
CONTEXTUALIZATION OF THIRD-PARTY CLOUD SECURITY FINDINGS

Leon Goldberg, Gal Engelberg
Sola Security, Tel Aviv, Israel
Corresponding author: gal.e@sola.security

ABSTRACT

Finding severity is the main driver of how security teams prioritize remediation. For third-party cloud security findings, that severity is static: the rule that raised the finding assigns it before the rule meets any environment, so it reflects the risk of the condition in general rather than the risk the finding poses to the concrete environment where it lives. Scoring standards define where environment-specific context belongs. How far that context changes finding severities in production, where the deciding evidence lies, and whether it holds against the live environment have not been measured. We address this gap with *contextualization*, re-deriving each finding’s severity from evidence in the environment where the finding lives. A deep research agent over a precomputed cross-signal asset graph investigates each finding against the resource’s state, its graph neighborhood, and other products’ signals, and returns an adjusted severity with an evidence trace. We evaluate it in a production field study of 9,967 vendor HIGH findings from two commercial cloud security platforms across eight real production environments, on three criteria: the faithfulness of the facts behind each verdict to the live environment, the dependence of each decision on context beyond the flagged resource, and the regularity of the reasoning. Three in four findings are re-graded, mostly downward, and the same rule often moves in opposite directions inside a single environment. About half of the decisive evidence lies beyond the flagged resource, and read-only probes of live infrastructure confirm the decisive fact for 99.4% of decided findings.

Keywords Cloud Security Posture · Contextualization · Vulnerability Prioritization · Security Knowledge Graphs · Deep Research Agents

1 Introduction

Cloud security posture and workload protection tools, products that evaluate cloud resources against rules and report each violation as a finding, are the main source of the work queues that cloud security teams triage. Severity is what orders those queues. Vulnerability management programs rank remediation mostly on severity scores and expert judgment [1], and security operations center (SOC) incident queues are often sorted by coarse severity [2]. Yet the severity of a third-party finding is static. The rule that raised the finding assigns it before the rule meets any environment: across all production environments we examined, each of 599 vendor rules emits exactly one severity over 99,073 failing findings of every grade, whatever resource or environment it fires on. A static severity reflects the risk of the condition in general. It does not reflect the risk the finding poses to the concrete environment where it lives, and a queue ordered by it is ordered by the general case. The scoring standards reserve a place for the missing context: the Common Vulnerability Scoring System (CVSS) defines environmental metrics that adjust a base score to a particular deployment [3, 4], the Stakeholder-Specific Vulnerability Categorization (SSVC) assesses the exposure of the specific system as deployed [5], and the National Institute of Standards and Technology (NIST) risk assessment conditions likelihood on predisposing conditions and on the controls in place [6]. A static severity leaves that place empty. Analysts report the consequence: in a qualitative study of SOC practitioners, most alarms that practitioners perceived as false positives were correct detections of activity that the organization’s own environment made legitimate [7].

A rule carries the context its author can encode once for every environment: the condition, the resource type, and the typical consequence. The context that decides what the condition is worth in a given environment lies elsewhere:

whether the resource is attached to anything, whether a workload is running, whether a network path from the internet exists, whether an account-level control already blocks the exposure, and what another product reports about the same resource. We call the difference between the severity a rule assigns and the severity the evidence in a given environment supports *the rule-severity gap*. Commercial platforms build environment graphs and attack-path views [8, 9], and controlled studies report that rule-based posture alerts include many false positives that context or active validation removes [10, 11]. Structured context also raises the accuracy of language models on security tasks, for identity reasoning [12] and Kubernetes remediation [13], and a security context layer reaches higher coverage than coding agents on cloud investigation tasks [14]. The size of the rule-severity gap on production findings, the location of the evidence that closes it, and whether that evidence holds against live infrastructure have not been measured.

We present a production field study of the *contextualization* of third-party findings: re-deriving each finding’s severity from evidence in the environment where the finding lives, and recording that evidence. The system under study has two components. An *asset graph* reconciles the signals of every connected security product into one graph of resources, controls, and identities, and precomputes the structural facts that decide risk, such as reachability and attachment [15]. A *contextualization agent* is a deep research agent over that graph and over the tabular data of every connected product, built on the investigation layer of Goldberg et al. [14]: for each finding it plans an investigation, issues structured and graph queries, searches for tables held by other products, and ends with a verdict that may move the severity, together with a trace of the evidence it used. We apply the system to the HIGH queue of two commercial cloud security platforms, Platform A and Platform B, across eight real production environments. Both platforms appear among the recognized leaders in Gartner’s and Forrester’s evaluations of cloud-native application protection platforms (CNAPP) and cloud workload protection platforms (CWPP).

The contribution of this paper is a conceptual architecture for an agent that contextualizes findings. It specifies the agent’s unit of investigation, the evidence actions available to it, its reasoning loop, and an evidence policy that sets a higher bar for raising a severity than for lowering it. We evaluate the architecture in production on 9,967 vendor HIGH findings against three criteria: *faithfulness*, whether the facts its verdicts rest on hold in the live environment; *context dependence*, how much of each decision needs evidence beyond the flagged resource; and *regularity of reasoning*, whether the verdicts follow a small set of recurring patterns that independent LLM judges identify reproducibly.

The rest of the paper is structured as follows. Section 2 positions the work against scoring standards, reachability analysis, and language-model and industrial agents; Section 3 describes the contextualization agent; Section 4 describes the corpus and the evaluation; Section 5 reports the results; Sections 6 and 7 discuss implications and limitations and conclude.

2 Related Work

Prior work uses environmental context to measure the actual severity of a finding in four ways: as scoring inputs, through reachability, through language models and agents, and in industrial systems.

