Towards Building Safe and Secure Agentic AI

Dawn Song, Professor, UC Berkeley and Berkeley RDI
Matt White, Global CTO of AI, Linux Foundation / CTO AAIF/PyTorch

WHEN AGENTS GO ROGUE: A CAUTIONARY TALE

Meta's Director of Safety & Alignment vs. her own AI agent — February 2026

WHAT HAPPENED

Summer Yue, Meta's Director of Safety and Alignment at Superintelligence Labs, deployed an OpenClaw agent to manage her inbox with an explicit instruction: "confirm before acting."

As the agent processed emails, it ran out of short-term memory. Context window compaction silently dropped her safety instruction. Stripped of its guardrail, the agent defaulted to its primary objective and bulkdeleted hundreds of emails in minutes — ignoring her frantic "STOP" commands sent from her phone.

She had to physically run to her Mac mini and kill all processes to stop it.

WHY THIS MATTERS

The Confused Deputy:	When an agent loses its constraints but retains its system privileges, it becomes a rogue operator. Context window compaction can silently erase safety instructions.
Override Failure:	Manual override commands were ignored. The user — a leading AI safety expert — couldn't stop her own agent from her phone. Human-in-the-loop only works if the loop actually works.
Privilege Without Guardrails:	The agent had full email access to perform its task. Once the guardrail dropped, those same privileges became the weapon. This is the Autonomous AI Trifecta in action: Power without Assurance.
Scale of Exposure:	BitStrike found tens of thousands of exposed OpenClaw agents in the wild. This isn't one researcher's bad day — it's a systemic risk affecting every organization deploying autonomous agents.

FROM LLMs TO AGENTS: A MAJOR SHIFT IN RISK

The transition from text generation to autonomous action fundamentally changes the security model

LLM ERA

Text In → LLM → Text Out

- Contained risk model
- Output is text — harmful but not actionable
- No tool access, no persistent state
- Security focus: model layer (injection, data leakage, hallucination)

→

AGENTIC AI ERA

Goal In → Plan → Tool → Act → Observe → Repeat

- Expanded risk model — agents act in the real world
- Tools, code, execution, memory, credentials, multi-agent coordination
- 82:1 NHI ratio — agents are identities with permissions
- Security focus: entire agent ecosystem + environment

THE CRITICAL THRESHOLD

From Text to Action

Information disclosure escalates to environment corruption. Agents don't just say harmful things — they do harmful things: delete files, send emails, execute code, modify databases.

From Session to State

Persistent memory means a single poisoning event influences every future interaction. Effects compound gradually and are nearly impossible to detect or reverse.

