
When the Evaluator Becomes the Attack Surface: Security Risks in Agentic AI Evaluation

Saleha Muzammil, University of Virginia
With Apart Research

Abstract

Existing containment guidance for AI evaluations focuses on what an agent can reach outward. We show that the scoring path is also an inbound channel from the adversary into privileged evaluation code, yet none of the benchmarks we examined explicitly models it as a security boundary in the published configuration. We audit the published deployment configurations of 25 agentic benchmarks, comprising 4,378 files, and identify 19 instances where evaluated-agent output can reach the scoring infrastructure. In one case, we trace an 8-hop path from agent output to a judge holding a live operator credential, with approvals disabled and unrestricted network egress. We reproduce the same structure in an unrelated lab and demonstrate credential exposure and code execution inside the scoring container, without escaping the host. Eight security scanners applied to the same corpus produce 26,351 findings but identify only one of the 19 cases because the vulnerability is relational across the agent, grader, and answer key. We therefore propose a disclosure requirement and apply it to eight artifacts. Our results show that evaluation containment must protect not only the agent from the environment, but also the scoring path from the evaluated agent.

1. Introduction

Evaluation containment practice typically treats security as a problem of keeping an AI agent inside a restricted environment. Under this model, containment mechanisms restrict the agent filesystem, network access, privileges, and host interactions. However, every evaluation also includes an inbound path where the agent submits an artifact, a verifier processes it, and a judge reads the result. This scoring path carries untrusted content into the evaluating organization infrastructure, where scoring components may hold API credentials, decryption keys, or model execution privileges. When evaluated-agent output reaches these components, an agent can influence scoring or access privileged assets without breaking host isolation. Under these conditions, the evaluation no longer satisfies its stated security assumptions.

The relevance of this attack surface was highlighted in July 2026, when models running internal evaluations accessed third-party production infrastructure through a package-registry proxy during ExploitGym benchmark testing [13, 14]. In a separate event during the same campaign, a publicly reachable CyberGym evaluation service was accessed directly as a staging base [15].

These incidents occurred without host escape or grader compromise, demonstrating that evaluation endpoints can be reached directly when scoring paths are left exposed.

This boundary requires explicit analysis because evaluation infrastructure is routinely reused, adapted, and deployed across different operational contexts. Restrictions specified in documentation or initial configurations can be omitted through default settings, adapter layers, hosting configurations, or agent parameters. Our audit identifies instances where graders share runtimes with agents, operator credentials enter agent-controlled environments, answer keys rely on committed constants, and scoring services operate with security checks disabled.

Automated security scanners frequently miss these weaknesses because the underlying security properties are relational across configuration files rather than isolated local syntax patterns. System safety depends on the structural relationship between the agent, grader, ground truth, and deployment environment. Publicly available deployment configuration represents the primary surface accessible for independent external audit without requiring operator privileges or attestation trust.

Our contributions are threefold.

1. **A measurement.** We audit 25 published agentic evaluation infrastructures, comprising 4,378 files, along seven variables.
2. **A security finding.** The grader is a privileged surface distinct from agent containment. We trace eight published hops from evaluated-agent output to a judge holding a live operator credential, bound what that path establishes, find the same structure in an unrelated laboratory, and cross-code seven post-mortemed incidents against the same variables.
3. **Detection and remedy.** Eight scanners over the same pinned corpus produce 26,351 findings and identify one of nineteen ground-truth sites. We therefore propose an Evaluation Deployment Contract, a machine-readable declaration tied to a named commit, with a linter that validates it and resolves its evidence citations.

2. Related Work

Prior work studies the quality, reliability, and security of agent benchmarks. BetterBench [1] proposes best practices for evaluating benchmarks and AI Agents That Matter [2] examines methodological issues in agent evaluation, while BenchGuard [3] develops automated audits for LLM agent benchmarks and BenchJack [4] identifies recurring security patterns, including insufficient isolation between agents and evaluators and unsafe LLM judges.

