

Agents of Chaos

A field guide to how agentic AI gets owned and what trust nothing looks like in practice

Amber Bennoui & Packguard





Amber Bennoui

AI, cybersecurity and engineering experience across companies including F5 Networks, Datadog, Chainguard and DataRobot. Co-founded the AI Security Alliance (AISECA), and conducts ongoing research into AI supply chain security.



Duc-Ly Vu, PhD

Security and AI researcher with a Ph.D. in software supply chain security from the University of Trento. Developed methods for detecting malicious Python packages at SAP, and previously worked on vulnerability risk metrics as an Early Stage Researcher with the European Network for Cyber Security.

The model was never the attack surface. The workflow is.

LANGUAGE MODEL

Input → output

Worst case is a bad sentence.

AGENTIC SYSTEM

Observe → plan → act → persist

Worst case is a privileged action with your credentials.

THE METHOD

We coded the whole literature so you can argue with one map instead of ninety-eight papers.

186

candidate sources

98

retained after
screening

396

attack-evidence
records, 11 fields
each

26

normalized attack
vectors

8

attack classes

TAXONOMY

Everything malicious enters through one of eight doors.

C1

Prompt injection & action hijacking

Untrusted content read as orders

C2

Tool, MCP & action layer

Poisoned descriptions and schemas

C3

Memory & RAG poisoning

Corrupting state the agent trusts later

C4

Multi-agent & collaboration

One agent's output is another's instruction

C5

Backdoors & persistence

Payloads waiting for a trigger

C6

Computer-use, browser & coding

Output becomes a shell command

C7

Agent skills & supply chain

Third-party artifacts crossing trust boundaries

C8

Goal & task manipulation

Benign substeps, malicious total



The most-discussed class is not the most-demonstrated one.

MOST PAPERS

17

papers on **tool, MCP & action-layer** attacks and the broadest coverage in the corpus.

MOST EVIDENCE

69

evidence records on **agent skills & supply chain** — the densest body of demonstrated attacks.

Normalized across all 396 records, the single most frequent mechanism is **instruction–data confusion**, with 46 records, 1 in 5 sources.

One markdown file, 6 stages, 0 malware.

1

Insertion

Skill published to a registry. It is prose.

2

Selection

The description persuades the agent to pick it.

3

Interpretation

Documentation is read as instruction.

4

Activation

Reads secrets, calls an external endpoint.

5

Persistence

Writes memory. Spreads with the workflow.

6

Recovery

Arrives after the state has already changed.

Attacks get in because agents cannot tell data from orders.

PATTERN 1

Instruction–data confusion

Untrusted content shares a context window with your system prompt.

PATTERN 2

Delayed activation

Planted now, fired later.
One–time scans prove nothing.

PATTERN 3

Metadata is executable

Descriptions and schemas steer behavior without any code.

They land hard because agents can act and because they remember.

PATTERN 4

Capability abuse

Severity is set by the agent's permissions, not by the attacker's phrasing.

PATTERN 5

Persistence across sessions

Memory, corpora, installed skills and dependencies outlive the interaction.

And they compose faster than any single defense can hold.