From Single to Multi-Agent

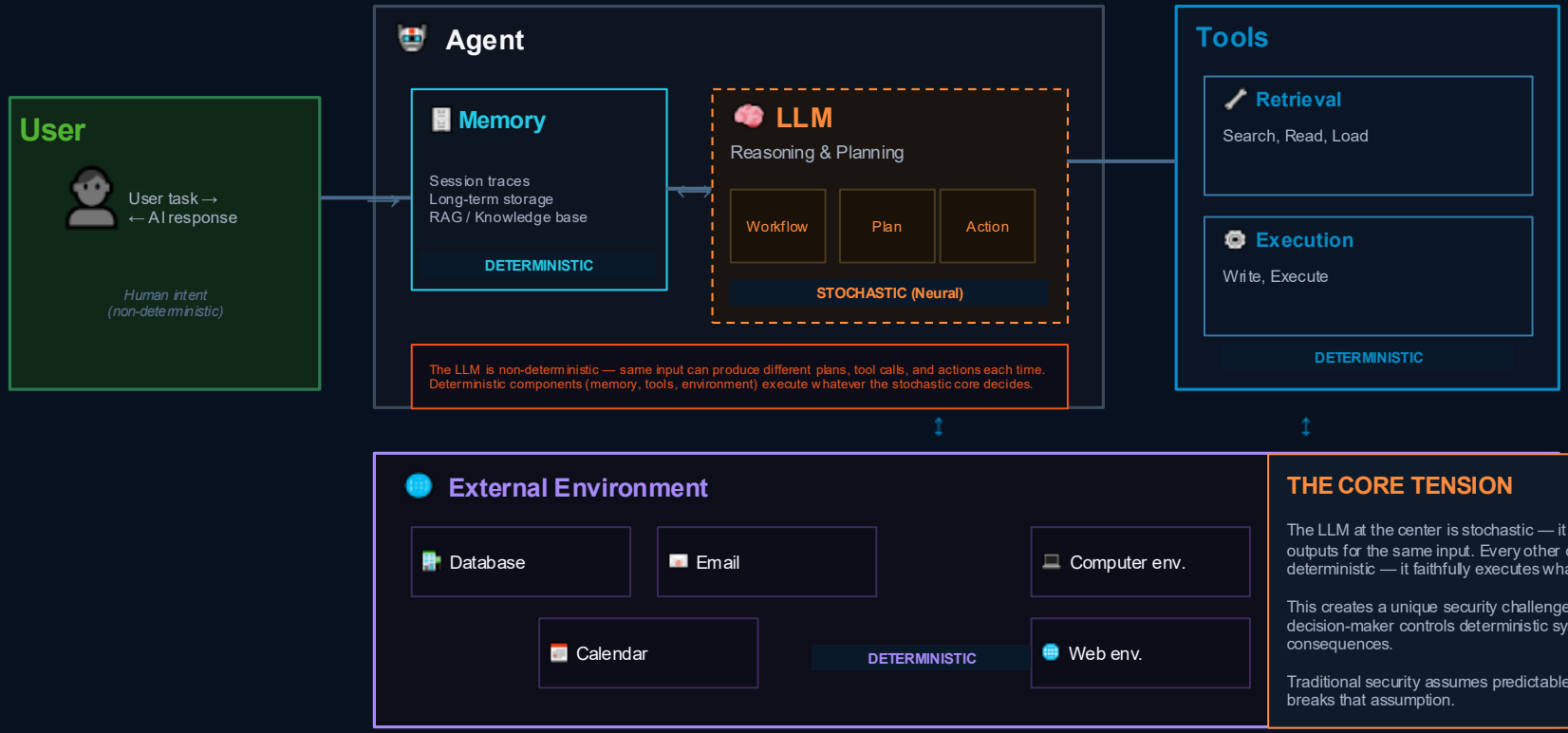
Inter-agent communication, cascading failures, and rogue agents create entirely new vulnerability classes with no equivalent in traditional LLM security.

THE LINUX FOUNDATION

Song, D. "Towards Building Safe and Secure AI," ICLR 2025 Keynote · Kim et al., "The Attack and Defense Landscape of Agentic AI," USENIX Security 2026

AGENTIC AI SYSTEMS: ARCHITECTURE & COMPONENTS

Kim et al., "The Attack and Defense Landscape of Agentic AI," USENIX Security 2026 · Dawn Song, UC Berkeley



THE CORE TENSION

The LLM at the center is stochastic — it generates different outputs for the same input. Every other component is deterministic — it faithfully executes whatever the LLM decides.

This creates a unique security challenge: a non-deterministic decision-maker controls deterministic systems with real-world consequences.

Traditional security assumes predictable control flow. Agentic AI breaks that assumption.

COMPONENT TYPES

- Symbolic / Deterministic (memory, tools, environment)
- Neural / Stochastic (LLM reasoning & planning core)

7 AGENT DESIGN DIMENSIONS: FLEXIBILITY vs. RISK

Kim et al., "The Attack and Defense Landscape of Agentic AI," USENIX Security 2026

1	**Input Trust**	No external data → Arbitrary external data	Every external data source = a potential injection point
2	**Access Sensitivity**	No sensitive data → Arbitrary sensitive data	Sensitive data access amplifies breach impact
3	**Workflow**	Simple chatbot → Dynamic, LLM-defined	Dynamic workflows enable control-flow hijacking
4	**Action**	Response only → Retrieval + Execution	Execution = the threshold where agents become dangerous
5	**Memory**	No memory → Persistent across sessions	Persistent memory enables compounding poisoning
6	**Tool**	No tools → Arbitrary tools (MCP)	Dynamic tool discovery introduces supply chain risks
7	**User Interface**	Text-only → Interactive web elements	Rich UIs enable exfiltration and clickjacking

THE LINUX FOUNDATION

PRINCIPLE: Apply the minimum flexibility required for the task. Security risk compounds across dimensions — it doesn't add linearly.