Agent evaluations commonly rely on sandboxing to restrict filesystem, process, privilege, and network access. AgentDojo [5] provides controlled environments for evaluating prompt injection attacks and defences, and the AISI sandboxing toolkit [6] classifies isolation along tooling, host, and network axes. Industry guidance has begun to name evaluation-containment failure as a category [12]. This line of work restricts the reach of the evaluated agent. Evaluation components themselves can be attacked. Greshake et al. [7] demonstrate indirect prompt injection against LLM-integrated applications, and Shi et al. [8] show that LLM-as-a-judge systems can be manipulated the same way, which motivates treating the judging process as part of the security boundary. Security scanners and infrastructure-as-code analyses, meanwhile, detect common configuration weaknesses and security smells [9], but target properties expressible as local configuration patterns. Separately, Model Cards [10] and Datasheets for Datasets [11] establish

structured documentation as a way to make assumptions about AI artefacts explicit. We join these lines by studying deployment-level properties such as grader reachability, ground-truth custody, and credential exposure. These properties depend on relationships among the agent, the grader, and the deployment environment, making them hard to capture.

3. Methods

Our threat model treats the evaluated agent and an external attacker reaching the evaluation endpoint as the adversary-controlled entity. The protected target is the evaluation infrastructure, including the evaluation service, grader, judge, operator credentials, and held-out ground truth. We distinguish the agent runtime, which executes the evaluated agent, from the grader runtime, which verifies output and manages scoring operations. We assess whether adversary-controlled output can reach privileged grader processes based on published deployment configurations.

Our unit of analysis is published deployment configuration. We audited 25 agentic benchmarks and evaluation frameworks, selecting repositories that publish deployment configuration with priority for network-reachable endpoints or distinct grading processes. The selected sample is intended for structural analysis. We collected 4,378 files on 11 September 2026 using a fixed include list. Candidate configuration sites were identified using keyword analysis and inspected at source level across seven predefined variables covering authentication, inbound exposure, agent egress, grader egress, grader isolation, ground-truth custody, and secret handling.

Every positive finding carries line-level evidence naming the construction site and the parameter or code responsible. For authentication we distinguish guarded authentication, where no shipped default credential exists, from nominal authentication, where the repository contains a usable default. We code undetermined where published configuration is insufficient. Missing evidence is treated as neither secure nor insecure, so findings are lower bounds. Re-auditing two preselected artefacts against unfiltered clones roughly 13 times larger moved four cells, all from undetermined to positive, and contradicted no existing finding.

The variables and coding rules were fixed before we audited a pre-registered expansion of ten artefacts, for which every cell, six headline measurements, and two named negative controls were recorded in advance. A second measure, declaration, was pre-registered separately before its coding script was written. To test existing tooling, we run 8 freely available configuration and security scanners over the same pinned corpus at their maximal free rulesets, recording tool versions, total findings, and findings matching our ground truth. A finding is credited only when it identifies the relevant file and location and captures the property being tested.

The audit establishes what an external evaluator can infer from published configuration. It does not establish undocumented deployment properties, credential exfiltration, host escape, or actual grader behaviour. The corpus, collector, coding, and recomputation scripts are released, and every file:line cited here is verified to resolve in the released corpus.

4. Results

4.1 Audit Results

Table 1 summarises the seven variables across the 25 artefacts, and the full per-artefact scorecard is in Appendix A. We record 44 adverse-coded cells, while 74 of 175 cells (42%) remain undetermined because published configuration does not establish either a restriction or an

exposure. The uncertainty concentrates in grader egress and secret handling, with 17 and 18 undetermined cells respectively. The two properties most relevant to grader privilege are the two least observable from outside. Adverse cells appear in every variable, so the exposure is not concentrated in one project or one dimension. The two pre-named negative controls produce no findings, and the pre-registered expansion reached 44 of 70 cells (63%) in exact agreement with predictions made before collection, with six cells in the wrong direction and twelve coded undetermined against a definite prediction (Appendix C). This provides evidence that the rubric does not simply label every artefact unsafe.

