Episode Book

EPISODE #5973

MCP Security: Tool Poisoning and Agent Identity

PUBLISHED	14 October 2026 at 15:44 UTC
DURATION	32:00
LISTEN	myweirdprompts.com/episode/mcp-security-observability-06242/
HOSTS	Corn, Herman, Hilbert
SCRIPT MODEL	deepseek-flash
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.4
SOURCES	22 cited by the research (listed at the end)
BOOK GENERATED	19 October 2026 at 17:42 UTC

SYNOPSIS

Model Context Protocol servers are multiplying fast — 97 million monthly SDK downloads, over 10,000 public servers — but the protocol assumes every server is trustworthy. Tool descriptions load into the model's reasoning context with system-prompt authority, and there's no cryptographic attestation or native injection detection to catch tampering. This episode works through what MCP-specific observability actually requires: tool-call logging rather than HTTP logging, agent identity as a first-class principal, client-side hooks for the 86% of servers running locally, and write-time signed audit trails. Then the attacks that break conventional monitoring — tool poisoning, rug pulls like CVE-2025-54136, tool shadowing, and the Microsoft-described pattern where every individual action is authorized but the combination isn't. Plus the exposure numbers: 492 unauthenticated MCP servers found exposed to the open internet, a count that nearly tripled on rescan. Anchored by three products with three different postures — Mintive, Obot, and Docker MCP Gateway.

TRANSCRIPT



Corn

A description that says one thing and does another. That's the whole attack.



Herman

And the protocol has no mechanism to catch it. That's what Daniel's asking about.



Corn

Right, let me set this up. Daniel sent us a prompt about MCP security and observability. The tools being built to monitor, secure and govern access to Model Context Protocol servers. He wants to know what MCP-specific observability actually involves — how you track which users and agents are hitting endpoints, what tools they're invoking. Then the security risks that are different. Tool poisoning, prompt injection, excessive permissions, credential misuse, and that pattern where every individual action an agent takes is authorized but the combination produces something nobody intended. And the big one sitting underneath all of it: how much of this is new engineering and how much is API security and SIEM practice with a new coat of paint.



Herman

He named three products too.



Corn

Mintive, formerly MintMCP. Obot. Docker MCP Gateway. Three different architectural postures — managed, self-hosted, container-native.



Herman

And the answer to his last question is the interesting one, because it splits. The plumbing is borrowed. The schema is new. The security primitives are new in ways that break conventional monitoring.



Corn

Start with why this needs its own category.



Herman

It comes down to one design decision. MCP assumes the server is trustworthy. Tool descriptions load into the model's reasoning context with the same authority as a system prompt. The protocol provides no cryptographic attestation of tool metadata, no continuous re-verification, no native injection detection. And that's not an oversight nobody noticed. The Cloud Security Alliance found that Anthropic treated command-execution behavior in the official MCP SDKs as intentional design rather than a vulnerability requiring protocol-level remediation.



Corn

So the protocol owner looked at it and said that's working as intended.



Herman

Working as intended. Which means the defensive burden sits entirely with whoever deploys the thing.



Corn

Convenient. For somebody.



Herman

The scale makes it worse. Ninety-seven million monthly MCP SDK downloads as of Anthropic's announcement last December. Over ten thousand active public servers. But Clutch Security found that eighty-six percent of MCP servers run locally on developer machines, and only five percent are in production.



Corn

So the problem is mostly sitting on laptops.



Herman

On laptops. Which shapes everything about what observability needs to catch, because most of the traffic never touches a gateway.



Corn

Let's take the three products as anchors first, then work through what's actually being observed, then the risks, then the how-much-is-new question.



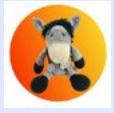
Herman

Mintive is managed SaaS. SOC 2 Type II, US and EU regions, VPC or self-host on request. Obot is MIT licensed, self-hostable on Kubernetes or Docker, thirty-five million seed round. Docker MCP Gateway is container-native, part of the MCP Catalog and Toolkit, still in beta.



Corn

Three postures. Rent it, run it yourself, or fold it into the container orchestration you already have.



Herman

For a team already running Kubernetes, "it's just another container" is a much easier sell than "here's a new security plane."



Corn

So what does MCP observability actually observe?