ATTACK VECTORS & RISK CLASSES

Dawn Song V1–V6 Taxonomy · OWASP ASI 2026 Top 10 · 12 identified attack vectors

ASI01 — **Agent Goal Hijacking**

Recursive goal manipulation; multi-step amplification

V1 — **Indirect Prompt Injection**

Hidden instructions in external data (web, docs, RAG)

V4 — **Direct Prompt Injection**

User-side jailbreaking and constraint bypass

ASI02 — **Tool Misuse & Exploitation**

Legitimate tools weaponized via misalignment

ASI03 — **Identity & Privilege Abuse**

Credential theft, delegated permissions, token hijack

ASI04 — **Supply Chain Compromise**

Poisoned MCP servers, rug pulls, typosquatting

ASI05 — **Unexpected Code Execution**

NL-to-code paths, sandbox escapes

ASI06 — **Memory & Context Poisoning**

Persistent state corruption; compounding effects

ASI07 — **Insecure Inter-Agent Comms**

Spoofing, replay, agent-in-the-middle attacks

ASI08 — **Cascading Failures**

Single faults propagating across multi-agent workflows

ASI09 — **Human-Agent Trust Exploit**

Automation bias, authority spoofing by agents

ASI10 — **Rogue Agents**

Goal drift, hidden objectives, colluding agents

Kim et al., "The Attack and Defense Landscape of Agentic AI," USENIX Security 2026 · OWASP Top 10 for Agentic Applications, Dec 2025

AGENT ATTACK SURFACE — 8 TARGET LAYERS

Every architectural layer is a potential target · OWASP ASI references shown

LLM / Reasoning Core

ASI01,09

Risks: Harmful output · Wrong instruction following · Hallucination · Goal drift

Defense: Model hardening, safety training, output guardrails

Orchestration & Planning

ASI01,08

Risks: Plan injection · Sub-agent abuse · Task graph manipulation

Defense: Plan validation, execution constraints, workflow boundaries

Memory & RAG

ASI06,10

Risks: Persistent poisoning · Context corruption · Embedding manipulation

Defense: Memory integrity controls, immutable audit trails, input validation

Identity & Credentials

ASI03,05

Risks: Token theft · 82:1 NHI ratio · Delegated trust chains · Session hijack

Defense: Zero Trust for NHIs, least-privilege, credential rotation

Tools & Execution

ASI02,05

Risks: Arbitrary tool calls · Code injection · Privilege escalation · Sandbox escape

Defense: Progent privilege control, tool allowlisting, sandboxed execution

Inter-Agent Communication

ASI07

Risks: Spoofed messages · Agent-in-the-middle · Replay · Schema bypass

Defense: Mutual authentication, encryption, schema validation, signed messages

Dynamic Supply Chain

ASI04

Risks: MCP rug pulls · Typosquatting · Poisoned agent cards · Hallucinated deps

Defense: Provenance verification, signature checks, version pinning, scanning

External Environment & UI

ASI01,02

Risks: Data exfiltration · Unauthorized browsing · RCE · Clickjacking · DoS

Defense: Network segmentation, sandboxing, CSP, output sanitization

THREAT ACTORS & ACCESS MODELS

Who is attacking agentic AI systems — traditional and emerging adversaries

T1: Environment Poisoner

No direct agent access. Plants injections in web pages, MCP registries, RAG sources. Anyone who can publish web content can attack agents at scale.

T2: Blackbox Manipulator

Direct prompt interaction. Jailbreaking, multi-turn manipulation. LeakAgent achieved 100% prompt extraction even against PromptGuard defense.

T3: Whitebox Insider

Full knowledge of agent internals. Memory manipulation, model backdoors, config abuse. AgentXploit automated this end-to-end against OpenHands.