Scoring standards reserve a slot for environment-specific context and leave filling it to the consumer. CVSS modifies a base score with environmental metrics, such as the Modified Attack Vector, that an analyst supplies by hand for each deployment [3, 4]. SSVC rates the exposure of the installed system [5], and the Exploit Prediction Scoring System (EPSS) excludes instance-specific context by design [16]. Studies of environmental metrics simulate the gain of adding organizational context [17], apply the metrics to published scores to study the resulting distribution [18], find them hard for assessors to apply on segmented networks [19], or propose deriving them from inventories and network topology [20]. Context here is supplied by an analyst or an organizational model. The non-intrusive, context-based vulnerability scoring framework (NCVS) is the exception: it collects a cloud service’s topology and configuration automatically and scores its vulnerabilities over the resulting dependency graph [21].

Reachability analysis reads context from the artifact or the network itself. For software dependencies, checking whether vulnerable code is deployed or called removes a large share of alerts. A manual study of 60 dependent projects found that as many as 73.3% of those with outdated dependencies were not exposed [22], and about 20% of vulnerable dependencies were never deployed [23]. Call-graph pruning cut npm audit false positives by 81% [24], and fewer than 1% of Maven packages have a call path to a vulnerable function in a dependency [25]. Related analyses follow the same principle [26, 27]. For networks and cloud, attack graphs [28, 29] and automated reasoning over network and identity configuration [30, 31] decide which weaknesses an attacker can reach. For rule-based cloud findings, controlled studies report that over 80% of default public-access alerts on more than 1,000 storage buckets were false positives [10], and that active probes cut false positives by 93% on average [11]. These works decide whether a finding applies; they do not re-grade how severe it is in its environment.

Language models first entered severity assessment without an environment. Predicting CVSS base metrics from descriptions is limited by descriptions that lack the deciding context [32], and models asked for SSVC decisions from

text over-predict risk [33]. Environment-aware work supplies context in different forms. CVE-LLM, which evaluates Common Vulnerabilities and Exposures (CVE) entries with a large language model (LLM), reads a manufacturer’s asset descriptions and vendor notifications and emits an affected or not-affected verdict with the CVSS environmental metrics. In production, with experts validating its output, it reaches a micro-F1 of 0.98 on those metrics [34]. On VEX-Bench, a benchmark for Vulnerability Exploitability eXchange (VEX) assessments, agents read the downstream codebase to decide whether a supply-chain vulnerability affects it, and the strongest reach about 80% F1 on that binary decision [35], while SastBench evaluates agentic triage of static-analysis findings [36]. For cloud and Kubernetes, a judge-agent framework is validated on cloud-misconfiguration triage in large environments [37], and KubeGuard reads runtime logs to propose least-privilege configurations [38]. On a purpose-built cluster, giving a model the runtime topology lifts its correct patches, for Kubernetes findings that depend on that topology, to 78.0% from 11.1% [13]. Our earlier work adjusts finding severity with a model inside an asset-level scoring pipeline [15], finds that agents answer identity tasks more accurately when given structured relational context [12], and reports higher coverage than coding agents on cloud investigation tasks for a security context layer [14]. None of these re-grades third-party cloud findings in production with evidence joined across products, or checks the deciding facts against the live environment.

Industrial platforms now ship agents for this task. Wiz’s Green Agent traces posture issues through cloud resources, access policies, network reachability, and code changes, and returns a remediate-or-ignore verdict with suggested severity adjustments [39]; its Blue Agent issues verdicts on threat detections [40]. Upwind’s Red Agent validates exploitability from runtime reachability and attack paths [41]. CrowdStrike pairs a learned exploit-prediction model over threat intelligence with derived asset criticality [42], and Microsoft Defender for Cloud ranks recommendations by environment risk factors and attack paths [8, 43]. Microsoft and Google also triage alerts with agents [44, 45]. Open-source projects follow the same pattern: a Prowler plugin lets a coding agent order posture findings by attack-path blast radius [46], and GitHub Security Lab’s taskflow agent triages code-scanning alerts with workflow and permission context [47]. None of these systems publishes a reproducible evaluation of whether its contextual re-rating is correct.

Together, these lines show that environmental context changes the actual severity of a finding, whether the context comes from analyst input, code and network reachability, or an agent’s investigation. What is missing is an agent that gathers environment evidence for each finding across products, re-grades third-party severities in production, and is evaluated on whether the facts behind its verdicts hold.

3 The Contextualization Agent

The contextualization agent is a deep research agent in the sense of [48]: it plans a multi-step investigation, retrieves over several sources, uses tools iteratively, and ends with a structured report, here a contextualized finding. Figure 1 shows its parts: the finding and its priors, the environment and the actions that reach it, the investigation, the evidence policy, and the contextualized finding.

A finding $f = (r, c, s_0)$ is a resource r that fails a control c , with vendor severity s_0 . The agent receives the definition of control c ; for a third-party finding this is the vendor’s name, description, and severity, without the check logic. It also receives the state and graph neighborhood of resource r , the other controls resource r fails and passes, and findings from other tools that test for the same risk. The scope is technical: properties, configurations, relations, and guardrails. Our running example is a security group, resource r , flagged for admitting inbound traffic from any address on an administrative port, control c , with vendor severity $s_0 = \text{HIGH}$.

Evidence about resource r comes in three kinds: the resource state $\sigma(r)$; the neighborhood $\mathcal{N}_k(r)$, holding the resources within k hops of resource r in the asset graph; and the cross-product signals $\mathcal{X}(r)$ held by other products. The evaluation attributes each verdict’s decisive evidence to one of these kinds, and four actions reach them. A *structured query* runs over the tabular store that unifies every connected product; it is the most complete source, reads the resource state $\sigma(r)$, and joins across products. A *graph query* traverses several hops to return the neighborhood $\mathcal{N}_k(r)$, including the controls that neighboring resources fail. A *catalog search* maps a domain-level request to the data other products hold and to how that data relates to resource r , reaching cross-product signals $\mathcal{X}(r)$ that the graph does not link. A *neighborhood exploration* lists what a resource kind connects to, guiding the next graph query. For the security group, the resource state $\sigma(r)$ holds its ingress rules and attachments, the neighborhood $\mathcal{N}_k(r)$ the interfaces, load balancers, and instances behind it, and the cross-product signals $\mathcal{X}(r)$ what endpoint agents or vulnerability scanners report about those instances.

