MCP Dev Summit Seoul

Beyond APIs: Making MCP Services Faster and Secure With WebAssembly

Brandon Kang
Principal Technical Solutions Architect
Akamai Technologies, Cloud Technology Group
August 2026

MCP is more than an API

MCP Dev Summit Seoul

Definitions are good — but the security boundary shifts when **the agent picks what runs**.

▲ TRADITIONAL API · PREDICTABLE

A request follows predefined logic.

Application

API endpoint

Backend

The developer chose this path *before* the request arrived.

▲ MCP · AGENT-DRIVEN

A tool may be chosen dynamically.

AI Agent

MCP server

MCP Tool

└ Files └ Network └ DB └ APIs └ Credentials

The agent picks. The host runs.

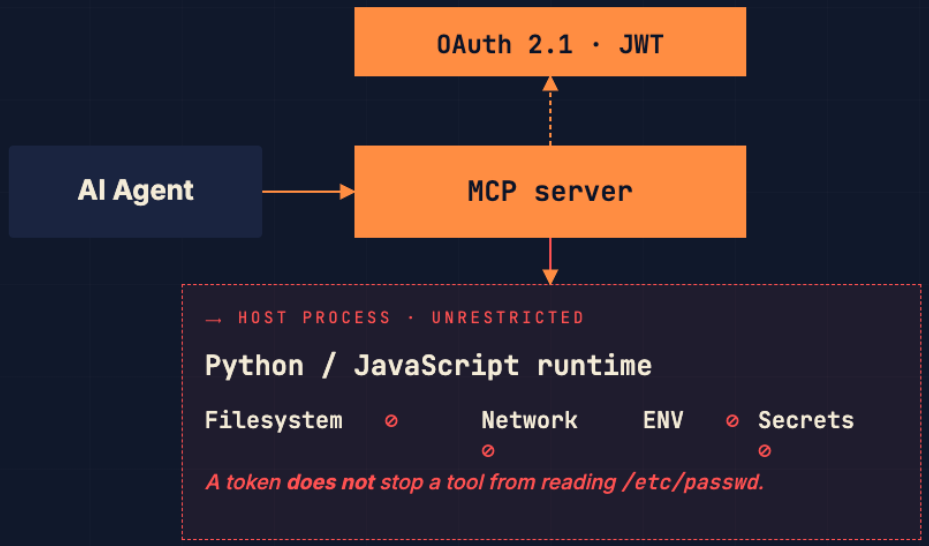
> **"The security boundary is no longer only the API endpoint.**
> **It is the execution environment behind the MCP tool."**

We secured the protocol

MCP Dev Summit Seoul

What about execution?

× **Authentication with OAuth ≠ *Execution isolation.***



WHAT MCP'S HTTP SPEC COVERS

- OAuth 2.1 authorization
- Access token flow
- Audience validation
- MCP server as resource server

WHAT IT DOES NOT COVER

- What the tool reads from disk
- Which network it calls
- Which env vars it sees
- Which credentials it holds

"Authorized" ≠ "safe to run."

Spec: MCP HTTP Authorization — OAuth 2.1, token-based, audience-bound. The protocol does not govern runtime behavior.

An MCP tool is code

MCP Dev Summit Seoul

A request does **not** arrive alone — it arrives with **side effects**.

SCENARIO · POISONED INPUT

Agent

"Analyze this file"

MCP Tool

```
└ read( "/etc/passwd" ) ?
└ getenv( "AWS_SECRET_KEY" ) ?
└ connect( "unknown-server.com" ) ?
└ exec( "/bin/bash" ) ?
```

Source: MCP-SandboxScan (2026) demonstrates Wasm/WASI sandboxing for tool-run analysis & vuln research.

FOUR ATTACK SURFACES

01 · FILESYSTEM
Read host files, paths outside the intended input.

02 · NETWORK
Exfiltrate data, call arbitrary endpoints.

