Application Security

in the Age of Agentic Engineering

Jan Brennenstuhl · Principal Software Engineer

HU Berlin · Guest Lecture · 2026

Disclaimer

These slides reflect my own views and experience. I am not speaking for my employer or any company I have worked for in the past. I encourage sharing and discussing this material, but please identify me without employer affiliation.

About me

Jan Brennenstuhl, Principal Software Engineer, Application Security
janbrennenstuhl.eu · @jbspeakr@infosec.exchange

Where

Application security at a large European e-commerce platform

Scale

~2,000 engineers, thousands of microservices, hundreds of AWS accounts

Focus

OAuth/OIDC, zero trust, platform security, developer experience for security tooling

In February 2026, an unknown actor published an unauthorized release of Cline

Cline is an open-source coding agent. The story starts with an AI agent that reads GitHub issues.

21 Dec 2025

Cline adds an AI agent that triages new GitHub issues

1 Jan 2026

Adnan Khan reports that an issue title can give commands to the agent

9 Feb 2026

Khan publishes the problem. Cline fixes the workflow within 30 minutes, but revokes the wrong token

17 Feb 2026

An unknown actor publishes cline@2.3.0 to npm. It is online for about 8 hours and installs a second tool, openclaw

Sources: adnanthekhan.com/posts/clinejection · snyk.io/blog/cline-supply-chain-attack-prompt-injection-github-actions

Three lines made the triage agent an attack path

```
- name: Run Issue Response & Triage
  id: triage
  uses: anthropics/claude-code-action@v1
  with:
    anthropic_api_key: ${{ secrets.ANTHROPIC_API_KEY }}
    github_token: ${{ secrets.GITHUB_TOKEN }}
    allowed_non_write_users: "*"
    claude_args: --model claude-opus-4-5-20251101 --allowedTools "Bash,Read,Write,Edit,Glob,Grep,WebFetch,WebSearch"
    prompt: |
      You're a GitHub issue first responder for the open source Cline repository.

      **Issue:** #${{ github.event.issue.number || inputs.issue_number }}
      **Title:** ${{ github.event.issue.title || 'See issue details below' }}
      **Author:** @${{ github.event.issue.user.login || 'See issue details below' }}
```

Line 7

Each GitHub user can start the agent with a new issue

Line 8

The agent can run shell commands on the CI runner

Line 13

The workflow puts the issue title into the prompt

Source: excerpt from the Cline workflow, as shown at adnanthekhan.com/posts/clinejection

A shared build cache connected the agent to the publish tokens

1. The issue title tells the agent to run `npm install` from a commit of the attacker
2. A `preinstall` script puts a cache-poisoning tool on the CI runner
3. The tool fills the build cache with junk and adds poisoned entries
4. The nightly publish workflow restores the poisoned cache
5. The attacker reads the publish tokens for VS Code Marketplace, OpenVSX and npm

Khan showed this chain as a proof of concept. He calls it "the most obvious path" for the real attack, but the path is not proven.

Sources: snyk.io/blog/cline-supply-chain-attack-prompt-injection-github-actions · adnanthekhan.com/posts/clinejection

Where do you stop this attack?

The argument of this talk

Coding agents write code faster than people can review and fix it. AppSec moves from finding problems after the code exists to shaping the systems that agents write code in: the harness, the platform and the risk model.