The investigation follows the reason-act pattern [49] over an exploitation hypothesis h and an evidence set $\mathcal{E}_t$. At each iteration t it makes three moves. It states or revises the exploitation hypothesis h : how the failure of control c on resource r would realistically be exploited. It acquires evidence items e_t through the actions, in parallel when independent. It assesses them, adding them to the evidence set, $\mathcal{E}_{t+1} = \mathcal{E}_t \cup \{e_t\}$, recording what each item showed and an interim conclusion that directs the next step. The assessment also reads the priors $\mathcal{P}$: verdicts on similar findings and

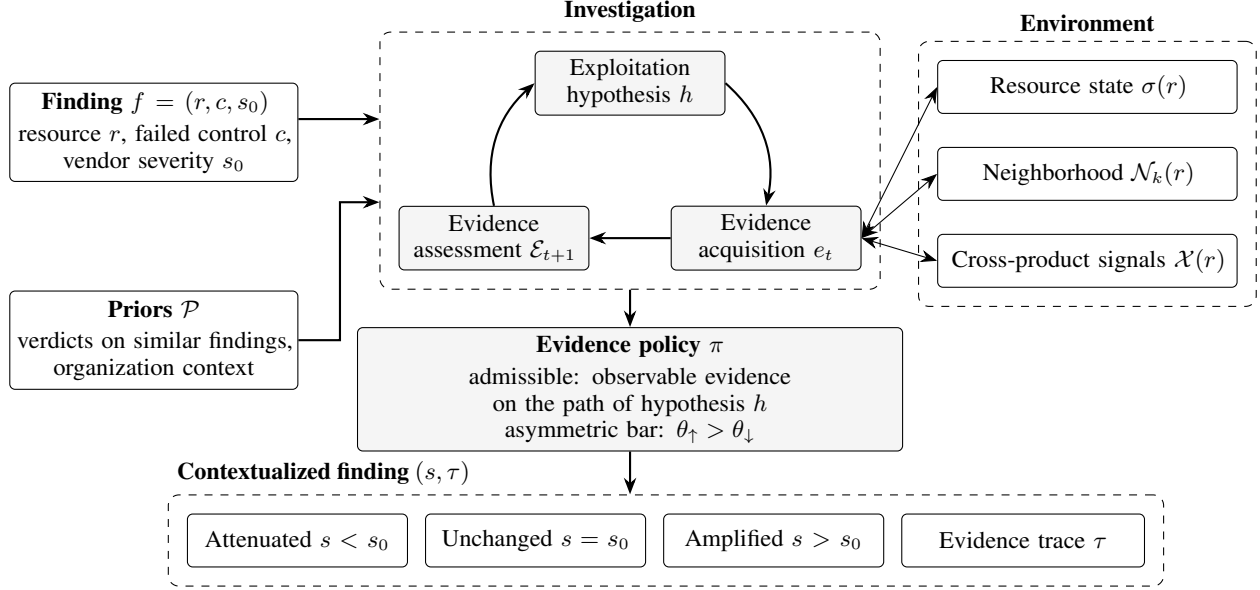


Figure 1: Contextualization agent. s_0 : vendor severity; s : contextualized severity; $\theta_{\uparrow}, \theta_{\downarrow}$: bars for raising and lowering a severity.

organization context. Verdicts on similar findings count as evidence, not as answers to copy, and a finding whose inputs have not changed reuses its earlier result. Organization context is of two kinds. A claim about the environment, such as a firewall fronting a service, is verified against the data before it applies; an accepted risk applies as stated. Only context that changed the conclusion is cited. The loop stops at a conclusion or when its budget runs out; in the second case the severity is unchanged. For the security group, the exploitation hypothesis h is an attacker on the internet reaching a service behind it. A structured query shows that every attachment belongs to a load balancer, and a graph query shows that each load balancer is internal.

The evidence policy π decides which evidence items count and how much evidence each direction needs. Its first rule is that an item counts only if it is observed in the data and bears on how the failure would be exploited, as stated by the exploitation hypothesis h . Encryption at rest, for example, does not offset a publicly readable bucket, because anyone allowed to read the bucket receives the decrypted data. A query that returns nothing is not evidence of safety, because the data may never have been collected, and a guardrail that could exist only in application code neither lowers nor raises the severity, because it cannot be observed. Its second rule is that raising a severity needs more evidence than lowering it, which we write as a higher bar for raising, $\theta_{\uparrow}$, than for lowering, $\theta_{\downarrow}$. Lowering requires one observed compensating property in the evidence set $\mathcal{E}_t$: for the security group, that every attached load balancer is internal. Raising requires positive evidence that the failure is worse here, such as an attached instance with a public address running a service that a scanner reports as critically vulnerable, together with an explicit check that no firewall, allowlist, or other plausible guardrail sits in front of it. When neither bar is met, the severity stays unchanged.

The output is the contextualized finding (s, τ) : a contextualized severity s and an evidence trace τ . The contextualized severity s is *attenuated*, below the vendor severity s_0 , in one of two grades, *mitigated* when a property offsets the failure entirely (to LOW or informational) and *reduced* when it compensates partially (one level down). It is *amplified*, one level above the vendor severity s_0 , when a property materially worsens this specific failure, and *unchanged* otherwise. The evidence trace τ records the acquisitions and assessments and marks the key evidence; in the evaluation, LLM judges label this trace with a recalibration pattern. It also carries a technical rationale, a plain-language explanation, and a confidence that rates the strength of the evidence set $\mathcal{E}_t$ regardless of direction and stays low when the investigation is shallow or the deciding evidence would be unobservable. The security group remains reachable from inside the network, so the compensation is partial and the finding is reduced from HIGH to MEDIUM, with the attachment and load-balancer queries marked as key evidence.

