

From Artifacts to Risk: Auditing Instruction Surfaces in Agent Systems

A Bottom-Up, Artifact-Centric Approach to Agent Security

Sergey Gordeychik
CyberOK Research
gordey@cyberok.ru

April 2026

Abstract

Agentic systems increasingly rely on persistent instruction artifacts, tool integrations, and repository-level configuration that shape behavior beyond individual prompts. Prior work has established prompt injection, indirect instruction attacks, tool poisoning, and agent hijacking as practical security concerns. Less attention, however, has been given to the repository layer as a persistent and auditable source of agent behavior.

This paper presents a bottom-up, artifact-centric audit of instruction surfaces in agent systems. We analyze a purposive corpus of 509 instruction-rich repositories containing agent guidance files, skills, plugin manifests, and Model Context Protocol (MCP) related artifacts. The scan produced 4,882 medium-or-higher raw findings and 4,637 clustered issue instances. The resulting distribution is highly concentrated: the top six repositories, approximately 1.2% of the corpus, account for 63.2% of raw findings; the top 10% account for 90.0%. The raw-finding and clustered-issue Gini values are 0.940240 and 0.939578, respectively, both rounding to 0.940.

The contribution is not a new prompt-injection benchmark or a replacement for existing scanners. Instead, this study integrates heterogeneous signature sources, applies them to real repositories, correlates raw detections into artifact-level issue instances, and maps the resulting evidence to an ASAMM-aligned agent-security interpretation layer. We explicitly treat detector outputs as candidate evidence rather than proof of exploitability. The paper positions instruction surfaces as repository-level control-plane artifacts and argues that agent security practice needs artifact-level auditing alongside runtime testing and defense.

1 Introduction

Agentic AI systems increasingly perform multi-step tasks by combining model reasoning, tool invocation, local files, network resources, and persistent project context. This changes the security boundary. A conventional prompt is transient, but many modern agent environments also consume repository-level artifacts that persist across sessions and are versioned with the project. Examples include `AGENTS.md`, `CLAUDE.md`, `SKILL.md`, plugin manifests, tool descriptions, and MCP configuration files.

This paper treats such artifacts as *instruction surfaces*: persistent repository objects that can influence agent behavior, tool choice, authorization assumptions, or task execution. The central claim is deliberately narrow. Instruction surfaces are not automatically vulnerabilities, and static matches are not exploit proofs. Instead, they are auditable evidence about how a repository exposes instructions, capabilities, and trust assumptions to agentic tools.

The study is motivated by a practical assurance problem. Security teams need to understand where agent risk enters the development workflow before a runtime incident occurs. Prompt-injection benchmarks and runtime harnesses are essential, but they usually evaluate specific tasks or interaction traces. Repository artifacts are different: they are reusable, reviewable, and often shared across teams and tools. They therefore provide a natural bottom-up measurement layer for agent security.

Contributions. This paper makes four contributions.

1. **Repository-level measurement of instruction surfaces.** We analyze 509 instruction-rich repositories containing guidance files, skills, plugin manifests, and MCP-related artifacts.
2. **In-the-wild evaluation of heterogeneous signature sources.** We apply native ASAMM detectors together with ATR-derived, Aguara-derived, and Cisco AI Defense skill-scanner / PromptGuard-style rule families, treating their outputs as candidate evidence rather than ground truth.
3. **Correlation from raw findings to artifact-level issue instances.** We introduce a post-processing layer that harmonizes severity, clusters overlapping rule hits, maps findings to canonical classes, and separates raw detector output from interpreted evidence.
4. **ASAMM-aligned interpretation of repository artifacts as control-plane surfaces.** We map concrete artifact-level signals to concerns such as autonomy boundaries, tool governance, context handling, trust-boundary expansion, and self-modification.

Scope. This is an exploratory empirical study of a purposive public-repository corpus. It should not be read as an estimate of GitHub-wide prevalence, a corpus-wide precision measurement, or an exploitability study. The goal is to show how bottom-up repository evidence can support agent-security triage and framework mapping.

2 Background: Instruction Surfaces as Control-Plane Artifacts

Agent systems combine at least four layers: the model, transient prompts, persistent context, and tool interfaces. The last two layers are increasingly stored in repositories. Instruction files tell agents how to behave in a project; skills encode reusable procedures; plugin and MCP manifests expose actions, endpoints, or tool metadata.

This paper uses the phrase *control plane* by analogy to infrastructure and distributed systems: a layer that shapes what actions are possible and how they are authorized or interpreted. Instruction artifacts do not execute by themselves, but they shape execution when consumed by an agent runtime.

Table 1: Instruction-surface artifact types analyzed in the study.

Artifact type	Security-relevant function	Representative risk signal
Agent guidance files	Define project-specific behavior, priorities, task norms, and constraints.	Instruction override, role manipulation, unsafe delegation.
Skills and reusable prompts	Encode repeatable agent procedures and domain-specific capabilities.	Overbroad autonomy, unsafe command guidance, hidden exfiltration paths.
Plugin manifests	Declare external tools, APIs, actions, or integration affordances.	Confused-deputy behavior, excessive tool scope, unreviewed external action.
MCP manifests and configuration	Expose tools and resources through agent-accessible protocol surfaces.	Tool poisoning, remote capability expansion, authorization mismatch.
Persistent identity or memory artifacts	Influence behavior across sessions or project contexts.	Persistent identity rewrite, policy drift, cross-session manipulation.

3 Related Work

3.1 Prompt Injection and Indirect Instruction Attacks

Prompt injection has become one of the central security problems for LLM-integrated applications. Greshake et al. introduced indirect prompt injection as an attack class in which untrusted external content is interpreted as instructions by an LLM-integrated application, enabling remote manipulation of application behavior [1]. Liu et al. later formalized prompt injection attacks and defenses and evaluated multiple attacks, defenses, models, and tasks in a benchmark setting [2]. These works establish the core security problem that natural-language content can cross the boundary between data and instruction.

Our unit of analysis is different. Rather than evaluating prompt-injection attacks as runtime scenarios, we analyze persistent repository artifacts that may later be consumed by agents as instructions, policies, skills, or tool definitions. This shifts the measurement layer from transient prompts to version-controlled artifacts.

3.2 Agent Benchmarks and Runtime Security Evaluation

Recent agent-security benchmarks evaluate how agents behave under adversarial runtime conditions. AgentDojo provides an extensible benchmark for prompt-injection attacks and defenses in tool-using agents and reports 97 realistic tasks with 629 security test cases [3]. Agent Security Bench (ASB) similarly evaluates agent attacks and defenses across multiple LLM backbones, attack methods, and defense methods at runtime [4]. Such benchmarks are essential because they measure whether an agent fails during execution.

