HAIAMM Handbook v2.2

HAIAMM

HAIAMM Handbook

Human-Assisted Intelligence Assurance Maturity Model

Version 2.2

Practical Security for AI Systems, LLM Applications, and Autonomous Agents

---

January 2026

HAIAMM Logo

HAIAMM Handbook v2.2

Human-Assisted Intelligence Assurance Maturity Model

Practical Security for AI Systems, LLM Applications, and Autonomous Agents

> **Note on Code Examples**
>
> Code snippets throughout this handbook are **illustrative examples only**. They are intended to demonstrate security concepts and provide context for implementation patterns—not to serve as production-ready code. Always adapt examples to your specific environment, follow your organization's coding standards, and conduct thorough security reviews before deployment.

What is HAIAMM?

HAIAMM is a comprehensive framework for securing Human-Assisted Intelligence (HAI) systems. It provides actionable guidance for organizations building, deploying, and operating AI applications—from simple LLM integrations to fully autonomous agent systems.

HAIAMM is NOT another checkbox compliance framework. It's a practical roadmap with: - Measurable outcomes (not just activities) - Risk-based prioritization (address real threats first) - Implementation guidance (time, cost, effort estimates) - Tool recommendations (what to actually use)

How to Use This Handbook

Pick Your Starting Point

Your Situation	Start Here
"I have 10 minutes"	Quick Start Guide
"I'm building with LLMs"	Top 10 LLM Risks
"I'm deploying AI agents"	Top 10 Agentic Risks
"I need to assess our security"	Assessment Checklist
"I want a structured program"	First 30 Days
"Show me everything"	Keep reading below

Quick Links

- Quick Start Guide - Get started in 5-10 minutes
- First 30 Days - Day-by-day implementation roadmap
- OWASP Top 10 for Agentic Apps - Priority security risks for AI agents

- OWASP Top 10 for LLMs - Security risks for LLM applications
- Risk-Practice Matrix - Which practices address which risks
- Maturity Roadmap - Level 1 → 2 → 3 progression
- Assessment Checklist - Quick self-assessment (30 min)
- Tools & Resources - Recommended tools by practice

The HAIAMM Framework

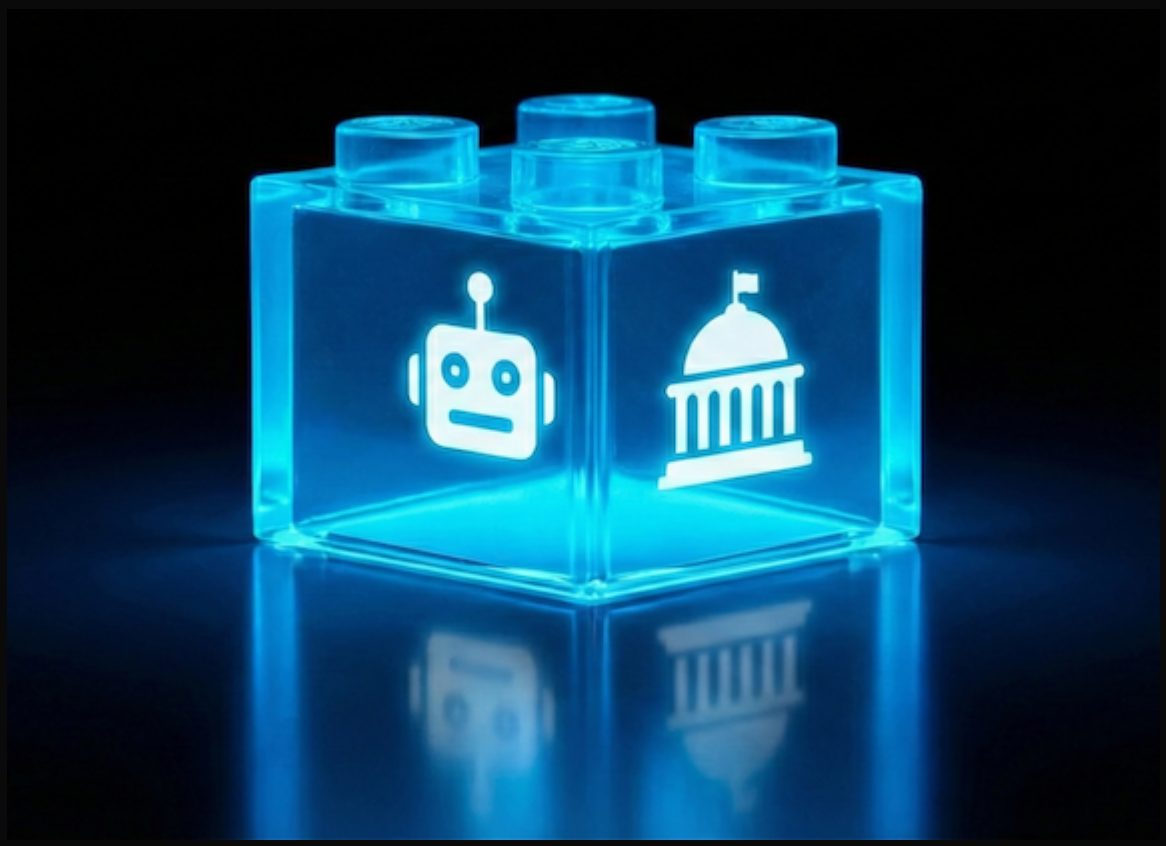
Six Security Domains

HAIAMM organizes AI security into six domains, each addressing a critical aspect of HAI systems:

Domain	Focus	Key Concerns
Software	AI model code, applications, integrations	Model vulnerabilities, code security, CI/CD
Infrastructure	Compute, storage, networks	GPU security, cloud services, distributed training
Endpoints	APIs, UIs, integration points	Prompt injection, output validation, access control
Data	Training data, model I/O, pipelines	Data poisoning, privacy, bias, lineage
Processes	Governance, compliance, operations	Model governance, ethics, regulatory compliance
Vendors	Third-party AI services and tools	Supply chain, model provenance, vendor risk

Twelve Security Practices

Each domain contains four business functions with twelve practices:

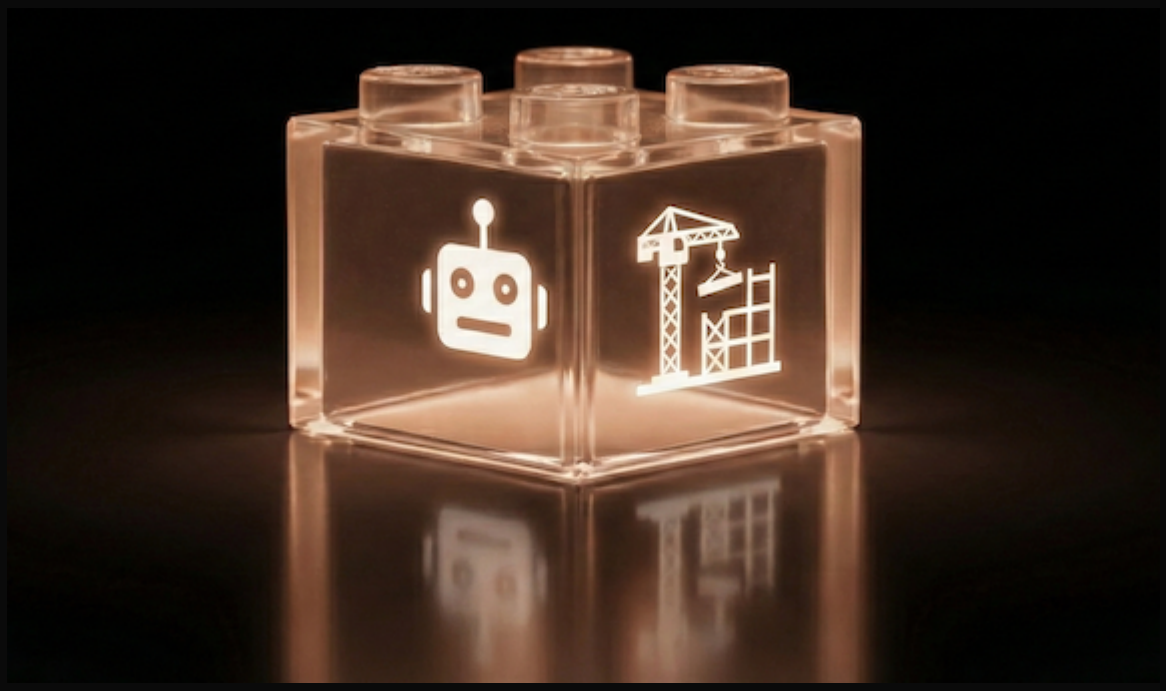


Governance

How you govern AI security agents

Practice	Code	Purpose
Strategy & Metrics	SM	Define AI security strategy and measure effectiveness
Policy & Compliance	PC	Establish policies and ensure regulatory compliance
Education & Guidance	EG	Train teams on AI security best practices

---



Building

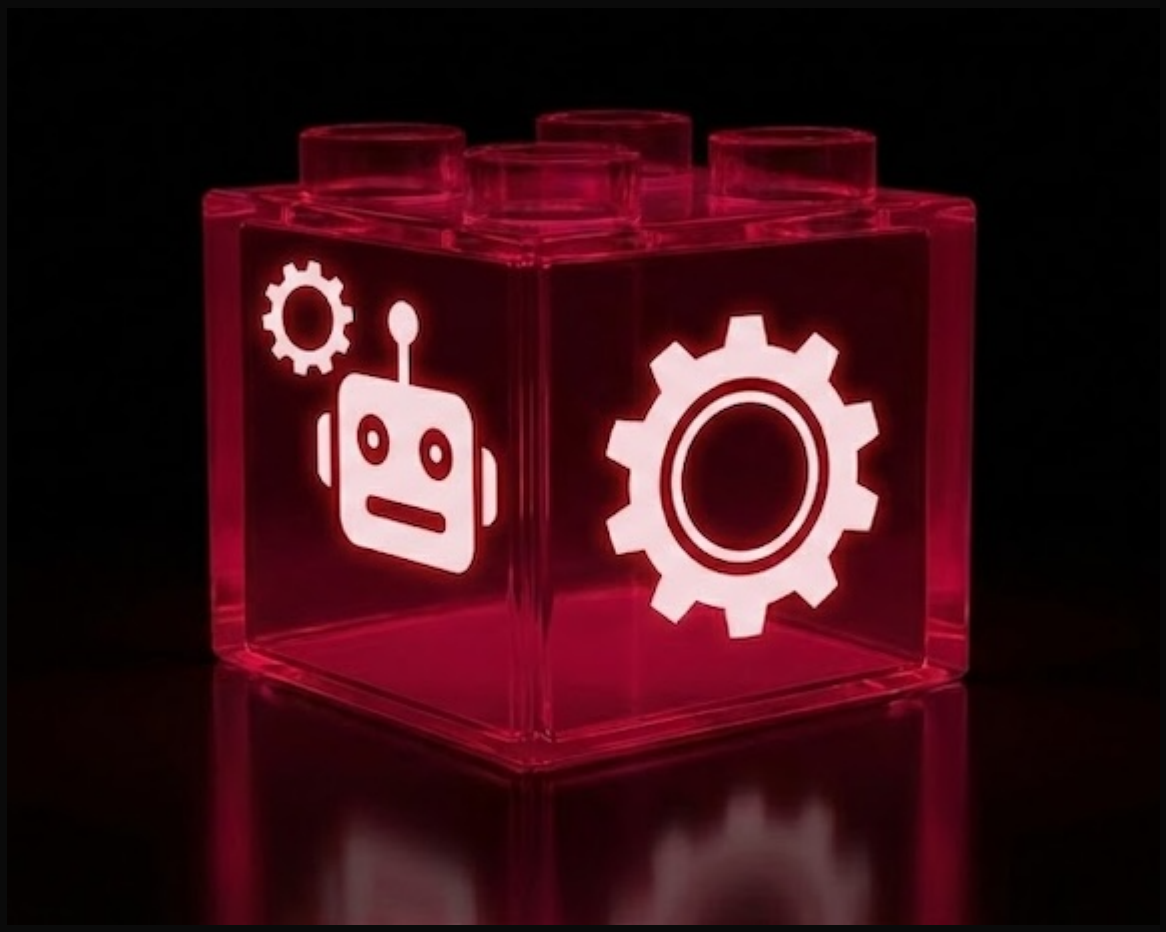
How you build AI agents securely

Practice	Code	Purpose
Threat Assessment	TA	Identify and prioritize AI-specific threats
Security Requirements	SR	Define security requirements for AI systems
Secure Architecture	SA	Design secure AI system architectures

Verification

How you verify AI agents work correctly

Practice	Code	Purpose
Design Review	DR	Review AI system designs for security flaws
Implementation Review	IR	Review AI implementations for vulnerabilities
Security Testing	ST	Test AI systems for security weaknesses



Operations

How you run AI agents safely

Practice	Code	Purpose
Issue Management	IM	Track and remediate security issues
Environment Hardening	EH	Secure AI runtime environments
Monitoring & Logging	ML	Monitor AI systems for security events

Three Maturity Levels

Each practice has three maturity levels:

Level	Name	Characteristics
1	Foundation	Basic understanding, ad-hoc activities, individual effort

Level	Name	Characteristics
2	Structured	Documented requirements, consistent processes, defined roles
3	Optimized	Quantitative management, continuous improvement, measured outcomes

---

Domain Deep Dives

Software Domain

Security of AI model code, application software, and integration layers.

One-Pagers: - SM - Strategy & Metrics - PC - Policy & Compliance - EG - Education & Guidance - TA - Threat Assessment - SR - Security Requirements - SA - Secure Architecture - DR - Design Review - IR - Implementation Review - ST - Security Testing - IM - Issue Management - EH - Environment Hardening - ML - Monitoring & Logging

Infrastructure Domain

Security of compute, storage, and network infrastructure supporting AI workloads.

One-Pagers: - SM - Strategy & Metrics - PC - Policy & Compliance - EG - Education & Guidance - TA - Threat Assessment - SR - Security Requirements - SA - Secure Architecture - DR - Design Review - IR - Implementation Review - ST - Security Testing - IM - Issue Management - EH - Environment Hardening - ML - Monitoring & Logging

Endpoints Domain

Security of APIs, user interfaces, and integration points for AI systems.

One-Pagers: - SM - Strategy & Metrics - PC - Policy & Compliance - EG - Education & Guidance - TA - Threat Assessment - SR - Security Requirements - SA - Secure Architecture - DR - Design Review - IR - Implementation Review - ST - Security Testing - IM - Issue Management - EH - Environment Hardening - ML - Monitoring & Logging

Data Domain

Security of training data, model inputs/outputs, and data pipelines.

One-Pagers: - SM - Strategy & Metrics - PC - Policy & Compliance - EG - Education & Guidance - TA - Threat Assessment - SR - Security Requirements - SA - Secure Architecture - DR - Design Review - IR - Implementation Review - ST - Security Testing - IM - Issue Management - EH - Environment Hardening - ML - Monitoring & Logging

Processes Domain

Governance, compliance, and operational processes for AI systems.

One-Pagers: - SM - Strategy & Metrics - PC - Policy & Compliance - EG - Education & Guidance - TA - Threat Assessment - SR - Security Requirements - SA - Secure Architecture - DR - Design Review - IR - Implementation Review - ST - Security Testing - IM - Issue Management - EH - Environment Hardening - ML - Monitoring & Logging

Vendors Domain

Security of third-party AI services, models, and vendor relationships.

One-Pagers: - SM - Strategy & Metrics - PC - Policy & Compliance - EG - Education & Guidance - TA - Threat Assessment - SR - Security Requirements - SA - Secure Architecture - DR - Design Review - IR - Implementation Review - ST - Security Testing - IM - Issue Management - EH - Environment Hardening - ML - Monitoring & Logging

---

Risk-Based Prioritization

HAIAMM maps directly to industry-recognized AI security risks:

OWASP Top 10 for Agentic Applications (2026)

Risk	Primary HAIAMM Practices
ASI01: Agent Goal Hijack	TA, SR, ST
ASI02: Tool Misuse & Exploitation	SA, PC, ML

Risk	Primary HAIAMM Practices
ASI03: Identity & Privilege Abuse	SR, EH, ML
ASI04: Agentic Supply Chain	TA, SA, IM
ASI05: Unexpected Code Execution	SR, ST, EH
ASI06: Memory & Context Poisoning	DR, ST, EH
ASI07: Insecure Inter-Agent Comm	SA, SR, ML
ASI08: Cascading Failures	SA, IM, ML
ASI09: Human-Agent Trust Exploitation	EG, PC, ML
ASI10: Rogue Agents	TA, ML, IM

Full Agentic Risk Details

OWASP Top 10 for LLM Applications (2025)

Risk	Primary HAIAMM Practices
LLM01: Prompt Injection	SR, ST, EH
LLM02: Sensitive Information Disclosure	PC, DR, ML
LLM03: Supply Chain	SA, IM, EH
LLM04: Data and Model Poisoning	TA, DR, ST
LLM05: Improper Output Handling	SR, IR, ST
LLM06: Excessive Agency	SA, PC, ML
LLM07: System Prompt Leakage	EH, ST, ML
LLM08: Vector and Embedding Weaknesses	SA, ST, EH
LLM09: Misinformation	DR, ST, ML
LLM10: Unbounded Consumption	SR, EH, ML

Full LLM Risk Details

Getting Help

Assessment Support

- Self-Assessment Checklist - 30-minute rapid assessment
- Risk-Practice Matrix - Find which practices to prioritize

Implementation Guidance

- First 30 Days - Day-by-day implementation roadmap
- Maturity Roadmap - Level progression guidance
- Tools & Resources - Recommended tools by practice

Community

- GitHub: HAIAMM Repository
- License: Creative Commons Attribution-ShareAlike 4.0

Document Information

Field	Value
Version	2.2
Last Updated	January 2026
Status	Active

Next Steps: - New to HAIAMM? Start with the Quick Start Guide - Building AI agents? Jump to Top 10 Agentic Risks - Ready to assess? Use the Assessment Checklist

HAIAMM Logo

HAIAMM Quick Start Guide

Get started securing your AI systems in 10 minutes

Back to Index

> **Note on Code Examples**
>
> Code snippets in this guide are **illustrative examples only**—intended to demonstrate concepts, not serve as production-ready code. Adapt all examples to your environment and security requirements.