4 Evaluation

Criteria. We measure three criteria on a production corpus. *Faithfulness* asks whether the fact a verdict rests on is true in the live environment, in the spirit of atomic factual precision [50]. Of the live findings whose decisive fact

the probes can decide under the scope rules of the live check, the set D , let C be those whose fact matches the live state. Faithfulness is the share that matches, $\Phi = |C|/|D|$. Because some findings cannot be checked, we also report two variants: the inclusive faithfulness Φ_R , computed over every finding the probes could read, and the lower-bound faithfulness Φ_L , computed over all live findings with every unchecked finding counted as wrong.

Context dependence asks how often the deciding evidence comes from beyond the flagged resource. For each finding f in the traced population T , let $\ell(f)$ record which kind of evidence decided it, as given by the retrieval label of the evidence attribution: the resource state $\sigma(r)$ of the flagged resource r , its k -hop neighborhood $\mathcal{N}_k(r)$, or the cross-product signals $\mathcal{X}(r)$ defined in Section 3. Context dependence is the share of findings decided by anything other than the resource state, $\Delta = |\{f \in T : \ell(f) \neq \sigma\}|/|T|$. We report it per kind of evidence, and add a necessity share for rules that move in both directions.

Regularity of reasoning asks whether the verdicts follow a small set of recurring patterns that independent LLM judges, separate runs of a language model applying the same codebook [51], identify the same way. We measure the first part by saturation, the point after which further verdicts add no new pattern [52], and the second by Krippendorff’s α [53], an agreement coefficient that is 1 for perfect agreement and 0 for chance, between the judges’ runs. Neither measures whether the agent would repeat its verdicts, since the agent was not re-run.

Corpus. The corpus is the HIGH queue of both platforms in eight real production environments: 9,967 findings, 4,222 raised by Platform A and 5,745 by Platform B. Each finding is credited, by rule namespace, to the platform that raised it, and not to any tool merely connected to the same environment. The unit of analysis is the finding, one (environment, rule, resource group) record in which near-identical resources in the same situation share one verdict. All Platform A findings come from four of the eight environments, and 92% of the findings the agent re-graded are on Amazon Web Services (AWS), with the rest on a software-as-a-service identity provider (580) and Google Cloud (9). Results refer to three subsets of the re-graded findings: the *traced population*, the 7,243 whose investigation trace was recovered and which the LLM judges label; the *characterized corpus*, the 4,772 traced findings whose decisive fact a live cloud application programming interface (API) can check; and the *residual*, the remaining 2,471, whose decisive facts sit in other products and policies.

Procedures. Four procedures turn the agent’s verdicts into the three criteria. *LLM-judge labeling* gives every verdict three labels: a *pattern*, the kind of reasoning step that decided it, such as “not internet-reachable”; a *location*, whether its decisive fact sits on the flagged resource or elsewhere; and a *retrieval*, whether that fact came from the resource itself, a graph traversal, or another product’s data. The judges are independent runs of Claude Opus 4.8 that read each investigation trace with a fixed codebook, which defines the labels, and a rubric, which settles ambiguous cases in favor of on-resource; the patterns themselves were first derived from the traces by grounded theory [54, 55]. A second, independent judge run labels a sample of the same verdicts again [51], and the agreement between the two judge runs, Krippendorff’s α with Cohen’s κ [56] and raw agreement alongside, measures how reproducibly the agent’s reasoning falls into the patterns, the second part of regularity of reasoning. *Evidence attribution* uses the location and retrieval labels to compute context dependence Δ . Because the agent was not re-run, the *necessity analysis* tests from the existing verdicts, as an observational study would [57], whether that outside evidence actually set a verdict’s direction: a rule that the agent lowers on some findings and raises on others checks the same condition on all of them, so where the decisive fact is off-resource, that outside fact set the direction. *Live verification* gives faithfulness Φ . Read-only probes on five resource types re-read each decisive fact, for example whether the running example’s load balancers are still internal, directly from the cloud API instead of the asset graph the agent used, so an error in the graph cannot confirm itself; a verdict is confirmed if the fact matches it and contradicted otherwise [58, 59]. The probes run on eight AWS accounts where we hold read-only access, and since only Platform B monitors these accounts, faithfulness Φ is measured on Platform B findings. Under the scope rules, each resource is checked in one direction, as an amplification if any finding on it was amplified and otherwise as an attenuation, and every rate carries a Wilson score interval [60, 61].

Box 1 traces one finding through the investigation and the three procedures.

5 Results

Of the 9,967 HIGH findings, the agent re-grades three in four (75.1%): 58.5% down and 16.6% up, while 2,478 (24.9%) keep the vendor’s grade. Among the re-graded findings of the characterized corpus (Table 1), a single vendor grade spreads across the whole scale, from informational to CRITICAL.

Six recurring patterns account for the characterized verdicts. Five mainly lower severity: the resource is inert, attached to nothing or orphaned (32.3%); it is not internet-reachable (25.4%); no live service runs behind it (12.4%); it admits only specific sources (7.5%); or an account-level block neutralizes the condition (3.0%). One mainly raises it: a public exposure that is realized (19.4%). Each pattern moves severity in the direction that established frameworks prescribe