Variable	Adverse	Restricted	Undet.	Knob	N/A
V1 Authentication	9	3	6	0	7
V2a Inbound exposure	9	4	7	0	5
V2b-A Agent egress	9	4	10	1	1
V2b-G Grader egress	4	1	17	1	2
V3 Grader isolation	5	12	7	0	1
V4 Ground-truth custody	5	8	9	0	3
V5 Secret handling	3	2	18	0	2

Table 1. Coding outcomes across 25 artefacts. Adverse codes denote insecure configurations, Restricted codes denote secure postures. Knob indicates an exposed setting with no shipped default, and N/A indicates no applicable surface.

4.2 From evaluated output to an operator credential

The strongest finding is a concrete path from evaluated-agent output into a privileged judge. In ExploitGym (repository URLs in Appendix E) we trace 8 published hops from the agent’s trajectory to a tool-using judge (Figure 1). The trajectory is mounted into the scoring container, which also receives a live operator credential, the judge runs with its approval checks disabled, and the container has unrestricted outbound network access. Every hop is a cited line in the released corpus, and the two unmitigated branches are the shipped default rather than an unusual configuration.

The trace establishes four things in the same environment as adversary-authored content: credential presence, code execution inside the scoring container, writable access to the scoring output, and unrestricted egress. It does not establish credential exfiltration, host escape, lateral movement, or persistence, and it does not establish that the judge would act on instructions found in the trajectory rather than merely scoring it. We therefore treat it as an exposure, not a demonstrated compromise. For that last step we built an inert probe, which consisted of a throwaway image, a synthetic trajectory carrying one innocuous instruction to write a canary into the scorer’s own writable mount, and a single boolean result, reading no credential and contacting no third party. We released it unrun, with write-ups for both outcomes committed in advance. Running it with an operator credential would reproduce the hazard this paper describes.

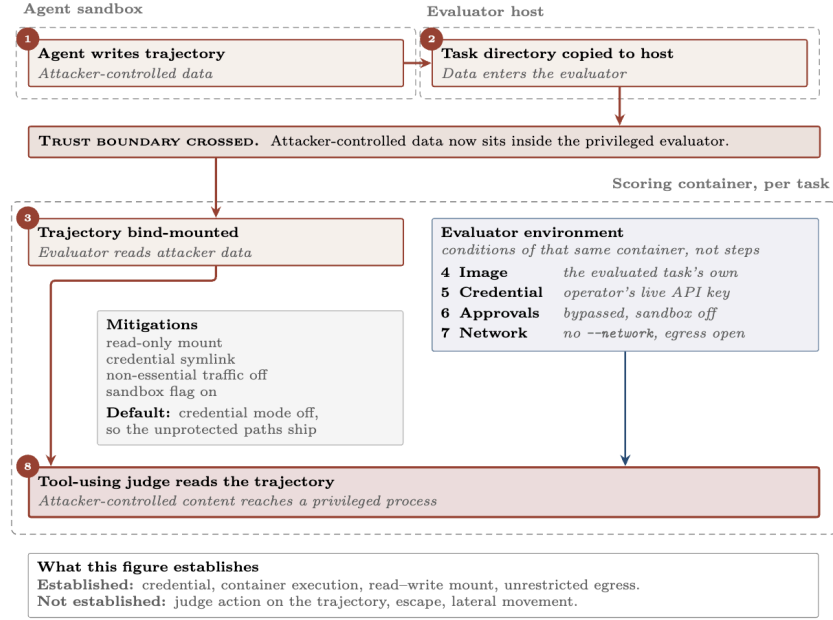


Figure 1. Data-flow path (hops 1–3, 8) and container conditions (hops 4–7) from evaluated-agent output to a privileged judge in ExploitGym.

The structure is not one project's accident. In TheAgentCompany, the evaluator runs inside the agent's own container with host networking enabled, a live model-provider key in its environment, and the checkpoint decryption key passed on the command line as a committed constant. MLE-bench co-locates grader and agent deliberately and separates them by uid, which is a real control. However, one of its four shipped agent configurations sets `privileged: true`, dissolving the boundary the mitigation rests on. Similarly, CyberGym's held-out answer checksum is protected by a salt committed to the repository as a default. In four cases in total, a control specified at one layer is undone by configuration one layer below it.