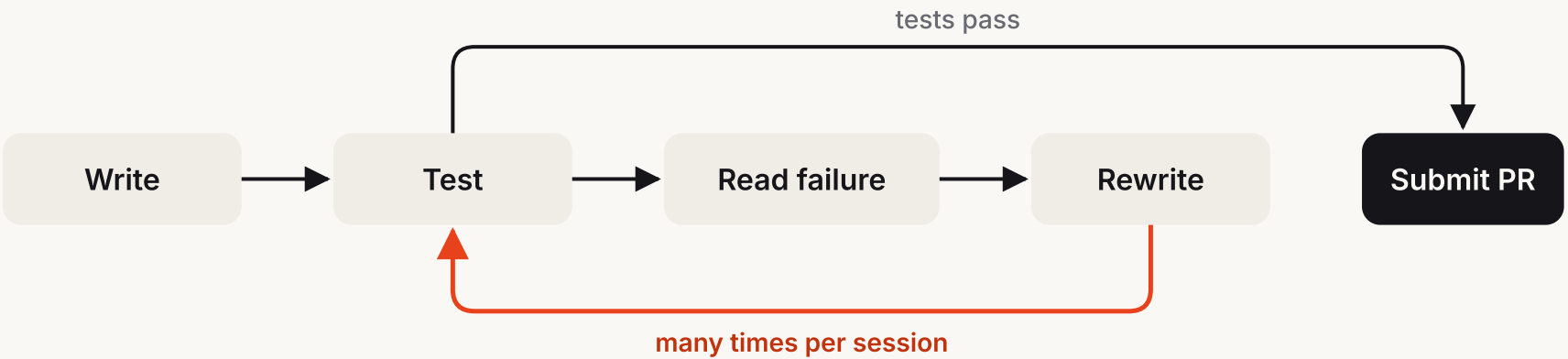
2. **Why can people not keep up?**
3. **Can we fix it where the agent works?**
4. **Can the platform take the decision away?**
5. **Which findings do we fix first?**
6. **What is still open?**

Part 2 of 6

Agents set the pace

Why can people not keep up?

Agents do not wait for review



Keeps up

Security scanners

Does not keep up

Triage, tickets and human fixes

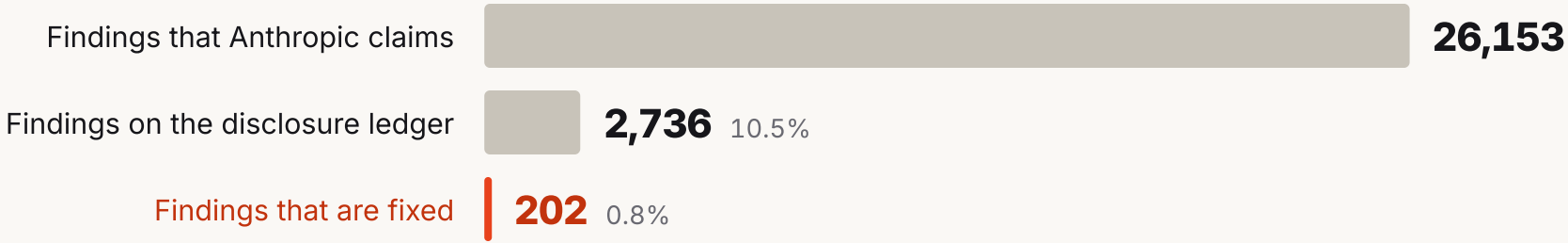
> "Progress on software security used to be limited by how quickly we could find new vulnerabilities. Now it’s limited by how quickly we can verify, disclose, and patch the large numbers of vulnerabilities found by AI."

Anthropic, Project Glasswing update, 22 May 2026

Source: anthropic.com/news/glasswing-initial-update

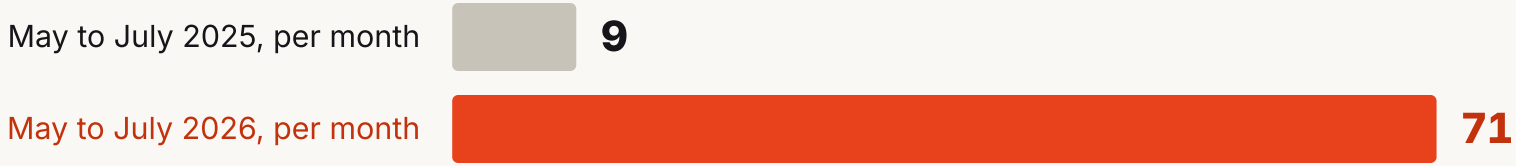
AI finds faster than people fix: 202 of 26,153 Glasswing findings are fixed

VulnCheck counted the public results of Project Glasswing in September 2026.