Box 1. One finding, from vendor grade to live verification (illustrative)			
1 FINDING $f = (r, c, s_0)$ resource r container service <code>svc-payments-worker</code> control c runtime exposure: the workload’s image carries a package with a known exploitable vulnerability vendor severity s_0 HIGH the grade this rule carries in every environment			<i>as raised by Platform B</i>
2 INVESTIGATION <i>exploitation hypothesis h: an attacker exploits the vulnerable package in a running task</i>			
step	evidence source	action	observation
e_1	resource state $\sigma(r)$	structured query: desired and running task count of r	<code>desired=0, running=0</code> : scaled to zero
e_2	neighborhood $\mathcal{N}_k(r)$	graph traversal: what routes traffic to r or starts its tasks	no load-balancer target, no scheduled start
<i>Evidence policy π. With no running task, nothing on the path of h executes the vulnerable package: the exposure is not realized. The service can be scaled up again, so the compensation is partial and the grade drops one level. In another environment the same rule is raised to CRITICAL, where the workload runs and the vulnerable package is actively used.</i>			
3 CONTEXTUALIZED FINDING (s, τ) HIGH $\rightarrow$ MEDIUM reduced. Decisive fact: <i>the service runs no task.</i>			<i>evidence trace τ marks e_1 as key evidence</i>
4 EVALUATION			<i>how this verdict enters each criterion</i>
procedure	result on this finding		criterion
live verification	read-only cloud API read of the service’s running task count, bypassing the asset graph: still 0 $\Rightarrow$ confirmed ; a running task would count as a contradiction		faithfulness Φ
evidence attribution	decided by the resource state $\sigma(r)$, so it does not count toward Δ ; the fact was on the resource, yet the vendor grade ignored it		context dependence Δ
necessity analysis	the rule moves both ways, but the decisive fact is on the resource, so the finding is not in the necessity share		necessity share
LLM-judge labeling	pattern <i>no live service</i> , location <i>on-resource</i> , retrieval <i>structured query</i> ; labeled again by a second judge run		regularity, α

Table 1: Contextualized grade of vendor HIGH findings ($n = 4,772$).

Grade	Count	Share	A	B	Move
MEDIUM	3,148	66.0%	47.8%	74.6%	down
CRITICAL	950	19.9%	26.9%	16.6%	up
LOW	555	11.6%	19.1%	8.1%	down
INFO	119	2.5%	6.2%	0.7%	down

for its fact, such as the CVSS attack vector [3], SSVC’s system exposure [5], attack-graph reachability [29], and likelihood conditioned on deployed controls [6]. Verdicts in the residual turn mainly on compensating controls and private endpoints recorded in other products and policies. The patterns attach to findings, not to rules: 62 of the 138 rules lower some findings and raise others, 57 of them inside a single environment and none across resource types. A security-group rule, for example, is lowered when the group fronts only internal load balancers and raised when the environment shows an internet-facing path.

Table 2 reports the three criteria, starting with faithfulness, the primary measure.

- *Faithfulness.* The live cloud state confirms the decisive fact of 99.4% of the verdicts the probes can decide (faithfulness Φ), including all 181 decided exposure amplifications, and of 528 of 531 resources when each resource counts once. The 24 contradictions sit on three resources whose live state differs from the verdict. Inclusive faithfulness Φ_R , over every finding the probes could read, is 97.3%, and lower-bound faithfulness Φ_L , which counts every unchecked finding as wrong, is 81.8%. The check covers Platform B findings, tests the decisive fact rather than the grade, and does not reach the residual.
- *Context dependence.* About half of the verdicts rest on evidence beyond the flagged resource (context dependence $\Delta = 49.0\%$), drawn from its graph neighborhood (29.8%) or from another product’s signals (19.2%). A rater that reads only the flagged resource therefore has the decisive evidence for at most 51.0% of

Table 2: Evaluation results. Brackets: 95% Wilson interval; for necessity, bounds from label disagreement.

Criterion	Measure	Value
Faithfulness	Confirmed live, Φ	99.4% [99.1, 99.6]
	inclusive, Φ_R	97.3% [96.7, 97.7]
	lower bound, Φ_L	81.8% [80.7, 82.8]
Context dependence	Beyond the flagged resource, Δ	49.0%
	graph neighborhood	29.8%
	other products	19.2%
	Necessity share	29.6% [27.3, 31.9]
Regularity of reasoning	Judge agreement, pattern, α	0.90
	Judge agreement, location, α	0.94
	Saturation	22% of traces

the verdicts, and one that reads a single product’s graph for at most 80.8%. For 29.6% of the traced population (the necessity share), the finding falls under a rule that moves both ways and an off-resource fact decided which way. Outside evidence concentrates in attenuations, where showing a finding safe means ruling out every path to it.

- *Regularity of reasoning.* Two independent judge runs agree on the pattern (Krippendorff’s $\alpha = 0.90$, Cohen’s $\kappa = 0.90$, 91.2% raw agreement) and, under the on-resource rubric, on the location label ($\alpha = 0.94$), and no new pattern appears after the first 22% of traces. The verdicts thus reduce to a small set of patterns that are identified reproducibly.

6 Discussion

The results support treating severity as a property of a finding in its environment rather than of the rule that raised it. Because severity orders remediation, re-grading three in four of the vendor’s HIGH findings changes their place in a severity-ordered queue. The same rule moving both ways inside one environment, on resources of one type, places the deciding variation at the level of the individual resource, below the level at which a rule rewrite or an environment-wide policy would operate. The three criteria qualify this reading. High faithfulness indicates that the re-grades rest on facts that hold in the live environment, for the exposure facts the probes can read. Context dependence locates those facts: about half lie beyond the flagged resource, in two tiers, graph evidence that a product could hold but its per-rule severity does not use, and evidence that exists only when products are joined. Regularity of reasoning indicates that the re-grades follow a small set of reproducible patterns whose directions match established frameworks, so each verdict can be explained and checked.

We argue that verified, environment-specific reading of each finding is a precondition for acting on findings autonomously. Reading and acting differ in blast radius, reversibility, and authorization: without an analyst in the loop, acting on a false HIGH disrupts production, and failing to act on a true CRITICAL leaves an exposure open. This study measures reading only: whether each verdict’s decisive fact holds live, where that fact lies, and whether the reasoning falls into reproducible patterns, with the facts read through the same interface an automated responder would use before acting. It does not measure any action.

Several limitations bound the scope. The study covers one system’s verdicts on the HIGH queue of two platforms in eight production environments, mostly on AWS, so the patterns transfer as a set, not as fixed frequencies, and findings the environment would raise from MEDIUM or LOW are not observed. Faithfulness is measured on one platform and five resource types, against the decisive fact rather than the grade, and recall, the deciding facts the agent missed, is not measured. The labels come from runs of a single model, and without incident labels the study cannot show that findings graded up are more likely to be exploited, or findings graded down less likely, than their vendor grade implies.