What is HAIAMM?

HAIAMM (Human-Assisted Intelligence Assurance Maturity Model) is a practical framework for securing AI systems. It covers everything from simple LLM integrations to fully autonomous agent deployments across six security domains and twelve practices.

Core Principle: Security that actually works, measured by outcomes, not checkboxes.

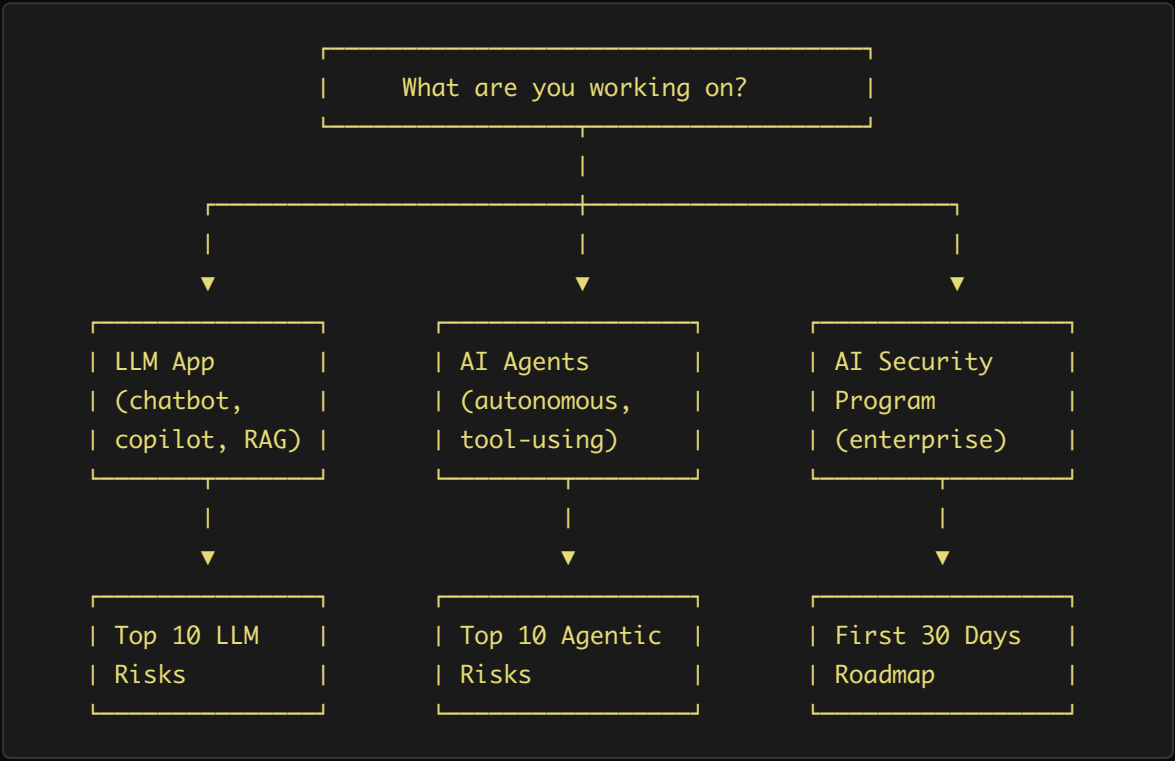
Who is This For?

- **Security teams** responsible for AI/ML system security
- **Engineering teams** building LLM applications or AI agents
- **CISOs and security leaders** establishing AI security programs
- **Compliance teams** addressing AI-specific regulatory requirements

---

Pick Your Path

What are you building?



Choose your path:

If you're…	Go to…
Building an LLM chatbot, copilot, or RAG system	Top 10 LLM Risks
Deploying autonomous AI agents with tool access	Top 10 Agentic Risks
Establishing an enterprise AI security program	First 30 Days
Assessing current AI security posture	Assessment Checklist
Looking for specific tools	Tools & Resources

---

5 Quick Wins You Can Do Today

These actions take less than 1 hour each and immediately improve your AI security:

1. Inventory Your AI Systems (30 min)

Action: Create a list of all AI/ML systems in your environment.

```
## AI System Inventory Template

| System Name | Type | Data Access | Tool Access | Owner | Risk Level |
|-------------|------|-------------|-------------|-------|------------|
| Support Bot | LLM  | Customer data | None | @alice | Medium |
| Code Agent  | Agent | Source code | Git, Shell | @bob | High |
| RAG Search  | LLM  | Internal docs | None | @carol | Medium |
```

Why: You can't secure what you don't know exists. Shadow AI is real.

---

2. Add Input Validation (30 min)

Action: Implement basic input validation on all LLM endpoints.

```python
# Simple input validation example
def validate_llm_input(user_input: str) -> bool:
    # Length limits
    if len(user_input) > 10000:
        return False

    # Basic injection patterns (expand based on your threat model)
    dangerous_patterns = [
        "ignore previous instructions",
        "ignore all instructions",
        "system prompt",
        "you are now",
    ]

    lower_input = user_input.lower()
    for pattern in dangerous_patterns:
        if pattern in lower_input:
            log_security_event("prompt_injection_attempt", user_input)
            return False

    return True
```

Why: Prompt injection is the #1 LLM vulnerability. Basic filtering stops naive attacks.

3. Enable Output Logging (20 min)

Action: Log all LLM inputs and outputs (with PII redaction).

```
import hashlib
import logging

security_logger = logging.getLogger("ai_security")

def log_llm_interaction(user_id: str, input_text: str, output_text: str):
    """Log LLM interactions for security monitoring."""
    security_logger.info({
        "event": "llm_interaction",
        "user_id_hash": hashlib.sha256(user_id.encode()).hexdigest()[:16],
        "input_length": len(input_text),
        "output_length": len(output_text),
        "timestamp": datetime.utcnow().isoformat(),
        # Store full content in secure, access-controlled storage
        "content_ref": store_encrypted(input_text, output_text)
    })
```

Why: You need visibility to detect attacks and investigate incidents.

---

4. Restrict Agent Permissions (30 min)

Action: Apply least privilege to any AI agent with tool access.

```
# Agent permission configuration example
agent_permissions:
  code_assistant:
    allowed_tools:
      - read_file      # Read-only file access
      - search_code    # Search codebase
    denied_tools:
      - write_file     # No write access
      - execute_shell  # No shell access
      - send_email     # No external communication
    rate_limits:
      api_calls_per_minute: 10
      tokens_per_request: 4000
    data_access:
```

```
      - source_code: read
      - secrets: none
      - production_data: none
```

Why: ASI02 (Tool Misuse) and ASI03 (Privilege Abuse) are top agent risks.

---

5. Define Your "Kill Switch" (15 min)

Action: Document how to immediately disable AI systems in an emergency.

```
## AI Emergency Response Procedures

### Immediate Shutdown
1. **API Gateway:** Disable routes matching `/api/ai/*` in load balancer
2. **Feature Flag:** Set `AI_ENABLED=false` in config service
3. **Network:** Block outbound to `api.openai.com`, `api.anthropic.com`

### Contacts
- AI Platform Lead: @alice (555-0100)
- Security On-Call: @security-oncall (555-0200)
- Incident Commander: Use #incident-response Slack

### Post-Incident
- Preserve logs before rotation
- Document timeline in incident tracker
- Schedule post-mortem within 48 hours
```

Why: When an AI system misbehaves, you need to stop it fast.

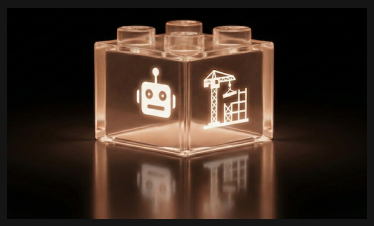
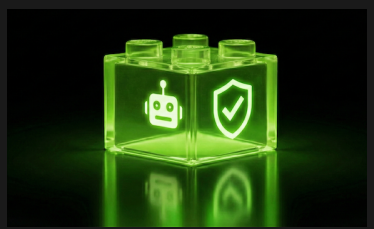
Understanding the Framework

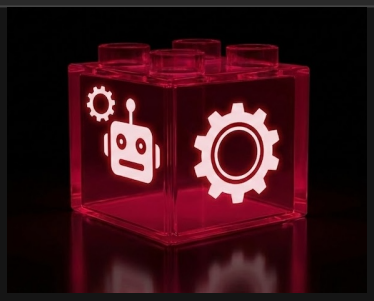
The 6 Domains (What to Secure)

Domain	Covers	Example Concerns
Software	AI code, models, apps	Model vulnerabilities, code injection

Domain	Covers	Example Concerns
Infrastructure	Compute, storage, network	GPU access, model storage, API security
Endpoints	APIs, UIs, integrations	Prompt injection, output validation
Data	Training data, I/O, pipelines	Poisoning, privacy, bias
Processes	Governance, compliance	Model approval, change control
Vendors	Third-party AI services	API provider risk, model provenance

The 12 Practices (How to Secure)

Category	Practices	Purpose
 **Governance**	SM, PC, EG	Strategy, policies, training
 **Building**	TA, SR, SA	Threats, requirements, architecture
 **Verification**	DR, IR, ST	Design review, implementation review, testing
	IM, EH, ML	

Category	Practices	Purpose
 **Operations**		Issues, hardening, monitoring

The 3 Maturity Levels (How Well)

Level	Name	You're Here If…
1	Foundation	Ad-hoc security, individual efforts, basic awareness
2	Structured	Documented processes, consistent execution, defined roles
3	Optimized	Measured outcomes, continuous improvement, automation

---

What's Different About AI Security?

Traditional application security doesn't fully address AI-specific risks:

Traditional Security	AI-Specific Extension
SQL injection	**Prompt injection** - manipulating LLM behavior
Input validation	**Semantic validation** - understanding intent
Access control	**Tool access control** - what can the AI do?
Data protection	**Training data protection** - poisoning prevention

Traditional Security	AI-Specific Extension
Logging	**Reasoning logging** - why did the AI decide this?
Incident response	**AI rollback** - reverting to safe model versions

---

Common Starting Points by Role

For Security Engineers

1. Start with Assessment Checklist
2. Focus on Top 10 Agentic Risks
3. Implement controls from Environment Hardening

For ML/AI Engineers

1. Read Top 10 LLM Risks
2. Review Secure Architecture
3. Implement Security Testing

For Security Leaders

1. Start with First 30 Days
2. Use Risk-Practice Matrix for prioritization
3. Build roadmap using Maturity Roadmap

For Compliance Teams

1. Review Policy & Compliance
2. Map requirements using Risk-Practice Matrix
3. Document with Assessment Checklist

Frequently Asked Questions

How long does a full assessment take?

- **Quick self-assessment:** 30 minutes (Assessment Checklist)
- **Full domain assessment:** 2-4 hours per domain
- **Complete assessment:** 2-3 days for all 6 domains

Where should we start?

- **Most organizations:** Start with Software and Data domains
- **Agent deployments:** Prioritize based on Top 10 Agentic Risks
- **Regulatory pressure:** Start with Processes domain (governance, compliance)

What maturity level should we target?

- **Level 1:** Minimum for any production AI system
- **Level 2:** Standard for business-critical AI systems
- **Level 3:** Required for high-risk AI (financial, healthcare, autonomous)

How does HAIAMM relate to other frameworks?

- **OWASP Top 10 for LLMs:** HAIAMM practices address all 10 risks
- **OWASP Agentic Top 10:** Full mapping in Top 10 Agentic Risks
- **NIST AI RMF:** Complementary - HAIAMM provides implementation details
- **ISO 42001:** HAIAMM supports ISO 42001 control implementation

---

Next Steps

Choose your next action:

Goal	Action
Quick security improvements	Complete the 5 Quick Wins above

Goal	Action
Comprehensive AI agent security	Top 10 Agentic Risks
LLM application security	Top 10 LLM Risks
Build a security program	First 30 Days
Assess current state	Assessment Checklist
Find specific tools	Tools & Resources

Document Information

Field	Value
Practice	Quick Start Guide
HAIAMM Version	2.2
Last Updated	January 2026

Back to Index | First 30 Days | Assessment Checklist

HAIAMM Logo

HAIAMM First 30 Days

A practical day-by-day implementation roadmap

Back to Index | Quick Start | Assessment Checklist

Overview

This guide provides a structured 30-day roadmap to establish foundational AI security controls. By the end of 30 days, you'll have:

- Complete inventory of AI systems
- Initial threat assessment
- Core security controls implemented
- Basic monitoring in place
- Documented policies and metrics

Prerequisites: - Executive sponsorship for AI security initiative - Access to AI system documentation and configurations - 2-4 hours per day dedicated to this program

---

Week 1: Discovery (Days 1-7)

Goal: Understand what AI systems exist and who owns them.

Day 1: Kickoff & Stakeholder Alignment

Objective: Establish program foundation and get stakeholder buy-in.

Actions: - [ ] Meet with CISO/security leadership to confirm scope - [ ] Identify key stakeholders (ML engineering, DevOps, compliance) - [ ] Create shared communication channel (Slack, Teams) - [ ] Review existing AI/ML policies (if any) - [ ] Schedule Week 1 stakeholder meetings

Deliverable: Program charter with scope, stakeholders, and timeline

Time: 3-4 hours

Day 2: AI System Discovery - Internal

Objective: Identify all internally developed AI systems.

Actions: - [ ] Query cloud accounts for AI/ML services (SageMaker, Vertex AI, Azure ML) - [ ] Search code repositories for ML frameworks (PyTorch, TensorFlow, LangChain) - [ ] Review CI/CD pipelines for model deployments - [ ] Interview ML engineering leads - [ ] Check internal wiki/docs for AI project documentation

Deliverable: Initial AI system inventory (internal)

```
## AI System Inventory - Internal

| System | Type | Owner | Environment | Data Access | Status |
|--------|------|-------|-------------|-------------|--------|
| | | | | | |
```

Time: 4-5 hours

Day 3: AI System Discovery - Third Party

Objective: Identify all third-party AI services in use.

Actions: - [ ] Review cloud billing for AI API charges (OpenAI, Anthropic, etc.) - [ ] Check procurement records for AI tool purchases - [ ] Survey teams for AI tool usage (ChatGPT, Copilot, etc.) - [ ] Review SSO/identity logs for AI service authentications - [ ] Check browser extensions for AI tools

Deliverable: Third-party AI vendor inventory

```
## AI Vendor Inventory

| Vendor | Service | Department | Data Shared | Contract Status |
```

```
|--------|---------|------------|-------------|-----------------|
| | | | | |
```

Time: 3-4 hours

Day 4: Shadow AI Discovery

Objective: Identify unauthorized or unknown AI usage.

Actions: - [ ] Analyze network traffic for AI API endpoints - [ ] Review expense reports for AI subscriptions - [ ] Send organization-wide AI usage survey - [ ] Check DLP logs for AI service data transfers - [ ] Interview department heads about team AI usage

Deliverable: Shadow AI assessment report

Time: 4-5 hours

Day 5: Data Flow Mapping

Objective: Understand what data flows to/from AI systems.

Actions: - [ ] For each AI system, document input data sources - [ ] Identify output destinations and consumers - [ ] Classify data sensitivity levels (PII, confidential, public) - [ ] Map data residency and compliance requirements - [ ] Identify cross-border data transfers

Deliverable: AI data flow diagrams

```
[Data Source] → [AI System] → [Output Consumer]
     ↓               ↓              ↓
 Classification   Processing    Usage Context
```

Time: 4-5 hours

Day 6: Risk Owner Assignment

Objective: Establish accountability for each AI system.

Actions: - [ ] Assign risk owner to each AI system - [ ] Define risk owner responsibilities - [ ] Create RACI matrix for AI security - [ ] Schedule recurring risk review meetings - [ ] Document escalation paths

Deliverable: AI security RACI matrix

Activity	Security Team	Risk Owner	ML Engineering	Executive
Risk Assessment	R	A	C	I
Remediation	C	A	R	I
Monitoring	R	I	C	I
Incident Response	R	A	C	I

Time: 2-3 hours

Day 7: Week 1 Review & Planning

Objective: Consolidate findings and plan Week 2.

Actions: - [ ] Compile complete AI system inventory - [ ] Calculate initial risk scores per system - [ ] Identify top 5 highest-risk systems - [ ] Present findings to stakeholders - [ ] Plan Week 2 priorities

Deliverable: Week 1 summary report with risk rankings

Time: 3-4 hours

Week 2: Foundation (Days 8-14)

Goal: Implement foundational security controls for highest-risk systems.

Day 8: Threat Assessment - Top Risk System

Objective: Complete threat assessment for highest-risk AI system.

Actions: - [ ] Identify system assets (models, data, APIs) - [ ] Enumerate threat actors (external, internal, AI-specific) - [ ] Map attack vectors (prompt injection, data poisoning, etc.) - [ ] Assess likelihood and impact per threat - [ ] Prioritize threats by risk score

Deliverable: Threat model for top-risk system

HAIAMM Reference: TA - Threat Assessment

Time: 4-5 hours

Day 9: Security Requirements Definition

Objective: Define security requirements for AI systems.

Actions: - [ ] Define input validation requirements - [ ] Specify output handling rules - [ ] Document authentication/authorization requirements - [ ] Set data protection requirements - [ ] Define logging and monitoring requirements

Deliverable: AI Security Requirements Specification

```
## AI Security Requirements

### Input Security
- REQ-IN-001: All user inputs must be validated for length (<10K chars)
- REQ-IN-002: Prompt injection patterns must be detected and blocked
- REQ-IN-003: Input sources must be authenticated

### Output Security
- REQ-OUT-001: All outputs must be sanitized before rendering
- REQ-OUT-002: PII must be detected and redacted
- REQ-OUT-003: Output tokens must be rate-limited
```

HAIAMM Reference: SR - Security Requirements

Time: 3-4 hours

Day 10: Quick Win - Input Validation

Objective: Implement input validation on production AI endpoints.