This work is complementary. It measures whether repositories contain persistent instruction and integration artifacts that encode risky patterns before execution occurs. In this sense, the study is closer to static software analysis than to behavioral benchmarking.

3.3 Agent Skills and Persistent Instruction Artifacts

Agent skills and repository-level instruction files have recently emerged as a distinct security concern. Skill-Inject studies attacks delivered through skill files and reports that frontier agents can execute harmful instructions, including exfiltration and destructive behavior, when skill files contain malicious or context-dependent injections [5]. Li et al. analyze the architecture and lifecycle of agent skills, identifying structural risks such as weak data-instruction boundaries, persistent trust after approval, and limited marketplace security review [6].

These papers provide direct evidence that instruction artifacts are not merely documentation. They can function as persistent behavioral inputs to agent systems. Our work contributes a complementary empirical layer: rather than crafting benchmark attacks against skill files, we measure instruction-surface patterns across a corpus of public repositories and interpret them through an artifact-centric audit model.

3.4 Tool Use, MCP, and Tool-Poisoning Risks

Tool integration changes the threat model for LLM applications. In agent systems, natural-language instructions can influence external actions through tool calls. MCP has accelerated this pattern by standardizing how assistants discover and invoke external tools. Hou et al. analyze the MCP ecosystem and identify security and privacy risks across the MCP server lifecycle [7]. Huang et al. focus on MCP threat modeling and tool poisoning, finding that malicious instructions embedded in tool metadata are a major client-side vulnerability [8].

This study includes plugin and MCP manifests as instruction surfaces because they encode capabilities, trust assumptions, and action boundaries. This extends the repository audit beyond prompt files and skills to tool-integration artifacts.

3.5 Rule-Based Scanners and AI Security Testing Tools

A growing ecosystem of AI-security tools detects prompt injection, exfiltration, excessive autonomy, unsafe tool execution, and related risks. Agura scans AI agent skills and MCP server configurations using static analysis [9]. Agent Threat Rules (ATR) provides a YAML-based rule format for agent-security threats such as prompt injection, credential theft, supply-chain manipulation, privilege escalation, and data exfiltration [10]. Cisco AI Defense publishes tools for scanning agent skills, MCP servers, and IDE-level AI assets [11]. NVIDIA Garak and Microsoft PyRIT address broader LLM and generative-AI red teaming, while Promptfoo provides application-level adversarial testing workflows [12, 13, 14].

This work does not claim to replace these tools. Instead, it uses heterogeneous signature sources as candidate evidence, then adds a repository-level correlation layer: severity harmonization, deduplication, canonical class mapping, and ASAMM-aligned interpretation. The contribution is therefore not a new signature pack alone, but an empirical application and interpretation layer over multiple rule families.

3.6 Frameworks, Risk Management, and ASAMM Positioning

General AI risk-management frameworks such as the NIST AI Risk Management Framework provide governance vocabulary for mapping, measuring, and managing AI risk [15]. Secure AI development guidance from NCSC and partner agencies emphasizes secure design, secure development, secure deployment, and secure operation for AI systems [16]. OWASP SAMM provides a broader maturity model for software assurance [17].

ASAMM [18] is used in this paper as an agentic extension and organizing framework for mapping artifact-level findings to concerns such as autonomy boundaries, tool governance, context handling, and self-modification. Because ASAMM and the scanner used in this study (**agent-audit** [19]) are developed by the author, this paper should be read as an application and stress test of an author-proposed framework rather than independent validation of an established standard. We therefore make the mapping explicit, preserve third-party rule provenance, and separate detector output from interpreted risk claims.

4 Comparator Baselines and Positioning

The closest comparators are not generic LLM red-team frameworks alone, but tools that inspect skills, MCP configurations, or instruction-like artifacts. Table 2 summarizes the relationship. This is a positioning comparison, not an apples-to-apples scanner-output benchmark; output-overlap benchmarking is listed as future work and a threat to validity.

Table 2: Comparison with adjacent tools and studies. The study is positioned as a repository-level empirical audit and correlation layer over instruction surfaces, not as a replacement for runtime benchmarks or existing scanners.

System or study	Primary unit of analysis	Main strength	Difference in this work
Aguara	Agent skills and MCP server configurations.	Static scanning of skills and MCP artifacts.	Repository-level measurement and ASAMM-aligned canonical class mapping.
ATR / Panguard	YAML detection rules for agent threats.	Open rule format for agent threat patterns.	ATR-derived rules are treated as candidate evidence and correlated across repositories.
Cisco AI Defense skill-scanner	Skills, MCP configs, and IDE-level agent assets.	Multi-engine skill and MCP scanning in developer workflows.	This study integrates PromptGuard-style patterns with other sources and evaluates them in repository context.

System or study	Primary unit of analysis	Main strength	Difference in this work
NVIDIA Garak	LLM or LLM-application probing.	Broad red-team probes for leakage, prompt injection, jailbreaks, and model failures.	Static artifact analysis rather than model-interaction probing.
Microsoft PyRIT	Generative-AI red-team orchestration.	Repeatable adversarial testing workflows.	Repository artifacts are measured before runtime deployment.
Promptfoo	Application-level adversarial testing.	Practical red-team tests and policy checks.	Persistent instruction artifacts are the measurement object.
AgentDojo	Runtime tasks and tool-using agents.	Realistic benchmark environment for attacks and defenses.	Static repository evidence rather than runtime attack success.
Skill-Inject	Skill-file attack benchmark.	Demonstrates harmful runtime behavior from crafted skill files.	Provides prevalence-oriented corpus measurement across broader instruction surfaces.
Towards Secure Agent Skills	Skill architecture and threat taxonomy.	Lifecycle and taxonomy for skill risks.	Adds empirical corpus evidence and ASAMM-aligned issue mapping.

5 Methodology

5.1 Corpus Construction

The study analyzes a corpus of 509 public repositories selected for instruction-rich agent development artifacts. The corpus includes a curated instruction-centric cohort and a broader discovery cohort. The corpus is purposive: it is intended to characterize repositories where instruction surfaces are present, not to estimate prevalence across all public repositories.

The scan covers 20,241 files and emits 4,882 medium-or-higher raw findings. After clustering and deduplication, these correspond to 4,637 clustered issue instances. Repository identities and path-heavy raw sidecars are omitted from the public summary bundle; aggregate analysis artifacts, sanitized triage outputs, and figure sources are released as the public `article-support-dataset-v1` [20], including the corpus list, repo metadata, sanitized corpus-wide `scan-project` output, summary analysis files, and the per-record adjudication exports used in Section 5.5.

5.2 Signature Sources and Rule Provenance

The scanner combines native ASAMM detectors with imported or adapted rule packs. Imported signatures are treated as candidate evidence. Severity labels are harmonized for analysis, but the study does not assume that severity is calibrated identically across sources. The counts in Table 3 are the bundled or filtered counts used in this study, not necessarily current upstream project counts. The current artifact package does not pin upstream commit SHAs for all imported rule families, which limits exact third-party reproduction; pinned snapshots and filter criteria are recorded as a reproducibility limitation in Section 8.