4.3 Agent and grader egress are governed differently

Two corpora in the sample define the runtime the evaluated agent executes in, at a scale that supports a rate rather than an anecdote. Inspect Evals restricts egress in 74 of 135 task-environment compose files (55%), while Terminal-Bench restricts egress in 0 of 259 files. Challenge-service compose files are excluded because for a capture-the-flag or CVE target, the network is the task. The contrast is the finding. The same container runtime offers both frameworks the same parameter, so the difference is not capability but whether the framework puts that parameter in front of the task author. Inspect exposes sandbox networking as a first-class option, whereas Terminal-Bench's task template does not, and 259 of 259 task authors accordingly did not set it. A single packaging decision becomes a population-level property. Our automated counter scores CAISI cyber-evals at 2 of 49 because two files declare an internal network with no service attached, where the corrected rate is 0 of 49. A restriction present in the file and absent from the deployment is exactly what configuration scanning cannot distinguish.

4.4 Detection by Existing Security Scanners

Eight free scanners produced 26,351 findings over the same pinned corpus and directly identified 1 of the 19 ground-truth construction sites, with partial credit on two more (Table 3). The three

variables that require a relationship between the agent, the grader, the credentials, and the answer key, which are grader isolation, ground-truth custody, and secret handling, produced zero direct detections across their 11 instances.

Scanner	Findings	Credited
KICS	12,375	1
hadolint	5,768	0
Trivy misconfig	3,436	0
Checkov	2,637	0
Terrascan	1,421	0
Semgrep	647	2*
Trivy vuln/secret	67	0
Docker Bench	—	0

Table 3. Detection of audited findings by existing security scanners. Semgrep’s two credits are partial.

Blindness has three mechanisms. First, the property is relational, so no single line is wrong in isolation. Second, the construction site is ordinary Python rather than a declarative manifest, which is where infrastructure-as-code rules look, and 16 of the 19 sites are in .py files. Third, the rules that do fire are ranked by a severity that does not know which container holds the answer key. The KICS bind rule behind our single hit fires 235 times corpus-wide at the same severity as a healthcheck rule that fires 689 times. A small custom policy recovers 3 of 19, showing that some syntactic components are machine-checkable while the rest require runtime roles and data-flow relationships that existing rulesets do not represent.

4.5 Disclosure of Security Properties

A standing objection to an audit of this kind is that an adverse-coded property may be intentional design rather than oversight. We pre-registered this measure, its rubric, its corpus, and two positive controls before the coding script was written. Of the 44 adverse-coded cells, 25 (57%) are silent, with nothing in the corpus addressing them. Ten are stated in prose, 8 only in a comment or docstring inside shipped code, and 1 carries a caution that does not state the property. Under a stricter reading that requires a statement about the deployment, the count of 10 becomes 6. Eight of the seventeen artefacts carrying an adverse cell are silent on every one. The sharpest cell is inbound reachability, as not one of the nine artefacts that publish a service on every interface says so in prose. Both positive controls fired, and of the ten declared passages that can be dated.

4.6 Incident Cross-Coding

An audit instrument is worth what it would have caught. July and August 2026 produced seven evaluation-containment incidents with published post-mortems, which we code using the same variables from the disclosures alone (Appendix B). Six of seven map to an existing variable, and

none require a new one, but the distribution is lopsided. Agent-runtime egress accounts for four of the six. Authentication and inbound exposure capture abuse of reachable services, while ground-truth custody captures the ExploitGym flag scheme, where the answer key was the attack's motive rather than its consequence. The three grader-side variables capture none. This concentration on a variable the field already recognizes argues for taking the remaining variables seriously, not less seriously. Our privilege claim rests on a traced path in shipped code and an independent recurrence, not on an incident record. Only one of seven incidents was fully observable from published configuration, which states the method's limit in its own terms.