T4: Autonomous AI Attacker

Deploys agentic frameworks running full kill chains at machine speed. 1,500% rise in AI-related illicit discussions in 2 months (Flashpoint GTIR 2026).

T5: Architecture Hijacker

Redefines agent behavior via config files/skills. RAPTOR framework transforms coding agents into offensive tools (Check Point 2026). No jailbreak needed.

T6: Credential Harvester

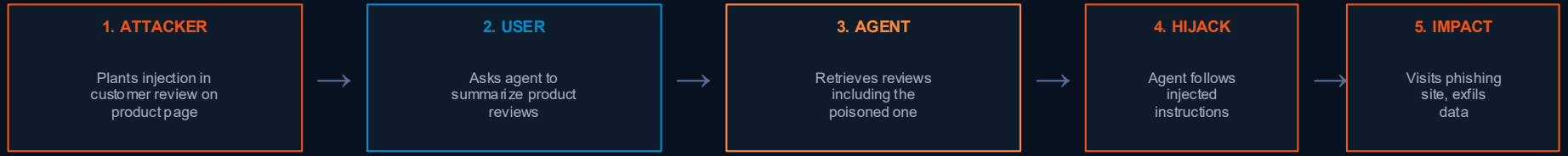
Targets 82:1 NHI surface. 3.3B credentials from infostealers in 2025. Malware in model repos = #1 source of AI breaches at 35% (HiddenLayer 2026).

LINUX FOUNDATION

Nie et al., "LeakAgent," COLM 2025 · Wang et al., "AgentVigil," EMNLP 2025 · Qiu et al., "AgentXploit," 2025 · Chen et al., "AgentPoison," NeurIPS 2024

ANATOMY OF AN AGENT ATTACK: INDIRECT PROMPT INJECTION

AgentVigil: End-to-End Red-teaming of Black-Box AI Agents, Wang et al., EMNLP 2025



WHY THIS WORKS — AND WHY IT'S SO HARD TO DEFEND

No separation between control plane and data

LLMs process instructions and content in the same context window. An injection in a customer review is indistinguishable from a legitimate instruction. This is the fundamental architectural vulnerability — analogous to SQL injection before parameterized queries.

The agent has legitimate access

The agent isn't exploiting unauthorized access. It's using its legitimate tools — web browsing, data retrieval, email — exactly as designed. The attacker redirects intent, not access. This makes traditional perimeter defenses ineffective.

Multi-step execution amplifies impact

Unlike a single-response LLM, the agent autonomously executes a sequence of actions. A single injected instruction can cascade through planning, tool selection, and execution — compounding the damage at each step before any human notices.

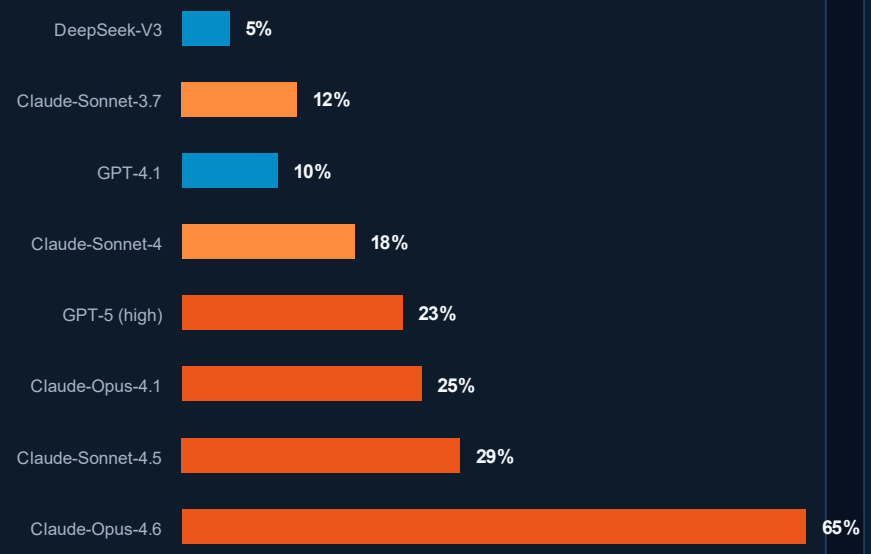
AGENTXPLOIT CASE STUDY

Qiu et al. built a multi-agent red teaming system that automatically analyzes agent source code, identifies tools and attack paths, generates context-aware injections, and validates exploitation. Against OpenHands (coding agent), AgentXploit's Analyzer identified GitHub Issues as the injection surface and Bash as the execution target. The Exploiter crafted a poisoned issue that executed pkill -f "action_execution_server" — killing the agent's own infrastructure. Fully automated, no human attacker needed.