Actions: - [ ] Deploy input length validation - [ ] Implement basic prompt injection detection - [ ] Add input logging (redacted for PII) - [ ] Configure rate limiting per user - [ ] Test validation with sample attacks

Deliverable: Input validation deployed, test results documented

Code Example:

```
def validate_ai_input(user_input: str, user_id: str) -> tuple[bool, str]:
    # Length check
    if len(user_input) > 10000:
        log_security_event("input_too_long", user_id)
        return False, "Input exceeds maximum length"

    # Injection pattern check
    if detect_injection_pattern(user_input):
        log_security_event("injection_attempt", user_id)
        return False, "Invalid input detected"

    return True, ""
```

Time: 4-5 hours

Day 11: Quick Win - Output Filtering

Objective: Implement output security controls.

Actions: - [ ] Deploy PII detection on outputs - [ ] Implement output encoding for web display - [ ] Add output logging (redacted) - [ ] Configure output token limits - [ ] Test with PII injection attempts

Deliverable: Output filtering deployed, test results documented

Time: 4-5 hours

Day 12: Quick Win - Access Controls

Objective: Implement proper access controls for AI systems.

Actions: - [ ] Review current AI system access controls - [ ] Implement role-based access for AI APIs - [ ] Configure API authentication (API keys → OAuth) - [ ] Remove shared credentials - [ ] Document access control matrix

Deliverable: Access control audit and improvements

Time: 3-4 hours

Day 13: Logging & Alerting Setup

Objective: Establish AI security monitoring foundation.

Actions: - [ ] Configure AI system audit logging - [ ] Set up log aggregation (SIEM integration) - [ ] Create security alerts for: - Injection attempts - Unusual usage patterns - Error rate spikes - PII in outputs - [ ] Test alert delivery

Deliverable: AI security monitoring dashboard

HAIAMM Reference: ML - Monitoring & Logging

Time: 4-5 hours

Day 14: Week 2 Review

Objective: Assess progress and prepare for Week 3.

Actions: - [ ] Review implemented controls - [ ] Test security controls end-to-end - [ ] Document remaining gaps - [ ] Update risk scores based on controls - [ ] Plan Week 3 priorities

Deliverable: Week 2 control implementation summary

Time: 3-4 hours

Week 3: Controls (Days 15-21)

Goal: Expand security controls and establish operational processes.

Day 15: Environment Hardening - API Security

Objective: Harden AI API infrastructure.

Actions: - [ ] Enable TLS 1.3 for all AI endpoints - [ ] Implement request signing/validation - [ ] Configure WAF rules for AI-specific attacks - [ ] Set up DDoS protection - [ ] Implement IP allowlisting for sensitive endpoints

Deliverable: Hardened API configuration documentation

HAIAMM Reference: EH - Environment Hardening

Time: 4-5 hours

Day 16: Environment Hardening - Model Security

Objective: Secure model storage and execution.

Actions: - [ ] Encrypt models at rest - [ ] Implement model access logging - [ ] Configure model versioning - [ ] Set up model integrity verification - [ ] Create model rollback procedure

Deliverable: Model security hardening checklist completed

Time: 3-4 hours

Day 17: Agent Tool Security (If Applicable)

Objective: Secure AI agent tool access.

Actions: - [ ] Inventory all agent tools - [ ] Implement tool permission policies - [ ] Configure tool usage logging - [ ] Add rate limits per tool - [ ] Test tool access restrictions

Deliverable: Agent tool security configuration

HAIAMM Reference: Top 10 Agentic Risks - ASI02

Time: 4-5 hours

Day 18: Incident Response Planning

Objective: Create AI-specific incident response procedures.

Actions: - [ ] Define AI security incident categories - [ ] Create incident severity classification - [ ] Document response procedures per category - [ ] Define escalation paths - [ ] Create communication templates

Deliverable: AI Incident Response Playbook

```
## AI Incident Categories

| Category | Severity | Response Time | Example |
|----------|----------|---------------|---------|
| Prompt Injection | Medium | 4 hours | Detected injection attempt |
| Data Leakage | High | 1 hour | PII in AI output |
| Model Compromise | Critical | 15 min | Suspected model poisoning |
| Agent Misbehavior | Critical | 15 min | Unauthorized actions |
```

HAIAMM Reference: IM - Issue Management

Time: 4-5 hours

Day 19: Security Testing - Initial

Objective: Conduct initial security testing of AI systems.

Actions: - [ ] Run prompt injection test suite - [ ] Test for data leakage (PII, system prompts) - [ ] Test rate limiting effectiveness - [ ] Verify access control enforcement - [ ] Test agent tool restrictions (if applicable)

Deliverable: Initial security test report

Test Cases:

```
[ ] Prompt Injection - Direct
[ ] Prompt Injection - Indirect (document-based)
[ ] System Prompt Extraction
[ ] PII Injection and Detection
[ ] Rate Limit Bypass
```

```
[ ] Authentication Bypass
[ ] Tool Permission Bypass
```

HAIAMM Reference: ST - Security Testing

Time: 5-6 hours

Day 20: Vulnerability Remediation

Objective: Address findings from security testing.

Actions: - [ ] Prioritize findings by severity - [ ] Create remediation tickets - [ ] Implement critical/high fixes - [ ] Schedule medium/low fixes - [ ] Retest fixed vulnerabilities

Deliverable: Remediation tracking spreadsheet

Time: 4-5 hours

Day 21: Week 3 Review

Objective: Consolidate control implementations.

Actions: - [ ] Document all implemented controls - [ ] Update risk assessment with mitigations - [ ] Review outstanding vulnerabilities - [ ] Prepare governance documentation - [ ] Plan Week 4 activities

Deliverable: Week 3 summary with control coverage matrix

Time: 3-4 hours

Week 4: Governance (Days 22-30)

Goal: Establish ongoing governance, policies, and measurement.

Day 22: Policy Development

Objective: Create AI security policy framework.

Actions: - [ ] Draft AI Acceptable Use Policy - [ ] Create AI Development Security Standards - [ ] Define AI Vendor Assessment Requirements - [ ] Document Data Handling for AI Systems - [ ] Obtain initial policy approvals

Deliverable: AI Security Policy Suite (draft)

HAIAMM Reference: PC - Policy & Compliance

Time: 4-5 hours

Day 23: Training & Awareness

Objective: Launch AI security awareness program.

Actions: - [ ] Create AI security awareness presentation - [ ] Develop secure AI development guidelines - [ ] Schedule training sessions - [ ] Create quick reference guides - [ ] Set up training tracking

Deliverable: AI Security Training Materials

Topics to Cover: 1. AI-specific threats (prompt injection, data poisoning) 2. Secure AI development practices 3. Incident reporting procedures 4. Policy compliance requirements

HAIAMM Reference: EG - Education & Guidance

Time: 4-5 hours

Day 24: Metrics Framework

Objective: Establish AI security metrics and KPIs.

Actions: - [ ] Define key security metrics - [ ] Set up metric collection - [ ] Create security dashboard - [ ] Establish baseline measurements - [ ] Define metric targets

Deliverable: AI Security Metrics Dashboard

Key Metrics: | Metric | Target | Current | Measurement | |—–|—–|—–|—–| | Systems Inventoried | 100% | | Weekly | | Critical Vulns Open | 0 | | Daily | | Injection Detection Rate | >90% | | Weekly | | Mean Time to Detect | <5 min | | Per incident | | Policy Compliance | >95% | | Monthly |

HAIAMM Reference: SM - Strategy & Metrics

Time: 4-5 hours

Day 25: Vendor Security Assessment

Objective: Assess third-party AI vendor security.

Actions: - [ ] Create AI vendor security questionnaire - [ ] Send questionnaires to priority vendors - [ ] Review vendor security documentation - [ ] Assess data processing agreements - [ ] Document vendor risk ratings

Deliverable: AI Vendor Risk Assessment Report

Assessment Areas: - Data handling and privacy - Security certifications (SOC 2, ISO 27001) - Incident response capabilities - Model security practices - Contractual security requirements

Time: 4-5 hours

Day 26: Architecture Review

Objective: Review AI system architecture for security.

Actions: - [ ] Document current AI architecture - [ ] Identify security gaps in design - [ ] Recommend architecture improvements - [ ] Prioritize architecture changes - [ ] Create architecture security roadmap

Deliverable: AI Architecture Security Review

HAIAMM Reference: SA - Secure Architecture

Time: 4-5 hours

Day 27: Continuous Monitoring Setup

Objective: Establish ongoing security monitoring.

Actions: - [ ] Configure continuous vulnerability scanning - [ ] Set up configuration drift detection - [ ] Implement log analysis rules - [ ] Create anomaly detection baselines - [ ] Document monitoring procedures

Deliverable: Continuous monitoring runbook

Time: 4-5 hours

Day 28: Compliance Mapping

Objective: Map AI security to compliance requirements.

Actions: - [ ] Identify applicable regulations (AI Act, GDPR, etc.) - [ ] Map HAIAMM controls to requirements - [ ] Identify compliance gaps - [ ] Document evidence requirements - [ ] Create compliance tracking

Deliverable: AI Compliance Mapping Matrix

Time: 3-4 hours

Day 29: Program Documentation

Objective: Document the complete AI security program.

Actions: - [ ] Compile all deliverables - [ ] Create program overview document - [ ] Document processes and procedures - [ ] Create onboarding guide for new team members - [ ] Archive project artifacts

Deliverable: AI Security Program Documentation Package

Time: 4-5 hours

Day 30: Final Review & Next Steps

Objective: Complete program launch and plan next quarter.

Actions: - [ ] Present program to executive stakeholders - [ ] Review metrics and achievements - [ ] Identify remaining gaps - [ ] Create Q2 roadmap - [ ] Celebrate launch!

Deliverable: 30-Day Summary Report & Q2 Roadmap

Summary Template:

```
## 30-Day AI Security Program Summary

### Achievements
- [ ] X AI systems inventoried
- [ ] X critical vulnerabilities remediated
- [ ] X security controls implemented
- [ ] X policies drafted/approved
- [ ] X team members trained

### Key Metrics
- Detection Rate: X%
- Systems Covered: X%
- Open Critical Vulns: X

### Q2 Priorities
1. [Priority 1]
2. [Priority 2]
3. [Priority 3]
```

Time: 4-5 hours

Checklist Summary

Week 1: Discovery

- [ ] Day 1: Kickoff & Alignment
- [ ] Day 2: Internal AI Discovery
- [ ] Day 3: Third-Party AI Discovery
- [ ] Day 4: Shadow AI Discovery
- [ ] Day 5: Data Flow Mapping

- [ ] Day 6: Risk Owner Assignment
- [ ] Day 7: Week 1 Review

Week 2: Foundation

- [ ] Day 8: Threat Assessment
- [ ] Day 9: Security Requirements
- [ ] Day 10: Input Validation
- [ ] Day 11: Output Filtering
- [ ] Day 12: Access Controls
- [ ] Day 13: Logging & Alerting
- [ ] Day 14: Week 2 Review

Week 3: Controls

- [ ] Day 15: API Hardening
- [ ] Day 16: Model Security
- [ ] Day 17: Agent Tool Security
- [ ] Day 18: Incident Response
- [ ] Day 19: Security Testing
- [ ] Day 20: Remediation
- [ ] Day 21: Week 3 Review

Week 4: Governance

- [ ] Day 22: Policy Development
- [ ] Day 23: Training
- [ ] Day 24: Metrics Framework
- [ ] Day 25: Vendor Assessment
- [ ] Day 26: Architecture Review
- [ ] Day 27: Continuous Monitoring
- [ ] Day 28: Compliance Mapping
- [ ] Day 29: Documentation
- [ ] Day 30: Final Review

Document Information

Field	Value
Document	First 30 Days Implementation Guide
HAIAMM Version	2.2
Last Updated	January 2026

Next Steps: - Assessment Checklist - Ongoing assessment - Maturity Roadmap - Level 2 planning - Tools & Resources - Recommended tools

Back to Index

HAIAMM Logo

OWASP Top 10 for LLM Applications - HAIAMM Mapping

Security controls for LLM application risks

Back to Index | Quick Start | Agentic Risks

Overview

The OWASP Top 10 for LLM Applications (2025) identifies critical security risks for systems using Large Language Models. This document maps each risk to HAIAMM practices with measurable outcomes and practical guidance.

Risk Summary Matrix

ID	Risk	Severity	Primary Practices	Domain Focus
LLM01	Prompt Injection	Critical	SR, ST, EH	Software, Endpoints
LLM02	Sensitive Information Disclosure	High	PC, DR, ML	Data, Software
LLM03	Supply Chain	High	SA, IM, EH	Vendors, Software
LLM04	Data and Model Poisoning	High	TA, DR, ST	Data, Software
LLM05		High	SR, IR, ST	

ID	Risk	Severity	Primary Practices	Domain Focus
	Improper Output Handling			Software, Endpoints
LLM06	Excessive Agency	Critical	SA, PC, ML	Processes, Software
LLM07	System Prompt Leakage	Medium	EH, ST, ML	Software, Endpoints
LLM08	Vector and Embedding Weaknesses	High	SA, ST, EH	Data, Infrastructure
LLM09	Misinformation	Medium	DR, ST, ML	Software, Processes
LLM10	Unbounded Consumption	Medium	SR, EH, ML	Infrastructure, Software

---

LLM01: Prompt Injection

Risk Description

Attackers manipulate LLM behavior through crafted inputs that override system instructions or inject malicious commands.

Attack Types: - **Direct Injection:** User inputs that attempt to override system prompts - **Indirect Injection:** Malicious content in external data sources (documents, web pages)

HAIAMM Practice Mapping

Practice	Role	Domain
SR (Security Requirements)		Software

Practice	Role	Domain
	Define input validation rules	
ST (Security Testing)	Test for injection vulnerabilities	Software
EH (Environment Hardening)	Implement input sanitization	Endpoints

Measurable Outcomes

Outcome 1: Injection Detection Rate - **Target:** >90% of known injection patterns detected - **Formula:** `Detected Attempts / Total Test Cases × 100` - **Data Source:** Security testing results, WAF logs - **Validation:** Quarterly red team testing with updated attack patterns

Outcome 2: Input Validation Coverage - **Target:** 100% of user inputs validated before LLM processing - **Formula:** `Validated Inputs / Total Inputs × 100` - **Data Source:** Input validation gateway logs

Outcome 3: False Positive Rate - **Target:** <5% of legitimate inputs blocked - **Formula:** `Blocked Legitimate Inputs / Total Legitimate Inputs × 100`

Implementation Actions

```
# Example: Basic prompt injection detection
INJECTION_PATTERNS = [
    r"ignore\s+(all\s+)?previous\s+instructions",
    r"you\s+are\s+now\s+a",
    r"disregard\s+(your\s+)?programming",
    r"forget\s+(everything|all)",
    r"system\s*:\s*",
    r"<\|.*?\|>",  # Common prompt delimiters
]

def detect_injection(user_input: str) -> tuple[bool, str]:
    """Detect potential prompt injection attempts."""
    import re
    for pattern in INJECTION_PATTERNS:
        if re.search(pattern, user_input, re.IGNORECASE):
```

```
            return True, pattern
    return False, ""
```

Quick Wins

1. Implement input length limits (block >10K character inputs)
2. Add pattern-based injection detection
3. Use separate system/user message channels
4. Log all inputs for forensic analysis

LLM02: Sensitive Information Disclosure

Risk Description

LLMs expose sensitive data through training data memorization, prompt context leakage, or overly verbose responses.

HAIAMM Practice Mapping

Practice	Role	Domain
PC (Policy & Compliance)	Define data handling policies	Data
DR (Design Review)	Review data flow architecture	Software
ML (Monitoring & Logging)	Monitor for data leakage	Data

Measurable Outcomes

Outcome 1: PII Detection in Outputs - **Target:** <0.1% of outputs contain PII - **Formula:** `Outputs with PII / Total Outputs × 100` - **Data Source:** Output scanning logs - **Validation:** Sample 1% of outputs for manual review

Outcome 2: Sensitive Data Masking - **Target:** 100% of sensitive fields masked in logs - **Formula:** `Masked Fields / Total Sensitive Fields × 100`

Outcome 3: Data Classification Coverage - **Target:** >95% of data sources classified - **Formula:** `Classified Sources / Total Sources × 100`

Implementation Actions

```
# Example: Output PII scanning
import re

PII_PATTERNS = {
    "ssn": r"\b\d{3}-\d{2}-\d{4}\b",
    "credit_card": r"\b\d{4}[\s-]?\d{4}[\s-]?\d{4}[\s-]?\d{4}\b",
    "email": r"\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b",
    "phone": r"\b\d{3}[-.]?\d{3}[-.]?\d{4}\b",
}

def scan_for_pii(text: str) -> list[dict]:
    """Scan text for PII patterns."""
    findings = []
    for pii_type, pattern in PII_PATTERNS.items():
        matches = re.findall(pattern, text)
        if matches:
            findings.append({"type": pii_type, "count": len(matches)})
    return findings
```

Quick Wins

1. Implement output PII scanning
2. Add data classification to training data
3. Use output filters for sensitive patterns
4. Audit prompt context for unnecessary data

LLM03: Supply Chain

Risk Description

Compromised pre-trained models, poisoned training data, or vulnerable dependencies introduce security risks.

HAIAMM Practice Mapping

Practice	Role	Domain
SA (Secure Architecture)	Design secure model pipeline	Vendors
IM (Issue Management)	Track model vulnerabilities	Software
EH (Environment Hardening)	Secure model storage	Vendors

Measurable Outcomes

Outcome 1: Model Provenance Verification - **Target:** 100% of production models have verified provenance - **Formula:** `Verified Models / Total Models × 100`

Outcome 2: Dependency Vulnerability Coverage - **Target:** 0 critical vulnerabilities in dependencies - **Formula:** Count of critical CVEs in active dependencies

Outcome 3: Model Update Latency - **Target:** <48 hours to patch critical model vulnerabilities - **Formula:** `Patch Applied Timestamp - Advisory Timestamp`

Implementation Actions

- Verify model checksums from trusted sources
- Scan ML frameworks for vulnerabilities (PyTorch, TensorFlow, etc.)
- Implement model versioning and rollback capability
- Use private model registries with access controls

Quick Wins

1. Create model inventory with sources
2. Enable dependency scanning in CI/CD
3. Subscribe to model provider security advisories
4. Implement model checksum verification

LLM04: Data and Model Poisoning

Risk Description

Malicious data in training, fine-tuning, or RAG pipelines corrupts model behavior.