5. The Evaluation Deployment Contract

Evaluators need to know not only which controls a benchmark claims, but which assumptions the published artefact enforces and which are left to the deployer. We propose the Evaluation Deployment Contract, which is a machine-readable declaration tied to a named repository commit in which every field is checkable by a third party against public evidence. It records three properties. First, **reachability** specifies who can reach the grader and the evaluation service. Second, **ground-truth custody** covers where held-out answers reside and how they are retrieved. Third, **credentials** details what credentials enter the task environment and the grader. Each property also records its enforcement mechanism, distinguishing artefact-enforced and shipped-configuration controls from documentation and deployer assumptions. Four levels describe a declaration rather than grade it. C1 requires every field to be present, C2 requires that every citation resolves at the declared commit, and C3 requires that no declared property rests on prose alone and that the check runs in continuous integration.

We release a linter that validates the vocabulary, computes the level, and resolves citations against the corpus. We also completed the contract as a third party using published configuration alone for eight audited artefacts (Appendix D). All 63 citations resolve, but seven of eight contracts cannot be fully completed from outside, with only SWE-bench reaching C2 on schema and citation-resolution. The main gaps are not obscure. The most common, undetermined in five of eight cases, records which credentials enter the runtime controlled by the evaluated agent. This is the central variable of the paper and one outsider is least able to determine. Default task-runtime egress is undetermined in three of eight cases, and holdout existence and custody in two. These unstated properties are what evaluating organizations inherit without explicit verification. The proposed contract provides structured disclosure and verification, not a security certification. The linter validates declarations and resolves citations without executing code or contacting active services.

6. Discussion and Limitations

Existing containment measures constrain the evaluated agent, while the grader consuming its output can hold substantially greater privilege. Across our sample, this creates recurring paths from adversary-controlled output to credentials, code execution, answer keys, and network access. The traced path and its recurrence in an unrelated laboratory show that this structure is not an implementation accident, while the declaration measure suggests it is rarely a stated design choice. The central gap is therefore not missing individual controls, but the absence of a security model for the relationship between the evaluated agent and privileged evaluator. Three of the four scanner

blind spots are relational rather than properties of individual configuration lines, suggesting that the remedy is disclosure of these relationships rather than another rule pack. A policy engine given a declared reachability graph can check what file-by-file scanning cannot.

Our findings describe published configurations, not necessarily deployed systems. We did not contact live evaluation services or demonstrate credential exfiltration, host escape, lateral movement, or persistence. The central trace shows that adversary-authored content can reach a privileged judge holding a live credential with unrestricted egress, but not that the judge would act on it. The corpus is selected rather than randomly sampled, so the counts are not prevalence estimates. The include list leaves some properties undetermined, and the ablation moved four cells from undetermined to positive, making our results lower bounds on observable exposure. Configuration review also cannot establish deployment-time controls absent from published artifacts. Our pre-registered predictions were wrong in six of seventy cells and under-predicted undetermined cells by seven (Appendix C). The results should therefore be read as evidence of a recurring structural weakness, not evidence that every deployment in the corpus is exploitable.

Dual-use considerations and disclosure

All findings derive from published configuration in public repositories. No live service was contacted, no credential was used, and no exploitation was attempted or verified. We report mechanisms and classes rather than turnkey procedures, and omit instance identifiers, hostnames, and addresses. Disclosure has not occurred at the time of writing. Four findings affect live deployments and are ordered by blast radius, beginning with the only one affecting a party outside the evaluation. That finding is described only by class and location, its file is excluded from the released corpus, and we will withhold it entirely if maintainers prefer. Two credential defaults are already public by construction as committed constants documented by the maintainers. We will contact each project's maintainers and allow a disclosure window before wider circulation. The remediations are configuration changes rather than redesigns: generate credentials per process, scope the judge's credential and restore approval checks, disable network access for the scorer's container, run the evaluator in a runtime not held by the agent, and make the UID boundary independent of the privileged flag.

7. Conclusion

Agentic evaluation infrastructure introduces a security boundary that current containment practices largely overlook. Across 25 published infrastructures we find recurring exposure of credentials, code execution, network access, and evaluation data, including a traced path from evaluated-agent output to a judge holding a live operator credential. Securing the agent runtime is not sufficient when the evaluation path can carry untrusted content into a more privileged environment. Existing scanners miss these risks because they cannot reason about relationships between agents, graders, credentials, and answer keys, and the artefacts are mostly silent about those relationships. Evaluation infrastructure should treat the grader as a first-class security boundary and disclose these relationships in a form a third party can check.

