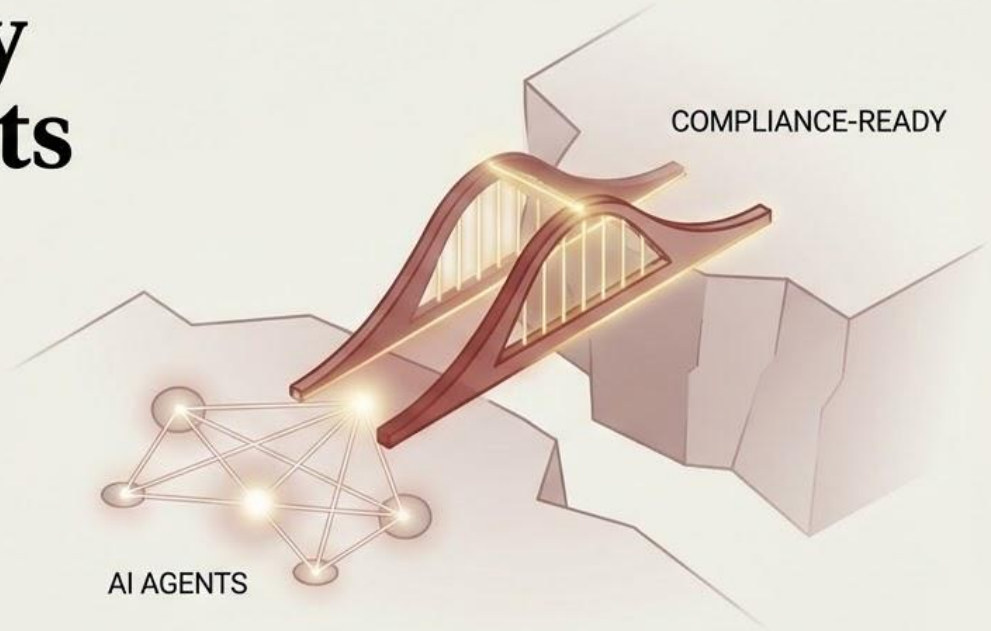
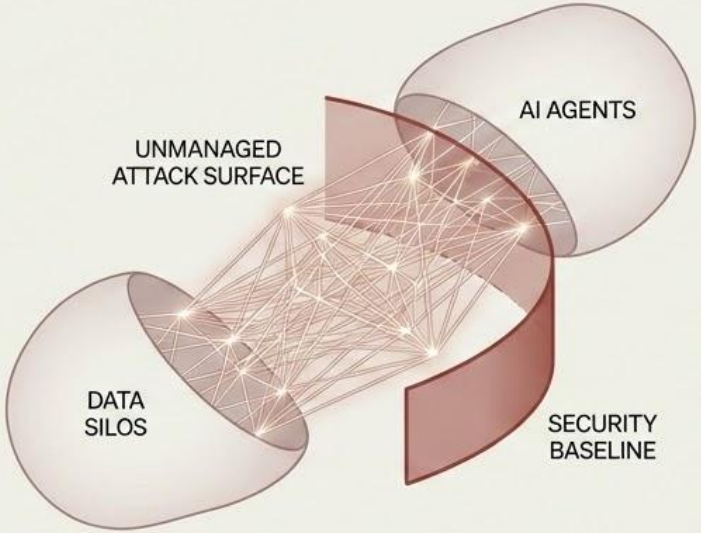
Beyond the Hype: Building a Compliance-Ready Bridge for AI Agents with SAFE-MCP



The New Perimeter: AI Agents and the Connectivity Gap



Speaker Intro:

Frederick Kautz, bringing 20+ years of cloud-native security, Zero Trust (zt.dev), and SPIFFE leadership to the AI era.

The Problem:

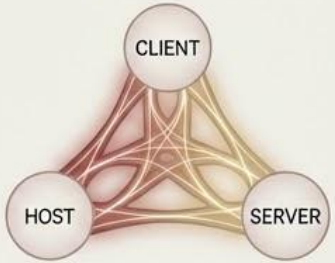
The Model Context Protocol (MCP) has solved the “data silo” problem, but created a massive, unmanaged attack surface.

The Mission:

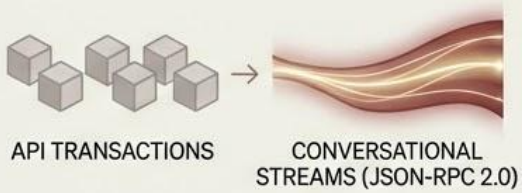
Why we need a community-driven security baseline to move agents from “useful” to “safe for enterprise”.

Architecture of the Model Context Protocol (MCP)

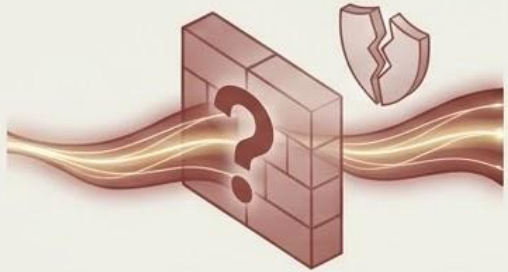
A Unified Standard:
Host, Client, and Server as the “connective tissue” of the agentic era.