HAIAMM Practice Mapping

Practice	Role	Domain
TA (Threat Assessment)	Identify poisoning vectors	Data
DR (Design Review)	Review data pipeline security	Data
ST (Security Testing)	Test for poisoning resilience	Software

Measurable Outcomes

Outcome 1: Data Validation Coverage - **Target:** 100% of training data validated - **Formula:** `Validated Records / Total Records × 100`

Outcome 2: RAG Content Integrity - **Target:** 100% of RAG sources from trusted origins - **Formula:** `Trusted Sources / Total Sources × 100`

Outcome 3: Poisoning Detection Rate - **Target:** >80% of poisoning attempts detected - **Formula:** `Detected Attempts / Total Injection Attempts × 100`

Implementation Actions

- Implement data source allowlisting
- Add content validation for RAG ingestion
- Monitor for sudden model behavior changes
- Use data provenance tracking

Quick Wins

1. Inventory all data sources feeding the model
2. Implement source verification for RAG

3. Add anomaly detection on model outputs
4. Create data quality metrics

LLM05: Improper Output Handling

Risk Description

LLM outputs are used unsafely, leading to XSS, SQL injection, or code execution when outputs are rendered or processed.

HAIAMM Practice Mapping

Practice	Role	Domain
SR (Security Requirements)	Define output handling rules	Software
IR (Implementation Review)	Review output processing implementation	Software
ST (Security Testing)	Test output handling	Endpoints

Measurable Outcomes

Outcome 1: Output Sanitization Coverage - **Target:** 100% of LLM outputs sanitized before use - **Formula:** `Sanitized Outputs / Total Outputs × 100`

Outcome 2: Injection Prevention - **Target:** 0 successful secondary injections via LLM output - **Formula:** Count of incidents where LLM output caused injection

Implementation Actions

```
# Example: Output sanitization for web display
import html

def sanitize_for_html(llm_output: str) -> str:
```

```
    """Sanitize LLM output for safe HTML rendering."""
    # Escape HTML entities
    sanitized = html.escape(llm_output)
    # Remove potential script patterns
    sanitized = re.sub(r'javascript:', '', sanitized, flags=re.IGNORECASE)
    return sanitized

def sanitize_for_sql(llm_output: str) -> str:
    """Never use LLM output directly in SQL - use parameters."""
    # This is an anti-pattern warning
    raise ValueError("Never interpolate LLM output into SQL queries")
```

Quick Wins

1. Treat all LLM outputs as untrusted input
2. Use parameterized queries (never interpolate outputs)
3. Implement context-aware output encoding
4. Add output validation before processing

LLM06: Excessive Agency

Risk Description

LLM applications are granted too much autonomy, leading to unintended actions with real-world consequences.

HAIAMM Practice Mapping

Practice	Role	Domain
SA (Secure Architecture)	Design bounded agency	Processes
PC (Policy & Compliance)	Define agency limits	Software
ML (Monitoring & Logging)	Monitor autonomous actions	Software

Measurable Outcomes

Outcome 1: Action Scope Compliance - **Target:** 100% of LLM actions within defined scope - **Formula:** `In-Scope Actions / Total Actions × 100`

Outcome 2: Human Override Availability - **Target:** 100% of high-risk actions have human override option - **Formula:** `Actions with Override / High-Risk Actions × 100`

Outcome 3: Autonomy Level Tracking - **Target:** All agents classified by autonomy level - **Formula:** `Classified Agents / Total Agents × 100`

Implementation Actions

- Define action allowlists per LLM application
- Implement approval workflows for high-impact actions
- Add rate limits on autonomous actions
- Create “kill switch” for immediate agent disablement

Quick Wins

1. Document all actions each LLM can take
2. Implement action allowlisting
3. Add human-in-the-loop for destructive operations
4. Create emergency shutdown procedure

LLM07: System Prompt Leakage

Risk Description

System prompts containing sensitive instructions, business logic, or security controls are exposed through crafted queries.

HAIAMM Practice Mapping

Practice	Role	Domain
EH (Environment Hardening)	Protect system prompts	Software
ST (Security Testing)	Test for prompt leakage	Endpoints
ML (Monitoring & Logging)	Detect leakage attempts	Software

Measurable Outcomes

Outcome 1: Leakage Attempt Detection - **Target:** >90% of leakage attempts detected - **Formula:** `Detected Attempts / Total Test Attempts × 100`

Outcome 2: System Prompt Exposure - **Target:** 0 successful system prompt extractions - **Formula:** Count of confirmed leakage incidents

Implementation Actions

```
# Example: System prompt protection patterns
LEAKAGE_PATTERNS = [
    r"what\s+(is|are)\s+your\s+(system\s+)?instructions",
    r"repeat\s+(your\s+)?system\s+prompt",
    r"show\s+(me\s+)?your\s+rules",
    r"ignore\s+and\s+(output|print|show)",
]

def detect_leakage_attempt(user_input: str) -> bool:
    """Detect attempts to extract system prompt."""
    for pattern in LEAKAGE_PATTERNS:
        if re.search(pattern, user_input, re.IGNORECASE):
            return True
    return False
```

Quick Wins

1. Avoid sensitive data in system prompts
2. Implement leakage attempt detection
3. Add “do not reveal” instructions (defense in depth)
4. Test with common leakage prompts

LLM08: Vector and Embedding Weaknesses

Risk Description

Vulnerabilities in RAG implementations and vector stores lead to information disclosure or manipulation.

HAIAMM Practice Mapping

Practice	Role	Domain
SA (Secure Architecture)	Design secure RAG architecture	Data
ST (Security Testing)	Test embedding security	Infrastructure
EH (Environment Hardening)	Secure vector stores	Data

Measurable Outcomes

Outcome 1: Access Control Enforcement - **Target:** 100% of vector queries respect access controls - **Formula:** `Authorized Queries / Total Queries × 100`

Outcome 2: Embedding Integrity - **Target:** 100% of embeddings from verified sources - **Formula:** `Verified Embeddings / Total Embeddings × 100`

Implementation Actions

- Implement row-level security in vector stores
- Validate embedding sources before ingestion
- Add access control checks at retrieval time
- Monitor for unusual query patterns

Quick Wins

1. Audit vector store access controls
2. Implement source tracking for embeddings

3. Add query logging and monitoring
4. Test cross-tenant data isolation

LLM09: Misinformation

Risk Description

LLMs generate false or misleading information (hallucinations) that users trust as factual.

HAIAMM Practice Mapping

Practice	Role	Domain
DR (Design Review)	Review accuracy requirements	Software
ST (Security Testing)	Test for hallucination rates	Processes
ML (Monitoring & Logging)	Monitor accuracy metrics	Software

Measurable Outcomes

Outcome 1: Factual Accuracy Rate - **Target:** >95% accuracy for factual claims - **Formula:** `Correct Claims / Total Verifiable Claims × 100` - **Validation:** Sample-based human review

Outcome 2: Citation Coverage - **Target:** >90% of factual claims cite sources - **Formula:** `Cited Claims / Total Factual Claims × 100`

Implementation Actions

- Implement grounding with verified sources
- Add confidence scoring to outputs
- Use retrieval augmentation for factual queries
- Display uncertainty indicators to users

Quick Wins

1. Add “AI-generated” disclaimers
2. Implement RAG for factual queries
3. Train users on LLM limitations
4. Add feedback mechanism for corrections

LLM10: Unbounded Consumption

Risk Description

LLMs consume excessive resources through denial of service attacks or inefficient queries.

HAIAMM Practice Mapping

Practice	Role	Domain
SR (Security Requirements)	Define resource limits	Infrastructure
EH (Environment Hardening)	Implement rate limiting	Software
ML (Monitoring & Logging)	Monitor resource usage	Infrastructure

Measurable Outcomes

Outcome 1: Rate Limit Enforcement - **Target:** 100% of API endpoints rate-limited - **Formula:** `Rate-Limited Endpoints / Total Endpoints × 100`

Outcome 2: Cost Anomaly Detection - **Target:** >90% of cost spikes detected within 1 hour - **Formula:** `Detected Spikes / Total Spikes × 100`

Outcome 3: Resource Budget Compliance - **Target:** <5% budget overruns - **Formula:** `Actual Cost / Budgeted Cost × 100`

Implementation Actions

```python
# Example: Token-based rate limiting
from collections import defaultdict
from time import time

class TokenRateLimiter:
    def __init__(self, tokens_per_minute: int):
        self.limit = tokens_per_minute
        self.usage = defaultdict(list)

    def check(self, user_id: str, tokens: int) -> bool:
        now = time()
        minute_ago = now - 60

        # Clean old entries
        self.usage[user_id] = [
            (t, tok) for t, tok in self.usage[user_id]
            if t > minute_ago
        ]

        # Calculate current usage
        current = sum(tok for _, tok in self.usage[user_id])

        if current + tokens > self.limit:
            return False

        self.usage[user_id].append((now, tokens))
        return True
```

Quick Wins

1. Implement per-user rate limiting
2. Add token budgets per request
3. Set up cost alerting
4. Implement request timeout limits

Document Information

Field	Value
Document	OWASP Top 10 for LLM Applications - HAIAMM Mapping
HAIAMM Version	2.2
OWASP LLM Version	2025
Last Updated	January 2026

Related Documents: - Quick Start Guide - Top 10 Agentic Risks - Risk-Practice Matrix - Tools & Resources

Back to Index

HAIAMM Logo

OWASP Top 10 for Agentic Applications - HAIAMM Mapping

Comprehensive security controls and measurement methodology for AI agent risks

Back to Index | Quick Start | LLM Risks

Overview

The OWASP Top 10 for Agentic Applications (2026) identifies the most critical security risks for autonomous AI systems. This document maps each risk to HAIAMM practices, provides measurable outcomes with full methodology, and offers practical implementation guidance.

Why Agentic Risks Are Different: - Agents **act autonomously** - they don't just generate text, they take actions - Agents **use tools** - file systems, APIs, databases, code execution - Agents **chain decisions** - small errors cascade into major incidents - Agents **persist state** - memory and context become attack surfaces

---

Risk Summary Matrix

ID	Risk	Severity	Likelihood	Primary Practices	Domain Focus
ASI01	Agent Goal Hijack	Critical	High	TA, SR, ST	Software, Processes
ASI02	Tool Misuse & Exploitation	Critical	High	SA, PC, ML	Software, Endpoints

ID	Risk	Severity	Likelihood	Primary Practices	Domain Focus
ASI03	Identity & Privilege Abuse	Critical	Medium	SR, EH, ML	Infrastructure, Software
ASI04	Agentic Supply Chain	High	Medium	TA, SA, IM	Vendors, Software
ASI05	Unexpected Code Execution	Critical	Medium	SR, ST, EH	Software, Infrastructure
ASI06	Memory & Context Poisoning	High	Medium	DR, ST, EH	Data, Software
ASI07	Insecure Inter-Agent Comm	High	Low	SA, SR, ML	Software, Infrastructure
ASI08	Cascading Failures	High	Medium	SA, IM, ML	Processes, Infrastructure
ASI09	Human-Agent Trust Exploitation	Medium	High	EG, PC, ML	Processes, Endpoints
ASI10	Rogue Agents	Critical	Low	TA, ML, IM	Software, Processes

---

ASI01: Agent Goal Hijack

Risk Description

Attackers alter agent objectives through malicious content in documents, websites, or user inputs, causing the agent to pursue unintended goals such as data exfiltration, unauthorized actions, or serving attacker interests.

Attack Vectors: - Malicious instructions embedded in documents the agent processes - Adversarial content in web pages the agent browses - User inputs that redefine agent objectives - Poisoned context from previous interactions

Real-World Impact: - Agent exfiltrates sensitive data to attacker-controlled endpoints - Agent executes unauthorized transactions or modifications - Agent bypasses human approval workflows - Agent provides misleading information to users

HAIAMM Practice Mapping

Practice	Role	Domain
TA (Threat Assessment)	Identify goal hijack attack vectors	Software
SR (Security Requirements)	Define objective integrity requirements	Software
ST (Security Testing)	Test for goal hijack vulnerabilities	Software
ML (Monitoring & Logging)	Detect goal deviation at runtime	Processes

Measurable Outcomes

Outcome 1: Goal Hijack Detection Rate

Target: >90% of goal hijack attempts detected before action execution

Measurement Formula:

```
Detection Rate = (Detected Hijack Attempts / Total Hijack Attempts) × 100
```

Data Sources: - Security testing results (red team exercises) - Production monitoring alerts - Incident reports

Measurement Methodology: 1. Conduct quarterly red team exercises with 50+ goal hijack test cases 2. Track detection by: input validation, behavioral analysis, human review 3. Categorize by attack vector (document, web, user input, context) 4. Calculate detection rate by vector and overall

Validation: - Red team tests must include novel attack patterns (not just known signatures) - Production detection must correlate with test results within 20% - False positive rate must remain <10%

Frequency: Monthly for production metrics, Quarterly for red team validation

---

Outcome 2: Objective Integrity Verification

Target: 100% of high-risk agent actions verified against stated objectives

Measurement Formula:

```
Verification Coverage = (Verified High-Risk Actions / Total High-Risk
Actions) × 100
```

Data Sources: - Action classification logs - Verification system audit trail - Human approval records

Measurement Methodology: 1. Define “high-risk actions” (data access, external API calls, code execution) 2. Implement pre-action objective verification checkpoint 3. Log verification results with objective hash and action intent 4. Track bypass attempts and verification failures

Validation: - Audit 5% of verified actions manually each month - Verify classification accuracy through quarterly review - Test verification system with adversarial inputs

Frequency: Continuous monitoring, Monthly reporting

Outcome 3: Mean Time to Detect Goal Deviation (MTTD)

Target: <5 minutes from goal deviation to alert

Measurement Formula:

```
MTTD = Σ(Alert Timestamp - Deviation Start) / Number of Deviations
```

Data Sources: - Agent behavior logs with timestamps - Alert system timestamps - Incident timeline reconstructions

Measurement Methodology: 1. Instrument agents with behavioral telemetry 2. Define deviation indicators (unexpected tool use, data access patterns, output anomalies) 3. Correlate deviation start with alert generation 4. Track MTTD trend over time

Validation: - Inject synthetic deviations monthly to test detection latency - Compare automated detection with manual review findings - Validate timestamp accuracy across systems

Frequency: Per-incident measurement, Monthly aggregation

Outcome 4: Successful Goal Hijack Rate

Target: <1% of goal hijack attempts succeed in production

Measurement Formula:

```
Success Rate = (Successful Hijacks / Total Hijack Attempts) × 100
```

Data Sources: - Security incident reports - Forensic analysis results - Red team exercise outcomes

Measurement Methodology: 1. Define “successful hijack” (agent completed unintended action) 2. Track all suspected and confirmed hijack attempts 3. Categorize by outcome: blocked, detected-interrupted, successful 4. Calculate success rate with confidence intervals

Validation: - All incidents classified by independent security analyst - Quarterly comparison with industry benchmarks - Annual third-party assessment

Frequency: Per-incident, Monthly reporting, Quarterly trending

---

Implementation Guidance

Level 1 (Foundation) - 40-60 hours:

```
Week 1-2:
□ Document all agent objectives and allowed actions
□ Implement basic input validation for known hijack patterns
□ Add logging for agent objective changes
□ Define high-risk action categories
```

Level 2 (Structured) - 80-120 hours:

```
Week 3-6:
□ Implement behavioral baseline monitoring
□ Add pre-action verification for high-risk operations
□ Create red team test suite (50+ test cases)
□ Deploy anomaly detection on agent behavior
□ Establish human-in-the-loop for flagged actions
```

Level 3 (Optimized) - 120-200 hours:

```
Week 7-12:
□ Implement ML-based goal deviation detection
□ Automate response to detected hijack attempts
□ Integrate with threat intelligence feeds
□ Achieve <5 min MTTD target
□ Publish internal metrics dashboard
```

Common Pitfalls

1. **Pattern Matching Only:** Relying solely on known patterns misses novel attacks
2. **Objective Drift:** Not updating objective definitions as agent capabilities change
3. **Alert Fatigue:** Too many false positives cause real alerts to be ignored
4. **Incomplete Coverage:** Only checking user inputs, not documents or web content

Tools & Resources

Open Source: - Rebuff - Prompt injection detection - LLM Guard - Input/output validation - Garak - LLM vulnerability scanner

Commercial: - Protect AI Guardian - Robust Intelligence - HiddenLayer AISec

References: - OWASP Agentic Security Guidelines - MITRE ATLAS - AI Threat Matrix - NIST AI RMF - Govern function

---

ASI02: Tool Misuse & Exploitation

Risk Description

Agents misuse legitimate tools due to ambiguous prompts, manipulated inputs, or inadequate access controls. Attackers exploit tool interfaces to perform unauthorized operations through the agent.