Table 3: Signature sources integrated by the scanner. Counts reflect the snapshot used for this study and should be read as bundled or filtered rule counts, not necessarily current upstream project counts.

Source	Rules	Primary coverage	Role in this study
ATR-derived	233	Prompt injection, agent manipulation, excessive autonomy, skill compromise, tool poisoning, context exfiltration, and related agent-centric attack patterns.	Largest imported signature family; broad agent-threat coverage.
Aguara-derived	37	External download/install behavior, third-party content ingestion, SSRF-cloud patterns, and remote input or remote execution surfaces.	Trust-boundary and remote-content coverage.
Cisco AI Defense skill-scanner / PromptGuard-style	26	PII harvesting, secret patterns, markdown/data-URI exfiltration, and prompt/output abuse patterns.	Data exposure and exfiltration-oriented coverage.
Native ASAMM detectors	–	Structural gaps such as broad external action without approval, autonomous loops with writes, and persistent identity rewrite.	Framework-aligned structural detection.
Post-processing layer	–	Severity harmonization, clustering, canonical class mapping, surface aggregation, and ASAMM mapping.	Converts raw hits into artifact-level issue instances and aggregate evidence.

5.3 Evidence Hierarchy

The study uses a four-layer evidence hierarchy to avoid overclaiming from raw matches. Table 4 defines these layers.

5.4 Clustering and Canonical Classes

Raw detector outputs can overlap when several rules identify related text or behavior in the same artifact. The post-processing layer clusters findings into artifact-level issue instances and maps them to canonical classes. The clustering key is intentionally conservative: it collapses overlapping signatures that point to the same artifact-level class, but it does not merge across repositories or aggressively

Table 4: Bottom-up evidence hierarchy used in the study.

Evidence layer	Definition	Interpretation	Primary use
Raw finding	A single detector output produced by a native or imported rule.	Candidate evidence only; not equivalent to a vulnerability or exploit.	Initial scan signal.
Clustered issue instance	A grouped artifact-level issue after de-duplication and correlation across overlapping rules.	Primary unit for issue-centric analysis.	Main empirical analysis layer.
Aggregate signal	A repeated pattern observed across related artifacts, categories, or repositories.	Useful for navigation and trend discovery, but weaker than file-backed evidence.	Corpus-level interpretation.
Adjudicated label	A reviewed label on a stratified subset of evidence packets.	Interpretive stability check; not a corpus-wide precision estimate.	Validity and calibration.

merge sibling files. As a result, clustering improves interpretability and reduces cross-signature duplication without being expected to change repository-level rank ordering.

In this release, the corpus contains 4,882 medium-or-higher raw findings and 4,637 clustered issue instances, a reduction of 245 records (5.0%). This reduction is modest at corpus level, but methodologically important: it shows that the heavy-tail pattern is not primarily an artifact of repeated signatures firing on exactly the same evidence. In other words, clustering changed the evidence layer used for interpretation more than it changed the concentration curve.

Table 5: Effect of conservative clustering on corpus-level counts and concentration. Similar Gini values should be read as evidence that concentration is rank-stable after de-duplication, not as evidence that clustering is unnecessary.

View	Count	Gini	Interpretation
Raw detector findings	4,882	0.940240	Signature-level candidate evidence; includes overlapping rule hits.
Clustered issue instances	4,637	0.939578	Artifact/class-level issue view after conservative correlation.
Reduction	245 (5.0%)	—	Removes local cross-signature duplication while preserving repository-level heavy-tail structure.

In the issue-centric view, 4,629 of 4,637 clustered issue instances are mapped to canonical classes; 8 remain unmapped.

5.5 Three-LLM Auto-Triage and Adjudication Audit

A stratified subset of 54 evidence packets was reviewed by three LLM-based reviewer channels: OpenAI ChatGPT 5.5, Claude Opus 4.7, and OpenAI ChatGPT 5.4 via the Codex desktop interface.¹ The subset is used to assess interpretive stability and packaging quality. It is not a precision estimate, not human ground truth, and not a claim about the full corpus. The reviewer interfaces were treated as operationally separate triage channels, not statistically independent raters; two channels are from the same vendor family, and all labels may share correlated failure modes.

The reviewer-channel mapping used in the adjudication tables is: Channel 1 = OpenAI ChatGPT 5.5, Channel 2 = Claude Opus 4.7, and Channel 3 = OpenAI ChatGPT 5.4 via the Codex desktop interface. Stable API model strings, exact deployment snapshots, and decoding parameters were not available in the public artifact package; this is recorded as a reproducibility limitation rather than inferred.

The corpus-wide `scan-project` results reported in this paper were produced by rule-based static scanning and deterministic post-processing. They were not produced by running automatic LLM triage over the full corpus. LLM review was applied only to the stratified 54-packet adjudication subset. `agent-audit` [19] also implements optional LLM verifier paths for interactive and batch review through `llm_verifier.py` and `batch_verifier.py`; these are tool capabilities rather than the source of the corpus-wide measurements. The reviewer rubric and prompt templates were fixed in the artifact package, including `reviewer-prompt.txt`, `adjudication-guidelines.md`, and verifier prompt templates in `llm_verifier.py` and `batch_verifier.py`. For the CLI verifier path, operational constraints such as `-allowedTools ""`, `-max-turns 1`, and read-only/no-approval style execution were fixed. Temperature, `top_p`, seed, stable API model strings, and exact deployment snapshots were not uniformly fixed or visible for all adjudication interfaces.

Each reviewer assigned one of three labels: TP, FP, or UNCERTAIN. Channel 1 produced 34 TP, 17 FP, and 3 UNCERTAIN labels; Channel 2 produced 26 TP, 27 FP, and 1 UNCERTAIN; Channel 3 produced 21 TP, 25 FP, and 8 UNCERTAIN. Pairwise agreement ranged from 0.685 to 0.815 and pairwise Cohen’s kappa ranged from 0.476 to 0.656. Fleiss’ kappa across the three channels is 0.562, computed from the released per-record label table. These values indicate moderate agreement and reinforce the need to treat non-unanimous cases conservatively.

¹Model names are reproduced as displayed in the consumer interfaces used during the adjudication run (April 2026). Stable API model strings and exact deployment snapshot dates were not visible in those interfaces; this is recorded as a reproducibility limitation rather than inferred. Two of the three channels (ChatGPT 5.5 and ChatGPT 5.4 / Codex) are from the same vendor family, so their labels may share correlated failure modes.

Table 6: Three-LLM auto-triage label distribution and agreement. The table describes the adjudication harness, not corpus-wide scanner precision.