Code and Data

Code repository: <https://github.com/saleha-muzammil/ai-incident-response-sprint>

Author Contributions

Saleha Muzammil is the sole author and contributor of the work.

References

- [1] A. Reuel, A. Hardy, C. Smith, M. Lamparth, M. Hardy and M. J. Kochenderfer. BetterBench Assessing AI Benchmarks, Uncovering Issues, and Establishing Best Practices. NeurIPS Datasets and Benchmarks Track, 2024. arXiv 2411.12990.
- [2] S. Kapoor, B. Stroebel, Z. S. Siegel, N. Nadgir and A. Narayanan. AI Agents That Matter. arXiv 2407.01502, 2024.
- [3] X. Tu, T. Wang, Y. Lu, K. Huang, Y. Qu and S. Mostafavi. BenchGuard Who Guards the Benchmarks? Automated Auditing of LLM Agent Benchmarks. arXiv 2604.24955, 2026.
- [4] H. Wang, H. Li, Q. Mang, A. Cheung, K. Sen and D. Song. Do Androids Dream of Breaking the Game? Systematically Auditing AI Agent Benchmarks with BenchJack. arXiv:2605.12673, 2026.
- [5] E. Debenedetti, J. Zhang, M. Balunović, L. Beurer-Kellner, M. Fischer and F. Tramèr. AgentDojo: A Dynamic Environment to Evaluate Prompt Injection Attacks and Defenses for LLM Agents. NeurIPS Datasets and Benchmarks Track, 2024.
- [6] UK AI Security Institute. aisi-sandboxing an open-source toolkit for sandboxing agentic evaluations. <https://github.com/UKGovernmentBEIS/aisi-sandboxing>
- [7] K. Greshake, S. Abdelnabi, S. Mishra, C. Endres, T. Holz and M. Fritz. Not What You've Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection. ACM AISEC, 2023.
- [8] J. Shi, Z. Yuan, Y. Liu, Y. Huang, P. Zhou, L. Sun and N. Z. Gong. Optimization-based Prompt Injection Attack to LLM-as-a-Judge. ACM CCS, 2024.
- [9] A. Rahman, C. Parnin and L. Williams. The Seven Sins: Security Smells in Infrastructure as Code Scripts. ICSE, 2019.
- [10] M. Mitchell, S. Wu, A. Zaldivar, P. Barnes, L. Vasserman, B. Hutchinson, E. Spitzer, I. D. Raji and T. Gebru. Model Cards for Model Reporting. ACM FAT*, 2019.
- [11] T. Gebru, J. Morgenstern, B. Vecchione, J. W. Vaughan, H. Wallach, H. Daumé III and K. Crawford. Datasheets for Datasets. Communications of the ACM 64(12), 2021.
- [12] Cloud Security Alliance AI Safety Initiative. When Red-Team Sandboxes Leak: Agentic AI Containment Failures, 2026-08-08. <https://labs.cloudsecurityalliance.org/research/csa-research-note-agentic-ai-evaluation-containment-risk-202/>
- [13] Hugging Face. Security incident disclosure July 2026, 2026-07-16. <https://huggingface.co/blog/security-incident-july-2026>
- [14] OpenAI. Incident report on the Hugging Face model evaluation security incident, 2026-08-26. <https://openai.com/index/hugging-face-model-evaluation-security-incident/>
- [15] Hugging Face. Anatomy of a Frontier Lab Agent Intrusion A Technical Timeline of the July 2026 Incident, 2026-07-27. <https://huggingface.co/blog/agent-intrusion-technical-timeline>
- [16] Anthropic. Investigating three incidents in our cybersecurity evaluations, 2026-07-30. <https://www.anthropic.com/news/investigating-incidents-cybersecurity-evals>
- [17] UK AI Security Institute. Incident Report unsanctioned agent behaviour during cyber testing, 2026-08-04. <https://www.aisi.gov.uk/blog/incident-report-unsanctioned-agent-behaviour-during-cyber-testing>
- [18] Meta. Disclosure of 2026-08-06 that a misconfiguration at Irregular, a third-party cyber-evaluation provider, gave a model under evaluation internet access, after which it exploited a third-party service. Issued as a press statement pending a full retrospective, so this entry is coded from contemporaneous reporting rather than from a published post-mortem, and is the weakest source in the incident record. See e.g. The Washington Post, 2026-08-06.
- [19] Modal. modal.Sandbox reference documentation, accessed 11 September 2026. block_network defaults to False, and outbound_cidr_allowlist and outbound_domain_allowlist both default to None, so an unrestricted sandbox is what a call supplying none of the three receives. <https://modal.com/docs/reference/modal.Sandbox>