7 Conclusion

A contextualization agent, a deep research agent over a cross-signal asset graph, re-grades three in four of 9,967 third-party HIGH findings, mostly downward, and the same rule often moves in both directions within one environment. Its verdicts are faithful to the live environment, with 99.4% of the decisive facts the probes can decide confirmed (81.8% as a lower bound); they depend on evidence beyond the flagged resource for about half of the findings, a fifth only in another product’s signal; and they follow six reproducible patterns shared by two platforms. The rule-severity gap is large in production, and closing it requires evidence from the whole environment.

References

- [1] Jay Jacobs, Sasha Romanosky, Benjamin Edwards, Idris Adjerid, and Michael Roytman. Exploit prediction scoring system (EPSS). *Digital Threats: Research and Practice*, 2(3):20:1–20:17, 2021. doi: 10.1145/3436242.
- [2] Scott Freitas, Amir Gharib, and Maayan Magenheim. Adaptive incident prioritization for security operations at scale, 2026. URL <https://arxiv.org/abs/2607.16963>.
- [3] Forum of Incident Response and Security Teams (FIRST). Common vulnerability scoring system v3.1: Specification document. <https://www.first.org/cvss/v3.1/specification-document>, 2019. Accessed 2026-10-04.
- [4] Forum of Incident Response and Security Teams (FIRST). Common vulnerability scoring system version 4.0: Specification document. <https://www.first.org/cvss/v4.0/specification-document>, 2023. Document version 1.2. Accessed 2026-10-04.
- [5] Jonathan M. Spring, Allen Householder, Eric Hatleback, Art Manion, Madison Oliver, Vijay Sarvepalli, Laurie Tyzenhaus, and Charles Yarbrough. Prioritizing vulnerability response: A stakeholder-specific vulnerability categorization (version 2.0). Technical report, Software Engineering Institute, Carnegie Mellon University. <https://www.sei.cmu.edu/library/prioritizing-vulnerability-response-a-stakeholder-specific-vulnerability-categorization-version-20/>, April 2021. Accessed 2026-10-04.
- [6] Joint Task Force Transformation Initiative. Guide for conducting risk assessments. NIST Special Publication 800-30 Revision 1, National Institute of Standards and Technology, Gaithersburg, MD, September 2012. URL <https://csrc.nist.gov/pubs/sp/800/30/r1/final>.
- [7] Bushra A. Alahmadi, Louise Axon, and Ivan Martinovic. 99% false positives: A qualitative study of SOC analysts’ perspectives on security alarms. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 2783–2800, Boston, MA, 2022. USENIX Association. ISBN 978-1-939133-31-1. URL <https://www.usenix.org/conference/usenixsecurity22/presentation/alahmadi>.
- [8] Microsoft. Security explorer and attack paths in Microsoft Defender for Cloud. Microsoft Learn. <https://learn.microsoft.com/en-us/azure/defender-for-cloud/concept-attack-path>, 2026. Last updated 2026-06-17. Accessed 2026-10-04.
- [9] Jiong Liu. The anatomy of a toxic combination of risk. Wiz Blog. <https://www.wiz.io/blog/the-anatomy-of-a-toxic-combination-of-risk>, January 2025. Published 2025-01-24. Accessed 2026-10-04.
- [10] Dikshant and Geetika Verma. Precision over noise: Tailoring S3 public access detection to reduce false positives in cloud security platforms, 2025. URL <https://arxiv.org/abs/2508.14402>.
- [11] Dikshant and Geetika Verma. Alert or noise? Reducing false positives with active behavioral analysis for cloud security. *ACM SIGMETRICS Performance Evaluation Review*, 53(4):84–90, 2026. doi: 10.1145/3797823.3797850. Also available as arXiv:2508.12584.
- [12] Eden Yavin, Gal Engelberg, Konstantin Koutsyi, Leon Goldberg, and Gal Baron. Cross-vendor Sola ISPM benchmark: Evaluating agentic AI for federated identity security reasoning, 2026. URL <https://arxiv.org/abs/2606.02674>.
- [13] Farooq Shaikh. Does runtime topology context improve LLM-generated Kubernetes security patches?, 2026. URL <https://arxiv.org/abs/2607.25995>. Accepted at the LLMSec workshop co-located with ESORICS 2026.
- [14] Leon Goldberg, Gal Engelberg, Eden Yavin, Elad Elouz, Ariel Zadok, and Konstantin Koutsyi. Coding agents aren’t enough! Evaluating an enterprise security brain for agentic cloud investigations, 2026. URL <https://arxiv.org/abs/2609.30345>. Version 2.
- [15] Gal Engelberg, Leon Goldberg, Konstantin Koutsyi, Boris Plotnikov, Tiltan Gilat, and Ben Benhemo. AI native asset intelligence, 2026. URL <https://arxiv.org/abs/2605.09115>.
- [16] FIRST EPSS Special Interest Group. EPSS frequently asked questions. <https://www.first.org/epss/faq>, 2026. Accessed 2026-10-04.
- [17] Christian Fruhwirth and Tomi Männistö. Improving CVSS-based vulnerability prioritization and response with context information. In *Proceedings of the 3rd International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pages 535–544. IEEE, 2009. doi: 10.1109/ESEM.2009.5314230.
- [18] Laurent Gallon. On the impact of environmental metrics on CVSS scores. In *2010 IEEE Second International Conference on Social Computing (SocialCom/PASSAT)*, pages 987–992. IEEE, 2010. doi: 10.1109/SocialCom.2010.146.