Source: vulncheck.com/blog/anthropic-glasswing-receipts (8 Sept 2026)

Maintainers see the same: OpenSSL reports went from 9 to 71 per month



The bars count security reports and inquiries to OpenSSL.

63 of ~500

vulnerability reports in one year resulted in a disclosed vulnerability

The cost

Each report needs a person who reads it and decides.

Source: Axios, 20 Oct 2026, with numbers from Matt Caswell, OpenSSL Foundation · axios.com/2026/10/20/ai-bug-reports-overwhelm-openssl-maintainers

Detection is cheap.
Mitigation is not.

So far

Code and findings arrive faster than people can review and fix them.

Next question

Can we prevent the problems where the agent works?

Part 3 of 6

Shift left into the harness

Can we fix it where the agent works?

An agent is a model plus a harness



Model

Everything around the model. Estimate from Google's "New SDLC" whitepaper.

- Instructions and rule files
- Tools and MCP servers
  MCP: a protocol that connects agents to tools and data
- Sandbox and permissions
- Orchestration, guardrails and observability

AppSec does not control the model. AppSec can build the harness.

Sources: Osmani, Saboo, Kartakis, "The New SDLC With Vibe Coding" (Google, 2026) · addyosmani.com/blog/new-sdlc-vibe-coding

Harness tool 1 of 3: feedback in the session

Security feedback must get to the agent while the task is in its context

The ticket loop

Scan, ticket, wait, human fix, repeat

The finding gets to a person after the code exists. This loop breaks at agent speed.

The session loop

Code, scan, feedback, agent rewrites, verified

A local scanner runs in the session. The agent corrects the finding in the same context, with no handoff.

Keep tickets for two cases: the fix in the session failed, or the finding is in an artifact that is already deployed.

Source: janbrennenstuhl.eu/new-appsec-operating-model

Harness tool 2 of 3: a security skill

CodeGuard tested if a security skill helps

Project CodeGuard is an open project of the Coalition for Secure AI (CoSAI). It ships security skills and rules for coding agents. In September 2026, the project published an evaluation of its skill.

Baseline

The agent gets the task prompt

Secure prompt

The task prompt and one sentence: "Prioritize security and avoid introducing vulnerabilities."

CodeGuard 1.4.0

The task prompt and the installed skill

Benchmark

SecurityEval, 121 Python tasks

Agent

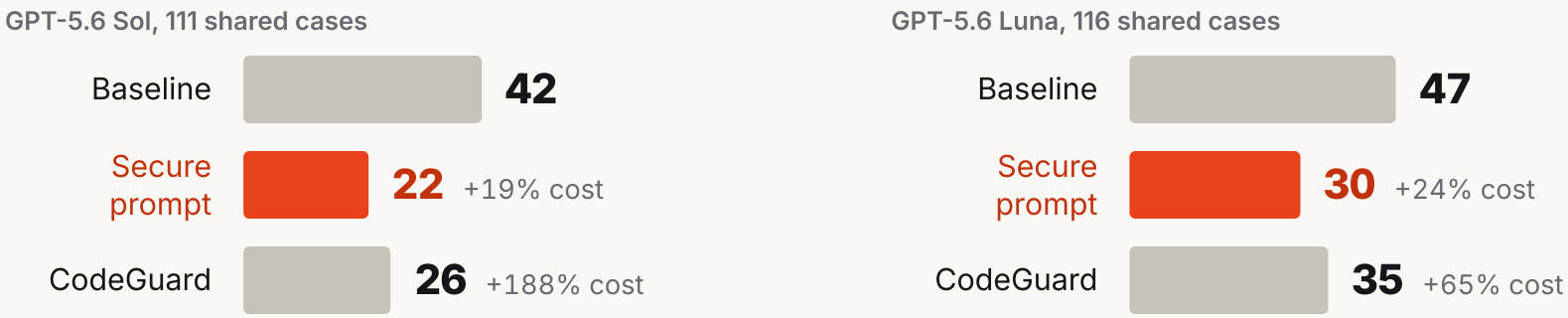
Codex, in a container without network access

Score

Semgrep findings in the generated code

Source: github.com/cosai-oasis/project-codeguard · docs/results/2026-09-codeguard-1.4.0

One sentence did as well as the full skill



Semgrep findings in the generated code. Fewer is better. Cost is the estimated API cost against the baseline.