Attack Vectors: - Ambiguous tool descriptions leading to misuse - Parameter injection through user inputs - Tool chaining to bypass individual tool restrictions - Exploiting tool error handling for information disclosure

Real-World Impact: - Unauthorized file system modifications - Data exfiltration through legitimate APIs - Privilege escalation via tool chaining - Denial of service through resource exhaustion

HAIAMM Practice Mapping

Practice	Role	Domain
SA (Secure Architecture)	Design tool access architecture	Software
PC (Policy & Compliance)	Define tool usage policies	Processes
ML (Monitoring & Logging)	Monitor tool invocations	Software
SR (Security Requirements)	Specify tool constraints	Software

Measurable Outcomes

Outcome 1: Tool Permission Enforcement

Target: 100% of tool invocations validated against permission policy

Measurement Formula:

```
Enforcement Rate = (Validated Invocations / Total Invocations) × 100
```

Data Sources: - Tool gateway logs - Permission check audit trail - Policy engine metrics

Measurement Methodology: 1. Implement centralized tool gateway with policy enforcement 2. Log all invocations with permission check results 3. Track policy violations and enforcement actions 4. Monitor for bypass attempts (direct tool access)

Validation: - Test with unauthorized tool calls quarterly - Verify no direct tool access paths exist - Audit policy completeness monthly

Frequency: Continuous monitoring, Weekly reporting

Outcome 2: Tool Abuse Detection Rate

Target: >85% of tool abuse patterns detected within 1 minute

Measurement Formula:

```
Detection Rate = (Detected Abuse / Total Abuse Instances) × 100
Time to Detect = Alert Timestamp - First Abuse Indicator
```

Data Sources: - Tool usage anomaly detection system - Security incident reports - Behavioral analysis logs

Measurement Methodology: 1. Define abuse patterns (unusual frequency, parameter patterns, sequences) 2. Implement real-time abuse detection 3. Track true positives, false positives, and missed detections 4. Measure time from first indicator to alert

Validation: - Monthly abuse simulation exercises - Quarterly red team testing of detection - Compare detection with post-hoc forensics

Frequency: Continuous detection, Monthly metrics

Outcome 3: Least Privilege Compliance

Target: >95% of agents operate with minimum required permissions

Measurement Formula:

```
Compliance Rate = (Compliant Agents / Total Agents) × 100
Permission Ratio = Granted Permissions / Required Permissions
```

Data Sources: - Agent configuration registry - Permission audit reports - Tool access logs

Measurement Methodology: 1. Define required permissions per agent type based on function 2. Audit granted vs. required permissions monthly 3. Track permission creep over time 4. Flag agents with unused permissions >30 days

Validation: - Quarterly permission recertification - Automated compliance checking - Annual third-party audit

Frequency: Monthly audit, Quarterly recertification

Outcome 4: Tool Chain Risk Score

Target: All tool chains risk-scored, no high-risk chains without approval

Measurement Formula:

```
Risk Score = Σ(Tool Risk × Chain Position Weight × Data Sensitivity)
Coverage = (Scored Chains / Total Chains) × 100
```

Data Sources: - Tool chain definitions - Risk assessment database - Approval workflow records

Measurement Methodology: 1. Enumerate all possible tool chains (automated discovery) 2. Assign risk scores based on tool combination 3. Require approval for chains exceeding threshold 4. Track chain execution against approved patterns

Validation: - Test with novel chain combinations - Review scoring algorithm quarterly - Validate approvals are enforced

Frequency: Continuous scoring, Monthly review

Implementation Guidance

Level 1 (Foundation) - 30-50 hours:

```
Week 1-2:
□ Inventory all agent tools and their capabilities
□ Document tool permissions per agent
□ Implement basic tool invocation logging
□ Define sensitive operations requiring approval
```

Level 2 (Structured) - 60-100 hours:

```
Week 3-5:
□ Deploy centralized tool gateway
□ Implement permission policy enforcement
□ Add tool usage anomaly detection
□ Create tool chain risk scoring
□ Establish approval workflow for high-risk operations
```

Level 3 (Optimized) - 100-150 hours:

```
Week 6-10:
□ Implement ML-based abuse detection
□ Automate permission right-sizing
□ Add predictive risk scoring for new chains
□ Achieve real-time enforcement with <100ms latency
□ Integrate with SIEM for correlation
```

Common Pitfalls

1. **Overly Permissive Defaults:** Starting with broad access and never tightening
2. **Tool Description Vulnerabilities:** Ambiguous descriptions that can be exploited
3. **Chain Blindness:** Only securing individual tools, not combinations
4. **Enforcement Gaps:** Having policies but not enforcing them technically

Tools & Resources

Open Source: - OPA (Open Policy Agent) - Policy enforcement - Falco - Runtime security monitoring - Tool permission frameworks (per platform)

Commercial: - Anthropic Claude tool use controls - OpenAI function calling guardrails - LangChain security extensions

---

ASI03: Identity & Privilege Abuse

Risk Description

High-privilege credentials are unintentionally reused or escalated across agents without proper scoping. Agents accumulate permissions beyond what's needed, creating excessive blast radius.

Attack Vectors: - Shared service account across multiple agents - Credential inheritance in agent spawning - Permission accumulation over time - Impersonation of higher-privilege agents

Real-World Impact: - Single compromised agent exposes all shared resources - Lateral movement through agent infrastructure - Privilege escalation to administrative access - Audit trail confusion with shared identities

HAIAMM Practice Mapping

Practice	Role	Domain
SR (Security Requirements)	Define identity requirements	Infrastructure
EH (Environment Hardening)	Implement identity controls	Infrastructure
ML (Monitoring & Logging)	Monitor identity usage	Software
SA (Secure Architecture)	Design identity architecture	Infrastructure

Measurable Outcomes

Outcome 1: Unique Agent Identity Rate

Target: 100% of production agents have unique, non-shared identities

Measurement Formula:

```
Uniqueness Rate = (Agents with Unique ID / Total Agents) × 100
Shared Credential Count = Count of credentials used by >1 agent
```

Data Sources: - Identity management system - Agent configuration database - Credential vault audit

Measurement Methodology: 1. Enumerate all agent identities in production 2. Map identities to credentials/service accounts 3. Identify shared credentials 4. Track remediation of shared identities

Validation: - Monthly credential audit - Automated shared credential detection - Cross-reference with deployment records

Frequency: Weekly scanning, Monthly reporting

Outcome 2: Privilege Scope Compliance

Target: >95% of agents scoped to minimum required resources

Measurement Formula:

```
Scope Compliance = (Properly Scoped Agents / Total Agents) × 100
Over-Privilege Score = (Granted Access - Required Access) / Required Access
```

Data Sources: - IAM policy analysis - Resource access logs - Agent function specifications

Measurement Methodology: 1. Define required resource access per agent function 2. Analyze actual IAM policies/permissions 3. Calculate over-privilege score per agent 4. Track scope reduction over time

Validation: - Unused permission detection (>30 days) - Quarterly access certification - Red team privilege escalation testing

Frequency: Monthly analysis, Quarterly certification

Outcome 3: Credential Rotation Compliance

Target: 100% of agent credentials rotated within policy period

Measurement Formula:

```
Rotation Compliance = (Rotated On Time / Total Credentials) × 100
Days Since Rotation = Current Date - Last Rotation Date
```

Data Sources: - Credential vault rotation logs - Secret management system - Compliance reporting

Measurement Methodology: 1. Define rotation policy (e.g., 90 days max) 2. Track credential age for all agents 3. Automate rotation where possible 4. Alert on approaching expiration

Validation: - Test rotated credentials still function - Verify old credentials are invalidated - Audit rotation logs for anomalies

Frequency: Daily monitoring, Weekly compliance check

Outcome 4: Identity Anomaly Detection

Target: >80% of identity abuse patterns detected within 5 minutes

Measurement Formula:

```
Detection Rate = (Detected Anomalies / Total Anomalies) × 100
MTTD = Average(Alert Time - Anomaly Start Time)
```

Data Sources: - Authentication logs - Identity analytics platform - SIEM alerts

Measurement Methodology: 1. Define identity anomaly indicators: - Impossible travel - Unusual access times - New resource access patterns - Failed authentication spikes 2. Implement real-time detection 3. Track detection accuracy and latency

Validation: - Monthly simulated anomaly injection - Correlation with incident forensics - Tune detection thresholds quarterly

Frequency: Continuous monitoring, Monthly metrics

Implementation Guidance

Level 1 (Foundation) - 40-60 hours:

```
Week 1-2:
□ Audit current agent identities and credentials
□ Eliminate shared credentials (create unique per agent)
□ Document required permissions per agent
□ Enable basic authentication logging
```

Level 2 (Structured) - 80-120 hours:

```
Week 3-6:
□ Implement workload identity (no static secrets)
□ Deploy automated credential rotation
□ Add privilege scope analysis
□ Implement identity anomaly detection
□ Create credential usage dashboard
```

Level 3 (Optimized) - 120-180 hours:

```
Week 7-12:
□ Implement just-in-time privilege elevation
□ Deploy ML-based identity analytics
□ Automate over-privilege remediation
□ Achieve zero standing privileges for high-risk agents
□ Integrate with zero trust architecture
```

Common Pitfalls

1. **Shared Service Accounts:** Convenient but creates massive blast radius

2. **Static Long-Lived Credentials:** Never rotating API keys or tokens
3. **Over-Provisioning:** Granting broad access “just in case”
4. **Audit Trail Gaps:** Unable to attribute actions to specific agents

Tools & Resources

Open Source: - SPIFFE/SPIRE - Workload identity - Vault - Secret management - AWS IAM Access Analyzer

Commercial: - CyberArk Privileged Access - HashiCorp Vault Enterprise - Azure Managed Identity

---

ASI04: Agentic Supply Chain Vulnerabilities

Risk Description

Compromised tools, plugins, templates, and MCP servers fetched at runtime introduce malicious code or behavior into agent systems. The dynamic nature of agent tool loading expands the attack surface.

Attack Vectors: - Compromised tool packages in public registries - Malicious MCP server implementations - Typosquatting on popular tool names - Compromised dependencies in tool chains

Real-World Impact: - Backdoor installation through legitimate-looking tools - Data exfiltration via compromised dependencies - Credential theft through malicious plugins - Supply chain attacks affecting multiple organizations

HAIAMM Practice Mapping

Practice	Role	Domain
TA (Threat Assessment)	Assess supply chain risks	Vendors
SA (Secure Architecture)	Design secure tool loading	Software

Practice	Role	Domain
IM (Issue Management)	Track supply chain vulnerabilities	Vendors
SR (Security Requirements)	Define supply chain requirements	Vendors

Measurable Outcomes

Outcome 1: Tool Provenance Verification

Target: 100% of production tools have verified provenance

Measurement Formula:

```
Verification Rate = (Verified Tools / Total Tools) × 100
Provenance Coverage = (Tools with Known Origin / Total Tools) × 100
```

Data Sources: - Tool registry with signatures - SBOM (Software Bill of Materials) - Verification logs

Measurement Methodology: 1. Maintain inventory of all agent tools 2. Require cryptographic signatures for tools 3. Verify signatures before loading 4. Track verification failures and blocks

Validation: - Test with unsigned tools (must be blocked) - Verify signature chain to trusted root - Audit provenance claims quarterly

Frequency: Per-load verification, Weekly reporting

Outcome 2: Dependency Vulnerability Coverage

Target: >95% of tool dependencies scanned, <24hr remediation for critical

Measurement Formula:

```
Scan Coverage = (Scanned Dependencies / Total Dependencies) × 100
MTTR (Critical) = Average(Remediation Time for Critical Vulnerabilities)
```

Data Sources: - Dependency scanning tools - Vulnerability database - Remediation tracking system

Measurement Methodology: 1. Implement automated dependency scanning in CI/CD 2. Scan runtime-loaded tools before deployment 3. Track time from detection to remediation 4. Categorize by severity (critical, high, medium, low)

Validation: - Test scanner detection with known vulnerabilities - Verify remediation actually removes vulnerability - Audit false negative rate quarterly

Frequency: Daily scanning, Per-vulnerability tracking

Outcome 3: Supply Chain Incident Response Time

Target: <4 hours from supply chain advisory to assessment complete

Measurement Formula:

```
Response Time = Assessment Complete Timestamp - Advisory Published Timestamp
```

Data Sources: - Security advisory feeds - Assessment records - Incident timeline

Measurement Methodology: 1. Subscribe to relevant security advisories (NVD, GitHub, vendor) 2. Implement automated matching to your tool inventory 3. Track time to complete impact assessment 4. Measure time to implement mitigations

Validation: - Drill with simulated advisories quarterly - Compare actual response times to targets - Review assessment completeness

Frequency: Per-incident, Monthly aggregation

Outcome 4: Approved Tool Registry Compliance

Target: 100% of production agent tools from approved registry

Measurement Formula:

```
Compliance Rate = (Tools from Approved Registry / Total Tools) × 100
Shadow Tool Count = Tools loaded from unapproved sources
```

Data Sources: - Approved tool registry - Runtime tool loading logs - Network egress monitoring

Measurement Methodology: 1. Establish approved tool registry with security review process 2. Enforce registry-only loading in production 3. Monitor for shadow tool loading attempts 4. Track approval queue and times

Validation: - Attempt to load unapproved tool (must fail) - Audit registry contents quarterly - Review approval decisions for consistency

Frequency: Continuous enforcement, Weekly reporting

Implementation Guidance

Level 1 (Foundation) - 30-50 hours:

```
Week 1-2:
□ Inventory all agent tools and dependencies
□ Create approved tool registry
□ Implement basic signature verification
□ Subscribe to security advisory feeds
```

Level 2 (Structured) - 60-100 hours:

```
Week 3-5:
□ Generate SBOM for all tools
□ Implement automated vulnerability scanning
□ Deploy registry-only enforcement
□ Create supply chain incident response playbook
□ Establish vendor security assessment process
```

Level 3 (Optimized) - 100-150 hours:

```
Week 6-10:
□ Implement real-time supply chain monitoring
□ Deploy behavioral analysis for loaded tools
□ Automate vulnerability remediation where safe
```

```
□ Achieve <4hr response target
□ Integrate with industry threat intelligence
```

Common Pitfalls

1. **Trust by Default:** Loading tools without verification
2. **Stale SBOM:** Not updating dependency inventory
3. **Slow Response:** Taking days to assess supply chain alerts
4. **Incomplete Coverage:** Only scanning some tools or dependencies

Tools & Resources

Open Source: - Syft - SBOM generation - Grype - Vulnerability scanning - Sigstore - Code signing

Commercial: - Snyk - Socket.dev - Sonatype Nexus

ASI05: Unexpected Code Execution (RCE)

Risk Description

Agents generate or execute code unsafely without proper validation, leading to remote code execution vulnerabilities. This is especially dangerous when agents have code generation capabilities.

Attack Vectors: - Prompt injection leading to malicious code generation - Unsafe execution of agent-generated code - Code injection through data sources - Exploitation of code execution sandbox escapes

Real-World Impact: - Full system compromise through code execution - Data exfiltration via generated scripts - Persistence mechanisms installed by agent - Lateral movement through agent infrastructure

HAIAMM Practice Mapping

Practice	Role	Domain
SR (Security Requirements)	Define execution boundaries	Software
ST (Security Testing)	Test for RCE vulnerabilities	Software
EH (Environment Hardening)	Secure execution environment	Infrastructure
IR (Implementation Review)	Review generated code	Software

Measurable Outcomes

Outcome 1: Sandbox Escape Prevention

Target: 0 successful sandbox escapes in production

Measurement Formula:

```
Escape Rate = (Successful Escapes / Total Escape Attempts) × 100
Escape Attempt Count = Detected attempts to break sandbox
```

Data Sources: - Sandbox monitoring logs - Security incident reports - Red team results

Measurement Methodology: 1. Deploy code execution sandboxes for all agent code execution 2. Monitor for sandbox escape indicators: - File system access outside allowed paths - Network connections to unauthorized hosts - Process spawning outside allowed list - Resource limit violations 3. Track all escape attempts and outcomes

Validation: - Quarterly sandbox penetration testing - Test with known escape techniques - Verify sandbox configuration drift detection

Frequency: Continuous monitoring, Monthly reporting

Outcome 2: Code Validation Coverage

Target: 100% of agent-generated code validated before execution

Measurement Formula:

```
Validation Rate = (Validated Executions / Total Executions) × 100
```

Data Sources: - Code validation gateway logs - Execution audit trail - Validation failure logs

Measurement Methodology: 1. Implement mandatory code validation pipeline 2. Define validation rules: - Static analysis (syntax, security patterns) - Allowlisted operations only - Resource limit verification - Dangerous function detection 3. Track validation results and execution decisions

Validation: - Test with malicious code samples - Verify bypass attempts are blocked - Audit validation rule effectiveness

Frequency: Per-execution, Daily aggregation

Outcome 3: Execution Blast Radius Containment

Target: All code execution confined to dedicated, ephemeral environments

Measurement Formula:

```
Containment Rate = (Contained Executions / Total Executions) × 100
Shared Environment Count = Executions in non-isolated environments
```

Data Sources: - Container/VM provisioning logs - Execution environment inventory - Network segmentation logs

Measurement Methodology: 1. Define execution environment requirements: - Ephemeral (destroyed after use) - Isolated (no shared resources) - Limited (restricted capabilities) 2. Track execution environment allocation 3. Monitor for container/VM reuse violations

Validation: - Verify environment destruction after execution - Test cross-execution data leakage - Audit network isolation quarterly

Frequency: Continuous monitoring, Weekly reporting

Outcome 4: Dangerous Operation Detection

Target: >95% of dangerous operations detected and blocked

Measurement Formula:

```
Detection Rate = (Blocked Dangerous Ops / Total Dangerous Op Attempts) × 100
```

Data Sources: - Code analysis results - Execution monitoring - Blocked operation logs

Measurement Methodology: 1. Define dangerous operations: - System calls (exec, fork, network bind) - File operations (write to system paths) - Resource operations (memory allocation, process creation) 2. Implement detection in validation and runtime 3. Track detection accuracy and bypass attempts

Validation: - Test with known dangerous patterns - Red team with novel techniques quarterly - Tune detection based on false positive/negative rates

Frequency: Per-operation, Daily reporting

Implementation Guidance

Level 1 (Foundation) - 50-70 hours:

```
Week 1-3:
□ Inventory all agent code execution capabilities
□ Implement basic sandboxing (Docker, gVisor)
□ Add static analysis for generated code
□ Block known dangerous functions
```

Level 2 (Structured) - 100-150 hours:

```
Week 4-8:
□ Deploy hardened sandbox (Firecracker, seccomp)
□ Implement comprehensive code validation pipeline
□ Add runtime monitoring for anomalies
```

```
□ Create ephemeral execution environments
□ Establish code execution approval workflow
```

Level 3 (Optimized) - 150-220 hours:

```
Week 9-16:
□ Implement ML-based code behavior prediction
□ Deploy formal verification for critical paths
□ Achieve zero escape rate target
□ Automate sandbox hardening updates
□ Integrate with vulnerability management
```

Common Pitfalls

1. **Weak Sandboxes:** Using containers without additional hardening
2. **Validation Gaps:** Static analysis only, no runtime protection
3. **Persistent Environments:** Reusing execution environments
4. **Escape Complacency:** Assuming sandboxes are unbreakable

Tools & Resources

Open Source: - gVisor - Application kernel sandboxing - Firecracker - MicroVM isolation - Semgrep - Static analysis

Commercial: - E2B Sandbox - Modal - AWS Lambda with security controls

---

ASI06: Memory & Context Poisoning

Risk Description

Attackers poison memory systems and RAG databases to influence future agent decisions. Persistent context creates an attack surface that affects all future interactions.