The Shift:
Moving from discrete API transactions to persistent, context-aware conversational streams (JSON-RPC 2.0).



The Security Gap:
Traditional firewalls and WAFs don’t understand the “intent” hidden in these streams.

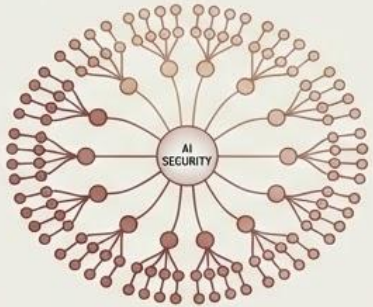


SAFE-MCP: The Framework for Evaluation

Defining the Baseline:
An open-source specification for documenting and mitigating threats in the MCP ecosystem.

Community Governance:
Under the Linux Foundation, ensuring a vendor-neutral standard for identity and safety.

The Taxonomy:
14 tactical categories and 80+ techniques, providing the structured ‘Why’ behind the ‘How’ of AI security.



Systematic Risk Identification

Adversarial Methodology:

Adapting the MITRE ATT&CK methodology for the agent lifecycle.

The TTP Catalog:

From Tool Poisoning (SAFE-T1001) to Instruction Steganography (SAFE-T1402).

Threat Modeling:

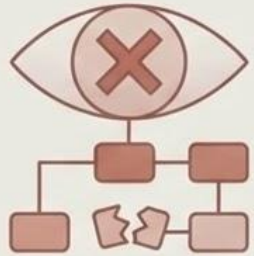
Using SAFE-MCP to move beyond generic “AI risk” to specific, exploitable vectors that developers can actually fix.

Manipulating Agent Intent

Prompt Injection (SAFE-T1102):
Malicious inputs that bypass system guardrails to execute unauthorized commands.

Indirect Injections:
When agents consume poisoned data from third-party sources (Email, Slack, Jira) that redirect their behavior.

The Visibility Problem:
If you can't see the context flow, you can't detect the shift in intent.



The Agentic Supply Chain

Tool Poisoning & Rug Pulls:
Benign tools that are updated with malicious logic or descriptions after they've been approved.

Shadow MCPs:
The rise of unmanaged, decentralized tools that bypass organizational governance.

Credential Harvesting:
Tricking agents into exfiltrating environment variables or API keys during tool calls.

Implementation & Mitigations (The SAFE-M Series)

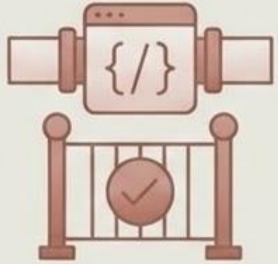
Actionable Defense:
Every threat in the framework is mapped to a specific mitigation (e.g., SAFE-M-4 for Unicode Sanitization).

Sandboxing & Isolation:
Using Docker-based isolation to restrict the “blast radius” of a compromised MCP server.

Input/Output Filtering:
Implementing real-time scanning of JSON-RPC traffic to neutralize injections before execution.

Integrating into the DevSecOps Pipeline

Security Gates as Code:
Moving from manual reviews to automated checks in CI/CD pipelines.

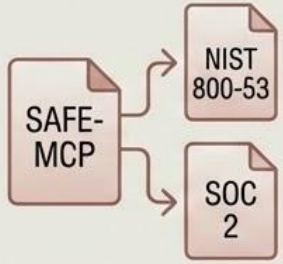


Automated Scanning:
Integrating tools to verify tool metadata and schemas before deployment.

Human-in-the-Loop (HITL):
Implementing gated approvals for high-risk tools (writing to DB, sending money) directly in the workflow.

Bridging to Compliance (NIST 800-53, SOC 2)

Mapping to Frameworks:
How SAFE-MCP techniques translate directly to NIST 800-53 and SOC 2 Trust Services Criteria.



Audit-Ready Evidence:
Automating the collection of metrics (Mean Time to Detection, Access Logs) to satisfy regulatory requirements.

NIST SP 800-204D Alignment:
Hardening the Software Supply Chain by treating AI agents as first-class identities.

Call to Action: Join the Community

Get Involved:
We meet bi-weekly under the Linux Foundation to sharpen these standards.

Contribute:
Submit a Pull Request, join the weekly hackathons, or share a new mitigation.