PATTERN 6

Composition & propagation

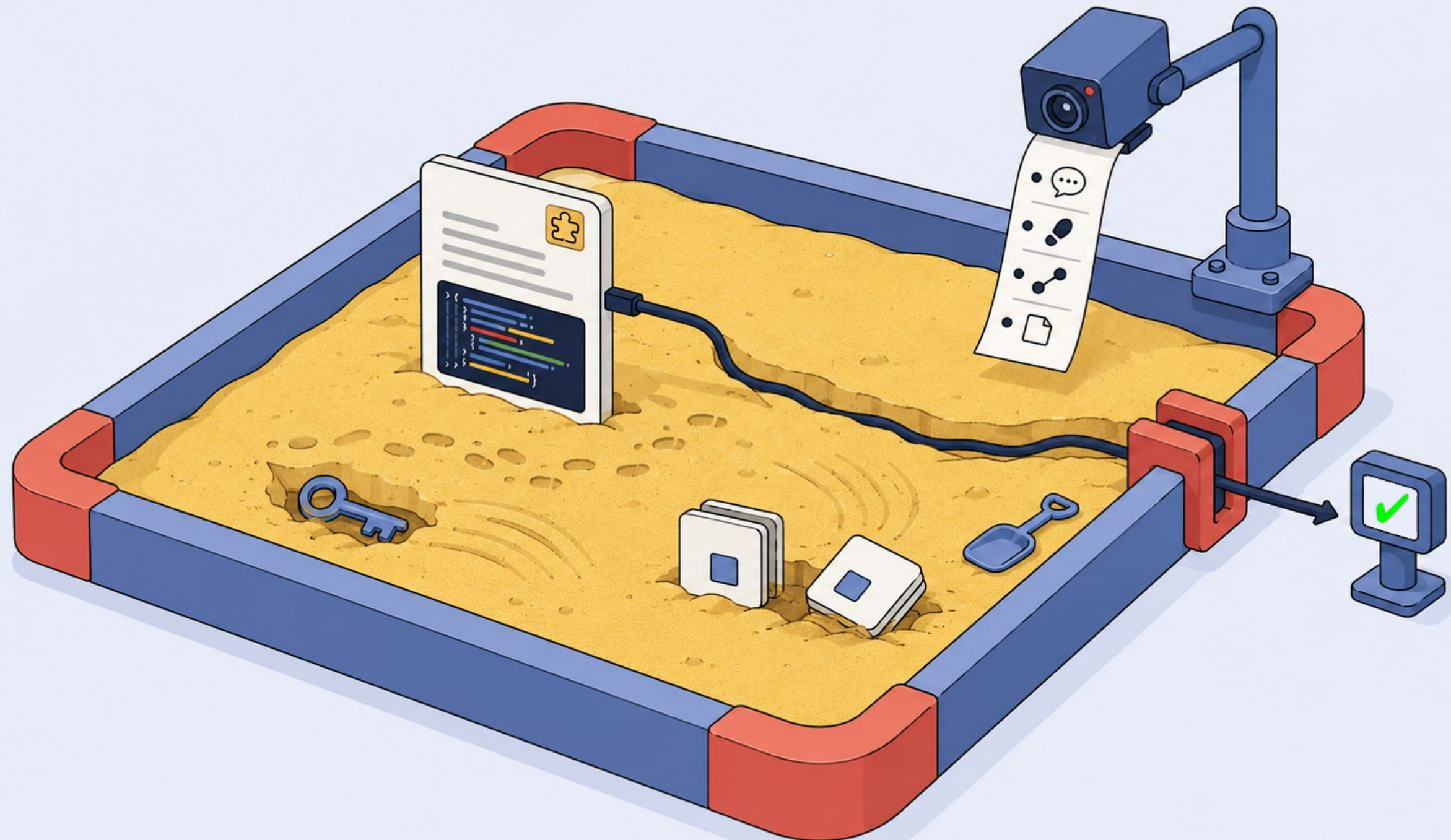
Benign parts, malicious whole — and in multi-agent systems it spreads.

PATTERN 7

Defense brittleness

Every single-layer control in the corpus has a documented bypass.

**Eight defense families, none of which is
a control on its own.**



Eight defense families, none of which is a control on its own.

D1

Prompt guards & context separation

D2

Tool validation & metadata vetting

D3

Memory & RAG governance

D4

Sandboxing & runtime isolation

D5

Least privilege & permission scoping

D6

Provenance, signing & registry governance

D7

Runtime monitoring & detection

D8

Recovery & revocation

Provenance proves origin, not behavior.

Sandboxing bounds damage, not entry.

We defend the moment of execution. We do not defend the lifecycle.

STRONG COVERAGE

Bounded execution risk

Unsafe tool invocation · command execution ·
package installation · sandboxable runtime behavior

WEAK COVERAGE

Semantic, delayed, compositional risk

Memory poisoning · multi-agent propagation · sleeper
triggers · tool-metadata manipulation · long-horizon
task manipulation

All 25 high-priority gaps sit in four classes.

8

**Backdoors &
persistent
compromise**

7

**Agent skills &
supply chain**

5

**Memory & RAG
poisoning**

5

**Multi-agent
propagation**

Eight defense families against eight attack classes

	D1 guards	D2 tool vet	D3 memory	D4 sandbox	D5 privilege	D6 provenance	D7 monitoring	D8 recovery
C1 prompt injection	●				○		○	
C2 tool / MCP		●		○	●	○	○	
C3 memory / RAG			●				○	○
C4 multi-agent	○				○		●	○
C5 backdoors		○		○		○	○	○
C6 computer use		○		●	●		○	
C7 skills / supply chain		○		○		●		○
C8 goal manipulation	○				○		●	

Four changes the evidence actually supports.

1 Threat-model the path, not the prompt

Trace every capability back to the untrusted content that can reach it.

2 Review the English, not just the code

Skill files, tool descriptions and registry metadata are executable in practice.

3 Test across state, not across turns

Vary sessions, retrieval events, updates and permission profiles.

4 Pin and re-verify anything mutable

Approval has an expiry date. Tools, deps and corpora change after review.

The unsolved problem is recovery, not detection.

Recovery & revocation

Removing the artifact does not undo the state it left behind.

Semantic artifact vetting

Tell documentation apart from an operational instruction.

System-level benchmarks

Real workflows, varied permissions, ground truth at every layer.

Scope: ecosystem as of July 2026 · peer-reviewed and grey literature, evidence levels preserved · many vectors legitimately span classes.

Assume insertion. Design for activation, persistence, and revocation.

ASK

What can it do?

What does it keep?

How would you undo it?

Thank you

`amberbennoui.com`

`packguard.dev`



`my linkedin!`