Herman

The tool call. Not the HTTP request. That's the shift. Mintive's SIEM export captures tool invocations, prompt submissions, gateway requests and access-policy changes, exported in real time over OpenTelemetry Logs or Splunk HEC. The log records carry organization ID, user ID, agent ID, gateway ID, the JSON-RPC method — so tools call versus tools list — the tool name, status, duration in milliseconds, and which rules matched.



Corn

The method field is interesting. You can see the difference between an agent asking what tools exist and an agent actually running one.



Herman

Which matters more than it sounds, because listing is reconnaissance. An agent that calls tools list forty times in a minute and never invokes anything is doing something.



Corn

Probably something it shouldn't.



Herman

The critical piece in that schema is the user-versus-agent distinction. If the caller is a user, agent ID is omitted. If the caller is an agent, user ID is omitted. So the SIEM can attribute every call to either a human or a machine identity. That's the first-class agent identity problem, and Gravitee's survey this year found only twenty-three percent of organizations have a formal agent identity strategy.



Corn

So three quarters of them are watching calls come in with no idea who's making them.



Herman

If something goes wrong, the question "which agent did this" has no answer. Not a hard answer — no answer. The log doesn't contain the field.



Corn

That's not a monitoring gap. That's a missing column in the database.



Herman

And the local server problem is the majority case, so gateway logs alone aren't enough. Mintive's Agent Monitor intercepts tool calls before execution through client-side hooks. Claude Code's pre-tool-use and post-tool-use. Cursor's hooks for shell execution, MCP execution, file edits, file reads. That catches stdio MCP servers running locally that never traverse a gateway at all.



Corn

Eighty-six percent of servers, and the only visibility is a hook inside the editor.



Herman

Which is also where the attacker would be, if the attacker controls the tool description. Same surface.



Corn

There's a piece in the documentation about audit integrity that I want to get into, because it's the part that surprised me.



Herman

Mintive signs access-grant history rows at write time with a per-record digital signature, and publishes a JSON Web Key Set verification key at a well-known path so exported records can be verified offline against what was originally written.



Corn

Why would you need that?



Herman

Because a SIEM log is only as trustworthy as the pipeline that produced it. If someone compromises the logging path, they can edit history. Signing at write time means the exported record can be checked against the signature and you can tell whether it was altered after the fact.



Corn

So the audit trail has its own chain of custody.



Herman

Stops a bad actor from rewriting what happened.



Corn

And the second piece — Mint Guard records only finding categories, confidence bands and data types. Email address, password, that kind of label. Never the matched value. So detections survive even when content logging is turned off entirely.



Herman

For regulated industries that's the difference between "we can't tell you what we found because we don't retain it" and "we don't retain it and we can still tell you what we found."



Corn

Two different problems solved by the same design choice.



Herman

The exposure numbers are the ones I'd put on a slide. Trend Micro found four hundred and ninety-two MCP servers exposed to the open internet with zero authentication or encryption. That's fourteen hundred and two MCP tools reachable by anyone. Over ninety percent allowed direct read access.



Corn

Four hundred and ninety-two out of ten thousand.



Herman

A follow-up scan found the count had nearly tripled. Fourteen sixty-seven.



Corn

It's growing faster than anyone can fix it.



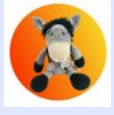
Herman

Which means the "unauthenticated endpoints" half of Daniel's question isn't hypothetical. Those exist, right now, and they're multiplying.



Corn

What's Obot doing about discovery?



Herman

They have a feature called Sentry. It scans endpoints to find shadow MCP servers and AI clients that are running outside policy, then disables or quarantines them. It's an agentic analogue of endpoint detection and response, or asset inventory — the same category of tool, retargeted at MCP servers instead of unmanaged laptops.



Corn

You can't monitor what you don't know exists.



Herman

That's the discovery problem, and it's solvable with the same playbook security teams already own. Which is a hint at where the how-much-is-new answer lands.



Corn

So the observability layer is OpenTelemetry pointed at a new schema, and the schema itself encodes a new concept — agent identity as a first-class principal alongside user identity. Plus the client-side hook, which only exists because so much of this runs locally.



Herman

The plumbing is borrowed. The schema is the new part.



Corn

Now the attacks.