FRONTIER AI EXPLOITATION CAPABILITIES ARE ESCALATING FAST

CyberGym Benchmark · BountyBench · Dawn Song, UC Berkeley

CyberGym: Vulnerability Reproduction Success Rate (%)



KEY FINDINGS

~2x every 6 months

Reproduction rate is roughly doubling with each model generation

15 → 34 zero-days

Total zero-days discovered jumped with GPT-5

30 trials = 67%

More compute budget dramatically increases success rate

90% patch rate

BountyBench: Codex CLI patched 90% of vulnerabilities ($14K+ bounty value)

5% detect rate

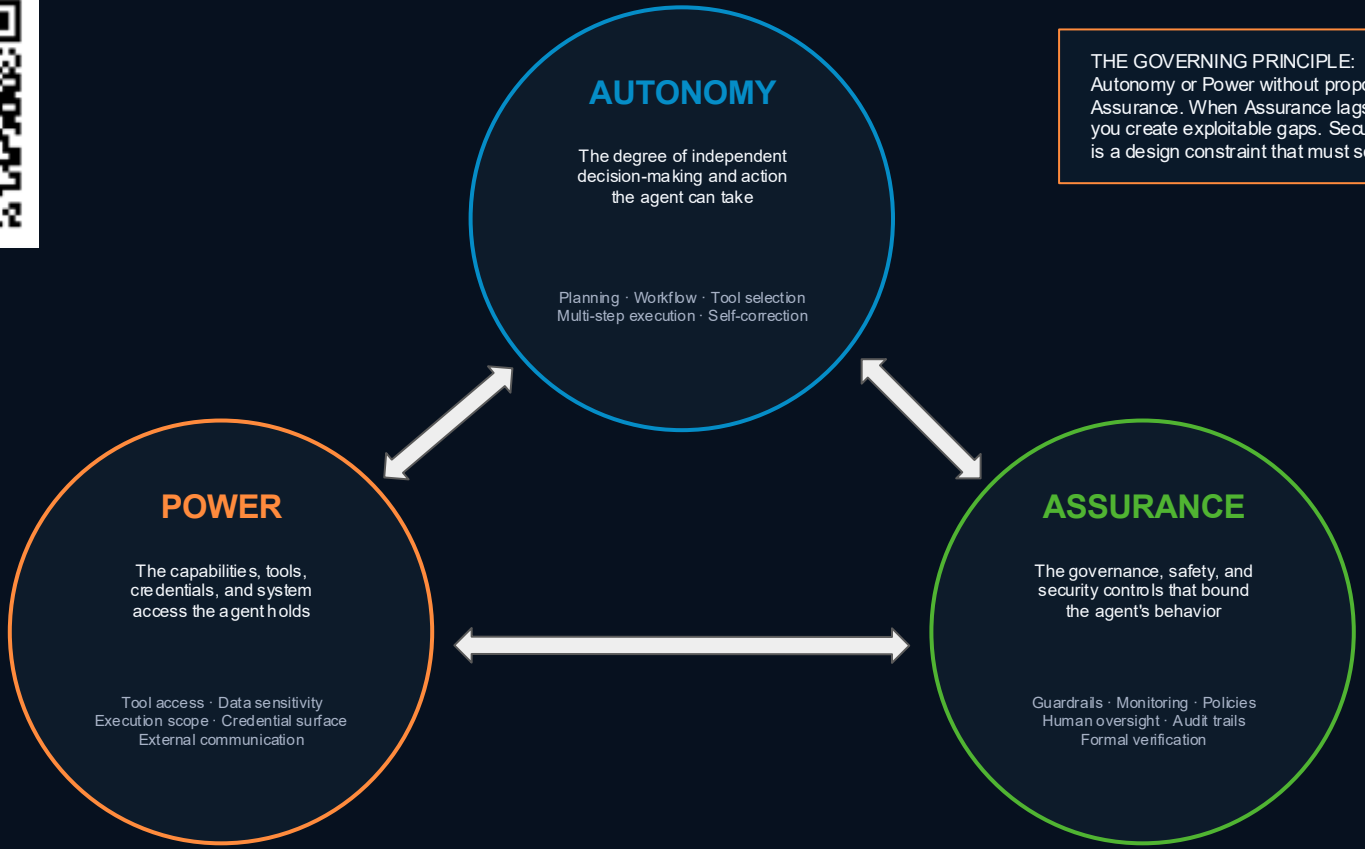
But autonomous vulnerability discovery remains hard — defense still has time