03 · ENVIRONMENT
Pull tokens, secrets, and config from env vars.

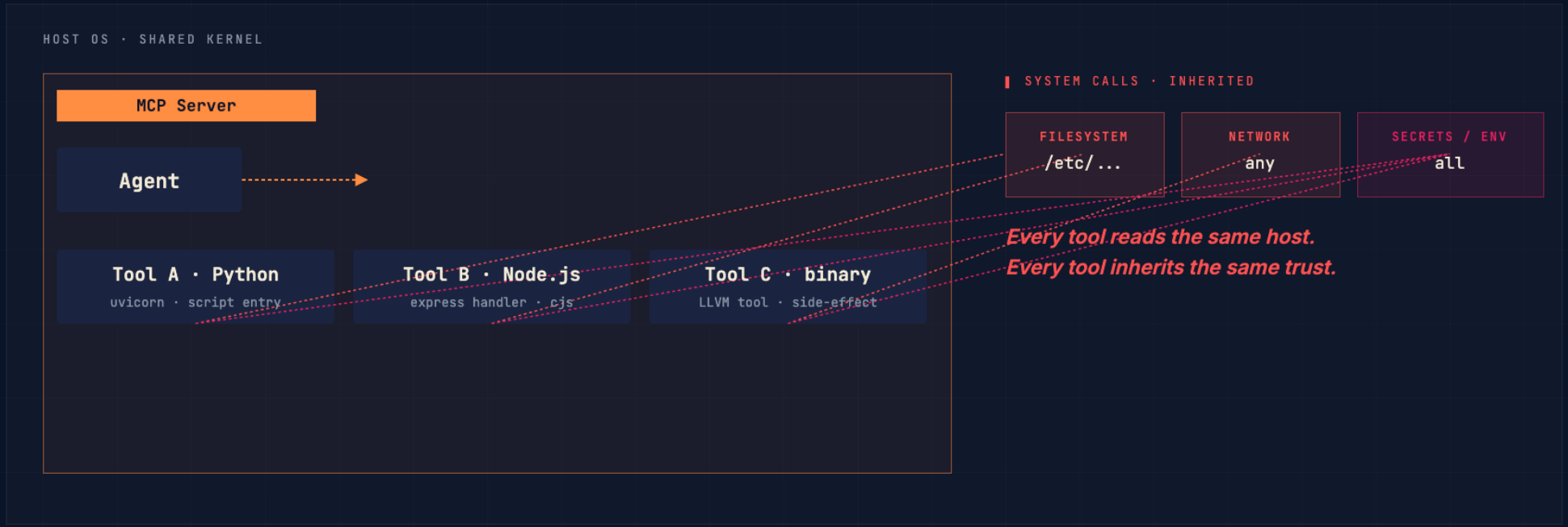
04 · SECRETS
Embedded keys, AWS_SECRET, DB_PASSWORD, outbound leaks.

The agent *doesn't* need to be malicious.
The tool — *or its input* — can be.

The traditional execution model

MCP Dev Summit Seoul

↳ One process. Many tools. Shared host context.



Tools inherit the security context of the host process.