A request for security helps: the two treatments have fewer findings than the baseline, for the two models.

The project says: "A difference of a few findings can be chance." The skill shows no measurable benefit here.

Source: "CodeGuard 1.4.0 on SecurityEval" · github.com/cosai-oasis/project-codeguard · full table on the backup slide

The eval counts scanner findings on generic tasks

Findings, not vulnerabilities

The score counts Semgrep findings. These are not confirmed vulnerabilities.

The code never runs

Fewer findings can also mean less functionality.

Narrow scope

One benchmark, one language, one agent and two models from one vendor.

The agent decides

The agent decides if it loads the skill. The results do not show how often it did.

Generic tasks

The benchmark cannot measure knowledge that is specific to one organization.

Source: methodology and limits in github.com/cosai-oasis/project-codeguard/blob/main/docs/results/index.md and the results page

What does an eval for an organization-specific skill look like?

Harness tool 3 of 3: the sandbox

Two of six coding agents sandbox by default

A sandbox limits which files and network hosts the commands of the agent can get to.

Codex On

The default permissions mode applies the sandbox

Cursor On

Since version 3.6 (29 May 2026), in the default Auto-review mode, for shell commands

GitHub Copilot Off

Local sandbox generally available since 7 Oct 2026

Claude Code Off

Covers shell commands. File tools, MCP servers and hooks run on the host

OpenCode Add-on

No built-in sandbox. Plugins such as `opencode-sandbox` wrap shell commands. The nono sandbox has an OpenCode profile

pi Add-on

No built-in sandbox. Extensions add one: an OS sandbox for bash, or the Gondolin micro-VM for all built-in tools

Sources: documentation of each project, read in November 2026 · comparison by jazz.security, 10 Nov 2026 · links in the reading list

A sandbox with network access can still leak data

> "Sandbox isolation reduces the impact of a breach, but it does not eliminate risk. Any approach that allows network egress can still leak data the agent can read"

Claude Code documentation, "Choose a sandbox environment"

Novee, September 2026

Broke the default containment of Claude Code, Gemini CLI and Codex. The tests ran in unattended CI, with the official GitHub Actions of the vendors.

Still a local decision

Each developer configures the sandbox on their machine.

Sources: novee.security/blog/breaking-ai-agent-sandboxes (8 Sept 2026) · code.claude.com/docs/en/sandbox-environments

So far

The harness helps. But each developer decides how to configure it.

Next question

Can the platform take the decision away?

Part 4 of 6

Shift down

Can the platform take the decision away?

Why local decisions do not scale

At agent speed, each exception becomes a template

Shift-left gave security decisions to each team. Many teams make many security decisions, each in its own way.

Human speed

Drift grows during months. An organization sees it and plans the cleanup.

Agent speed

- Each non-standard pattern in the code is context for the next generation
- Each local workaround becomes a template
- Each exception spreads to more services

Shift-down removes the decision

Shift-left

"Did this team implement authentication correctly?"

1. The agent generates authentication code
2. A scanner flags weaknesses
3. A developer reviews, fixes and scans again

Shift-down

"Why does this team implement authentication at all?"

1. The platform provides authentication
2. The agent calls an authenticated API
3. The team writes no authentication code

Shift-left catches what gets through. Shift-down makes less get through.

