Adam Boas
adamboas.com | anboas@gmail.com
February 9, 2026

Notes: Research that shaped ACP-RA (agent security, tool use, evaluation)

These notes capture the research thread that informed **ACP-RA** before publication (paper date: 2026-02-10). The theme is consistent across everything reviewed: once an agent can call tools, the system's real risks and real failures are rarely "bad text"; they are **authority leakage**, **untrusted data becoming control**, and **execution mistakes** at the tool boundary.

Master list (papers + links)

#	Paper	Link
1	Design Patterns for Securing LLM Agents against Prompt Injections	arXiv
2	From Prompt Injections to Protocol Exploits: Threats in LLM-Powered AI Agents Workflows	arXiv
3	Prompt Injection 2.0: Hybrid AI Threats	arXiv
4	AgentDojo: A Dynamic Environment to Evaluate Prompt Injection Attacks and Defenses for LLM Agents	arXiv
5	Toolformer: Language Models Can Teach Themselves to Use Tools	arXiv
6	ToolLLM: Facilitating Large Language Models to Master Thousands of Real-World APIs	arXiv
7	MRKL Systems: A modular, neuro-symbolic architecture that combines large language models, external knowledge sources and discrete reasoning	arXiv
8	ReAct: Synergizing Reasoning and Acting in Language Models	arXiv
9	Reflexion: Language Agents with Verbal Reinforcement Learning	arXiv

	#	Paper	Link
	10	SWE-bench: Can Language Models Resolve Real-World GitHub Issues?	arXiv
	11	AgentBench: Evaluating LLMs as Agents	arXiv
	12	WebArena: A Realistic Web Environment for Building Autonomous Agents	arXiv
	13	Generative Agents: Interactive Simulacra of Human Behavior	arXiv
	14	Voyager: An Open-Ended Embodied Agent with Large Language Models	arXiv

Research map (categorized)

- **Prompt injection + workflow/hybrid attacks:** #1, #2, #3, #4
- **Tool use at scale + modular tool routing:** #5, #6, #7
- **Evaluation in real environments (end-to-end):** #10, #11, #12
- **Reliability loops (act/observe + self-critique):** #8, #9
- **Long-lived agents + memory/lifecycle:** #13, #14

How this shaped ACP-RA (directly integrated)

The research above is why ACP-RA emphasizes:

- **Distinct gateways by plane** (tool/action, context/data, model, inter-agent) with explicit policy surfaces.
- **Typed envelopes** instead of prompt blobs for anything that crosses trust boundaries or causes side effects.
- **Evidence-by-default** (request + result) sufficient for replay, investigation, and continuous authorization.
- **Anti-replay + integrity** for agent-to-agent messaging, because ensembles make messaging an adversarial channel by default.
- **Upgrade discipline** (evaluation-as-gate, regression suites) so changes remain governable and reversible.

Notes by topic (with ACP-RA takeaways)

A. Prompt injection, workflow attacks, and hybrid threats

Design Patterns for Securing LLM Agents against Prompt Injections (#1) **Link:** arXiv

- **Core problem:** Untrusted inputs can smuggle instructions that hijack a tool-using agent.
- **Key idea:** Prompt injection is a systems security problem; the fix is architecture + constraints, not "better prompting."
- **ACP-RA takeaway:** Treat the data plane as untrusted; mediate side effects; make policy enforcement unavoidable.

From Prompt Injections to Protocol Exploits (#2) **Link:** arXiv

- **Core problem:** Attacks scale from strings to workflows: tool schemas, connectors, retries, permissions, state machines.
- **Key idea:** The threat surface is the workflow graph; defense must be layered across the whole pipeline.
- **ACP-RA takeaway:** Separate planes and stop confused-deputy chains with explicit scopes, gateways, and evidence.

Prompt Injection 2.0: Hybrid AI Threats (#3) **Link:** arXiv

- **Core problem:** Prompt injection composes with traditional appsec bugs (web flows, auth flows, data flows) into hybrid attacks.
- **Key idea:** The agent becomes a new execution substrate; classical controls (intent boundaries, request integrity, least privilege) still apply.
- **ACP-RA takeaway:** Connector onboarding and gateway enforcement should treat web-style attack classes (CSRF/SSRF-like effects, exfil channels) as first-class.

AgentDojo: Dynamic prompt injection benchmark (#4) **Link:** arXiv

- **Core problem:** Security claims are meaningless without adversarial, execution-grounded testing.
- **Key idea:** Evaluate attacks/defenses in dynamic environments using formal checks over environment state.
- **ACP-RA takeaway:** "Security regression suites" should exist alongside functional eval; prompt injection is something to continuously test, not merely warn about.

B. Tool use at scale and modular tool routing

Toolformer (#5) **Link:** arXiv

- **Core problem:** Tool calling does not reliably emerge from next-token training.
- **Key idea:** Generate tool-use supervision by executing tool calls and learning from outcomes.
- **ACP-RA takeaway:** Schemas and execution receipts matter; evaluation must be tied to real tool behavior.