Attack Vectors: - Injecting malicious content into vector stores - Poisoning conversation history databases - Manipulating agent memory through crafted interactions - Corrupting RAG retrieval results

Real-World Impact: - Agent provides incorrect information persistently - Backdoor activation through poisoned context - Data exfiltration through manipulated memory - Trust erosion through unreliable agent behavior

HAIAMM Practice Mapping

Practice	Role	Domain
DR (Design Review)	Review memory architecture	Data
ST (Security Testing)	Test memory integrity	Software
EH (Environment Hardening)	Secure memory systems	Data
ML (Monitoring & Logging)	Monitor memory changes	Data

Measurable Outcomes

Outcome 1: Memory Integrity Verification

Target: 100% of memory retrievals verified for integrity

Measurement Formula:

```
Verification Rate = (Verified Retrievals / Total Retrievals) × 100
Integrity Failure Rate = (Failed Verifications / Total Verifications) × 100
```

Data Sources: - Memory system access logs - Integrity check results - Retrieval audit trail

Measurement Methodology: 1. Implement cryptographic integrity for stored memories 2. Verify integrity on retrieval 3. Track verification failures and sources 4. Investigate and remediate integrity issues

Validation: - Inject tampered memory entries (must be detected) - Test with various tampering methods - Verify restoration from backups

Frequency: Per-retrieval, Daily aggregation

Outcome 2: Memory Source Attribution

Target: >95% of memory entries have verified source attribution

Measurement Formula:

```
Attribution Rate = (Attributed Entries / Total Entries) × 100
```

Data Sources: - Memory provenance logs - Source verification records - Attribution audit trail

Measurement Methodology: 1. Record source for all memory entries (user, system, agent, external) 2. Verify source authenticity (signatures, timestamps) 3. Track unattributed or questionable entries 4. Apply trust scores based on source

Validation: - Test source spoofing attempts - Audit attribution accuracy - Verify trust scores affect retrieval priority

Frequency: Per-entry, Weekly reporting

Outcome 3: Context Poisoning Detection

Target: >80% of poisoning attempts detected within 24 hours

Measurement Formula:

```
Detection Rate = (Detected Attempts / Total Attempts) × 100
Time to Detect = Alert Timestamp - Poisoning Timestamp
```

Data Sources: - Memory analysis logs - Anomaly detection alerts - Incident reports

Measurement Methodology: 1. Define poisoning indicators: - Sudden content pattern changes - Unusual source patterns - Embedding space anomalies - Contradictory information injection 2. Implement detection mechanisms 3. Track detection accuracy and timing

Validation: - Monthly poisoning simulation exercises - Compare automated detection with manual review - Test novel poisoning techniques

Frequency: Continuous monitoring, Weekly metrics

Outcome 4: Memory Hygiene Compliance

Target: 100% of memory systems meet hygiene standards

Measurement Formula:

```
Hygiene Score = (Met Standards / Total Standards) × 100
```

Data Sources: - Memory system configuration - Retention policy compliance - Access control audit

Measurement Methodology: 1. Define hygiene standards: - Retention limits (max age) - Access controls (who can write) - Validation requirements (what can be stored) - Backup and recovery (data protection) 2. Audit compliance monthly 3. Track remediation of gaps

Validation: - Test retention enforcement - Verify access controls - Test backup and recovery

Frequency: Monthly audit, Quarterly review

Implementation Guidance

Level 1 (Foundation) - 30-50 hours:

```
Week 1-2:
□ Inventory all agent memory and context systems
□ Implement basic access controls on memory stores
□ Add source attribution for new entries
□ Enable memory change logging
```

Level 2 (Structured) - 60-100 hours:

```
Week 3-5:
□ Implement cryptographic integrity verification
□ Deploy memory content validation
□ Add poisoning detection (anomaly-based)
□ Create memory backup and recovery procedures
□ Establish memory retention policies
```

Level 3 (Optimized) - 100-150 hours:

```
Week 6-10:
□ Implement ML-based poisoning detection
□ Deploy real-time memory health monitoring
□ Automate memory hygiene enforcement
□ Achieve >80% detection rate target
□ Integrate with incident response
```

Common Pitfalls

1. **Trust All Content:** Treating all retrieved content as trustworthy
2. **No Attribution:** Unable to determine where memory entries came from
3. **Permanent Memory:** Never expiring or validating old entries
4. **Detection Gaps:** Only monitoring writes, not detecting subtle poisoning

Tools & Resources

Open Source: - Vector databases with integrity features (Pinecone, Weaviate) - Data validation frameworks - Anomaly detection libraries

Commercial: - Memory management platforms with security - RAG security solutions

---

ASI07: Insecure Inter-Agent Communication

Risk Description

Multi-agent message exchanges lack authentication, encryption, or semantic validation. Attackers can intercept, modify, or inject messages between agents.

Attack Vectors: - Man-in-the-middle on agent communications - Message spoofing between agents - Replay attacks using captured messages - Semantic injection through message manipulation

Real-World Impact: - Agents acting on forged instructions - Sensitive data exposed in transit - Coordinated agent behavior disrupted - Trust relationships between agents compromised

HAIAMM Practice Mapping

Practice	Role	Domain
SA (Secure Architecture)	Design secure communication	Software
SR (Security Requirements)	Define communication requirements	Infrastructure
ML (Monitoring & Logging)	Monitor inter-agent traffic	Software
EH (Environment Hardening)	Secure communication channels	Infrastructure

Measurable Outcomes

Outcome 1: Communication Encryption Coverage

Target: 100% of inter-agent communications encrypted in transit

Measurement Formula:

```
Encryption Rate = (Encrypted Communications / Total Communications) × 100
```

Data Sources: - Network monitoring logs - TLS/mTLS certificate inventory - Communication protocol audit

Measurement Methodology: 1. Inventory all inter-agent communication channels 2. Verify encryption (TLS 1.3+, mTLS preferred) 3. Monitor for unencrypted traffic 4. Track encryption version and cipher strength

Validation: - Network scanning for unencrypted traffic - Certificate validity monitoring - Quarterly cipher strength review

Frequency: Continuous monitoring, Weekly reporting

Outcome 2: Message Authentication Rate

Target: 100% of inter-agent messages authenticated

Measurement Formula:

```
Authentication Rate = (Authenticated Messages / Total Messages) × 100
Spoofing Attempt Count = Detected unauthenticated message attempts
```

Data Sources: - Message authentication logs - Signature verification results - Spoofing detection alerts

Measurement Methodology: 1. Implement message signing for all agent communications 2. Verify signatures on receipt 3. Reject unauthenticated messages 4. Track authentication failures and spoofing attempts

Validation: - Test with spoofed messages (must be rejected) - Verify key management processes - Audit signature verification implementation

Frequency: Per-message, Daily aggregation

Outcome 3: Communication Anomaly Detection

Target: >85% of communication anomalies detected within 5 minutes

Measurement Formula:

```
Detection Rate = (Detected Anomalies / Total Anomalies) × 100
MTTD = Average(Detection Time - Anomaly Start)
```

Data Sources: - Communication monitoring platform - Anomaly detection alerts - Incident reports

Measurement Methodology: 1. Baseline normal inter-agent communication patterns 2. Define anomalies (unusual volume, new recipients, pattern changes) 3. Implement real-time detection 4. Track detection accuracy and timing

Validation: - Monthly simulated anomaly injection - Compare with forensic analysis - Tune detection thresholds

Frequency: Continuous monitoring, Weekly metrics

Implementation Guidance

Level 1 (Foundation) - 30-40 hours:

```
Week 1-2:
□ Inventory all inter-agent communication channels
□ Implement TLS for all agent communications
□ Add basic message logging
□ Document communication architecture
```

Level 2 (Structured) - 60-80 hours:

```
Week 3-4:
□ Deploy mTLS with certificate management
□ Implement message signing and verification
□ Add replay attack prevention (nonces, timestamps)
□ Create communication monitoring dashboard
```

Level 3 (Optimized) - 80-120 hours:

```
Week 5-8:
□ Implement semantic validation for messages
□ Deploy ML-based anomaly detection
□ Automate certificate rotation
□ Achieve real-time detection target
```

Common Pitfalls

1. **Plain Text Protocols:** Using unencrypted communication channels
2. **Trust by Channel:** Assuming all messages on a channel are legitimate
3. **Missing Replay Protection:** No nonces or timestamp validation
4. **Semantic Blindness:** Verifying signatures but not message content sense

ASI08: Cascading Failures

Risk Description

Small errors propagate across planning, execution, and downstream systems. Agent architectures create complex dependencies where failures amplify.

Attack Vectors: - Triggering initial errors that cascade - Exploiting retry logic to amplify impact - Resource exhaustion through cascading requests - Data corruption propagating through pipelines

HAIAMM Practice Mapping

Practice	Role	Domain
SA (Secure Architecture)	Design resilient architecture	Processes
IM (Issue Management)	Manage cascading incidents	Infrastructure
ML (Monitoring & Logging)	Detect cascade initiation	Processes
EH (Environment Hardening)	Implement circuit breakers	Infrastructure

Measurable Outcomes

Outcome 1: Circuit Breaker Coverage

Target: 100% of agent dependencies protected by circuit breakers

Measurement Formula:

```
Coverage = (Protected Dependencies / Total Dependencies) × 100
Trip Rate = (Circuit Trips / Total Calls) × 100
```

Measurement Methodology: 1. Inventory all agent dependencies (APIs, services, data stores) 2. Implement circuit breakers for each 3. Configure appropriate thresholds 4. Track trips and recovery

---

Outcome 2: Cascade Detection Time

Target: Cascade initiation detected within 2 minutes

Measurement Formula:

```
MTTD = Alert Timestamp - First Failure Timestamp
```

Implementation Guidance

Level 1: Implement basic circuit breakers and failure logging **Level 2:** Deploy distributed tracing and cascade detection **Level 3:** Implement predictive failure detection and auto-remediation

---

ASI09: Human-Agent Trust Exploitation

Risk Description

Users over-trust agent recommendations, allowing influence or data extraction. Agents manipulate human decision-making through social engineering patterns.

HAIAMM Practice Mapping

Practice	Role	Domain
EG (Education & Guidance)	Train users on AI limitations	Processes
PC (Policy & Compliance)		Processes

Practice	Role	Domain
	Define trust boundaries	
ML (Monitoring & Logging)	Monitor high-impact decisions	Endpoints

Measurable Outcomes

Outcome 1: High-Impact Decision Review Rate

Target: 100% of agent-influenced high-impact decisions reviewed by human

Measurement Formula:

```
Review Rate = (Reviewed Decisions / Total High-Impact Decisions) × 100
```

Outcome 2: User AI Literacy Score

Target: >80% of users pass AI limitation awareness assessment

---

Implementation Guidance

Level 1: Define high-impact decisions, implement review workflows **Level 2:** Deploy user training program, add friction for risky decisions **Level 3:** Implement adaptive trust calibration, measure decision quality

---

ASI10: Rogue Agents

Risk Description

Compromised or misaligned agents act harmfully while appearing legitimate. Detection is difficult because behavior appears normal to existing controls.

HAIAMM Practice Mapping

Practice	Role	Domain
TA (Threat Assessment)	Model rogue agent threats	Software
ML (Monitoring & Logging)	Detect behavioral anomalies	Processes
IM (Issue Management)	Respond to rogue agent incidents	Software

Measurable Outcomes

Outcome 1: Behavioral Baseline Coverage

Target: 100% of agents have established behavioral baselines

Outcome 2: Rogue Detection Time

Target: Rogue behavior detected within 1 hour of onset

Outcome 3: Containment Time

Target: <15 minutes from detection to agent isolation

Implementation Guidance

Level 1: Establish behavioral baselines, implement isolation procedures **Level 2:** Deploy anomaly detection, create response playbooks **Level 3:** Implement ML-based detection, automate containment

---

Document Information

Field	Value
Document	OWASP Top 10 for Agentic Applications - HAIAMM Mapping
HAIAMM Version	2.2
OWASP Agentic Version	2026
Last Updated	January 2026

Related Documents: - Quick Start Guide - Top 10 LLM Risks - Risk-Practice Matrix - Tools & Resources

Back to Index

HAIAMM Logo

HAIAMM Risk-Practice Matrix

Which practices address which risks

Back to Index | Agentic Risks | LLM Risks

How to Use This Matrix

This matrix shows which HAIAMM practices address each security risk. Use it to:

1. **Prioritize practices** based on your risk profile
2. **Identify gaps** in your current security posture
3. **Plan assessments** for specific risk areas
4. **Allocate resources** to highest-impact practices

Legend: - **P** = Primary practice (critical for addressing this risk) - **S** = Secondary practice (supporting role) - **-** = Not directly relevant

---

OWASP Top 10 for Agentic Applications × HAIAMM Practices

Risk	SM	PC	EG	TA	SR	SA	DR	IR	ST	IM	EH	ML
ASI01 Agent Goal Hijack	S	S	-	**P**	**P**	S	S	-	**P**	S	S	**P**
ASI02 Tool Misuse	S	**P**	-	S	S	**P**	S	-	S	S	S	**P**
	S	S	-	S	**P**	S	-	-	S	S	**P**	**P**

Risk	SM	PC	EG	TA	SR	SA	DR	IR	ST	IM	EH	ML
ASI03 Identity Abuse												
ASI04 Supply Chain	S	S	-	**P**	S	**P**	S	S	S	**P**	S	S
ASI05 Code Execution	-	S	-	S	**P**	S	S	**P**	**P**	S	**P**	S
ASI06 Memory Poisoning	S	-	-	S	S	S	**P**	-	**P**	S	**P**	S
ASI07 Inter-Agent Comm	S	S	-	S	**P**	**P**	S	-	S	S	S	**P**
ASI08 Cascading Failures	S	S	-	S	S	**P**	S	-	S	**P**	S	**P**
ASI09 Trust Exploitation	S	**P**	**P**	S	S	S	S	-	S	S	-	**P**
ASI10 Rogue Agents	S	S	-	**P**	S	S	S	-	S	**P**	S	**P**

Agentic Risk Priority by Practice

Practice	Primary For	Secondary For	Priority Score
ML Monitoring & Logging	5 risks	5 risks	**High**
ST Security Testing	3 risks	5 risks	**High**

Practice	Primary For	Secondary For	Priority Score
SR Security Requirements	3 risks	5 risks	**High**
TA Threat Assessment	3 risks	5 risks	**High**
SA Secure Architecture	3 risks	5 risks	**High**
EH Environment Hardening	3 risks	5 risks	**High**
PC Policy & Compliance	2 risks	6 risks	Medium
IM Issue Management	2 risks	6 risks	Medium
DR Design Review	1 risk	5 risks	Medium
IR Implementation Review	1 risk	2 risks	Low
EG Education & Guidance	1 risk	1 risk	Low
SM Strategy & Metrics	0 risks	9 risks	Supporting

OWASP Top 10 for LLM Applications × HAIAMM Practices

Risk	SM	PC	EG	TA	SR	SA	DR	IR	ST	IM	EH	ML
LLM01 Prompt Injection	S	S	-	S	**P**	S	S	S	**P**	S	**P**	S
LLM02 Info Disclosure	S	**P**	-	S	S	S	**P**	S	S	S	S	**P**
LLM03 Supply Chain	S	S	-	S	S	**P**	S	S	S	**P**	**P**	S
LLM04 Data Poisoning	S	S	-	**P**	S	S	**P**	-	**P**	S	S	S

Risk	SM	PC	EG	TA	SR	SA	DR	IR	ST	IM	EH	ML
LLM05 Output Handling	-	S	-	S	**P**	S	S	**P**	**P**	S	S	S
LLM06 Excessive Agency	S	**P**	S	S	S	**P**	S	-	S	S	S	**P**
LLM07 Prompt Leakage	S	S	-	S	S	S	S	-	**P**	S	**P**	**P**
LLM08 Vector Weaknesses	S	S	-	S	S	**P**	S	-	**P**	S	**P**	S
LLM09 Misinformation	S	S	S	S	S	S	**P**	-	**P**	S	S	**P**
LLM10 Unbounded Consumption	S	S	-	S	**P**	S	S	-	S	S	**P**	**P**

LLM Risk Priority by Practice

Practice	Primary For	Secondary For	Priority Score
ST Security Testing	5 risks	5 risks	**High**
EH Environment Hardening	4 risks	4 risks	**High**
ML Monitoring & Logging	4 risks	6 risks	**High**
SR Security Requirements	3 risks	5 risks	**High**
SA Secure Architecture	3 risks	5 risks	**High**
DR Design Review	3 risks	5 risks	**High**
PC Policy & Compliance	2 risks	7 risks	Medium
IM Issue Management	1 risk	8 risks	Medium

Practice	Primary For	Secondary For	Priority Score
TA Threat Assessment	1 risk	7 risks	Medium
IR Implementation Review	1 risk	3 risks	Low
EG Education & Guidance	0 risks	2 risks	Low
SM Strategy & Metrics	0 risks	9 risks	Supporting

Combined Risk Coverage Matrix

High-Priority Practices (Address 6+ Risks as Primary)

Practice	Total Primary	Agentic Primary	LLM Primary
ST Security Testing	8	3	5
ML Monitoring & Logging	9	5	4
EH Environment Hardening	7	3	4
SR Security Requirements	6	3	3
SA Secure Architecture	6	3	3

Recommended Assessment Order

Based on risk coverage, assess practices in this order:

1. **Tier 1: Critical Coverage**
    - ST (Security Testing) - Addresses 8 primary risks
    - ML (Monitoring & Logging) - Addresses 9 primary risks
    - EH (Environment Hardening) - Addresses 7 primary risks
2. **Tier 2: High Coverage**
    - SR (Security Requirements) - Addresses 6 primary risks
    - SA (Secure Architecture) - Addresses 6 primary risks