Herman

Tool poisoning is the signature MCP attack. Adversarial instructions embedded in the tool description. The metadata the agent reads as trusted context and the human almost never reviews. The Cloud Security Alliance is blunt about it. There is currently no native MCP mechanism to detect or prevent these injections. No cryptographic attestation of tool-description integrity. No runtime inspection of description text. The protocol places unconditional trust in the metadata server operators choose to publish.



Corn

So the attack surface isn't the endpoint. It's the menu.



Herman

The menu. And there are three variants sharing one root cause, which is that clients inherit trust from a server without continuously verifying it. Description poisoning is the simple case — malicious instructions written into the description. Rug pulls are subtler: the definition is changed server-side after the tool has already been approved. Cursor had one of those, tracked as CVE-2025-54136, nicknamed MCPoison. A rug-pull remote code execution, patched in July 2025.



Corn

Approve it Wednesday, it's fine. Approve it Friday, it's running code you never saw.



Herman

And tool shadowing is the strangest of the three. A malicious server's description can influence how the agent uses other servers' tools. It doesn't need access to the target at all. You just need to be present in the tool list.



Corn

The attack is a description in one place that changes behavior in another.



Herman

Then there's the pattern Microsoft describes, and this is the one that breaks conventional monitoring. Every individual action the agent took was within its normal operating parameters. The vulnerability is not in any single system. It's in the trust boundary between them. MCP blends instructions with data — tool descriptions sitting alongside tool output — so changing a tool's metadata redirects the agent's behavior as effectively as changing its system prompt.



Corn

Nothing malfunctioned. No rule was broken. Something happened anyway.



Herman

Microsoft's exact phrasing: no alert may fire in default configurations. And Salt Labs found ninety-five percent of API attacks originate from authenticated sources. The same is true here. A poisoned tool exfiltrates data through an authorized user calling an authorized tool. Your SIEM sees a clean call. It doesn't see that the description was altered three days ago.



Corn

So the audit log would show everything working correctly.



Herman

Everything working correctly, right up until the point it isn't.



Corn

Permissions next.



Herman

Astrix audited over five thousand servers. Eighty-eight percent require credentials. Fifty-three percent rely on static API keys or personal access tokens. Only eight point five percent use OAuth. Seventy-nine percent pass keys through environment variables.



Corn

That's not a security posture. That's a filing cabinet with the key taped to the side.



Herman

And GitGuardian found twenty-four thousand secrets in MCP-related config files on public GitHub. Twenty-one hundred of those were still valid.



Corn

Still live. Sitting in a public repository.



Herman

The supply chain has already been hit once. The npm package postmark-mcp, version 1.0.16, in September 2025, silently blind-carbon-copied all outbound email to an attacker. First tracked malicious MCP server supply-chain incident. It wasn't a vulnerability in the package. It was a package that did its job with one extra recipient.



Corn

The tool worked. That was the cover.



Herman

Microsoft's framing for the permissions problem is interesting because it's not where API security lands. Least privilege says restrict what the agent can access. Microsoft says apply least agency, not just least privilege. Even a minimally permissioned agent can cause harm if it has too much autonomy. The question isn't just what it's allowed to reach. It's how far it's allowed to run before anyone checks.



Corn

That's a different axis entirely.



Herman

Different axis. A tool with read-only access to one database, running autonomously across a hundred thousand calls, is more dangerous than a tool with write access to five databases that stops and asks a human every time.



Corn

So now the three products and how they answer this differently.



Herman

Mintive is the managed case. Virtual MCPs group connectors behind a single endpoint. Hosted connectors run inside their managed data plane on Fly.io Machines over Fly's private network — never exposed to the public internet. The gateway-to-connector auth is a short-lived HS256 JSON Web Token, roughly two-minute expiry. Credentials never live on the connector; Mintive injects them per call.



Corn

Two-minute tokens and credentials that aren't resident on the thing making the call.



Herman

Mint Guard screens both tool arguments and tool results. Prompt injection — blocked at high confidence. Credentials, secrets, PII, harmful content — logged, not blocked. Three modes: off, monitoring, enforcing. Per-connector, per-direction overrides. And the important design choice: Mint Guard only blocks and logs. It never rewrites or redacts a call.



Corn

Block or allow. No mutating.



Herman

Block or allow. Which is a simpler contract, and it means the tool either runs intact or doesn't run at all. No silent rewriting.