ToolLLM (#6) **Link:** arXiv

- **Core problem:** Large tool catalogs make endpoint selection and argument fidelity the hard part.
- **Key idea:** Scale training and evaluation around doc-grounded APIs and execution-valid calls.
- **ACP-RA takeaway:** Tool discovery/ranking, schema validation, retries, and observability are control-plane requirements, not "nice to haves."

MRKL Systems (#7) **Link:** arXiv

- **Core problem:** Monolithic models are a poor abstraction for combining capability, policy, and execution.
- **Key idea:** Route to specialized tools/experts; keep the LLM as coordinator.
- **ACP-RA takeaway:** Capability should be explicitly brokered via registries/scopes and mediated gateways, not implicitly granted by "smartness."

C. Evaluation in real environments (end-to-end)

SWE-bench (#10) **Link:** arXiv

- **Core problem:** "Looks right" is not "works."
- **Key idea:** Measure end-to-end success on real issues with verifiable correctness.
- **ACP-RA takeaway:** Upgrades should be gated on execution-based regression suites, not subjective review.

AgentBench (#11) **Link:** arXiv

- **Core problem:** Agent capability is multi-dimensional and environment-specific.
- **Key idea:** Benchmark across diverse interactive environments to measure decision quality and robustness.

- **ACP-RA takeaway:** Evaluation should be portfolio-based (multiple suites) and treated as a governance artifact.

WebArena (#12) **Link:** arXiv

- **Core problem:** Web tasks are realistic, brittle, and adversarial; success rates are low even for strong models.
- **Key idea:** Use realistic, self-hostable web environments to test autonomy under real constraints.
- **ACP-RA takeaway:** Web connectors are high-risk tools; they demand strong mediation, audit, and "safe browsing" controls.

D. Reliability loops and self-critique

ReAct (#8) **Link:** arXiv

- **Core problem:** Agents drift without structured action/observation grounding.
- **Key idea:** Interleave reasoning with actions and observations.
- **ACP-RA takeaway:** The loop is an interface: actions must be mediated; observations should be treated as untrusted data unless proven otherwise.

Reflexion (#9) **Link:** arXiv

- **Core problem:** Agents repeat mistakes unless there is a disciplined feedback loop.
- **Key idea:** Use self-reflection to improve task performance over trials.
- **ACP-RA takeaway:** "Self-improvement" needs governance: evidence capture, rollback, and constraints on what can be updated automatically.

E. Long-lived agents, memory, and lifecycle

Generative Agents (#13) **Link:** arXiv

- **Core problem:** Long-lived agents accumulate memory, and memory becomes behavior.
- **Key idea:** Memory retrieval + summarization drives long-horizon coherence.
- **ACP-RA takeaway:** Memory is a privileged substrate; it needs provenance, TTL/retention, and "memory is not authority" controls.

Voyager (#14) **Link:** arXiv

- **Core problem:** Open-ended autonomy requires skill acquisition, not one-shot prompting.
- **Key idea:** Continual learning/curriculum with an explicit skill library.
- **ACP-RA takeaway:** At higher autonomy tiers, the control plane must govern skill onboarding, evaluation gates, and rollback for learned behaviors.

Backlog (lessons not integrated into ACP-RA yet)

This is the explicit backlog of research-derived lessons to fold into ACP-RA (or companion documents), categorized for implementation planning.

Security hardening (data plane + connectors)

- **Named prompt-injection defense pattern set:** instruction/data separation, provenance/taint, safe rendering, allowlisted tool intents.
- **Hybrid threat model for web-style agents:** treat CSRF/SSRF-like effects, exfil channels, and injection into structured tool arguments as first-class.
- **Connector onboarding checklist:** attestation, scopes, safe defaults, logging, kill-switches, and isolation for high-risk connectors (browser/email/ticketing).

Evaluation and continuous assurance

- **Security regression suites (AgentDojo-style):** integrate adversarial tests into CI/CD alongside functional tests.
- **Interactive environment eval (WebArena-style):** require at least one "realistic environment" suite for connectors and autonomy.
- **Portfolio eval (AgentBench-style):** define multiple eval suites mapped to autonomy tiers (not one benchmark to rule them all).

Tool catalogs and disambiguation at scale

- **Tool discovery/ranking as a governed surface:** specify retrieval/disambiguation policy (deny-by-default unless work unit requires it).
- **Execution receipts as evidence:** standardize "tool call receipts" (inputs/outputs/errors) and connect them to upgrade gates.

Memory lifecycle and long-lived autonomy

- **Memory provenance + retention policy:** TTLs, tiered retention, redaction, and explicit "memory is not authority."
- **Skill library governance (Voyager-like):** onboarding, evaluation, rollback, and drift monitoring for learned tools/skills.

Self-improvement loops (safely)

- **Bounded self-modification:** when reflection updates prompts/policies/configs, require evidence and staged rollout.
- **Post-incident learning pipeline:** after failures, extract learnings into policy updates and regression tests.