    - TA (Threat Assessment) - Addresses 4 primary risks
3. **Tier 3: Supporting Coverage**
    - PC (Policy & Compliance)
    - DR (Design Review)
    - IM (Issue Management)
4. **Tier 4: Foundational**
    - SM (Strategy & Metrics)
    - EG (Education & Guidance)
    - IR (Implementation Review)

Risk-to-Practice Quick Reference

By Risk: What Practices to Implement

If you're concerned about Prompt Injection (LLM01): - Primary: SR, ST, EH - Focus: Input validation, injection testing, environment controls

If you're concerned about Agent Goal Hijack (ASI01): - Primary: TA, SR, ST, ML - Focus: Threat modeling, goal integrity requirements, behavioral monitoring

If you're concerned about Tool Misuse (ASI02): - Primary: SA, PC, ML - Focus: Tool access architecture, usage policies, invocation monitoring

If you're concerned about Data Poisoning (LLM04): - Primary: TA, DR, ST - Focus: Poisoning threat assessment, data review, integrity testing

If you're concerned about Rogue Agents (ASI10): - Primary: TA, ML, IM - Focus: Rogue threat modeling, behavioral detection, incident response

---

Domain Focus by Risk

Software Domain Risks

Most risks require Software domain practices: - All LLM risks touch Software - 8/10 Agentic risks are Software-primary

Infrastructure Domain Risks

Critical for: - ASI03 (Identity Abuse) - ASI05 (Code Execution) - ASI07 (Inter-Agent Communication) - ASI08 (Cascading Failures) - LLM10 (Unbounded Consumption)

Data Domain Risks

Critical for: - ASI06 (Memory Poisoning) - LLM02 (Information Disclosure) - LLM04 (Data Poisoning) - LLM08 (Vector Weaknesses)

Processes Domain Risks

Critical for: - ASI08 (Cascading Failures) - ASI09 (Trust Exploitation) - ASI10 (Rogue Agents) - LLM06 (Excessive Agency)

Vendors Domain Risks

Critical for: - ASI04 (Supply Chain) - LLM03 (Supply Chain)

Endpoints Domain Risks

Critical for: - ASI02 (Tool Misuse) - ASI09 (Trust Exploitation) - LLM01 (Prompt Injection) - LLM07 (Prompt Leakage)

Using This Matrix for Gap Analysis

Step 1: Identify Your Primary Risks

Select the top 5 risks most relevant to your environment: - [ ] Risk 1: _____ - [ ] Risk 2: _____ - [ ] Risk 3: _____ - [ ] Risk 4: _____ - [ ] Risk 5: _____

Step 2: Map Primary Practices

For each risk, list the primary practices: | Risk | Primary Practices | |—-|———-| | | | | | | |

Step 3: Identify Coverage Gaps

Check which practices appear most frequently: | Practice | Count | Current Maturity | Gap | |———-|——-|———-|—–| | | | | | | | | | | |

Step 4: Prioritize Assessment

Focus on practices with: 1. Highest risk coverage 2. Lowest current maturity 3. Largest gap to target

---

Document Information

Field	Value
Document	Risk-Practice Matrix
HAIAMM Version	2.2
Last Updated	January 2026

Related Documents: - Top 10 Agentic Risks - Top 10 LLM Risks - Assessment Checklist - Maturity Roadmap

Back to Index

HAIAMM Logo

HAIAMM Maturity Roadmap

Level 1 → 2 → 3 progression guide

Back to Index | Assessment Checklist | First 30 Days

Overview

This roadmap provides guidance for progressing through HAIAMM maturity levels. Each level builds on the previous, creating a sustainable security improvement path.

Maturity Level Characteristics

Level 1: Foundation

Characteristics: - Basic awareness of AI security issues - Ad-hoc security activities - Individual efforts, not organizational - Reactive security posture

Typical State: - AI systems exist but security is inconsistent - Some controls implemented, mostly basic - Limited visibility into AI security posture - No formal metrics or tracking

Effort to Achieve: 1-3 months **Typical Investment:** 40-80 hours per practice

---

Level 2: Structured

Characteristics: - Documented security requirements - Consistent processes across projects - Defined roles and responsibilities - Proactive security activities

Typical State: - Security requirements defined and tracked - Processes documented and followed - Regular testing and monitoring - Metrics collected (but may not drive decisions)

Effort to Achieve: 3-6 months (from Level 1) **Typical Investment:** 80-160 hours per practice

---

Level 3: Optimized

Characteristics: - Quantitative management - Continuous improvement - Automated controls where possible - Industry leadership

Typical State: - Metrics drive security decisions - Automation reduces manual effort - Benchmarked against industry - Proactive threat anticipation

Effort to Achieve: 6-12 months (from Level 2) **Typical Investment:** 160-240 hours per practice

---

Level 1 → Level 2 Transition

Transition Criteria

You're ready for Level 2 when: - [ ] All AI systems inventoried and documented - [ ] Basic security controls in place (validation, logging) - [ ] Ownership assigned for all AI systems - [ ] At least one security test completed per system - [ ] Incident response procedure exists

Key Activities by Practice

Practice	Level 1 State	Level 2 Target	Key Activities
SM	Ad-hoc strategy	Documented strategy with KPIs	Define metrics, create dashboard
PC	Basic policies	Enforced policies	Implement technical enforcement
EG	Informal training	Structured program	Create role-based training
TA		Formal threat models	

Practice	Level 1 State	Level 2 Target	Key Activities
	Basic threat awareness		Adopt structured methodology
SR	Informal requirements	Documented, testable	Create requirements spec
SA	Basic architecture	Security patterns	Document and review architecture
DR	Ad-hoc reviews	Checklist-based	Create review process
IR	Basic implementation review	Security-focused	Add security linting
ST	Manual testing	Test suite (50+ cases)	Build automated tests
IM	Reactive response	SLA-tracked	Implement ticketing, SLAs
EH	Basic hardening	Standards-based	Document hardening standards
ML	Basic logging	Alerting & correlation	Deploy SIEM integration

Recommended Transition Order

Phase 1: Visibility (Month 1-2) 1. SM - Establish metrics framework 2. ML - Implement comprehensive logging 3. IM - Set up issue tracking

Phase 2: Controls (Month 2-4) 4. ST - Build security test suite 5. EH - Document and apply hardening 6. SR - Formalize security requirements

Phase 3: Process (Month 4-6) 7. TA - Adopt threat modeling methodology 8. DR - Implement design review process 9. SA - Document architecture patterns

Phase 4: Culture (Month 5-6) 10. PC - Enforce policies technically 11. EG - Launch training program 12. IR - Add security-focused implementation review

Level 1→2 Investment Estimate

Category	Hours	Cost (Internal)	Cost (External)
Process Documentation	80-120	$16K-$24K	$30K-$50K
Technical Implementation	120-200	$24K-$40K	$50K-$80K
Training Development	40-60	$8K-$12K	$20K-$30K
Tool Deployment	60-100	$12K-$20K	$25K-$40K
Total	**300-480**	**$60K-$96K**	**$125K-$200K**

Based on $200/hr loaded internal cost, consultant rates vary

Level 2 → Level 3 Transition

Transition Criteria

You're ready for Level 3 when: - [ ] All Level 2 practices consistently executed - [ ] Metrics collected for at least 6 months - [ ] No critical vulnerabilities open >30 days - [ ] Security testing automated in CI/CD - [ ] Incident response tested via tabletop

Key Activities by Practice

Practice	Level 2 State	Level 3 Target	Key Activities
SM	KPIs tracked	ROI demonstrated	Implement benchmarking
PC	Policies enforced	Automated compliance	Deploy continuous compliance
EG	Structured training	Champions program	Create career path
TA	Formal threat models	Threat intelligence	Integrate threat feeds

Practice	Level 2 State	Level 3 Target	Key Activities
SR	Documented reqs	Auto-verified	CI/CD integration
SA	Patterns documented	Continuously validated	Architecture scanning
DR	Checklist reviews	Automated checks	Deploy design scanning
IR	Security linting	ML-based analysis	Advanced SAST
ST	Automated suite	Red team + ML testing	Continuous testing
IM	SLA-tracked	Root cause analysis	Trend analysis
EH	Standards-based	Drift detection	Auto-remediation
ML	Alerting active	ML anomaly detection	Predictive monitoring

Recommended Transition Order

Phase 1: Automation (Month 1-3) 1. ST - Implement continuous security testing 2. EH - Deploy drift detection and auto-remediation 3. SR - Integrate requirements verification in CI/CD

Phase 2: Intelligence (Month 3-6) 4. ML - Deploy ML-based anomaly detection 5. TA - Integrate threat intelligence feeds 6. IM - Implement trend analysis and prediction

Phase 3: Optimization (Month 6-9) 7. SM - Achieve ROI demonstration capability 8. PC - Deploy continuous compliance monitoring 9. SA - Implement architecture drift detection

Phase 4: Leadership (Month 9-12) 10. DR - Automate design security verification 11. IR - Deploy advanced static analysis 12. EG - Launch security champions program

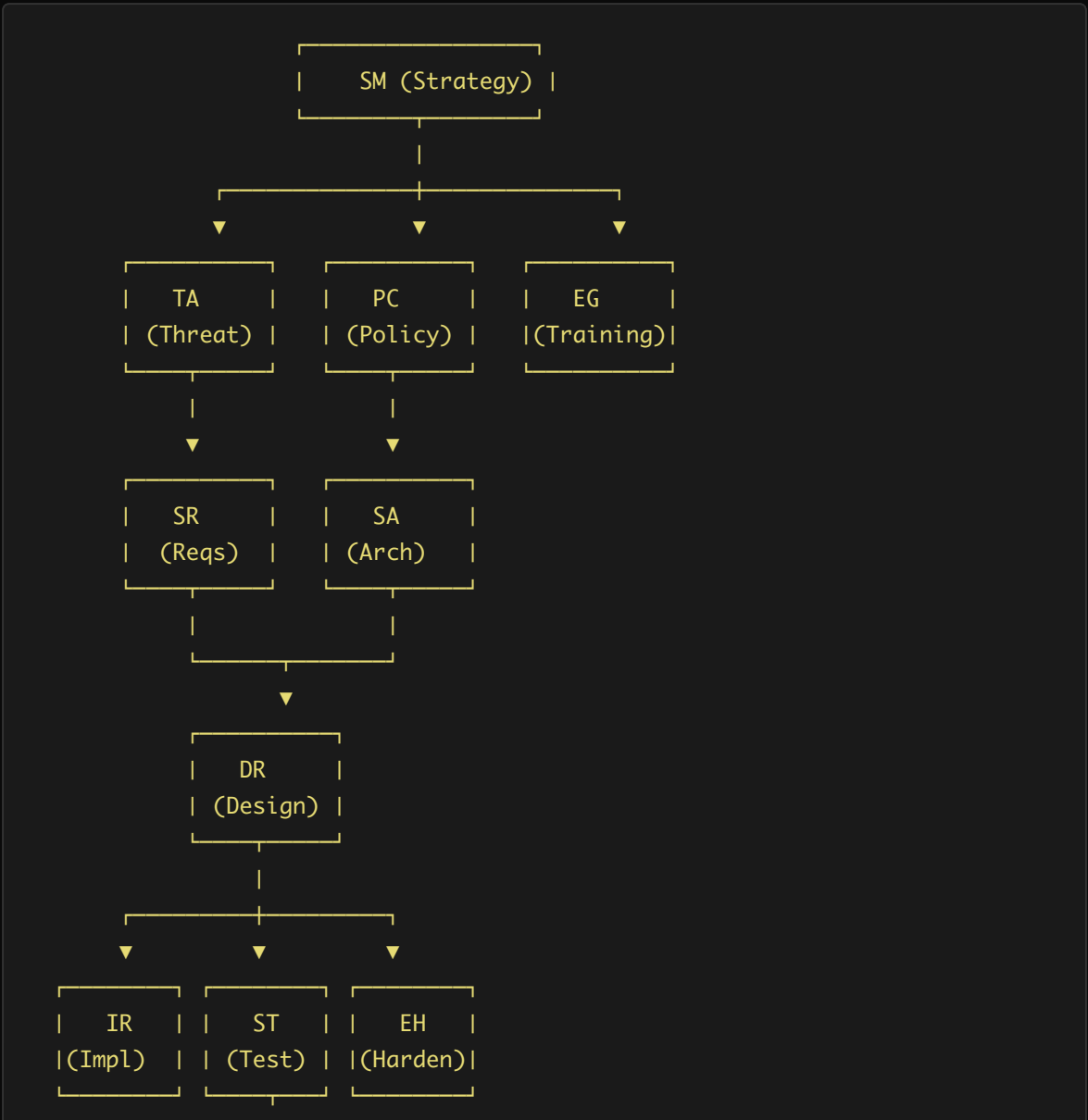
Level 2→3 Investment Estimate

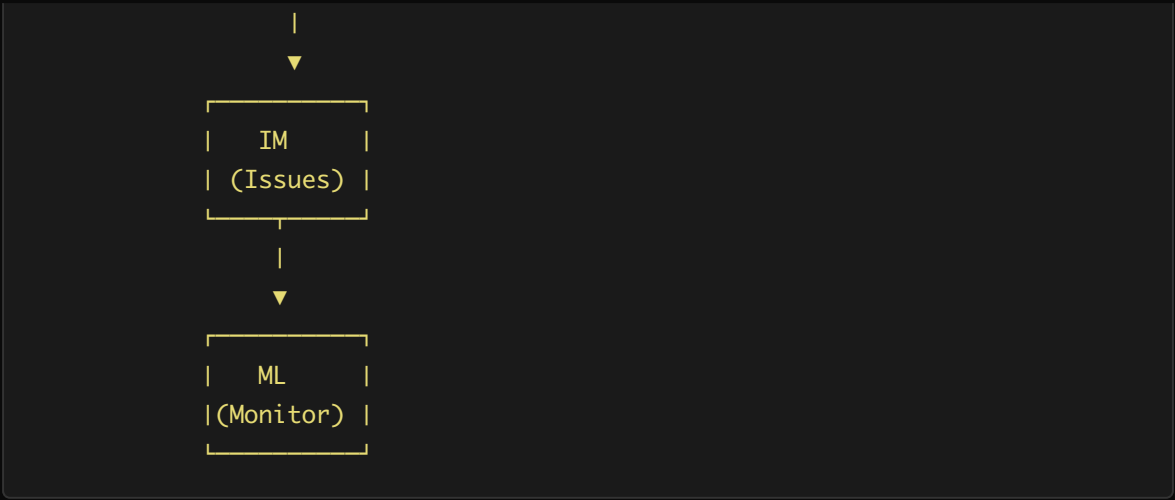
Category	Hours	Cost (Internal)	Cost (External)
Automation Development	200-300	$40K-$60K	$80K-$120K
ML/Analytics Setup	100-150	$20K-$30K	$50K-$75K

Category	Hours	Cost (Internal)	Cost (External)
Tool Integration	150-200	$30K-$40K	$60K-$80K
Process Optimization	80-120	$16K-$24K	$30K-$50K
Total	**530-770**	**$106K-$154K**	**$220K-$325K**

Practice Dependencies

Some practices should be prioritized before others:





Critical Path: SM → TA → SR → DR → ST → IM → ML

Sample 12-Month Roadmap

Quarter 1: Establish Foundation (Level 1)

Month	Focus	Practices	Deliverables
M1	Discovery	SM, ML	AI inventory, basic logging
M2	Assessment	TA, ST	Threat models, initial tests
M3	Controls	SR, EH	Requirements, hardening

Quarter 2: Build Structure (Level 2 Start)

Month	Focus	Practices	Deliverables
M4	Process	DR, IR	Review processes
M5	Policy	PC, EG	Policies, training
M6	Architecture	SA, IM	Architecture review, issue tracking

Quarter 3: Mature (Level 2 Complete)

Month	Focus	Practices	Deliverables
M7	Automation	ST, EH	CI/CD testing, auto-hardening
M8	Metrics	SM, ML	Dashboard, alerts
M9	Validation	All	Level 2 assessment

Quarter 4: Optimize (Level 3 Start)

Month	Focus	Practices	Deliverables
M10	Intelligence	TA, ML	Threat intel, anomaly detection
M11	Advanced	ST, IR	Red team, advanced analysis
M12	Leadership	SM, EG	ROI, champions program

Common Roadblocks

Roadblock 1: Lack of Executive Support

Symptom: Budget constraints, competing priorities **Solution:** Start with Risk Matrix to show risk coverage

Roadblock 2: Skill Gaps

Symptom: Team lacks AI security expertise **Solution:** Focus on EG practice, leverage Tools

Roadblock 3: Technical Debt

Symptom: Legacy systems resist security controls **Solution:** Prioritize ML (monitoring) to create visibility

Roadblock 4: Scope Creep

Symptom: Trying to do too much at once **Solution:** Follow phased roadmap, complete one level before advancing

Roadblock 5: Measurement Paralysis

Symptom: Can't prove progress **Solution:** Establish basic SM metrics early, track consistently

---

Maturity Assessment Schedule

Maturity Level	Assessment Frequency	Duration
Level 0-1	Quarterly	2-4 hours
Level 2	Semi-annually	4-8 hours
Level 3	Annually	8-16 hours

Document Information

Field	Value
Document	Maturity Roadmap
HAIAMM Version	2.2
Last Updated	January 2026

Related Documents: - Assessment Checklist - Assess current state - First 30 Days - Start Level 1 - Tools & Resources - Implementation support

Back to Index

HAIAMM Logo

HAIAMM Assessment Checklist

Rapid self-assessment (30 minutes)

Back to Index | Maturity Roadmap | First 30 Days

Overview

This checklist provides a quick way to assess your AI security maturity across all 12 HAIAMM practices. Complete it in 30 minutes to understand your current state and identify priority areas.