Reviewer channel	TP	FP	UNCERTAIN	Notes
OpenAI ChatGPT 5.5	34	17	3	Most permissive on artifact-backed risk signals.
Claude Opus 4.7	26	27	1	Balanced TP/FP split with few UNCERTAIN labels.
OpenAI ChatGPT 5.4 / Codex desktop	21	25	8	Most conservative and most likely to mark ambiguity.
Agreement summary	Pairwise agreement: 0.685–0.815			Pairwise Cohen’s kappa: 0.476–0.656; Fleiss’ kappa: 0.562.

6 Results

6.1 Surface Prevalence

Table 7 summarizes artifact coverage in the corpus. The difference between 418 instruction-file repositories and 422 repositories with any counted instruction surface is intentional: the latter also includes plugin and MCP-related surfaces.

Table 7: Instruction-surface coverage in the 509-repository corpus.

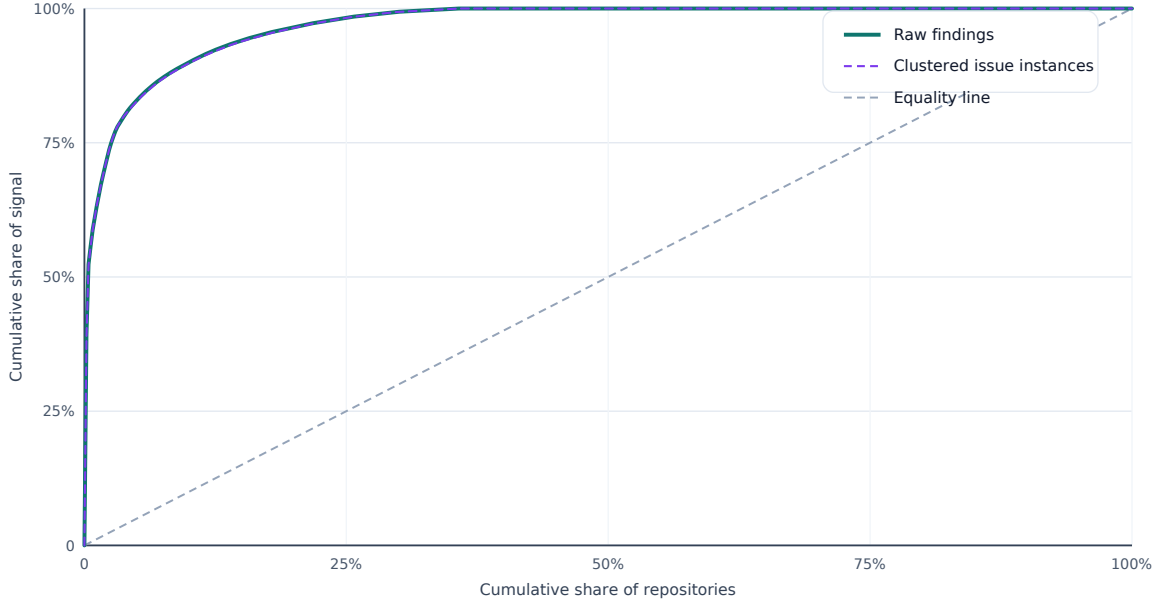
Surface indicator	Repos	Rate	Interpretation
Any instruction file	418	82.1%	File-centric instruction surface.
Any counted instruction surface	422	82.9%	Broader count including plugin/MCP surfaces.
AGENTS.md	264	51.9%	Project-level agent guidance.
SKILL.md / skill files	261	51.3%	Reusable capability surface.
CLAUDE.md	246	48.3%	Persistent project instruction file.
Plugin manifest	78	15.3%	Tool or integration declaration.
MCP manifest	58	11.4%	Protocol-level tool/resource surface.

6.2 Risk Concentration

The corpus exhibits a heavy-tailed distribution. The top six repositories, approximately 1.2% of the corpus, account for 63.2% of medium-or-higher raw findings. The top 26 repositories, approximately 5.1%, account for 83.1%; the top 51 repositories, approximately 10.0%, account for 90.0%. The raw-finding Gini is 0.940240 and the clustered-issue Gini is 0.939578; both round to 0.940.

Figure 1. Findings Concentration Curve

Cumulative share of repositories versus cumulative share of findings and clustered issue instances



Top six repos (~1.2%) account for ~63% of findings; top 10% account for ~90%. The two curves are nearly coincident.

Figure 1: Concentration of medium-or-higher repository-level findings in a purposive corpus of 509 instruction-rich repositories. Repositories are ordered by finding count. The solid curve shows the cumulative share of 4,882 raw findings; the dashed curve shows the cumulative share of 4,637 clustered issue instances. The two curves are nearly coincident, consistent with the rank-stable Gini values reported in Table 5. Source: `article-support-dataset-v1`.

6.3 Instruction-Surface Density and Normalized Outcomes

Figure 3 shows a positive exposure gradient across skill-count buckets in both normalizations reported. Per instruction-surface file, average normalized finding density rises from approximately 0.07 in the zero-skill bucket to 0.37 in the 20–99 bucket, then settles at 0.33 in the 100+ bucket; per skill file, density rises from 0.20 in the 1–4 bucket to 0.44 in the 20–99 bucket and then to 0.36 in the 100+ bucket. The shapes are similar: density increases sharply through the small and medium skill ranges and levels off in the largest bucket. This should be interpreted as an exposure-size association, not a causal security relationship: more instruction-bearing files create more scan opportunities for the same rule pack. A repository-level Spearman correlation against raw finding counts would partly recapitulate this, since the rules trigger on the same skill files; we therefore report the gradient and leave a per-token or per-instruction-line normalization to future work. The exposure gradient is useful for triage and prioritization but not sufficient to infer exploitability.