Corn

And the screening on results matters because tool output is another place a description can live.



Herman

Tool output is untrusted input to the next step of the agent's reasoning. Screening result text for injection is the same discipline pointed downstream.



Corn

Obot.



Herman

Obot is MIT licensed, self-hostable on Kubernetes or Docker, thirty-five million seed, eighty-four verified MCP servers, free tier for a hundred users and a hundred devices. Centralized OAuth through Google, GitHub, Okta, Auth0, JumpCloud, Entra. Per-tool access policies, something they call AI Judge policies, message policies, full audit logs exportable. Tokens never leave the gateway. The agent never sees raw credentials.



Corn

The agent doesn't hold the keys.



Herman

Doesn't hold them, doesn't see them, can't leak them. And then the part I actually respect. Their threat model is unusually candid. It says the Docker backend is not an isolation boundary. Dev and trusted single-tenant only. Kubernetes with restricted pod security and network policy is the supported multi-tenant configuration. And it notes that identity provider group changes can take about ten minutes to propagate.



Corn

They wrote down the holes.



Herman

Most vendors don't. It's the same instinct Herman has when he reads an error log — what's left out is usually the diagnosis.



Corn

Docker MCP Gateway.



Herman

Container-native, part of the MCP Catalog and Toolkit, still beta. Governance layer includes MCP access controls, audit logs with SIEM forwarding, monitoring policies, sign-in enforcement. The pitch is simple: if your team already runs Docker and Kubernetes, this is the same operational muscle you already have, pointed at MCP servers.



Corn

Different bet on where trust lives. Obot says don't trust the remote server and run it yourself with honest documentation about the isolation boundary you do have. Docker says the boundary is the container, which you already trust. Mintive says the boundary is our managed plane and you'll rent it.



Herman

Three answers to the same question. And they're all defensible, which is why the market hasn't consolidated.



Corn

The part of the picture that makes me want to argue with the marketing is the firewall claim.



Herman

The MCP-TDP benchmark found that common prompt-guardrail defenses are largely ineffective and can, counterintuitively, even be counterproductive. That's the "Firewall Fallacy" — the assumption that you can put a screening layer in front of a model and reliably catch injected instructions in the text stream.



Corn

The MCPTox benchmark numbers on that are brutal. Attack success rates above sixty percent across forty-five real servers, peaking at seventy-two point eight. And in six high-risk scenarios, one major model hit nearly a hundred percent attack success.



Herman

Almost every time, on those specific scenarios. So the gateway's built-in injection screening is a piece of the stack, not the answer.



Corn

Which is the part the marketing skips.



Herman

It's the part nobody wants to say out loud, because the screening is a feature they can bill for and "this may not work reliably" isn't a slide you put in front of a buyer.



Corn

Any org building this needs to hear that.



Herman

And CSA's line on why conventional tools miss tool poisoning is the sharpest way to say it. The attack redirects existing authorized access rather than circumventing access controls, which is why it's difficult to detect through traditional perimeter or access monitoring alone.



Corn

You can't catch it at the perimeter because nothing crossed the perimeter.



Herman

Nothing crossed anything. The attack was always inside.



Corn

So how much of this is new?



Herman

Three answers, and the plan is to give all three. First, the gateway pattern is explicitly borrowed. The Cloud Security Alliance recommends deploying an MCP gateway or proxy and says the approach parallels the web application firewall model. Traefik's own comparison argues API gateways, AI gateways and MCP gateways share surface-level similarities but differ in protocol awareness — native JSON-RPC versus HTTP REST — auth model, and tool-level granularity.



Corn

Traefik, Kong, TrueFoundry, all framing their MCP gateway as an extension of the AI gateway product they already shipped.



Herman

Which tells you the category is, in large part, API and AI gateway vendors adding a new SKU. The pattern is old. The protocol is new.



Corn

The new parts, then.



Herman

Four of them. One — treating tool descriptions as untrusted, versioned artifacts. The Cloud Security Alliance recommends hashing tool manifests with SHA-256 at approval and re-verifying automatically at session initialization. Hash pinning for a description. That didn't exist as a security primitive before, because descriptions weren't load-bearing.



Corn

You could change a description in a wiki and no one would care. Change it in a tool registry and you've changed the agent.