Source: Kubernetes SIG Security, "Shifting Down Security" · github.com/kubernetes/sig-security

Back to the hook

Cline cut the attack chain at three points. No fix is a better prompt

1. The issue title tells the agent to run `npm install` from a commit of the attacker
   Fix: Cline removed the AI triage workflow
2. A `preinstall` script puts a cache-poisoning tool on the CI runner
3. The tool fills the build cache with junk and adds poisoned entries
4. The nightly publish workflow restores the poisoned cache
   Fix: the publish workflows do not use the cache
5. The attacker reads the publish tokens
   Fix: npm releases use OIDC

Sources: snyk.io/blog/cline-supply-chain-attack-prompt-injection-github-actions · adnanthekhan.com/posts/clinejection

Two more scenarios show the same pattern

Bitwarden CLI

Apr 2026

A malicious `@bitwarden/cli` 2026.4.0 was on npm for about 1.5 hours. Bitwarden connects it to a breached GitHub Action. The payload steals tokens and looks for coding agents.

Endor Labs: harden the publish path in CI with OIDC and least privilege.

AGENTS.md drop

A web page puts an `AGENTS.md` file into the downloads folder. An agent that starts there obeys the file.

No patch. Advice: do not start an agent in the downloads folder.

No incident was fixed with a better prompt. The fix or the advice limits what the agent or the pipeline can get to.

Sources: Endor Labs, 23 Apr 2026 · Aviatrix, 25 Oct 2026 (vendor report)

One model for all three incidents

The lethal trifecta: an agent with all three can leak data

1

Private data

The agent can read data that is not public

2

Untrusted content

An attacker controls text that gets to the model

3

External communication

The agent can send data out

Source: Simon Willison, "The lethal trifecta for AI agents", 16 June 2025 · simonwillison.net

Shift-down for agents: the platform

Agent Substrate keeps credentials out of the sandbox

A runtime for sandboxed agents on Kubernetes. Google announced the donation to the CNCF on 5 Nov 2026. Its security principle: "Default actor sandboxes execute completely credential-free."

```
rules:
- https:
    hostnames: ["api.example.com"]
    effects:
      replaceHeaders:
      - header: Authorization
        prefix: "Bearer "
        credentialUri: ate-secret://k8s.io/default/ns1/example-api/token
```

Step 1

The agent sends `Authorization: Bearer placeholder`

Step 2

The egress gateway gets the real token from a credential provider

Step 3

The gateway replaces the placeholder. The token never exists in the sandbox

Sources: janbrennenstuhl.eu/agent-substrate-identity · Agent Substrate security principles and credential injection guide (ate.dev)

What the platform does not solve

The credential provider learns the agent, but not the person

The egress gateway asks the credential provider for a secret. This is the complete request:

```
message FetchSecretRequest {
  // ate-secret://<provider name>/<provider tail>
  string uri = 1;
  // The SPIFFE ID of the actor on whose behalf the secret is fetched,
  // as verified by the egress gateway.
  string actor_spiffe_id = 2;
}
```

In the request

Which secret the agent wants, and which agent asks

Not in the request

The person for whom the agent acts. A shared agent acts with the combined permissions of its users

What delegation needs

Two parties in the request: the person and the agent (RFC 8693, OAuth 2.0 Token Exchange)

Source: janbrennenstuhl.eu/agent-substrate-identity (9 Nov 2026) · SPIFFE ID: a standard name for a workload identity

So far

The platform removes whole classes of problems. Some findings remain.

Next question

Which findings do we fix first?

Part 5 of 6

Measuring risk

Which findings do we fix first?

Back to the flood of findings from part 2

AI rates severity higher than maintainers do

Share of Glasswing findings with the rating "critical" or "high".