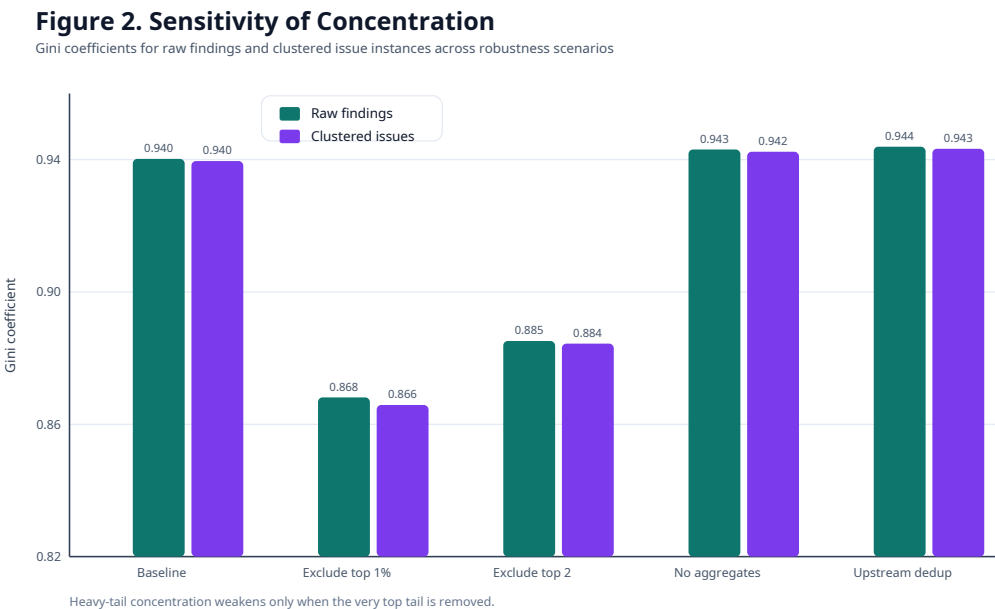


Figure 2: Robustness of repository-level concentration under sensitivity scenarios. The figure should be read as robustness evidence for a heavy-tail pattern within the constructed corpus, not as a population-level estimate beyond that corpus.

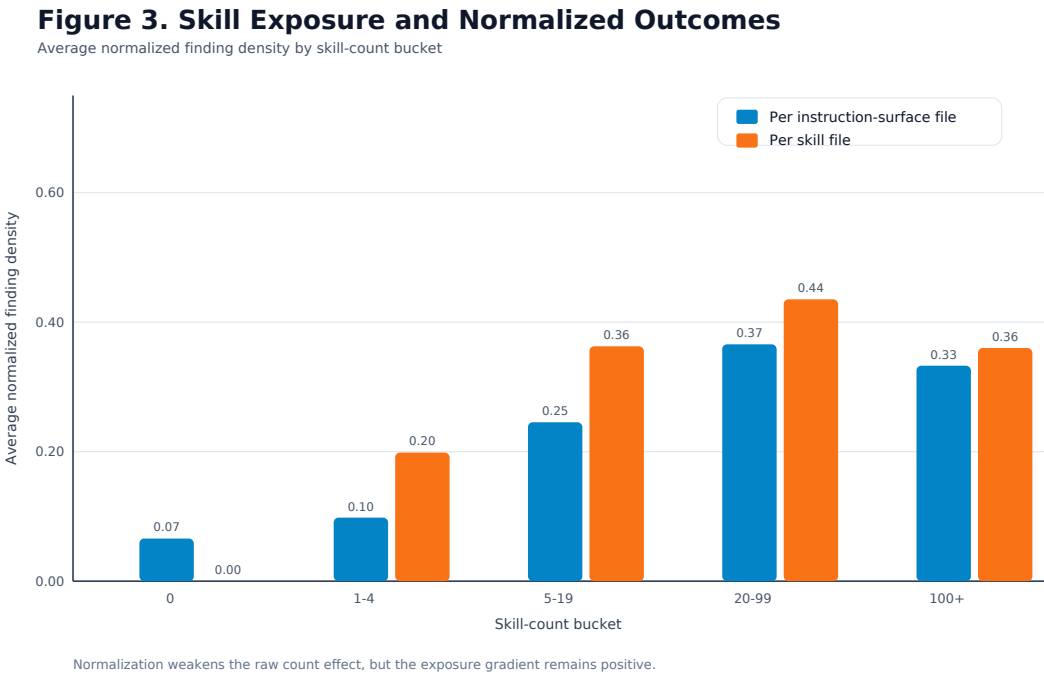


Figure 3: Association between instruction-surface density and raw or normalized outcomes. The result supports triage by exposure size but should not be interpreted as evidence that skill files alone cause risk.

6.4 Dominant Canonical Classes

The top five canonical classes account for 3,932 of 4,629 mapped clustered issue instances, or 84.95% of mapped instances. Relative to all 4,637 clustered issue instances, including the 8 unmapped cases, the same top five account for 84.80%. This supports the claim that the issue-centric picture is dominated by recurring structural patterns rather than one-off anomalies.

Table 8: Canonical class distribution for mapped clustered issue instances.

Canonical class	Count	Share of mapped
Autonomous execution or looping	1,155	24.95%
Unsafe command or execution surface	1,004	21.69%
Prompt or instruction override surface	781	16.87%
Agent role or goal manipulation	573	12.38%
Remote fetch or install expands trust boundary	419	9.05%
Tool or skill poisoning surface	369	7.97%
Credential or PII exposure surface	216	4.67%
Broad external action without approval	64	1.38%
SSRF or internal service reachability	37	0.80%
Identity rewrite with persistent effect	11	0.24%