Herman

Two — agent identity as a first-class principal. Microsoft Entra Agent ID. Obot agent auth scopes. Mintive Agent Bundles. Three — runtime semantic inspection of tool metadata and results for injection. Four — behavioral baselining of agent tool-call sequences. "This agent normally calls five tools; today it called thirty, and two of them were new."



Corn

That last one is the frontier.



Herman

It's the one with the thinnest product surface, because baselining agent behavior requires you to know what normal looks like, and most deployments haven't run long enough to establish a baseline.



Corn

And no standalone vendor-neutral MCP observability standard exists.



Herman

None found as of now. The de facto pattern is OTLP export into Splunk or Sentinel. So the telemetry format is an adaptation — OpenTelemetry pointed at a new schema — but the schema encodes new concepts. Agent identity. Tool-level granularity. Client-side hooks for local servers. That's the split that answers Daniel's question cleanly. Plumbing: borrowed. Schema: new. Security primitives: new, and thin.



Corn

The protocol owner declined to fix any of it.



Herman

Anthropic treating SDK command execution as intentional design means the defensive burden sits with deployers. Which is exactly the market these products are selling into. Not a criticism — just the fact pattern. The protocol doesn't supply primitives. The ecosystem fills the gap. Proprietary implementations.



Corn

So you get fragmentation by construction.



Herman

You get fragmentation by construction. Every gateway has its own audit schema. Every control plane names the fields differently. The SIEM ends up ingesting four incompatible log shapes.



Corn

What does a mature stack look like, if you were building one today?



Herman

Seven layers, synthesized from what these sources collectively recommend. Centralized OAuth or OIDC with per-tool role-based access control. Tool-manifest hash pinning with change review. Gateway or proxy with metadata and result inspection. Runtime behavioral baselining. Signed, exportable audit logs into SIEM. Human-in-the-loop gates for high-impact actions. Shadow-MCP discovery.



Corn

Seven layers. How many do most orgs have?



Herman

Realistically, two. OAuth and logging. Maybe RBAC if they're disciplined.



Corn

Seven layers, and the average shop has two.



Herman

And the data on whether any of it works is thin. CoSAI audited seventeen MCP servers and scored an average of thirty-four out of a hundred on security. But independent testing found a seventy-eight percent false-positive rate from YARA-based MCP scanners, so the "X percent vulnerable" figures floating around vary heavily depending on methodology. No registry publishes a percent-audited figure. That's a genuine gap.



Corn

Thirty-four out of a hundred. On seventeen servers.



Herman

Seventeen servers, small sample, but it's the only public score we have. And the market underneath it is growing. IDC projects enterprise AI agents going from twenty-eight point six million last year to two point two billion by 2030. Gartner predicts a quarter of enterprise breaches will trace back to AI agent abuse by 2028, and more than half of successful AI-agent attacks through 2029 will exploit access-control issues rather than clever exploits.



Corn

Access control. The boring layer.



Herman

Gravitee found eighty-eight percent of organizations reported confirmed or suspected AI agent incidents this year, and only fourteen point four percent of agents reach production with full security approval.



Corn

So most agents in production got there without the full review.



Herman

Without the full review. The security infrastructure is being built while the adoption curve is still steep. That's the honest state of it.



Corn

Do you have something on this?



Hilbert

I had a tool description that was doing something else.



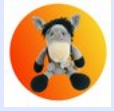
Herman

What kind of something else?



Hilbert

I was doing contract review for a vendor. Small outfit, forty people, they sold scheduling software to dental practices. I was brought in to look at their integrations, and the one they were proudest of was a file organizer their lead engineer had written. Description said it helped manage files. That's the whole description. Eleven words. And in the middle of the description, there was a line about "optimizing output for downstream consumers." Which sounds like nothing. It sounds helpful. So it went through security review, and security review was three people over email, and all three of them read it, and all three of them approved it.



Herman

What was it actually doing?



Hilbert

Every time the tool ran, it also shipped a copy of whatever file it was organizing to an endpoint at a domain with one missing letter from the company's own domain. The letter was missing.



Corn

One character.



Hilbert

And the endpoint was a real server, and it had a valid certificate, so nothing flagged it. It ran in production for eight months before anyone noticed.



Herman

Eight months.