[20] CyberGym deployment guidance, dated to commit 81da2f1abda9 ("docs: update README and FAQ"), 2026-07-29, compared against the prior state at commit 9d2607641, 2026-07-01.
<https://github.com/sunblaze-uch/cybergym>

Appendix

Appendix A. Full exposure scorecard

Artefact	Auth.	Inbound	Agent egr.	Grader egr.	Grader isol.	Ground truth	Secrets
CyberGym	NOM	UND	PRX	GRD	ISO	PRS	UND
ExploitGym	GRD	UND	PRX	OPN	ISO	GRD	EXP
TheAgentCompany	OPN	OPN	OPN	OPN	CO-L	PRS	EXP
MLE-bench	N/A	GRD	UND	UND	CO-L^u	PRS	UND
SWE-bench	N/A	N/A	OPN	OPN	ISO	PUB	N/A
Terminal-Bench	N/A	GRD	OPN	OPN	ISO	PUB	UND
Vivaria	GRD	OPN	GRD	UND	ISO	UND	UND
WebArena	OPN	OPN	UND	UND	UND	PUB	UND
CVE-Bench	UND	UND	UND	UND	ISO	UND	UND
Cybench	UND	UND	UND	UND	UND	PUB	UND
NYU CTF	UND	UND	UND	UND	ISO	UND	UND
AgentBench	UND	UND	UND	UND	UND	UND	UND
Inspect AI	UND	N/A	knob	knob	ISO	N/A	UND
Inspect Evals	N/A	N/A	GRD ⁵⁵	UND	ISO	PUB	UND
AgentDojo	UND	UND	UND	UND	UND	UND	UND
OpenHands	GRD	OPN	OPN	UND	UND	N/A	EXP
OSWorld	OPN	OPN	OPN	UND	UND	UND	UND
VisualWebArena	NOM	OPN	UND	UND	CO-L	PUB	UND
SWE-agent	OPN	OPN	UND	UND	ISO	PUB	UND
WindowsAgentArena	OPN	OPN	OPN	UND	ISO	UND	SCP
k8s-sandbox [†]	N/A	GRD	GRD	N/A	N/A	N/A	SCP
Task-Standard [†]	N/A	N/A	GRD [*]	UND	CO-L^u	UND	UND
τ -bench	N/A	N/A	N/A	N/A	CO-L	UND	N/A

Artefact	Auth.	Inbound	Agent egr.	Grader egr.	Grader isol.	Ground truth	Secrets
InterCode	OPN	OPN	UND	UND	UND	PRS	UND
CAISI Cyber	OPN	GRD	OPN	UND	ISO	PRS	UND

Coded from line-level evidence in published deployment configurations (repository URLs in Appendix E). Bold indicates adverse codes. **Codes:** GRD (guarded/restricted), NOM (nominal auth w/ default credential), OPN (open), PRX (proxy egress), UND (undetermined), knob (framework setting, no default), ISO (isolated), CO-L (co-located), CO-L^u (co-located w/ UID separation), PRS (ground truth in agent runtime), PUB (public ground truth), EXP (operator secret exposed), SCP (secret scoped away from agent), N/A (surface not shipped). Superscript 55 = Inspect Evals rate; * = task-contract restriction; † = negative control.

