CONF42 CLOUD NATIVE 2026

Cloud MCP Servers Under Fire

Security Risks, Vulnerability Patterns, and Practical Defenses

Aleksandr Pinaev

Swordfish Security

A practical look at architecture risk in MCP-based cloud systems

SPEAKER INTRO

Why I am giving this talk

- I work at the intersection of AppSec, cloud security, DevSecOps, and AI security.
- I analyze real-world risks in AI-connected architectures, including emerging MCP-style integration layers.
- I focus on architecture-level risk, where the most dangerous MCP weaknesses often appear.
- My goal is to translate complex security problems into practical defensive guidance.

Architecture risk | Practical threat modeling | Actionable defenses

20,000+

Active public MCP servers

97M+

Monthly SDK downloads

Architecture-level thinking is where MCP security becomes concrete.

MCP OVERVIEW

What an MCP server actually is

The MCP server is the structured interface between an AI client and external tools, resources, prompts, and data sources.

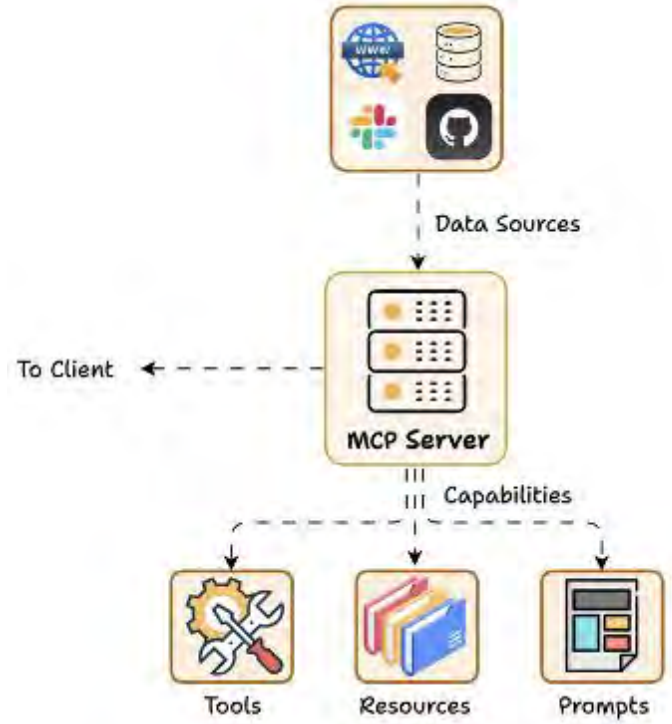
- It advertises capabilities the model can use, rather than embedding tool logic inside the model itself.
- Those capabilities typically include tools, resources, and prompts exposed through a standard protocol.
- That separation improves interoperability — but it also creates a security boundary that must be designed deliberately.

How MCP works in practice

Capabilities advertised to the client

Resources, prompts, tools, and data access

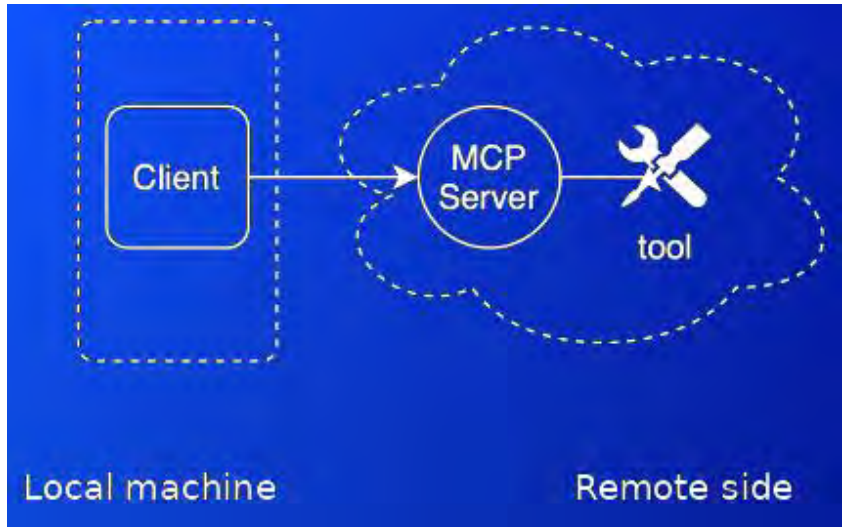
A separate server boundary for execution



DEPLOYMENT MECHANICS

From local helpers to remote control points

The same MCP abstraction that simplifies integration can also concentrate trust and execution power.



Local client

Remote server

Execution boundary

Capability container

The server declares what it can do — scan code, validate policy, fetch data, or trigger actions.

Unified protocol

A standard interface lets clients discover and invoke capabilities consistently.

Bridge between product and LLM

The model does not learn tool behavior itself — it calls prebuilt functions through MCP.