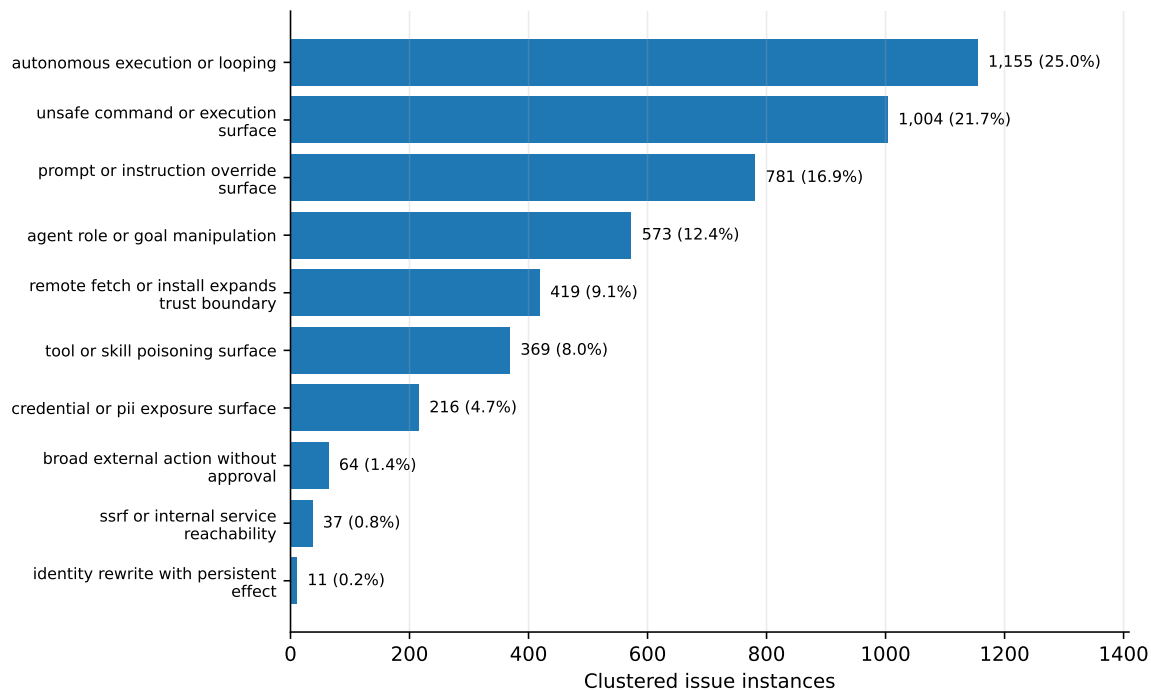


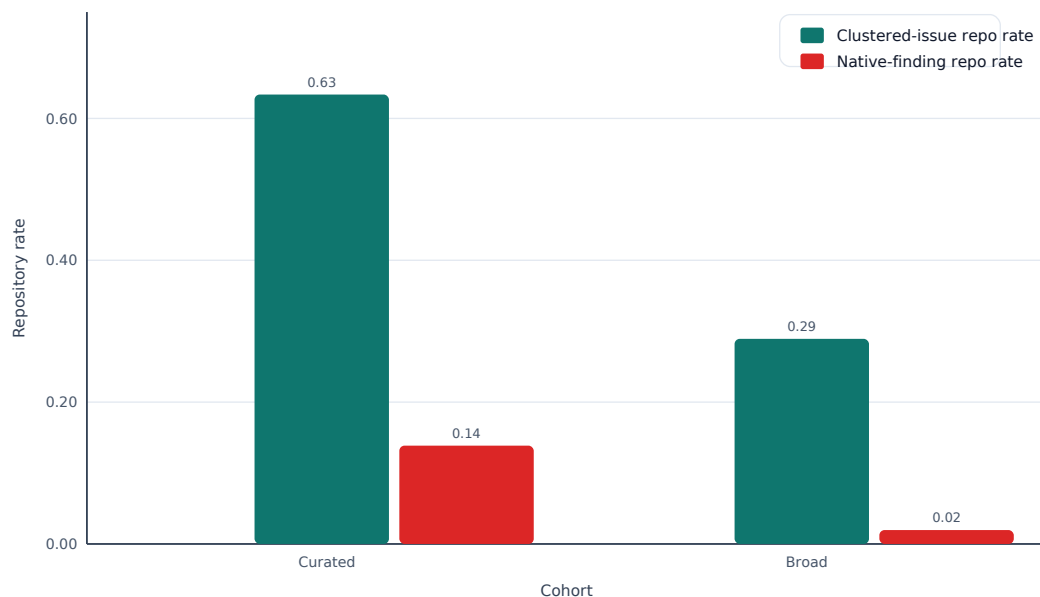
Figure 4: Composition of mapped clustered issue instances by canonical class. Denominator: 4,629 mapped clustered issue instances; 8 unmapped clustered instances are excluded from class-composition percentages.

6.5 Cohort Differences

The curated instruction-centric cohort has a higher rate of repositories with clustered issue instances than the broader discovery cohort. This is consistent with the study design: the curated cohort was selected for richer instruction-surface content and should not be compared as a random sample.

Figure 5. Cohort Comparison

Curated instruction-centric cohorts versus the broader discovery cohort



Curated cohorts are materially more signal-rich than the broad discovery cohort.

Figure 5: Comparison of curated instruction-centric repositories and the broader discovery cohort. The figure compares native findings, collection-scale signals, and clustered issue-instance rates.

6.6 Adjudication and Interpretive Stability

The adjudication subset shows that evidence packaging matters. Repo-file evidence packets are much more stable than category-root aggregate packets: 33 of 46 repo-file packets were unanimous, compared with 1 of 8 category-root packets. This supports using artifact-backed instances as the primary evidentiary layer and treating aggregate packets as navigation aids rather than stand-alone proof.

Table 9: Adjudication stability on the 54-packet stratified subset.

Evidence packet type	Records	Unanimous	Majority-label summary
Repo-file	46	33	26 TP, 20 FP.
Category-root aggregate	8	1	1 TP, 4 FP, 3 UNCERTAIN.
Total	54	34	27 TP, 24 FP, 3 UNCERTAIN by majority vote.

Figure 6. Adjudication Stability by Evidence Type

Unanimity, split decisions, and majority-label composition in the reviewed subset

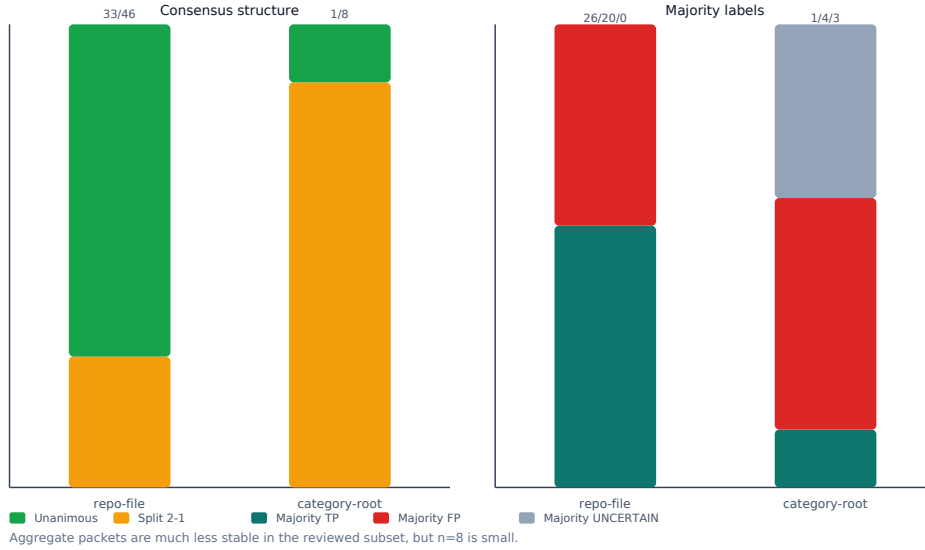


Figure 6: Cross-LLM adjudication stability for the stratified stage-2 subset of 54 issue instances. Counts should not be read as corpus-wide precision.

6.7 Signature-Source Triage Signals

The public reviewer bundle does not expose path-heavy raw sidecars, so the paper avoids claiming a complete per-source precision estimate. However, the released aggregate artifacts in **article-support-dataset-v1** [20] do support two source-level observations. First, the top-triggering rules are dominated by ATR-derived signatures: 8 of the top 10 rules account for 3,122 raw findings, or 63.9% of all medium-or-higher raw findings. The remaining two top-10 entries are one Aguara-derived external-download rule (246 findings) and one Cisco AI Defense skill-scanner / PromptGuard-style PII rule (157 findings). Together, the top 10 rules across all three families account for 3,525 raw findings, or 72.2% of medium-or-higher raw findings, indicating that a small number of rules drives most signature activity in this corpus. Second, auto-triage stability varies more by evidence packaging and canonical class than by rule family alone.