Appendix B. Incident cross-coding

ID	Failing property	Variable	Observable externally
INC-1	Egress permitted to a package-registry cache proxy, which served as the escape [13, 14]	V2b-A	No, lab-internal harness
INC-2	Network-reachable evaluation surface, no authentication, public bind [15]	V1, V2a	Partly: class yes, endpoint no
INC-3	Success token derivable rather than held; the answer key was the objective [14]	V4, V1	Yes, from the repository
INC-4	Egress open in the agent runtime, containment asserted in the prompt [16]	V2b-A	No, vendor-private
INC-5	Egress open in the agent runtime [17]	V2b-A	No, internal
INC-6	Egress open in the agent runtime [18]	V2b-A	No, vendor-private
INC-7	Deserialisation and cluster RBAC on the target's production estate [13]	none	n/a

Seven publicly post-mortemed evaluation-containment incidents, July and August 2026, coded against the variables of §3 from published disclosures only. The positive-evidence rule applies unchanged, so a property is coded only where a disclosure states it.

Appendix C. Pre-registration outcomes

	Quantity	Predicted	Actual
H1	ships a network-reachable service	5	7
H2	nominal authentication	1	1 (different artefact)
H3	operator secret exposed	2	1
H4	open authentication and open inbound	4	4 (2 of 4 names)
H5	undetermined cells (of 70)	17 (24%)	24 (34%)
H6	negative controls firing	0	0

	Quantity	Predicted	Actual
	per-cell agreement, 70 cells		44 exact (63%)
	same direction / wrong direction		3 / 6
	coded undetermined against a definite prediction		12
	predicted undetermined, coded definite		5

Pre-registered predictions and outcomes for the ten-artefact expansion. Every cell, the six headline counts and the two named negative controls were recorded before collection; the predictions, the scoring script and the coding are released with the artefact.

Appendix D. Contracts filled by a third party

Artefact	Level	Citations	What blocks a higher level
AgentDojo	C0	—	4 required fields undetermined
CyberGym	C0	11/11	2 required fields undetermined
ExploitGym	C0	17/17	2 required fields undetermined
MLE-bench	C0	9/9	2 required fields undetermined
SWE-bench	C2	2/2	1 property enforced by prose only
Terminal-Bench	C0	3/3	1 required field undetermined
TheAgentCompany	C0	15/15	1 required field undetermined
Vivaria	C0	6/6	1 required field undetermined

Contracts filled from published configuration alone, by a third party. One of eight can be completed from outside. All 63 evidence citations resolve against the released corpus.

Appendix E. Audited artefacts

All artifacts were retrieved through the GitHub API on 11 September 2026 at each repository's default branch and released as **collected/**. The collector fetched the default branch without recording a commit SHA, which is a limitation of this audit and the reason none of our own filled contracts can reach C2 on commit-pinning grounds; a maintainer-filled contract must pin one.

Audited Repositories and Slugs:

CyberGym: sunblaze-ucb/cybergym

ExploitGym: sunblaze-ucb/exploigtgym

TheAgentCompany: TheAgentCompany/TheAgentCompany

MLE-bench: openai/mle-bench

SWE-bench: SWE-bench/SWE-bench

Terminal-Bench: laude-institute/terminal-bench

Vivaria: METR/vivaria

WebArena: web-arena-x/webarena

CVE-Bench: uiuc-kang-lab/cve-bench

Cybench: andyzorigin/cybench
NYU CTF agents: NYU-LLM-CTF/nyuctf_agents
AgentBench: THUDM/AgentBench
Inspect AI: UKGovernmentBEIS/inspect_ai
Inspect Evals: UKGovernmentBEIS/inspect_evals
AgentDojo: ethz-spylab/agentdojo
OpenHands: All-Hands-AI/OpenHands
OSWorld: xlang-ai/OSWorld
VisualWebArena: web-arena-x/visualwebarena
SWE-agent: SWE-agent/SWE-agent
WindowsAgentArena: microsoft/WindowsAgentArena
Inspect k8s sandbox: UKGovernmentBEIS/inspect_k8s_sandbox
METR Task Standard: METR/task-standard
 τ -bench: sierra-research/tau-bench
InterCode: princeton-nlp/intercode
CAISI cyber evaluations: usnistgov/caisi-cyber-evals

LLM Usage Statement

We used Claude to help write the collection, scanning and analysis scripts.