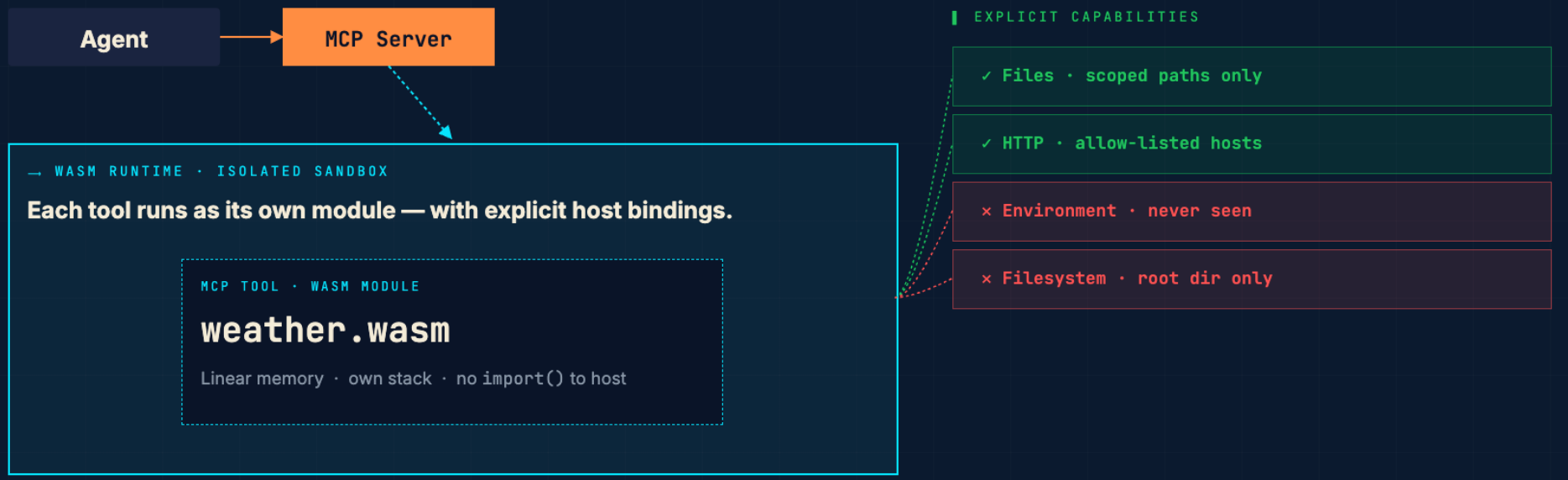
Isolation is possible — but it becomes an infrastructure responsibility.

What if the tool had..

MCP Dev Summit Seoul

no access by default?

WebAssembly ships modules in a sandbox — and passes host resources through explicit gates.



***WebAssembly's security model* = module-by-module isolation *+ host-controlled* capability gating.**

Why WASM fits MCP

MCP Dev Summit Seoul

Four properties that line up with how agents actually use tools.

01 · SANDBOXED

Isolated memory.
Isolated execution.

Modules cannot reach host state indirectly.
Every syscall crosses a host boundary.

02 · FAST STARTUP

Lightweight, head-of-line ready.

Cold-start measured in ms, not seconds — order-of-magnitude lighter than VMs and containers.

03 · PORTABLE

Same module, anywhere.

Compile once. Run on Wasmtime, Wasmer, Spin, edge runtime — portability is the spec.