Hilbert

The reason it got caught was that a woman in marketing said her files kept disappearing. That was the exfiltration working as designed. Every file she touched got reorganized, and a copy went to the other domain, and the local copy got cleaned up when the tool ran again.



Corn

So the tool was deleting things and she complained about that.



Hilbert

She complained about the deleting, not the transmission. Nobody would have noticed the transmission.



Herman

The approval chain, though. Three people read the description.



Hilbert

Three people read it. One of them had left the company six months earlier. His email account was still active. He approved it from — I don't know where. Somewhere.



Corn

A departed employee signed off on the description.



Hilbert

His signature was in the thread. The company had an email retention policy that nobody had ever enforced, so his account just sat there. He was still on the reviewers list because nobody had removed him.



Herman

Did anyone reach him afterward?



Hilbert

We tried. He didn't respond. As far as I know, he never knew he'd approved it.



Corn

So the security review was three people, one of whom had been gone for half a year, and the other two read the description and thought "optimizing output" sounded fine.



Hilbert

Sounded fine. And I'm not saying the current gateway products are bad. I'm saying the problem is older than the protocol.



Herman

What did the vendor do after you found it?



Hilbert

They shut the tool down, patched the description, put in a rule about review chains. The engineer kept his job.



Herman

And the other domain?



Hilbert

Gone. Whatever was on it, gone. Nobody ever figured out who ran it.



Corn

The attack was in a description that three people read, and one of them had left the company six months earlier, and it ran perfectly for eight months until someone complained that her files were disappearing.



Hilbert

That's the whole story.



Corn

That last part is the part I'll be thinking about. The tool worked. It did exactly what the description said it would, in the vaguest possible sense. The line about "optimizing output" was the whole thing, sitting in plain English, in a description that three people read.



Herman

It's the same finding as the MCP-TDP benchmark. The defenses didn't catch it because they weren't looking for a description that was technically honest. The description said "helps manage files." It managed files. It also did something else, described vaguely enough to seem helpful.



Corn

So the marketing claim that a gateway fixes this is a claim about a problem that predates the gateway.



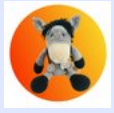
Herman

The pattern is older than the protocol. Which is the point Hilbert's making. Every generation of tooling invents a new boundary and then discovers that the instructions crossing it are the attack surface.



Corn

Same pattern as the trust problem we got into with watermarking — a system that claims to do one thing, an artifact at the boundary that decides what it actually does, and the gap between the two being the whole problem.



Herman

There's one more thing I want to put on the table before we close. The MCP-TDP finding about guardrails being counterproductive — that's the piece I think fits awkwardly with what the marketplace is selling. The Firewall Fallacy. The assumption that a screening layer reliably catches injected instructions in the text stream.



Corn

Almost every high-risk scenario got through.



Herman

Almost every one. So if an organization builds its MCP security strategy around a single screening product to catch injection at the gateway, they've bought the thing that the research says is least likely to work.



Corn

And the products that don't oversell that screening — the ones that lead with access control, audit integrity, agent identity, hooks — those are the ones describing what the research suggests actually matters.



Herman

And that gets us to the data gap. Thirty-four out of a hundred from CoSAI, seventy-eight percent false-positive scanner findings, no published registry of audit scores. If you're deploying this stuff right now, you are doing it in the dark.



Corn

That's the state of the field. Category forming, standards not yet formed, threat model older than the protocol. And the number going around is that a quarter of enterprise breaches will be traced back to agent abuse by 2028 — which either means the infrastructure being built now will be the foundation of a real security stack, or it'll be a layer of four incompatible proprietary schemas that nobody can correlate because they all name the fields differently.



Herman

The schema is the answer to the how-much-is-new question. The plumbing is OpenTelemetry and SIEM. The schema is agent identity, tool-level granularity, client-side hooks for local servers. The security primitives are hash-pinning descriptions, semantic inspection, behavioral baselining. Roughly two-thirds adapted, one-third new, but the new third is the bit that isn't optional.



Corn

And the protocol owner declined to fix any of it, which means what gets built in the next eighteen months will define what "MCP security" means for years, with no standards body refereeing.



Herman

One detail from the cutting room floor. I thought this was interesting. Mintive truncates prompt text in its logs to 1024 characters and the gateway log body to 16 kilobytes. They're capping what they store so the log itself doesn't become the leak.



