Implications for incident investigation

A defensible deployment would use the ranking to direct evidence collection. Each candidate should be accompanied by the parent values, residual magnitude, graph version and supporting telemetry. Analysts could then assess whether the proposed mechanism corresponds to an initiating action, an enabling condition or a downstream effect. These categories can overlap operationally even when the simulator assigns a unique injected location.

For example, an access anomaly may be upstream of execution in the graph, while an insecure policy enables both. Calling one the root cause depends partly on the investigator's intervention question. The present study avoids that ambiguity by defining ground truth as the location of an injected innovation. It does not identify a responsible person, a software vulnerability or an organisational failure. Operational studies must state their ground-truth definition with equal precision.

The procedure should also permit abstention when evidence is insufficient. A score gap can support a prioritisation rule, but it is not a calibrated causal confidence measure. This experiment neither selects an abstention threshold nor measures calibration. Learning such a threshold would require separate validation data and a cost model for missed causes, unnecessary investigation and harmful interventions. Table 4 specifies the principal boundaries and the evidence needed to cross them.

Table 4. Claim boundaries and required next validation

Current evidence	Boundary	Required next test
Known-graph simulation	No causal discovery claim	Estimate and validate structure independently
Single innovation shift	No multi-cause result	Vary shift size and inject simultaneous causes
Independent sensor noise	No missing-log robustness	Controlled missingness and delayed telemetry

Current evidence	Boundary	Required next test
Oracle-assisted repair	No learned action effect	Held-out interventions with fitted effect models
Abstract numerical signals	No enterprise effectiveness	Authorised testbed and operational validation
Uncalibrated score ranking	No autonomous response	Calibrated abstention and analyst assessment

5.3 Limitations and future validation

The graph has only six variables and seven edges. It cannot represent the scale, concurrency or identity ambiguity of enterprise telemetry. All mechanisms are linear, and ordinary least squares is well matched to the generator. Larger graphs with nonlinear, discrete or mixed mechanisms may change the relative performance of residual and marginal scoring. Comparing advanced models would be premature without first ensuring that each receives equivalent structural information.

The single four-unit shift is another important limitation. Smaller disturbances would be harder to distinguish from background variation, while multiple coordinated disturbances could make single-cause accuracy an inappropriate metric. Here, balanced cause frequencies simplify comparison and do not reproduce operational base rates. Incident detection precision and alert workload are therefore unmeasured. A ranking study of preselected incidents cannot establish detection quality.

No real audit logs were collected or analysed. The study cannot determine whether the six abstract variables are observable, whether an investigator would agree with the proposed causes or whether recommended actions improve recovery. A next validation stage should use an authorised isolated test environment with documented interventions, held-out attack scenarios and benign changes. It should assess label disagreement, time to localisation, computation cost and failures under missing or delayed observations.

Finally, deleting one edge tests only one form of misspecification. Additional studies should consider incorrect orientations, omitted common causes, poisoned training data and feedback. Correctly rejecting a misleading causal explanation may be more valuable than always returning a first-ranked mechanism. These extensions require new experiments and are not presented as completed work.

6 Conclusion

Mechanism residual scoring improves localisation of a single injected disturbance in a small, supplied linear causal graph. Across 30 seeded replications, the noiseless top-one accuracy is 98.41%, compared with 86.66% for marginal scoring. Sensor noise and a deleted edge reduce accuracy, demonstrating that structural knowledge and measurement quality are central to the result. The simulator repair check verifies internal propagation while relying on oracle information. The study provides reproducible synthetic validation and an explicit boundary between anomaly ranking, mechanism localisation and operational causal claims. Validation on authorised incident data and controlled interventions remains necessary before deployment.

Declarations

Ethics approval and consent to participate: The reported computations use generated numerical observations only and involve no human participants, animals, personal data or collected operational logs.

Consent for publication: No identifiable participant information is included.

Data availability: The generating equations, all parameter values and the executable numerical core are included in this manuscript. All reported observations are synthetic. No restricted dataset is required to reproduce the numerical results.

Code availability: Appendix A contains the numerical core for reproducing the summary metrics and simulator repair means. Figures are original plots generated from the described experiment.

Funding: Author confirmation required before submission.

Competing interests: Author declaration required before submission.

Author contributions: Rehana Devani must review the manuscript and confirm her contribution statement, accountability for the work and approval of the final version before submission.

Author information: The qualification supplied for Rehana Devani is presented as academic background. It does not establish a current institutional appointment. Current affiliation and corresponding author email must be confirmed before submission.

Use of artificial intelligence: Generative AI assisted with literature retrieval, manuscript drafting, simulation code and figure preparation. The named author must independently review the sources, methods, results and final wording and assumes responsibility for the submitted work. Generative AI is not listed as an author.

Acknowledgements: Any acknowledgements should be confirmed by the author before submission.