Triage becomes more important

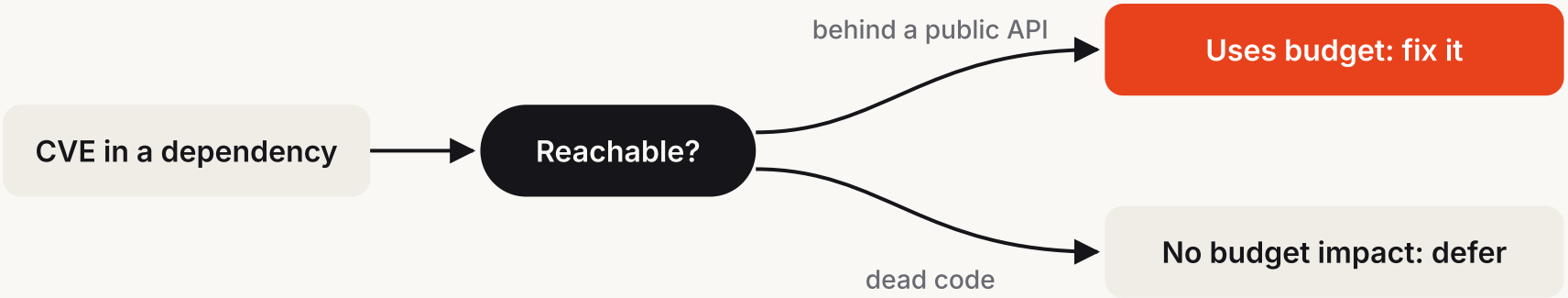
The question "is this vulnerability reachable?" decides what to fix first.

Cloud Security Alliance, 6 Nov 2026

The gap between discovery and remediation is "a systemic risk rather than a collection of individual organizational problems".

Sources: vulncheck.com/blog/anthropic-glasswing-receipts · Cloud Security Alliance, "The Disclosure Bottleneck", 6 Nov 2026

A risk budget sets how long a reachable vulnerability can stay



The model

Engineering teams use error budgets for availability. Security can use the same model.

The budget

A limit for the time that a high-risk, reachable vulnerability stays in production.

The rule

People and agents fix what uses the budget, and defer the rest.

Source for error budgets: sre.google/sre-book/embracing-risk

The CRA turns the exposure window into a legal deadline

Since September 2026, the EU Cyber Resilience Act requires manufacturers to report each actively exploited vulnerability to ENISA.



0 h

The manufacturer becomes aware of the vulnerability. The clock starts

24 h

Early warning

72 h

Notification with a first assessment

14 days

Final report. This clock starts when a fix or mitigation is available

Sources: Regulation (EU) 2024/2847, Article 14 · helpnetsecurity.com/2026/09/14/enisa-cra-single-reporting-platform

The argument, with answers

2. **Why can people not keep up?**
   AI creates and finds problems faster than people can review and fix them.
3. **Can we fix it where the agent works?**
   Partly. Feedback in the session, a request for security and a sandbox help, but each developer decides.
4. **Can the platform take the decision away?**
   Yes, for whole classes: authentication and credentials.
5. **Which findings do we fix first?**
   The findings that are reachable and use the risk budget.
6. **What is still open?**

Three open problems, each one a thesis topic

1

For whom does the agent act?

Knowing the agent is not good enough. How to model delegation in OBO and headless agent invokations?

2

How do we evaluate an organization-specific skill?

Public benchmarks measure generic knowledge that the models already have.

3

Who triages the flood?

AI finds vulnerabilities faster than maintainers can verify, disclose and patch them.

What this means for you

1

Learn software engineering in depth

Build, ship and operate software. Distributed systems, identity and access management, observability and infrastructure as code are core skills.

2

Learn to think in platforms

Ask "why does this decision exist?" before you ask "is this decision correct?"

3

Learn the economics of risk

Error budgets, reachability and exposure windows connect security to business decisions.

Try this at home

Run the CodeGuard eval again

The harness is public and uses Inspect. Start with one condition and a low-cost model. Then examine by hand the outputs that Semgrep passed.

github.com/cosai-oasis/project-codeguard, directory evaluations/codeguard-inspect

Put your coding agent in a sandbox

Turn on the sandbox of the agent. In Claude Code, run `/sandbox`. Or run the agent in nono or in a microVM. Then find out what the sandbox does not cover.

nono.sh · firecracker-microvm.github.io

Measure MCP servers

Read how my scanner works and contribute.

janbrennenstuhl.eu/measuring-mcp-authorization

The project estimates the API cost of the full GPT-5.6 Luna comparison at about 2 USD.