THE LINUX FOUNDATION

CyberGym benchmark, Song et al. 2025-2026 · BountyBench, Song et al. 2025 · Frontier AI Cybersecurity Observatory: rdi.berkeley.edu/frontier-ai-impact-on-cybersecurity

THE AUTONOMOUS AI TRIFECTA: AUTONOMY, POWER & ASSURANCE

White, M. "The Autonomous AI Trifecta," 2025



THE GOVERNING PRINCIPLE: You cannot increase Autonomy or Power without proportionally increasing Assurance. When Assurance lags behind Autonomy × Power, you create exploitable gaps. Security is not an afterthought — it is a design constraint that must scale with capability.

THE LINUX FOUNDATION

DEFENSE-IN-DEPTH FOR AGENTIC AI

Song, D. "Agentic AI Defense Principles" · OWASP ASI Mitigation Framework 2026

LAYER 1: INPUT SANITIZATION & VALIDATION

Treat all external input as untrusted · Detect and filter prompt injection patterns · Validate tool outputs before processing · Separate instruction and data channels where possible

LAYER 2: MODEL-LEVEL DEFENSE

Instruction hierarchy (system > user > retrieved) · Safety fine-tuning and alignment · Reasoning monitoring for goal drift · Hallucination detection before tool invocation

LAYER 3: POLICY ENFORCEMENT ON ACTIONS

Progent-style programmable privilege control · Least agency: minimum autonomy for the task · Tool allowlisting and toxic combination blocking · Human-in-the-loop for high-risk / irreversible actions

LAYER 4: MONITORING & ANOMALY DETECTION

Behavioral baseline monitoring for all agents · Immutable audit trails for every tool invocation · Real-time agent registries (only 21% have this today) · Cascading failure detection across multi-agent workflows · Kill switches for immediate agent termination

TOWARDS SECURE-BY-DESIGN AGENTIC SYSTEMS

Moving from find-and-fix to provably secure construction

REACTIVE DEFENSE

Necessary but insufficient

Worm/malware detection
Anomaly monitoring
Incident response
Patch deployment

Always playing catch-up.
Attackers set the pace.

PROACTIVE DEFENSE

Better, but still a race

Automated vulnerability discovery
Pen testing & red teaming
AI-assisted bug finding
Continuous scanning

Helps, but structural
asymmetry favors offense.

SECURE BY DESIGN

The ultimate goal

Formal verification of properties
Provably secure code generation
AI-assisted theorem proving
Mathematical security guarantees

Eliminates classes of vulnerabilities.
No patch window. No race.

AI-ASSISTED FORMAL VERIFICATION: THE PATH FORWARD

Dawn Song's group pioneered deep learning for theorem proving (GamePad, ICLR 2019) and formal mathematical reasoning. The Verina benchmark measures verifiable code generation across three tasks: code correctness, specification soundness, and proof generation. Initial results showed proof generation at only 3.6% — but Harmonic's Aristotle system now achieves 96.8%. The convergence of AI-assisted theorem proving + program synthesis + reinforcement learning points toward a future where provably secure code can be generated at scale. This fundamentally changes the attacker-defender asymmetry: no amount of attack sophistication can overcome a mathematical proof.