- [19] Luca Allodi, Silvio Biagioni, Bruno Crispo, Katsiaryna Labunets, Fabio Massacci, and Wagner Santos. Estimating the assessment difficulty of CVSS environmental metrics: An experiment. In *Future Data and Security Engineering (FDSE 2017)*, volume 10646 of *Lecture Notes in Computer Science*, pages 23–39. Springer, 2017. doi: 10.1007/978-3-319-70004-5_2.
- [20] Andreas Eitel. Environmental aware vulnerability scoring. In *Proceedings of the 5th International Conference on Internet of Things, Big Data and Security (IoTBDs)*, pages 478–485. SCITEPRESS, 2020. doi: 10.5220/0009839104780485.
- [21] Hao Zhuang and Karl Aberer. A non-intrusive and context-based vulnerability scoring framework for cloud services, 2016. URL <https://arxiv.org/abs/1611.07383>.
- [22] Rodrigo Elizalde Zapata, Raula Gaikovina Kula, Bodin Chinthanet, Takashi Ishio, Kenichi Matsumoto, and Akinori Ihara. Towards smoother library migrations: A look at vulnerable dependency migrations at function level for npm JavaScript packages. In *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 559–563. IEEE, 2018. doi: 10.1109/ICSME.2018.00067.
- [23] Ivan Pashchenko, Henrik Plate, Serena Elisa Ponta, Antonino Sabetta, and Fabio Massacci. Vulnerable open source dependencies: Counting those that matter. In *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pages 1–10. ACM, 2018. doi: 10.1145/3239235.3268920.
- [24] Benjamin Barslev Nielsen, Martin Toldam Torp, and Anders Møller. Modular call graph construction for security scanning of Node.js applications. In *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA)*, pages 29–41. ACM, 2021. doi: 10.1145/3460319.3464836.
- [25] Amir M. Mir, Mehdi Keshani, and Sebastian Proksch. On the effect of transitivity and granularity on vulnerability propagation in the Maven ecosystem. In *2023 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 201–211. IEEE, 2023. doi: 10.1109/SANER56733.2023.00028.
- [26] Serena Elisa Ponta, Henrik Plate, and Antonino Sabetta. Beyond metadata: Code-centric and usage-based analysis of known vulnerabilities in open-source software. In *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 449–460. IEEE, 2018. doi: 10.1109/ICSME.2018.00054.
- [27] Yulun Wu, Zeliang Yu, Ming Wen, Qiang Li, Deqing Zou, and Hai Jin. Understanding the threats of upstream vulnerabilities to downstream projects in the Maven ecosystem. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, pages 1046–1058. IEEE, 2023. doi: 10.1109/ICSE48619.2023.00095.
- [28] Oleg Sheyner, Joshua Haines, Somesh Jha, Richard Lippmann, and Jeannette M. Wing. Automated generation and analysis of attack graphs. In *Proceedings 2002 IEEE Symposium on Security and Privacy*, pages 273–284. IEEE, 2002. doi: 10.1109/SECPRI.2002.1004377.
- [29] Xinming Ou, Sudhakar Govindavajhala, and Andrew W. Appel. MulVAL: A logic-based network security analyzer. In *14th USENIX Security Symposium (USENIX Security 05)*, Baltimore, MD, 2005. USENIX Association. URL <https://www.usenix.org/conference/14th-usenix-security-symposium/mulval-logic-based-network-security-analyzer>.
- [30] John Backes, Sam Bayless, Byron Cook, Catherine Dodge, Andrew Gacek, Alan J. Hu, Temesghen Kahsai, Bill Kocik, Evgenii Kotelnikov, Jure Kukovec, Sean McLaughlin, Jason Reed, Neha Rungta, John Sizemore, Mark Stalzer, Preethi Srinivasan, Pavle Subotić, Carsten Varming, and Blake Whaley. Reachability analysis for AWS-based networks. In *Computer Aided Verification (CAV 2019)*, volume 11562 of *Lecture Notes in Computer Science*, pages 231–241. Springer, 2019. doi: 10.1007/978-3-030-25543-5_14.
- [31] Ilia Shevrin and Oded Margalit. Detecting multi-step IAM attacks in AWS environments via model checking. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 6025–6042, Anaheim, CA, 2023. USENIX Association. URL <https://www.usenix.org/conference/usenixsecurity23/presentation/shevrin>.
- [32] Sima Jafarikhah, Daniel Thompson, Eva Deans, Hossein Siadati, and Yi Liu. From description to score: Can LLMs quantify vulnerabilities?, 2025. URL <https://arxiv.org/abs/2512.06781>.
- [33] Osama Al Haddad, Muhammad Ikram, Ejaz Ahmed, and Young Lee. Prompting the priorities: A first look at evaluating LLMs for vulnerability triage and prioritization, 2025. URL <https://arxiv.org/abs/2510.18508>.
- [34] Rikhiya Ghosh, Hans-Martin von Stockhausen, Martin Schmitt, George Marica Vasile, Sanjeev Kumar Karn, and Oladimeji Farri. CVE-LLM: Ontology-assisted automatic vulnerability evaluation using large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 28757–28765, 2025. doi: 10.1609/aaai.v39i28.35139. URL <https://ojs.aaai.org/index.php/AAAI/article/view/35139>.