References

[1] Pearl J. Causality Models Reasoning and Inference. 2nd ed. Cambridge University Press; 2009. https://doi.org/10.1017/CBO9780511803161

[2] Peters J, Janzing D, Schölkopf B. Elements of Causal Inference Foundations and Learning Algorithms. MIT Press; 2017. https://mitpress.mit.edu/9780262037310/elements-of-causal-inference/

[3] Schölkopf B, Locatello F, Bauer S, Ke NR, Kalchbrenner N, Goyal A, Bengio Y. Toward causal representation learning. Proceedings of the IEEE. 2021;109:612–634. https://doi.org/10.1109/JPROC.2021.3058954

[4] Budhathoki K, Minorics L, Blöbaum P, Janzing D. Causal structure-based root cause analysis of outliers. Proceedings of Machine Learning Research. 2022;162:2357–2369. https://proceedings.mlr.press/v162/budhathoki22a.html

[5] Blöbaum P, Götz P, Budhathoki K, Mastakouri AA, Janzing D. DoWhy-GCM An extension of DoWhy for causal inference in graphical causal models. Journal of Machine Learning Research. 2024;25(147):1–7. https://jmlr.org/papers/v25/22-1258.html

[6] Sharma A, Kiciman E. DoWhy An end-to-end library for causal inference. arXiv preprint. 2020. https://doi.org/10.48550/arXiv.2011.04216

[7] Zheng X, Aragam B, Ravikumar PK, Xing EP. DAGs with NO TEARS Continuous optimization for structure learning. Advances in Neural Information Processing Systems. 2018;31. https://papers.nips.cc/paper/2018/hash/e347c51419ffb23ca3fd5050202f9c3d-Abstract.html

[8] Yu Y, Chen J, Gao T, Yu M. DAG-GNN DAG structure learning with graph neural networks. Proceedings of Machine Learning Research. 2019;97:7154–7163. https://proceedings.mlr.press/v97/yu19a.html

[9] Pamfil R, Sriwattanaworachai N, Desai S, Pilgerstorfer P, Georgatzis K, Beaumont P, Aragam B. DYNOTEARS Structure learning from time-series data. Proceedings of Machine Learning Research. 2020;108:1595–1605. https://proceedings.mlr.press/v108/pamfil20a.html

[10] Runge J, Nowack P, Kretschmer M, Flaxman S, Sejdinovic D. Detecting and quantifying causal associations in large nonlinear time series datasets. Science Advances. 2019;5:eaau4996. https://doi.org/10.1126/sciadv.aau4996

[11] Runge J. Discovering contemporaneous and lagged causal relations in autocorrelated nonlinear time series datasets. Proceedings of Machine Learning Research. 2020;124:1388–1397. https://proceedings.mlr.press/v124/runge20a.html

[12] Shimizu S, Hoyer PO, Hyvärinen A, Kerminen A. A linear non-Gaussian acyclic model for causal discovery. Journal of Machine Learning Research. 2006;7:2003–2030. https://jmlr.org/papers/v7/shimizu06a.html

[13] Peters J, Mooij JM, Janzing D, Schölkopf B. Causal discovery with continuous additive noise models. Journal of Machine Learning Research. 2014;15:2009–2053. https://jmlr.org/papers/v15/peters14a.html

[14] Chickering DM. Optimal structure identification with greedy search. Journal of Machine Learning Research. 2002;3:507–554. https://jmlr.org/papers/v3/chickering02b.html

[15] Peters J, Bühlmann P, Meinshausen N. Causal inference by using invariant prediction Identification and confidence intervals. Journal of the Royal Statistical Society Series B. 2016;78:947–1012. https://doi.org/10.1111/rssb.12167

[16] Lundberg SM, Lee SI. A unified approach to interpreting model predictions. Advances in Neural Information Processing Systems. 2017;30. https://papers.nips.cc/paper/2017/hash/8a20a8621978632d76c43dfd28b67767-Abstract.html

[17] Janzing D, Minorics L, Blöbaum P. Feature relevance quantification in explainable AI A causal problem. Proceedings of Machine Learning Research. 2020;108:2907–2916. https://proceedings.mlr.press/v108/janzing20a.html

[18] Rudin C. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nature Machine Intelligence. 2019;1:206–215. https://doi.org/10.1038/s42256-019-0048-x

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

[20] Arp D, Quiring E, Pendlebury F, Warnecke A, Pierazzi F, Wressnegger C, Cavallaro L, Rieck K. Dos and don'ts of machine learning in computer security. 31st USENIX Security Symposium. 2022:3971–3988. https://www.usenix.org/system/files/sec22-arp.pdf

[21] Hossain MN, Milajerdi SM, Wang J, Eshete B, Gjomemo R, Sekar R, Stoller S, Venkatakrishnan VN. SLEUTH Real-time attack scenario reconstruction from COTS audit data. 26th USENIX Security Symposium. 2017:487–504. https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/hossain

[22] Alsaheel A, Nan Y, Ma S, Yu L, Walkup G, Celik ZB, Zhang X, Xu D. ATLAS A sequence-based learning approach for attack investigation. 30th USENIX Security Symposium. 2021:3005–3022. https://www.usenix.org/conference/usenixsecurity21/presentation/alsaheel