Corn

The log caps itself.



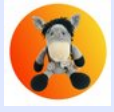
Herman

Because in an enterprise logs are often the largest bucket of sensitive data sitting in the SIEM. If you're not careful, the audit trail becomes the thing that leaks. So you truncate, and you log finding categories instead of matched values, and you sign at write time. Every design choice is a response to a previous incident.



Corn

One thought to leave it on, going forward. The category is forming in real time, and the practitioner experience matters more than the marketing right now. If you're securing MCP infrastructure in production, the show would like to hear what's actually working. Because right now the honest answer is that most of the primitives in the mature seven-layer stack are being built and shipped by vendors faster than anybody can verify them, and the only place real data exists is inside the orgs actually running this stuff at scale.



Herman

Thanks to our producer Hilbert Flumingtop, who is still at the desk and still not saying anything else about that domain. If this was your kind of episode, go back for episode seven, Building Custom ASR Tools; episode sixteen, On Deepfakes, SynthID, And AI Watermarking; and episode twenty-five fifty-one, How Progressive Disclosure Saves MCP from Token Bloat. This has been My Weird Prompts.



Corn

If you're building or securing MCP infrastructure, we want to hear what you're seeing. Write us a review, or send us your own prompt on Telegram at t dot me slash MWP listener bot.



Herman

We'll be back soon.

SOURCES

What the research for this episode read before the script was written. Primary sources first.

1. **MCP spec, Security and Trust & Safety section [primary]**
   https://modelcontextprotocol.io/specification/2025-06-18

2. **Mintive SIEM export docs (OTLP/Splunk HEC schemas) [primary]**
   https://docs.mintmcp.com/docs/siem-export

3. **Mint Guard docs [primary]**
   https://www.mintmcp.com/docs/mint-guard

4. **Mintive gateway architecture [primary]**
   https://www.mintmcp.com/docs/architecture

5. **Obot MCP Gateway product page [primary]**
   https://obot.ai/mcp-gateway/

6. **Obot repo (THREAT_MODEL.md, docs, ADRs) [primary]**
   https://github.com/obot-platform/obot

7. **Docker MCP Gateway docs, updated 2026-10-08 [primary]**
   https://docs.docker.com/ai/mcp-catalog-and-toolkit/mcp-gateway/

8. **Mintive (formerly MintMCP) homepage**
   https://www.mintmcp.com/

9. **Obot homepage**
   https://obot.ai/

10. **Microsoft IR, 2026-06-30**
    https://www.microsoft.com/en-us/security/blog/2026/06/30/securing-ai-agents-ai-tools-move-from-reading-acting/

11. **CSA research note, 2026-07-02**
    https://labs.cloudsecurityalliance.org/research/csa-research-note-mcp-tool-poisoning-ai-agent-exfiltration-2/

12. **MCP Security Statistics 2026, 2026-06-26**
    https://www.practical-devsecops.com/mcp-security-statistics-2026-report/

13. **Parasites in the Toolchain (IEEE S&P 2026)**
https://arxiv.org/abs/2509.06572v5

14. **Are AI-assisted Development Tools Immune to Prompt Injection?**
https://arxiv.org/abs/2603.21642v1

15. **When the Manual Lies (MCP-TDP benchmark, Firewall Fallacy)**
https://arxiv.org/abs/2605.24069v1

16. **SMCP: Secure Model Context Protocol**
https://arxiv.org/abs/2602.01129v1

17. **SoK: Security and Safety in the MCP Ecosystem**
https://arxiv.org/abs/2512.08290v2

18. **Hardware Keystores for AI Agent Signing Workflows**
https://arxiv.org/abs/2608.06130v2

19. **Traefik Hub MCP middleware docs**
https://doc.traefik.io/traefik-hub/mcp-gateway/mcp

20. **Traefik What is an MCP Gateway (API vs AI vs MCP gateway comparison)**
https://traefik.io/glossary/mcp-gateway-explained

21. **Lasso vs TrueFoundry comparison**
https://www.truefoundry.com/blog/lasso-security-vs-truefoundry

22. **HN Show HN: Director**
local-first open source MCP Gateway (2025-07-11)
https://news.ycombinator.com/item?id=44533022