04 · CAPABILITY-ORIENTED

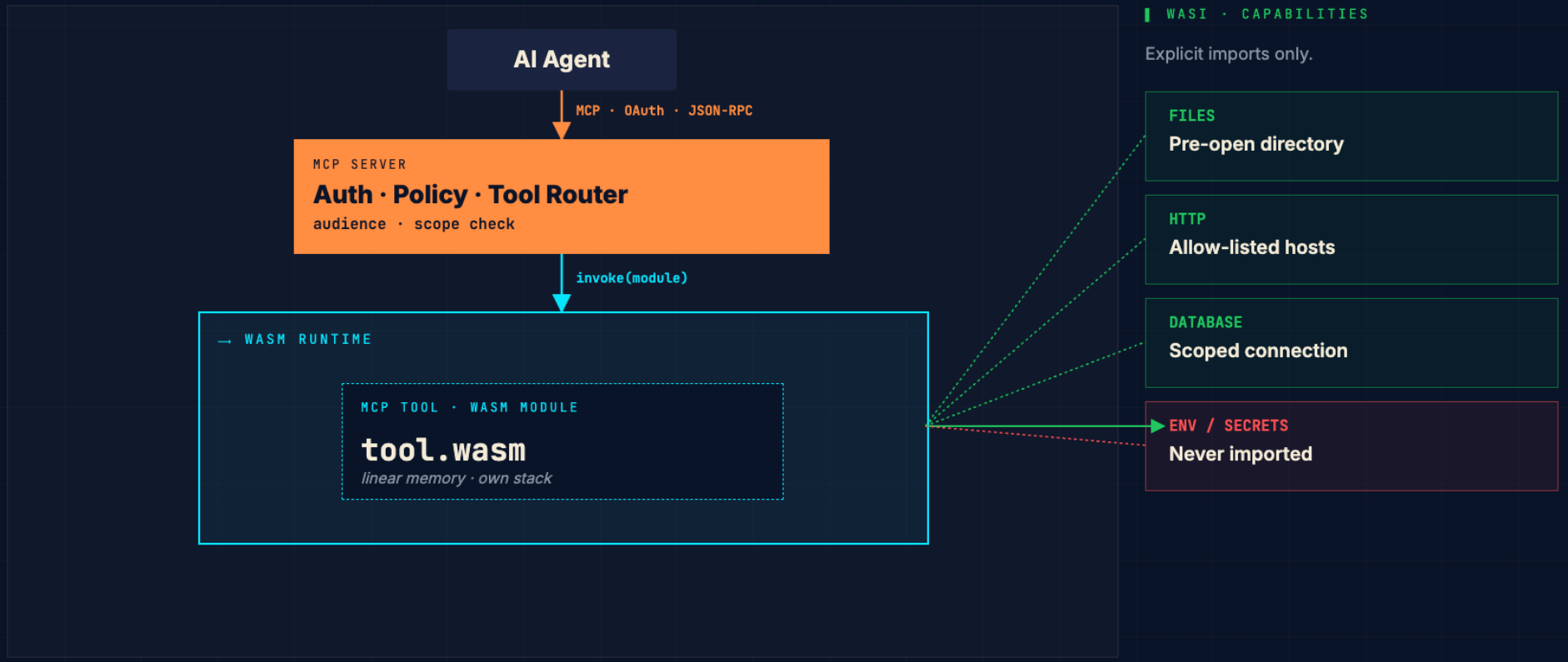
Deny by default.

WASI explicitly imports required capabilities — nothing ambient, nothing inherited.

The same primitive answer to four different agent pain points.

MCP + WASM Execution Architecture

MCP Dev Summit Seoul



MCP controls WHAT is called.
Wasm controls WHAT that tool can do.

From ambient authority to explicit capability

MCP Dev Summit Seoul

Each WASI import becomes an auditable, signed, scoped handle.

BEFORE · AMBIENT AUTHORITY

The tool inherits everything.

- Host filesystem · all paths
- Network · any destination
- Environment · full process env
- Credentials · embedded or env-loaded

A bug in the tool becomes a host-wide incident.

Blast radius = the entire process.

AFTER · WASI · EXPLICIT

The tool requests each binding.

- Filesystem ✓ /data/input only
- Network ✓ api.example.com
- Environment ×
- Secrets × never imported

Violation attempts fail at the runtime boundary.

Blast radius = the module.

Deny by default. **Grant only what the tool needs.**

The tool shouldn’t own the secret

MCP Dev Summit Seoul

Capabilities ≠ credentials. In a Wasm model, secrets stay with the host.

▲ ANTI-PATTERN · SECRET-IN-FUNCTION

Credentials live inside the tool.

MCP TOOL · INSIDE

AWS_SECRET_KEY
—— AKIA••••••••

DATABASE_PASSWORD
—— ••••••••

API_TOKEN
—— ••••••••

Compromised tool = leaked credential.

✓ PATTERN · CAPABILITY-BASED

Tool asks; host attaches.

MCP TOOL · WASM
tool.wasm
no secret

host_call("customer-api")

HOST · CREDENTIAL BROKER
Signs & attaches
Vault · metadata broker · policy-aware shortcut

customer-api.com

The tool receives capability, never credentials.