Client-controlled isolation

The server runs as a separate process or service, and the client decides what to connect.

TALK PREMISE

Why cloud MCP matters now

MCP is becoming a control layer between models, tools, and cloud services — and that makes it security-sensitive infrastructure.

New AI-driven attack surface

They create new paths for tool use, data access, and unintended execution.

Not just connectors

They mediate actions, data access, and trust decisions across systems.

Higher blast radius in cloud

Exposure, speed, and privilege concentration make mistakes scale faster.

Control layer
Models, tools, cloud services

Action mediation
Tool invocation and data access

Trust-sensitive execution
Where intent and privilege can diverge

ARCHITECTURE LENS

Where the risk really comes from

Traditional cloud assumptions do not fully apply when AI agents can dynamically invoke tools and trigger multi-step actions across systems.

Blurred trust boundaries

Intent can change in transit

Privileges can expand across hops

Abuse paths become multi-step

The whole chain has to be constrained — not just one endpoint.

ARCHITECTURE-LEVEL WEAKNESSES

The most dangerous weaknesses are architectural

These patterns matter even when the implementation looks secure enough at the code level.

Overprivileged access

Broad tool scopes and weak privilege design.

Misplaced trust boundaries

Too much trust between model, client, server, and tool.

Weak isolation

Tenant, environment, or context boundaries that leak.

Insecure exposure

Overexposed tools and brittle connector logic.

Weak validation

Inputs, context, or actions are accepted too easily.

Systemic risk > isolated bug

Privilege design | Isolation design | Validation design

VULNERABILITY PATTERNS

A practical classification for security teams

A usable taxonomy helps teams assess exposure, communicate impact, and prioritize defenses.

Identity and access

Weak auth, token misuse, confused identity, excessive scopes

Trust boundary failures

Unsafe delegation, hidden privilege transitions, implicit trust

Tool and connector exposure

Insecure tool registration, action sprawl, risky connectors

Isolation failures

Cross-tenant leakage, cross-environment drift, shared context abuse

Validation and execution flaws

Prompt/context injection, unsafe parameter handling, dangerous writes

Visibility and control gaps

Weak auditability, poor approval gates, missing behavioral monitoring

Why this helps

- Assess exposure
- Explain impact
- Design targeted mitigations

ABUSE SCENARIOS

How real attack paths emerge

Cloud MCP risk often comes from chaining individually small decisions into one high-impact outcome.

1

Injected context → model chooses tool → MCP server executes risky read → sensitive data exposed

2

Weak server authentication → unauthorized caller gains tool access → writes or retrieves data

3

Compromised connector or plugin → trusted execution channel abused → cross-system action taken

Common pattern	Low-friction invocation	High-trust execution	Hidden blast radius

CLOUD-SPECIFIC EXPOSURE

Why cloud MCP deployments amplify risk

Public reachability, token sprawl, and operational blind spots make cloud-hosted MCP materially different from local experimentation.

Public exposure

Publicly reachable endpoints and control surfaces.

Ephemeral infrastructure

Harder attribution, traceability, and forensics.

Credential sprawl

Tokens spread across agents, connectors, and environments.

Trust drift

Dev, staging, and prod boundaries blur over time.

Operational challenge

SECURE DESIGN

Design MCP as a privileged execution boundary

Security has to be designed into MCP architecture before scale arrives.

- Separate read paths from write paths and treat write actions as high risk.
- Constrain tool choices, scopes, and allowed parameter ranges.
- Validate inputs, context, and execution intent before action.
- Use explicit confirmation gates for dangerous or irreversible operations.
- Preserve isolation between tenants, agents, services, and environments.

Defensive mindset

Least privilege | Explicit trust boundaries | Safe-by-default execution

Design principle: treat MCP like a privileged control plane, not a harmless connector.

OPERATIONAL DEFENSES

What teams should implement in practice

The objective is not perfect prevention — it is reducing abuse paths, limiting blast radius, and making risky behavior visible.

Build-time and control-plane

- Strong authentication and short-lived credentials
- Scoped authorization for every tool and connector
- Allowlists for actions and resource classes

Runtime and operating model

- Audit trails for MCP calls and downstream effects
- Behavioral monitoring aligned to AI-enabled abuse scenarios
- Architecture reviews focused on trust boundaries and isolation

Security outcome → **Reduced abuse paths** → **Lower blast radius** → **Better incident visibility**

TAKEAWAYS

Reduce risk before MCP becomes the next blind spot

- Cloud MCP servers can unlock powerful AI integrations — but they must be treated as security-sensitive infrastructure from day one.
- The key risks are architectural: trust, privilege, isolation, exposure, and validation.
- A practical vulnerability classification helps security teams focus on what matters.
- Safer deployments require proactive design, not just post-factum monitoring.

MCP is powerful

MCP is security-sensitive

Build defenses early

Thank you

Aleksandr Pinaev • Swordfish Security