Where to find us:
GitHub: github.com/safe-agentic-framework/safe-mcp

> *"Security shouldn't be a bottleneck; it's the bridge that lets us move faster into the future of AI."*

ATK-TA0001	Initial Access	SAFE-T1009	Authorization Server Mix-up	Client follows redirect to look-alike AS domain (e.g., accounts-google.com vs accounts.google.com), causing authorization codes or tokens to be leaked to attacker-controlled server
ATK-TA0002	**Execution**	SAFE-T1101	Command Injection	Exploitation of unsanitized input in MCP server implementations leading to remote code execution
ATK-TA0002	Execution	SAFE-T1102	Prompt Injection (Multiple Vectors)	Malicious instructions injected through various vectors to manipulate AI behavior via MCP
ATK-TA0002	Execution	SAFE-T1103	Fake Tool Invocation (Function Spoofing)	Adversary forges JSON that mimics an MCP function-call message, tricking the host into running a tool that was never offered
ATK-TA0002	Execution	SAFE-T1104	Over-Privileged Tool Abuse	Legit tool (e.g. "Shell") runs with broader OS rights than necessary; LLM can be induced to perform arbitrary commands

SAFE-T1201: MCP Rug Pull Attack

Overview

Tactic: Persistence (ATK-TA0003)
Technique ID: SAFE-T1201
Severity: High
First Observed: April 2025 (Discovered by Invariant Labs)
Last Updated: 2025-01-15

Description

MCP Rug Pull Attack refers to a persistence technique where adversaries deploy legitimate-appearing MCP tools that later undergo time-delayed malicious modifications after gaining initial user approval and trust. This technique exploits the dynamic nature of MCP tool definitions and the tendency for users to approve tools based on initial functionality without ongoing verification.

The attack involves a multi-stage approach: first establishing trust through legitimate tool behavior, then introducing malicious functionality through server-side updates, configuration changes, or time-based triggers. This technique is particularly effective because it bypasses initial security reviews and user approval processes by leveraging the established trust relationship.

SAFE-K8S Public Security Control Catalog

This repository publishes the public SAFE-K8S security control catalog for Kubernetes and AI systems. It includes the public control set, knowledge area structure, and framework crosswalks under `SAFE-K8S-*` identifiers.

Purpose

- Publish a public SAFE-K8S control catalog for external use
- Preserve traceability between controls and mapped frameworks
- Support review, reuse, and downstream publication without internal-only fields

Contents

- YAML source files for domains, knowledge areas, controls, and crosswalks
- Generated markdown pages for controls and reverse mappings by framework

Basic Info

- Domains: 10
- Knowledge areas: 55
- Controls: 593
- Crosswalk rows: 4723

https://github.com/safe-agentic-framework/safe-k8s

Knowledge Areas

- 1.1 - Kubernetes API Server Security
- 1.2 - etcd and Cluster State Protection
- 1.3 - Controller-Manager, Scheduler, and Cloud Controller Security
- 1.4 - CIS Benchmarks and Patch Management
- 2.1 - Kubelet and Node Configuration Hardening
- 2.2 - Container Runtime Security
- 2.3 - Host OS and Kernel Hardening
- 2.4 - Runtime Threat Detection
- 2.5 - kube-proxy and Node Networking Security
- 3.1 - Pod Security Standards and Admission
- 3.2 - Security Contexts and Capabilities
- 3.3 - Mandatory Access Controls
- 3.4 - Secure Defaults and Resource Constraints
- 4.1 - Role-Based Access Control (RBAC)
- 4.2 - Service Accounts and Workload Identity
- 4.3 - Secrets Management
- 4.4 - Certificate Management
- 4.5 - Identity Abuse Detection and Mitigation
- 5.1 - Network Policies
- 5.2 - CNI Plugins and Pod Networking Security
- 5.3 - Ingress, Egress, and DNS Hardening
- 5.4 - Zero Trust Architecture and Service Mesh
- 5.5 - API Server and Service Exposure Protection
- 6.1 - Container Image and Registry Security
- 6.2 - Image Signing and Admission Enforcement
- 6.3 - Attestation, Provenance, and Cryptographic Assurance
- 6.4 - SBOMs and Vulnerability Intelligence
- 6.5 - Admission Control
- 6.6 - CI/CD and GitOps Pipeline Security

Control ID	Title	Maturity	Class
SAFE-K8S-0801-001	GPU device plugin security configuration and hardening	Practitioner	ai-specific
SAFE-K8S-0801-002	MIG partitioning for hardware-enforced GPU isolation	Advanced	ai-specific
SAFE-K8S-0801-004	vGPU virtualization security controls	Practitioner	ai-specific
SAFE-K8S-0801-006	GPU memory clearing between workload transitions	Advanced	ai-specific
SAFE-K8S-0801-009	GPU topology metadata protection and node label visibility restriction	Practitioner	ai-specific
SAFE-K8S-0801-010	Admission validation of authorized GPU resource requests	Practitioner	ai-specific
SAFE-K8S-0801-011	MPS and time-slicing residual-risk acceptance and compensating control approval	Advanced	ai-specific
SAFE-K8S-0801-012	MPS and time-slicing workload eligibility and same-trust co-location enforcement	Advanced	ai-specific
SAFE-K8S-0801-013	MPS and time-slicing memory remnant prevention verification	Advanced	ai-specific
SAFE-K8S-0801-014	MPS and time-slicing side-channel risk assessment	Advanced	ai-specific


Questions?