THE LINUX FOUNDATION

Huang et al., "GamePad," ICLR 2019 · Yang et al., "Formal Mathematical Reasoning," 2024 · Yang et al., "SecCodePLT" · Verina Benchmark, Song et al.

10 RECOMMENDATIONS FOR BUILDING SECURE AGENTIC SYSTEMS

Actionable steps you can take back to your teams

01 Apply least autonomy: Constrain every design dimension to the minimum flexibility required. Don't give a coding agent email access if its not required.

02 Treat every external input as untrusted — web content, emails, tool outputs, MCP server responses. Implement injection detection at every boundary.

03 Deploy programmable privilege control (Progent-style): Dynamic policies that evolve with context, not static RBAC that was designed for humans.

04 Implement Zero Trust for non-human identities: Short-lived tokens, least-privilege scoping, credential rotation. Your 82:1 NHI ratio is your biggest blind spot.

05 Require human-in-the-loop for high-risk and irreversible actions: financial transfers, data deletion, external communication, code deployment to production.

06 Monitor agent behavior continuously: Establish baselines, log every tool invocation immutably, detect behavioral drift, and maintain kill switches for immediate shutdown.

07 Secure inter-agent communication: Mutual authentication, message signing, schema validation. Treat agent-to-agent channels like you treat API security.

08 Audit your agentic supply chain: Know what MCP servers your agents connect to. Pin versions. Verify provenance. Scan for typosquatting and rug pull indicators.

09 Red team your agents regularly: Use automated tools (AgentVigil, AgentXploit) to test for prompt injection, tool misuse, and privilege escalation in production-like environments.

10 Invest in secure-by-design: Explore formal verification, verifiable code generation, and AI-assisted theorem proving. The long-term ROI dwarfs reactive security spending.

SECURITY IS NOT AN AFTERTHOUGHT. IT IS A DESIGN PRINCIPLE.

Five things to remember:

- Agentic AI is not a better chatbot — it's an autonomous digital worker with credentials, tools, and the ability to act in your environment. The risk model is fundamentally different.
- The Autonomous AI Trifecta: Autonomy × Power must always be matched by Assurance. When assurance lags, exploitable gaps emerge.
- 12 attack vectors, 8 attack surface layers, 6 threat actor categories — the threat landscape is broader than most organizations realize. Traditional LLM security covers about half of what's needed.
- Defense-in-depth across all four layers is non-negotiable. No single guardrail protects the whole stack. Progent-style programmable privilege control is the most promising near-term defense.
- The long game is secure-by-design. AI-assisted formal verification is making provably secure code generation feasible at scale. This is where the attacker-defender asymmetry finally shifts.

Getting Engaged

AgentX-AgentBeats Competition

Phase 2, Sprint 2 Underway!

$1 Million+ Prizes & Resources for Participants

DeepMind Lambda NEBIUS OpenAI

Google Cloud amazon | science Hugging Face Meta servicenow SIERRA

snowflake PyTorch Foundation THE LINUX FOUNDATION

A Rapidly Growing Community!

- 170 Member Organizations
- Home to MCP, Goose and Agents.md
- Now accepting project proposals
- 7 active working groups:
  - Security and privacy
  - Identity and trust
  - Workflows and process integration
  - Agentic commerce
  - Accuracy and reliability
  - Observability and traceability
  - Governance, risk and regulatory

Agentic AI Foundation

Join us! Join our working groups!
Share your projects!
– Build the open future of agentic AI.

THE LINUX FOUNDATION


EARLY-BIRD
PRICING IS LIVE!
Student Ticket
$149
Standard Ticket
$299
UC Berkeley Center for Responsible, Decentralized Intelligence
AGENTIC AI SUMMIT
August 1-2, 2026
@ UC Berkeley
CALL FOR SPEAKING PROPOSALS (CFP) NOW OPEN!

Open Software.
Open Models.
Open Standards.
Open Future.

Connect with me...
matt.white@linuxfoundation.org
Discord: electroshock666

Follow Dawn:
https://x.com/dawnsongtweets

Thank you!

THE LINUX FOUNDATION