[23] Cheng Z, Lv Q, Liang J, Wang Y, Sun D, Pasquier T, Han X. KAIROS Practical intrusion detection and investigation using whole-system provenance. IEEE Symposium on Security and Privacy. 2024:3533–3551. https://doi.org/10.1109/SP54263.2024.00005

[24] Jia Z, Xiong Y, Nan Y, Zhang Y, Zhao J, Wen M. MAGIC Detecting advanced persistent threats via masked graph representation learning. 33rd USENIX Security Symposium. 2024:5197–5214. https://www.usenix.org/conference/usenixsecurity24/presentation/jia-zian

[25] Harris CR, Millman KJ, van der Walt SJ, et al. Array programming with NumPy. Nature. 2020;585:357–362. https://doi.org/10.1038/s41586-020-2649-2

[26] Virtanen P, Gommers R, Oliphant TE, et al. SciPy 1.0 Fundamental algorithms for scientific computing in Python. Nature Methods. 2020;17:261–272. https://doi.org/10.1038/s41592-019-0686-2

[27] Hunter JD. Matplotlib A 2D graphics environment. Computing in Science and Engineering. 2007;9(3):90–95. https://doi.org/10.1109/MCSE.2007.55

[28] Pedregosa F, Varoquaux G, Gramfort A, et al. Scikit-learn Machine learning in Python. Journal of Machine Learning Research. 2011;12:2825–2830. https://jmlr.org/papers/v12/pedregosa11a.html

[29] Guo R, Cheng L, Li J, Hahn PR, Liu H. A survey of learning causality with data Problems and methods. ACM Computing Surveys. 2020;53(4):Article 75. https://doi.org/10.1145/3397269

[30] Vowels MJ, Camgoz NC, Bowden R. D'ya like DAGs? A survey on structure learning and causal discovery. ACM Computing Surveys. 2022;55(4):Article 82. https://doi.org/10.1145/3527154

Appendix A Reproducible numerical core

Run the following Python code with the library versions reported in Section 3.5. It reproduces all summary metrics and repair means. The same random draws and loop order are retained from the figure-generating implementation. Output includes all five noise levels; no private inputs are needed.

```
import numpy as np
from sklearn.linear_model import LinearRegression
from scipy.stats import t
W=np.zeros((6,6)); W[2,0]=.9; W[2,1]=.6; W[3,2]=1.1; W[4,1]=.7; W[4,3]=.8; W[5,3]=.8; W[5,4]=.9
A=np.linalg.inv(np.eye(6)-W)
labels=['Access','Policy','Execution','Privilege','Movement','Impact']
levels=[0,.25,.5,1,2]; records=[]; conf=np.zeros((3,5),int); repair=[]; example=None
for seed in range(30):
    rng=np.random.default_rng(seed)
    train=rng.normal(size=(2000,6))@A.T
    eps=rng.normal(size=(600,6)); root=np.tile(np.arange(3),200); eps[np.arange(600),root]+=4
    latent=eps@A.T
    for noise in levels:
        tr=train+rng.normal(scale=noise,size=train.shape)
        x=latent+rng.normal(scale=noise,size=latent.shape)
        z=np.abs((x-tr.mean(0))/tr.std(0))[:,:5]
        for variant in ['Marginal','Known graph','Missing edge']:
            if variant=='Marginal': score=z
            else:
                G=W.copy()
                if variant=='Missing edge': G[2,0]=0
                score=np.zeros((600,5))
                for j in range(5):
                    pa=np.flatnonzero(G[j])
                    if len(pa):
                        m=LinearRegression().fit(tr[:,pa],tr[:,j])
                        residual=tr[:,j]-m.predict(tr[:,pa])
                        score[:,j]=np.abs(x[:,j]-m.predict(x[:,pa]))/residual.std()
                    else: score[:,j]=z[:,j]
            order=np.argsort(-score,axis=1,kind='stable'); pred=order[:,0]
            ranks=(order==root[:,None]).argmax(1)+1
            records.append(dict(
                seed=seed, noise=noise, method=variant,
                top1=float(np.mean(pred==root)),
                top3=float(np.mean(ranks<=3)),
                mrr=float(np.mean(1/ranks))))
            if noise==0 and variant=='Known graph':
                for r,p in zip(root,pred): conf[r,p]+=1
                # Hypothetical four-unit reversal using the true propagation matrix.
                repair.extend(np.column_stack([
                    latent[:,5], latent[:,5]-4*A[5,root],
                    latent[:,5]-4*A[5,pred]]).tolist())
    if seed==0: example=train
summary=[]
for noise in levels:
 for method in ['Marginal','Known graph','Missing edge']:
  rr=[r for r in records if r['noise']==noise and r['method']==method]
  row={'noise':noise,'method':method}
  for metric in ['top1','top3','mrr']:
   v=np.array([r[metric] for r in rr]); row[metric]=float(v.mean());
row[metric+'_ci']=float(t.ppf(.975,29)*v.std(ddof=1)/np.sqrt(30))
  summary.append(row)

print(summary)
print("Confusion counts", conf)
print("Repair means", np.asarray(repair).mean(axis=0))
```