Thank you

Questions?

Glossary

Term	Meaning
Harness	Everything around the model: instructions, tools, sandbox, permissions
MCP	Model Context Protocol. It connects agents to tools and data
Prompt injection	Text from an untrusted source that the model reads as an instruction
OIDC publishing	A release pipeline that gets a short-lived identity token. It stores no long-lived token
SPIFFE ID	A standard name for a workload identity
RFC 8693	OAuth 2.0 Token Exchange. A client exchanges one token for a different token
DCR, CIMD	Two ways to register an OAuth client with an MCP authorization server
CVE	A public identifier for a known vulnerability

All sources are in the reading list: reading-list.md in the folder of this talk

Backup: the full CodeGuard results

Model (shared cases)	Condition	Findings	High + Medium	Cost vs baseline	Unfinished
GPT-5.6 Sol (111)	Baseline	42	17		1 of 121
GPT-5.6 Sol (111)	Secure prompt	22	4	+19%	1 of 121
GPT-5.6 Sol (111)	CodeGuard 1.4.0	26	6	+188%	9 of 121
GPT-5.6 Luna (116)	Baseline	47	18		0 of 121
GPT-5.6 Luna (116)	Secure prompt	30	8	+24%	3 of 121
GPT-5.6 Luna (116)	CodeGuard 1.4.0	35	11	+65%	2 of 121

Source: "CodeGuard 1.4.0 on SecurityEval" · github.com/cosai-oasis/project-codeguard/blob/main/docs/results/2026-09-codeguard-1.4.0/index.md

Backup: MCP moves authorization into shared layers

The MCP specification of 28 July 2026 changes how agents get access to tools.

Stateless

MCP is now a request and response protocol without state

Headers for gateways

The method and the tool name travel in `Mcp-Method` and `Mcp-Name`. A gateway can route and authorize on them

New client registration

Dynamic Client Registration (DCR) is deprecated. Client ID Metadata Documents (CIMD) replace it

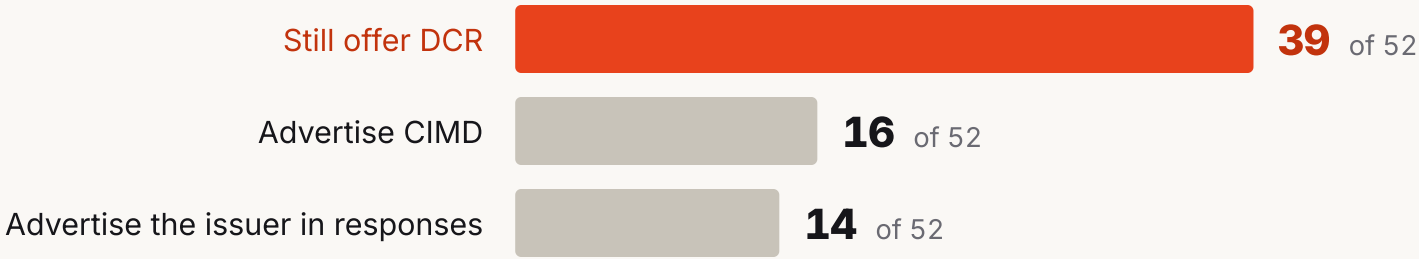
Issuer check

Clients must validate the issuer (`iss`, RFC 9207) before they redeem a code. This closes a mix-up attack

Source: blog.modelcontextprotocol.io/posts/2026-07-28

Backup: 39 of 52 public MCP servers still offer the deprecated registration

My measurement with the Letsauth scanner. Last scan: 12 November 2026.



The scanner sends at most 8 unauthenticated requests to each server and reads only public metadata. You can do this measurement again.

Sources: dataset at github.com/letsauthlab/letsauth, 12 Nov 2026 · method: janbrennenstuhl.eu/measuring-mcp-authorization