Table 10: Top-triggering rule families among the ten highest-count rules. This is a top-10 source-family summary, not a full per-source attribution for all findings.

Rule family in top 10	Top-10 rules	Raw findings	Dominant signal types
ATR-derived	8	3,122	Excessive autonomy, unsafe shell/tool arguments, prompt override, tool poisoning, and agent manipulation.
Aguara-derived	1	246	Remote SDK/script fetch as agent input; trust-boundary expansion through remote content.
Cisco AI Defense / PromptGuard-style	1	157	PII harvesting and data exposure patterns.

The triage subset also provides a class-level view of which signature families produce stable evidence packets. Autonomous execution and looping was relatively stable in auto-triage (9 of 12 unanimous; 10 TP by majority), while broad external action without approval was more ambiguous (12 of 21 unanimous; 11 TP, 9 FP, 1 UNCERTAIN by majority). Identity rewrite with persistent effect was fully unanimous, but mostly FP in this subset, suggesting that this structural detector is useful as a review trigger but should be interpreted conservatively.

Table 11: Auto-triage outcomes by major canonical class and primary source family. Counts are from the 54-packet adjudication subset and should not be generalized as corpus-wide precision.

Canonical class	Primary source family	Records	Unanimous	Majority labels
Autonomous execution or looping	Native ASAMM / ATR excessive-autonomy	12	9	10 TP, 1 FP, 1 UNCER-TAIN.
Broad external action without approval	Native ASAMM / tool-governance detectors	21	12	11 TP, 9 FP, 1 UNCER-TAIN.
Identity rewrite with persistent effect	Native ASAMM self-modification	11	11	2 TP, 9 FP.
Remote fetch or install expands trust boundary	Aguara external-download	4	2	3 TP, 1 FP.
Singleton classes	ATR, Aguara, and Cisco-derived families	6	0	Mixed: 1 TP, 4 FP, 1 UN-CERTAIN.

7 Discussion

7.1 What the Results Support

The results support three claims. First, instruction-rich repositories exhibit concentrated static-analysis evidence. Second, recurring issue classes dominate the issue-centric picture. Third, artifact-backed evidence is more interpretable than aggregate evidence packets. These claims are useful for triage, review ordering, and framework mapping.

7.2 What the Results Do Not Support

The results do not support claims about exploitability rates, GitHub-wide prevalence, or precision of the scanner across all repositories. A static finding is a signal. It becomes a security issue only after contextual review, and it becomes an exploitability claim only after runtime validation.

7.3 Implications for ASAMM

The study shows how ASAMM-style concerns can be operationalized as a repository audit. Context-handling controls map naturally to instruction files; tool-governance controls map to plugin and MCP artifacts; autonomy controls map to loop/write patterns; and self-modification governance maps to persistent identity and memory artifacts. The mapping is most useful where detector families converge

on the same concern, such as autonomy and tool-governance surfaces, where ATR-derived rules and native ASAMM detectors both contribute evidence at scale (Table 11). Conversely, classes such as identity rewrite with persistent effect show that ASAMM-native structural detectors can be valuable review triggers even when adjudication often marks individual packets as FP: in the stratified subset, the eleven identity-rewrite packets were unanimous across all three reviewer channels but mostly labeled FP by majority. This is consistent with using the detector as a high-priority review surface rather than as a standalone finding. The mapping is useful as a practical organizing layer, but it is not independent validation of ASAMM as a standard.

8 Threats to Validity

Corpus selection. The corpus is purposive and instruction-rich. Results should not be generalized to all repositories or all agent deployments.

Rule provenance and severity calibration. The detector uses heterogeneous rule sources with different design goals. Severity labels are harmonized but not independently calibrated. The current artifact package does not pin upstream commit SHAs for all imported rule families (ATR-derived, Aguara-derived, and Cisco AI Defense / data-protection rules), which limits exact third-party reproduction of the rule-bundle snapshot used in this study. This release also does not report sensitivity with severity removed or uniformly weighted; such an ablation is left to future work.

Rule overlap and clustering. Clustering reduces raw findings by approximately 5%, from 4,882 raw findings to 4,637 issue instances. Because clustering is conservative and repository-rank preserving, it has little visible effect on the concentration curve. This should be interpreted as robustness of the heavy-tail result, not as evidence that de-duplication is unnecessary. More aggressive clustering across sibling files, templates, or replicated upstream content would likely change issue counts and could change some per-repository rates.

Adjudication limitations. The adjudication subset is LLM-reviewed, not human-ground-truth labeled. Pairwise agreement is moderate rather than high (Cohen’s kappa 0.476–0.656; Fleiss’ kappa 0.562), and non-unanimous records should be treated conservatively. The signature-source analysis is also limited: the public bundle supports top-rule and canonical-class summaries, but not a full per-source precision estimate.

No runtime exploit validation. The study does not execute repositories in a sandboxed agent runtime. Therefore, it measures static evidence and artifact patterns, not exploit success.

Descriptive versus prescriptive intent. Static analysis of instruction text can confuse documentation or examples with operational instructions. This release does not report code-block, heading-context, or documentation-section ablations; future work should evaluate whether these context filters materially change finding rates.

Author-framework relationship. The scanner and ASAMM mapping were developed by the author. This introduces a conflict-of-interest risk and should be mitigated in future work through independent baseline runs, human expert adjudication, and third-party reproduction.

9 Ethics and Responsible Disclosure

The study analyzes public repositories and reports aggregate results. No exploitation, credential use, tool execution, or repository modification was performed. The public artifact package avoids path-heavy raw sidecars and emphasizes aggregate analysis and sanitized adjudication data.

Maintainers of the top six repositories (approximately 1.2% of the corpus) were notified before broader release. This manuscript does not report response outcomes or claim remediation, because response tracking is outside the released artifact package. This notification step is important because the same concentration result that helps defenders prioritize review could also help attackers prioritize inspection. Future releases should make the notification process reproducible by recording notification dates, response windows, and whether repository identifiers are anonymized, delayed, or reported only in aggregate.

10 Conclusion

Agent security is not only a runtime problem. Persistent instruction files, skills, plugin manifests, and MCP artifacts encode behavior, authority, and trust assumptions before an agent is executed. This paper presents a bottom-up, artifact-centric audit of such instruction surfaces across 509 instruction-rich repositories. The results show strong concentration, recurring structural classes, and a meaningful difference between artifact-backed and aggregate evidence. The resulting method is best understood as a triage and interpretation layer: it turns heterogeneous rule outputs into auditable repository evidence and maps that evidence to agent-security concerns.