The security layers, three questions

MCP Dev Summit Seoul

Authentication checks identity. Authorization checks reach. Execution checks behavior.

LAYER 1

Authentication

Who are you?

EXAMPLE

OAuth

bearer token · JWT validation

IDENTITY PROOF

LAYER 2

Authorization

What can you call?

EXAMPLE

JWT scopes · MCP policy

tool allowed by user role

REACH PROOF

LAYER 3

Execution

What can the tool access?

EXAMPLE

Wasm · WASI capabilities

filesystem, net, env, secrets

BEHAVIOR PROOF

`OAuth → MCP → Policy → Wasm → Capability`

OAuth protects access to the MCP service.

Wasm protects the execution behind it.

Let’s run an MCP tool inside WASM

MCP Dev Summit Seoul

A live stack: Client ↔ MCP Server ↔ Wasm runtime ↔ tool.wasm.

THREE PROBES

Same module. Three different outcomes.

PROBE 01

```
get_weather("Seoul")
✓   → 22°C · light rain
```

PROBE 02

```
read("/etc/passwd")
×   → EACCES: capability denied
```

PROBE 03

```
getenv("AWS_SECRET_KEY")
×   → symbol not exported
```

Probe ordering — allow → attack surface one → attack surface two. Same module, three outcomes.

$ MCP<weather> -> WASM

MCP Dev Summit Seoul

Three terminal outputs — same module, identical runtime, three outcomes.

STEP 01 · BINDING THE TOOL

```
$ mcp invoke --module=weather.wasm --cap=fs:/data:ro --cap=http:api.weather.example/*
[module] weather.wasm   ready
[cap] fs:/data   ✓ mounted
[cap] http:api.weather.example/*  ✓ allow-listed
[cap] env  × denied · module never imported
```

STEP 02 · ALLOWED CALL

```
$ weather --city "Seoul"
[wasm] http://api.weather.example/v1/forecast?city=Seoul  → 22°C, light rain
✓ call met capability policy   runtime elapsed: ms
```

STEP 03 · FILESYSTEM ATTEMPT

```
$ weather --read "/etc/passwd"
EACCES: capability "/etc/passwd" not in pre-opens
× invocation rejected at runtime boundary   module never saw the call
```

STEP 04 · SECRET EXFIL ATTEMPT

```
$ weather --getenv AWS_SECRET_KEY
UNDEFINED: symbol env::getenv not imported by module
× trap fires at runtime import resolution   credential was never visible
```

Two runtime boundary violations — one successful capability call. The tool didn't patch itself. The runtime said no.

Where MCP + WASM becomes interesting

MCP Dev Summit Seoul

Four deployment patterns — from a single tenant to the edge.

PATTERN 01

Multi-Tenant MCP

Many customers, same platform.
Per-tenant modules — per-tenant caps.

PATTERN 02

Enterprise Agents

Internal APIs · DBs · least privilege by tool.

PATTERN 03

Untrusted Tools

Community modules, third-party scripts.
Run them safely by design.

PATTERN 04 · AKAMAI

Edge MCP

Lightweight modules at POPs.
Cheap to scale, easy to isolate.

AT AKAMAI

MCP + Wasm is the natural pairing for the edge:

Fast cold-start · small footprint · capability-scoped · an isolation story that scales with your agent fleet.

Three things to remember

MCP Dev Summit Seoul

Secure what happens after.

01 MCP SECURITY · INCOMPLETE

MCP security doesn't stop at authentication. Tool execution is a security boundary.

02 WASM · ISOLATED EXECUTION

WebAssembly gives us an isolated execution layer. Explicit capabilities, not ambient access.

03 AGENTS · LIGHTWEIGHT RUN

Agents need lightweight execution: fast startup, isolation, portability. Wasm is a natural fit.

CLOSING

Don't just secure the MCP endpoint.

Secure what happens after the tool call.


MCP
Dev Summit
Seoul