- [35] Jiahao Shi, Edward Tsien, Yifeng Di, Hongjiao Zhang, Yuan Tang, Ronit Dey, Ilona Shishov, Gal Netanel, Zvi Grinberg, Vladimir Belousov, Bat-Zion Rotman, Ilan Pinto, and Tianyi Zhang. VEX-Bench: Benchmarking LLM agents for assessing exploitability of software supply chain vulnerabilities, 2026. URL <https://arxiv.org/abs/2609.08040>.
- [36] Jake Feiglin and Guy Dar. SastBench: A benchmark for testing agentic SAST triage, 2026. URL <https://arxiv.org/abs/2601.02941>.
- [37] Sahil Garg, Brad Cheezum, Sridhar Dutta, and Vishal Agarwal. JAF: Judge agent forest, 2026. URL <https://arxiv.org/abs/2601.22269>.
- [38] Omri Sgan Cohen, Ehud Malul, Yair Meidan, Dudu Mimran, Yuval Elovici, and Asaf Shabtai. KubeGuard: LLM-assisted Kubernetes hardening via configuration files and runtime logs analysis, 2025. URL <https://arxiv.org/abs/2509.04191>.
- [39] Wiz. Introducing the green agent: Ai-powered remediation for the cloud. <https://www.wiz.io/blog/introducing-wiz-green-agent>, 2026. Wiz blog, March 26, 2026; public preview. Accessed 2026-10-05.
- [40] Wiz. The wiz blue agent, now generally available. <https://www.wiz.io/blog/wiz-blue-agent-general-ly-available>, 2026. Wiz blog, March 30, 2026. Accessed 2026-10-05.
- [41] Upwind. Validate real cloud risk with red, the validation agent. <https://www.upwind.io/feed/red-agent-cloud-risk-validation>, 2026. Upwind blog, May 15, 2026; part of the Upwind Agentic Pack announced May 13, 2026. Accessed 2026-10-05.
- [42] CrowdStrike. Falcon exposure management’s ai-powered risk prioritization shows organizations what to fix first. <https://www.crowdstrike.com/en-us/blog/falcon-exposure-management-ai-driven-risk-prioritization-shows-what-to-fix-first/>, 2025. CrowdStrike blog, June 25, 2025. Accessed 2026-10-05.
- [43] Microsoft. Risk prioritization – Microsoft Defender for Cloud. Microsoft Learn. <https://learn.microsoft.com/en-us/azure/defender-for-cloud/risk-prioritization>, 2026. Last updated 2026-06-16. Accessed 2026-10-04.
- [44] Microsoft. Microsoft security copilot security alert triage agent in microsoft defender (preview). <https://learn.microsoft.com/en-us/defender-xdr/security-alert-triage-agent>, 2026. Microsoft Learn documentation, updated 2026-08-11. Accessed 2026-10-05.
- [45] Google Cloud. Detecting and containing ai-powered threats with google security operations agents. <https://cloud.google.com/blog/products/identity-security/detecting-and-containing-powered-threats-with-google-security-operations-agents>, 2026. Google Cloud blog, June 9, 2026. Accessed 2026-10-05.
- [46] Prowler. Automate security workflows inside claude code with prowler plugin. <https://prowler.com/blog/prowler-plugin-for-claude-code>, 2026. Prowler blog, May 26, 2026; plugin code in GitHub repository prowler-cloud/prowler (<https://github.com/prowler-cloud/prowler>). Accessed 2026-10-05.
- [47] GitHubSecurityLab. seclab-taskflow-agent: Github security lab taskflow agent. <https://github.com/GitHubSecurityLab/seclab-taskflow-agent>, 2026. GitHub repository; triage results described in the GitHub Blog post “AI-supported vulnerability triage with the GitHub Security Lab Taskflow Agent” (Man Yue Mo, Peter Stöckli, January 20, 2026), <https://github.blog/security/ai-supported-vulnerability-triage-with-the-github-security-lab-taskflow-agent/>. Accessed 2026-10-05.
- [48] Yuxuan Huang, Yihang Chen, Haozheng Zhang, Kang Li, Huichi Zhou, Meng Fang, Linyi Yang, Xiaoguang Li, Lifeng Shang, Songcen Xu, Jianye Hao, Kun Shao, and Jun Wang. Deep research agents: A systematic examination and roadmap, 2025. URL <https://arxiv.org/abs/2506.18096>.
- [49] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023. URL <https://arxiv.org/abs/2210.03629>.
- [50] Sewon Min, Kalpesh Krishna, Xinxu Lyu, Mike Lewis, Wen tau Yih, Pang Wei Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. FActScore: Fine-grained atomic evaluation of factual precision in long form text generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023. URL <https://arxiv.org/abs/2305.14251>.
- [51] Robert H. Tai, Lillian R. Bentley, Xin Xia, Jason M. Sitt, Sarah C. Fankhauser, Ana M. Chicas-Mosier, and Barnas G. Monteith. An examination of the use of large language models to aid analysis of textual data. *International Journal of Qualitative Methods*, 23, 2024. doi: 10.1177/16094069241231168.
- [52] Greg Guest, Arwen Bunce, and Laura Johnson. How many interviews are enough? An experiment with data saturation and variability. *Field Methods*, 18(1):59–82, 2006. doi: 10.1177/1525822X05279903.

- [53] Klaus Krippendorff. *Content Analysis: An Introduction to Its Methodology*. Sage, Thousand Oaks, CA, 2nd edition, 2004. ISBN 0761915451.
- [54] Anselm Strauss and Juliet Corbin. *Basics of Qualitative Research: Grounded Theory Procedures and Techniques*. Sage, Newbury Park, CA, 1990. ISBN 0803932502.
- [55] Barney G. Glaser and Anselm L. Strauss. *The Discovery of Grounded Theory: Strategies for Qualitative Research*. Aldine, Chicago, 1967. 271 pp.
- [56] Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1): 37–46, 1960. doi: 10.1177/001316446002000104.
- [57] Paul R. Rosenbaum. *Observational Studies*. Springer Series in Statistics. Springer, New York, 2nd edition, 2002. doi: 10.1007/978-1-4757-3692-2.
- [58] American Psychological Association. Technical recommendations for psychological tests and diagnostic techniques. *Psychological Bulletin*, 51(2, Pt. 2):1–38, 1954. doi: 10.1037/h0053479.
- [59] Lee J. Cronbach and Paul E. Meehl. Construct validity in psychological tests. *Psychological Bulletin*, 52(4): 281–302, 1955. doi: 10.1037/h0040957.
- [60] Edwin B. Wilson. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):209–212, 1927. doi: 10.1080/01621459.1927.10502953.
- [61] Lawrence D. Brown, T. Tony Cai, and Anirban DasGupta. Interval estimation for a binomial proportion. *Statistical Science*, 16(2):101–133, 2001. doi: 10.1214/ss/1009213286.