11 Use of AI Assistance in This Study

This study used AI assistance in a limited and methodologically relevant way. The disclosure is explicit because the paper studies artifacts that shape agent behavior while also using LLM-based review in one part of the methodology.

Scope of AI assistance in the research pipeline. In the research pipeline reported in this paper, AI assistance was used for the adjudication/triage layer and for the optional LLM verifier functionality implemented in `agent-audit` [19]. It was not used for corpus construction, repository discovery, artifact discovery, static scanning, rule matching, canonical clustering and de-duplication, normalized outcome computation, sensitivity analysis, or cohort/activity correlation analysis. Those corpus-wide measurements were produced by rule-based scanning and deterministic post-processing.

Adjudication and verifier paths. The 54-packet adjudication subset was reviewed by three LLM-based reviewer channels: OpenAI ChatGPT 5.5, Claude Opus 4.7, and OpenAI ChatGPT 5.4 via the Codex desktop interface. Their outputs are used as an interpretive-stability signal, not as human ground truth. The reviewer rubric and prompt templates were fixed and released with the artifact package, including `reviewer-prompt.txt` and `adjudication-guidelines.md` for reviewer packets, as well as verifier prompt templates in `llm_verifier.py` and `batch_verifier.py`. The CLI verifier path fixed operational constraints such as `-allowedTools ""`, `-max-turns 1`, and read-only/no-approval style execution. Temperature, `top_p`, seed, stable API model strings, and exact deployment snapshot

dates were not uniformly fixed or visible for all adjudication interfaces; this limits reproducibility and is treated as a threat to validity.

Manuscript preparation. The manuscript was revised with AI-assisted editing for structure, wording, consistency checks, and reviewer-response planning. This assistance did not generate the corpus-wide measurements. The author remains responsible for all claims, statistics, citations, and interpretations. Reference entries for recent preprints were rechecked against public metadata during revision, including Skill-Inject, Towards Secure Agent Skills, and the MCP tool-poisoning paper.

Reflexivity. The method is partly an agent-assisted audit of agent-facing artifacts: the measurements are static and deterministic, while the interpretation sample is LLM-reviewed. This is not merely a disclosure obligation; it reflects the practical setting the paper studies. As agent-assisted development becomes routine, agent-security audits are also likely to be agent-assisted. This makes validation, disclosure, and human calibration important parts of the assurance workflow rather than external administrative details.

Conflict of Interest Statement

The author developed the scanner and the ASAMM mapping used in this study. This relationship is disclosed because it affects interpretation of novelty, taxonomy design, and evaluation. The study should therefore be read as an application and stress test of an author-proposed framework rather than independent validation of an established standard. The LLM-based adjudication and AI-assisted manuscript revision are disclosed separately because they introduce a distinct methodological reflexivity risk.

References

- [1] Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not What You’ve Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection. *AISec ’23: Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security*, 2023.
- [2] Yupei Liu, Yuqi Jia, Runpeng Geng, Jinyuan Jia, and Neil Zhenqiang Gong. Formalizing and Benchmarking Prompt Injection Attacks and Defenses. *USENIX Security Symposium*, 2024.
- [3] Edoardo Debenedetti, Jie Zhang, Mislav Balunovic, Luca Beurer-Kellner, Marc Fischer, and Florian Tramer. AgentDojo: A Dynamic Environment to Evaluate Prompt Injection Attacks and Defenses for LLM Agents. *NeurIPS 2024 Datasets and Benchmarks Track*, 2024.
- [4] Hanrong Zhang, Jingyuan Huang, Kai Mei, Yifei Yao, Zhenting Wang, Chenlu Zhan, Hongwei Wang, and Yongfeng Zhang. Agent Security Bench (ASB): Formalizing and Benchmarking Attacks and Defenses in LLM-based Agents. arXiv:2410.02644, 2024 (ICLR 2025).
- [5] David Schmotz, Luca Beurer-Kellner, Sahar Abdelnabi, and Maksym Andriushchenko. Skill-Inject: Measuring Agent Vulnerability to Skill File Attacks. arXiv:2602.20156, 2026.
- [6] Zhiyuan Li, Jingzheng Wu, Xiang Ling, Xing Cui, and Tianyue Luo. Towards Secure Agent Skills: Architecture, Threat Taxonomy, and Security Analysis. arXiv:2604.02837, 2026.

- [7] Xinyi Hou, Yanjie Zhao, Shenao Wang, and Haoyu Wang. Model Context Protocol (MCP): Landscape, Security Threats, and Future Research Directions. arXiv:2503.23278, 2025.
- [8] Charoes Huang, Xin Huang, Ngoc Phu Tran, and Amin Milani Fard. Model Context Protocol Threat Modeling and Analyzing Vulnerabilities to Prompt Injection with Tool Poisoning. arXiv:2603.22489, 2026.
- [9] Aguara. Aguara AI Agent and MCP Security Scanner. Project documentation, 2026. <https://aguarascan.com/>
- [10] Panguard AI. Agent Threat Rules (ATR): an open detection standard for AI agent security threats. Project documentation, 2026. <https://docs.panguard.ai/>
- [11] Cisco AI Defense. Open Source AI Security Scanners and Tools, including Skill Scanner and MCP Scanner. Project documentation, 2026. <https://cisco-ai-defense.github.io/>
- [12] NVIDIA. garak: Generative AI Red-teaming and Assessment Kit. Project repository, 2026. <https://github.com/NVIDIA/garak>
- [13] Microsoft. PyRIT: The Python Risk Identification Tool for generative AI red teaming. Project repository and documentation, 2024–2026. <https://github.com/Azure/PyRIT>
- [14] Promptfoo. LLM red teaming and evaluation framework. Project documentation, 2026. <https://www.promptfoo.dev/>
- [15] National Institute of Standards and Technology. Artificial Intelligence Risk Management Framework (AI RMF 1.0). NIST AI 100-1, January 2023.
- [16] National Cyber Security Centre, Cybersecurity and Infrastructure Security Agency, and partner agencies. Guidelines for Secure AI System Development. November 2023.
- [17] OWASP Foundation. Software Assurance Maturity Model (SAMM). Project documentation. <https://owasp samm.org/model/>
- [18] Sergey Gordeychik. Agentic SAMM: An OWASP SAMM Extension for AI-Driven Development. CyberOK, 2026, version 0.2.0-draft. <https://github.com/scadastrangelove/asamm>
- [19] Sergey Gordeychik. agent-audit: Forensic auditor and project-surface scanner for local AI coding agents. CyberOK, 2026. <https://github.com/scadastrangelove/agent-audit>
- [20] Sergey Gordeychik. agent-audit article support dataset (v1): 509-repository corpus list, scan-project results, summary analysis, and sanitized triage / adjudication artifacts. CyberOK, 2026. <https://github.com/scadastrangelove/agent-audit/tree/main/artifacts/article-support-dataset-v1>