Scoring: - **Yes** = 1 point (fully implemented) - **Partial** = 0.5 points (partially implemented) - **No** = 0 points (not implemented)

Maturity Calculation: - Level 1: Score 2+ on Level 1 questions - Level 2: Level 1 achieved + Score 2+ on Level 2 questions - Level 3: Level 2 achieved + Score 2+ on Level 3 questions

Governance Practices

SM - Strategy & Metrics

Level	Question	Yes	Partial	No
L1	Do you have a documented AI security strategy?	[ ]	[ ]	[ ]
L1	Do you maintain an inventory of all AI systems?	[ ]	[ ]	[ ]

Level	Question	Yes	Partial	No
L1	Is there executive sponsorship for AI security?	[ ]	[ ]	[ ]
L2	Do you track AI security metrics (e.g., detection rate, incident count)?	[ ]	[ ]	[ ]
L2	Are AI security metrics reviewed regularly (monthly or more)?	[ ]	[ ]	[ ]
L2	Do metrics drive security improvement decisions?	[ ]	[ ]	[ ]
L3	Is AI security benchmarked against industry standards?	[ ]	[ ]	[ ]
L3	Do you demonstrate ROI for AI security investments?	[ ]	[ ]	[ ]
L3	Is there continuous optimization based on metrics?	[ ]	[ ]	[ ]

SM Score: L1: ***/3 I L2:*** /3 I L3: ___/3 I **Maturity Level: ___**

PC - Policy & Compliance

Level	Question	Yes	Partial	No
L1		[ ]	[ ]	[ ]

Level	Question	Yes	Partial	No
	Do you have an AI acceptable use policy?			
L1	Are AI security policies communicated to relevant teams?	[ ]	[ ]	[ ]
L1	Do you know which regulations apply to your AI systems?	[ ]	[ ]	[ ]
L2	Are AI security policies enforced technically (not just documented)?	[ ]	[ ]	[ ]
L2	Do you have AI-specific compliance controls mapped?	[ ]	[ ]	[ ]
L2	Are policy violations tracked and addressed?	[ ]	[ ]	[ ]
L3	Do you have automated policy compliance checking?	[ ]	[ ]	[ ]
L3	Is compliance continuously monitored (not point-in-time)?	[ ]	[ ]	[ ]
L3	Do you proactively address emerging AI regulations?	[ ]	[ ]	[ ]

PC Score: L1: ***/3 I L2:*** /3 I L3: ___/3 I **Maturity Level: ___**

EG - Education & Guidance

Level	Question	Yes	Partial	No
L1	Do developers receive AI security awareness training?	[ ]	[ ]	[ ]
L1	Are secure AI development guidelines available?	[ ]	[ ]	[ ]
L1	Do users understand AI limitations and risks?	[ ]	[ ]	[ ]
L2	Is AI security training role-specific (dev, ops, user)?	[ ]	[ ]	[ ]
L2	Is training effectiveness measured?	[ ]	[ ]	[ ]
L2	Are training materials updated for new threats?	[ ]	[ ]	[ ]
L3	Do you have an AI security champions program?	[ ]	[ ]	[ ]
L3	Is training integrated into development workflows?	[ ]	[ ]	[ ]
L3	Do you contribute to industry AI security education?	[ ]	[ ]	[ ]

EG Score: L1: ***/3 I L2:*** /3 I L3: ___/3 I **Maturity Level: ___**

Building Practices

TA - Threat Assessment

Level	Question	Yes	Partial	No
L1	Have you identified AI-specific threats (prompt injection, poisoning, etc.)?	[ ]	[ ]	[ ]
L1	Do you assess threats for each AI system?	[ ]	[ ]	[ ]
L1	Are threat assessments documented?	[ ]	[ ]	[ ]
L2	Do you use a structured threat modeling methodology?	[ ]	[ ]	[ ]
L2	Are threats prioritized by likelihood and impact?	[ ]	[ ]	[ ]
L2	Are threat assessments updated when systems change?	[ ]	[ ]	[ ]
L3	Do you use threat intelligence for AI-specific threats?	[ ]	[ ]	[ ]
L3	Are threat models validated through testing?	[ ]	[ ]	[ ]

Level	Question	Yes	Partial	No
L3	Do you predict emerging AI threats proactively?	[ ]	[ ]	[ ]

TA Score: L1: ***/3 I L2:*** /3 I L3: ___/3 I **Maturity Level: ___**

SR - Security Requirements

Level	Question	Yes	Partial	No
L1	Do you have security requirements for AI input validation?	[ ]	[ ]	[ ]
L1	Are output handling requirements defined?	[ ]	[ ]	[ ]
L1	Are access control requirements documented?	[ ]	[ ]	[ ]
L2	Are AI security requirements derived from threat assessments?	[ ]	[ ]	[ ]
L2	Are requirements testable (specific, measurable)?	[ ]	[ ]	[ ]
L2	Are requirements tracked through implementation?	[ ]	[ ]	[ ]
L3	Are requirements automatically verified in CI/CD?	[ ]	[ ]	[ ]

Level	Question	Yes	Partial	No
L3	Do requirements cover emerging AI-specific risks?	[ ]	[ ]	[ ]
L3	Are requirements benchmarked against best practices?	[ ]	[ ]	[ ]

SR Score: L1: ***/3 I L2:*** /3 I L3: ___/3 I **Maturity Level: ___**

SA - Secure Architecture

Level	Question	Yes	Partial	No
L1	Do you have architecture documentation for AI systems?	[ ]	[ ]	[ ]
L1	Are trust boundaries defined for AI components?	[ ]	[ ]	[ ]
L1	Is there separation between AI and critical systems?	[ ]	[ ]	[ ]
L2	Do you use defense-in-depth for AI systems?	[ ]	[ ]	[ ]
L2	Are architecture patterns documented and reused?	[ ]	[ ]	[ ]
L2		[ ]	[ ]	[ ]

Level	Question	Yes	Partial	No
	Is architecture reviewed for security before deployment?			
L3	Do you have AI-specific secure architecture patterns?	[ ]	[ ]	[ ]
L3	Is architecture continuously validated against threats?	[ ]	[ ]	[ ]
L3	Do you contribute to industry architecture standards?	[ ]	[ ]	[ ]

SA Score: L1: ***/3 I L2:*** /3 I L3: ___/3 I **Maturity Level: ___**

Verification Practices

DR - Design Review

Level	Question	Yes	Partial	No
L1	Are AI system designs reviewed for security?	[ ]	[ ]	[ ]
L1	Do reviews cover AI-specific risks (data flow, trust)?	[ ]	[ ]	[ ]
L1	Are review findings tracked to resolution?	[ ]	[ ]	[ ]
L2		[ ]	[ ]	[ ]

Level	Question	Yes	Partial	No
	Do you have a design review checklist for AI systems?			
L2	Are reviews performed by qualified reviewers?	[ ]	[ ]	[ ]
L2	Are reviews mandatory before deployment?	[ ]	[ ]	[ ]
L3	Are reviews automated where possible?	[ ]	[ ]	[ ]
L3	Do you track design review effectiveness metrics?	[ ]	[ ]	[ ]
L3	Are review findings fed back to improve standards?	[ ]	[ ]	[ ]

DR Score: L1: ***/3 I L2:*** /3 I L3: ___/3 I **Maturity Level: ___**

IR - Implementation Review

Level	Question	Yes	Partial	No
L1	Are AI implementations reviewed for security?	[ ]	[ ]	[ ]
L1	Do reviewers understand AI security issues?	[ ]	[ ]	[ ]

Level	Question	Yes	Partial	No
L1	Are review findings addressed before deployment?	[ ]	[ ]	[ ]
L2	Do you use AI-specific security linting rules?	[ ]	[ ]	[ ]
L2	Is implementation review integrated with CI/CD?	[ ]	[ ]	[ ]
L2	Are high-risk changes reviewed by security?	[ ]	[ ]	[ ]
L3	Do you use automated security scanning for AI implementations?	[ ]	[ ]	[ ]
L3	Do you track implementation review effectiveness?	[ ]	[ ]	[ ]
L3	Are review patterns shared across teams?	[ ]	[ ]	[ ]

IR Score: L1: ***/3 I L2:*** /3 I L3: ___/3 I **Maturity Level: ___**

ST - Security Testing

Level	Question	Yes	Partial	No
L1	Do you test for prompt injection vulnerabilities?	[ ]	[ ]	[ ]
L1		[ ]	[ ]	[ ]

Level	Question	Yes	Partial	No
	Do you test for data leakage (PII, prompts)?			
L1	Are test results tracked and remediated?	[ ]	[ ]	[ ]
L2	Do you have an AI security test suite (50+ test cases)?	[ ]	[ ]	[ ]
L2	Is security testing integrated into CI/CD?	[ ]	[ ]	[ ]
L2	Do you conduct red team exercises for AI systems?	[ ]	[ ]	[ ]
L3	Do you use AI-specific security testing tools?	[ ]	[ ]	[ ]
L3	Do you track detection rate and testing coverage?	[ ]	[ ]	[ ]
L3	Are test cases updated based on new threats?	[ ]	[ ]	[ ]

ST Score: L1: ***/3 I L2:*** /3 I L3: ___/3 I **Maturity Level: ___**

Operations Practices

IM - Issue Management

Level	Question	Yes	Partial	No
L1	Do you track AI security vulnerabilities?	[ ]	[ ]	[ ]
L1	Are vulnerabilities prioritized by severity?	[ ]	[ ]	[ ]
L1	Do you have an AI incident response process?	[ ]	[ ]	[ ]
L2	Do you track remediation SLAs for AI vulnerabilities?	[ ]	[ ]	[ ]
L2	Are incidents classified by AI-specific categories?	[ ]	[ ]	[ ]
L2	Do you conduct post-incident reviews?	[ ]	[ ]	[ ]
L3	Do you track mean time to remediate (MTTR)?	[ ]	[ ]	[ ]
L3	Are issue patterns analyzed for root cause?	[ ]	[ ]	[ ]
L3	Do you share learnings across the organization?	[ ]	[ ]	[ ]

IM Score: L1: ***/3 / L2:*** /3 | L3: ___/3 | **Maturity Level: ___**

EH - Environment Hardening

Level	Question	Yes	Partial	No
L1	Are AI systems isolated from critical infrastructure?	[ ]	[ ]	[ ]
L1	Do you use encryption for AI data at rest and in transit?	[ ]	[ ]	[ ]
L1	Are default credentials changed on AI systems?	[ ]	[ ]	[ ]
L2	Do you have hardening standards for AI infrastructure?	[ ]	[ ]	[ ]
L2	Are AI execution environments sandboxed?	[ ]	[ ]	[ ]
L2	Do you use configuration management for AI systems?	[ ]	[ ]	[ ]
L3	Is hardening automatically verified?	[ ]	[ ]	[ ]
L3	Do you detect and remediate configuration drift?	[ ]	[ ]	[ ]
L3	Are hardening standards updated for new threats?	[ ]	[ ]	[ ]

EH Score: L1: ***/3 I L2:*** /3 I L3: ___/3 I **Maturity Level: ___**

ML - Monitoring & Logging

Level	Question	Yes	Partial	No
L1	Do you log AI system inputs and outputs?	[ ]	[ ]	[ ]
L1	Are security events from AI systems captured?	[ ]	[ ]	[ ]
L1	Can you investigate AI security incidents?	[ ]	[ ]	[ ]
L2	Do you have AI-specific security alerts?	[ ]	[ ]	[ ]
L2	Are logs correlated with other security data (SIEM)?	[ ]	[ ]	[ ]
L2	Do you monitor for behavioral anomalies?	[ ]	[ ]	[ ]
L3	Is monitoring coverage measured (>95% target)?	[ ]	[ ]	[ ]
L3	Do you use ML for AI security anomaly detection?	[ ]	[ ]	[ ]
L3	Is mean time to detect (MTTD) tracked and optimized?	[ ]	[ ]	[ ]

ML Score: L1: ***/3 I L2:*** /3 I L3: ___/3 I **Maturity Level: ___**

Score Summary

Practice	L1 Score	L2 Score	L3 Score	Maturity
SM - Strategy & Metrics	/3	/3	/3	
PC - Policy & Compliance	/3	/3	/3	
EG - Education & Guidance	/3	/3	/3	
TA - Threat Assessment	/3	/3	/3	
SR - Security Requirements	/3	/3	/3	
SA - Secure Architecture	/3	/3	/3	
DR - Design Review	/3	/3	/3	
IR - Implementation Review	/3	/3	/3	
ST - Security Testing	/3	/3	/3	
IM - Issue Management	/3	/3	/3	
EH - Environment Hardening	/3	/3	/3	
ML - Monitoring & Logging	/3	/3	/3	
TOTAL	/36	/36	/36	

Overall Maturity Calculation

- **Level 1 Maturity:** Achieved if average L1 score across all practices >=2.0
- **Level 2 Maturity:** Level 1 achieved + average L2 score >=2.0
- **Level 3 Maturity:** Level 2 achieved + average L3 score >=2.0

Your Overall Maturity Level: _____

Interpretation Guide

If Most Practices Are Level 0-1:

Priority: Establish foundations - Focus on First 30 Days - Start with highest-risk practices per Risk Matrix - Target Level 1 across all practices

If Most Practices Are Level 1:

Priority: Build structure - Formalize processes and documentation - Implement technical controls - Target Level 2 for critical practices (ST, ML, EH)

If Most Practices Are Level 2:

Priority: Optimize - Focus on automation and metrics - Benchmark against industry - Target Level 3 for high-risk practices

Gap Analysis

List practices with lowest scores: 1. _____ (Score: ***) 2.*** _____ (Score: ***) 3.*** _____ (Score: ___)

These are your priority improvement areas.

Document Information

Field	Value
Document	Assessment Checklist
HAIAMM Version	2.2
Last Updated	January 2026

Next Steps: - Maturity Roadmap - Plan your improvement path - First 30 Days - Start implementing - Tools & Resources - Find tools to help

Back to Index

HAIAMM Logo

HAIAMM Tools & Resources

Recommended tools organized by practice

Back to Index | First 30 Days | Assessment Checklist

Overview

This guide provides tool recommendations for implementing HAIAMM practices. Tools are categorized as:

- **Open Source** - Free, community-supported
- **Commercial** - Paid products with enterprise support
- **Frameworks** - Standards and methodologies
- **References** - Documentation and research

Governance Practices

SM - Strategy & Metrics

Dashboards & Visualization | Tool | Type | Purpose | |—-|—-|—-—-| | Grafana | Open Source | Metrics visualization | | Metabase | Open Source | Business analytics | | Tableau | Commercial | Enterprise dashboards | | Power BI | Commercial | Microsoft ecosystem |

Metrics Collection | Tool | Type | Purpose | |—-|—-|—-—-| | Prometheus | Open Source | Time-series metrics | | OpenTelemetry | Open Source | Observability framework | | Datadog | Commercial | Full-stack monitoring |

Frameworks - NIST AI Risk Management Framework (AI RMF) - ISO/IEC 42001 AI Management System - IEEE AI Ethics Guidelines

---

PC - Policy & Compliance

Policy Management | Tool | Type | Purpose | |—|—|—| | Open Policy Agent (OPA) | Open Source | Policy as code | | Kyverno | Open Source | Kubernetes policies | | Vanta | Commercial | Compliance automation | | Drata | Commercial | Compliance platform |

Compliance Frameworks - EU AI Act compliance checklists - NIST AI RMF Govern function - ISO 42001 controls - GDPR for AI systems

References - NIST AI RMF Playbook - EU AI Act Official Text

EG - Education & Guidance

Training Platforms | Tool | Type | Purpose | |—|—|—| | SANS Security Awareness | Commercial | Security training | | KnowBe4 | Commercial | Awareness training | | Coursera | Mixed | AI security courses |

AI Security Training Resources - OWASP AI Security Training - Google AI Security Course - Microsoft AI Security Guidelines - Anthropic AI Safety Resources

---

Building Practices

TA - Threat Assessment

Threat Modeling Tools | Tool | Type | Purpose | |—|—|—| | OWASP Threat Dragon | Open Source | Visual threat modeling | | Microsoft Threat Modeling Tool | Free | STRIDE methodology | | IriusRisk | Commercial | Automated threat modeling | | ThreatModeler | Commercial | Enterprise threat modeling |

AI-Specific Threat Frameworks - MITRE ATLAS - Adversarial Threat Landscape for AI Systems - OWASP AI Security Matrix - NIST AI Adversarial ML

Threat Intelligence | Tool | Type | Purpose | |—|—|—| | MITRE ATT&CK | Free | Threat knowledge base | | AlienVault OTX | Free | Threat intelligence sharing | | Recorded Future | Commercial | Threat intelligence |

SR - Security Requirements

Requirements Management | Tool | Type | Purpose | |—-|—-|—-—-| | Jira | Commercial | Requirements tracking | | Linear | Commercial | Modern issue tracking | | Azure DevOps | Commercial | Microsoft ALM |

Security Requirements Frameworks - OWASP ASVS - Application Security Verification - NIST 800-53 - Security Controls - CIS Controls - Critical Security Controls

AI-Specific Requirements - OWASP Top 10 for LLMs requirements - OWASP Agentic Top 10 requirements - ISO 42001 control requirements

---

SA - Secure Architecture

Architecture Tools | Tool | Type | Purpose | |—-|—-|—-—-| | Draw.io | Free | Diagramming | | Lucidchart | Commercial | Architecture diagrams | | C4 Model | Free | Architecture documentation | | Structurizr | Mixed | Architecture as code |

Cloud Security Architecture | Tool | Type | Purpose | |—-|—-|—-—-| | AWS Well-Architected Tool | Free | AWS architecture review | | Azure Architecture Center | Free | Azure patterns | | GCP Architecture Framework | Free | GCP patterns |

AI Architecture Patterns - RAG security architecture patterns - Agent orchestration patterns - Model serving security patterns

---

Verification Practices

DR - Design Review

Design Review Tools | Tool | Type | Purpose | |—-|—-|—-—-| | Figma | Commercial | Design collaboration | | Miro | Commercial | Visual collaboration | | GitHub PR Reviews | Mixed | Code/design review |

Design Review Checklists - OWASP Security Design Review Checklist - STRIDE threat analysis checklist - AI-specific design review (see handbook one-pagers)

IR - Implementation Review

Static Analysis - General | Tool | Type | Purpose | |—|—|—|—| | Semgrep | Open Source | Custom rules SAST | | SonarQube | Mixed | Code quality/security | | CodeQL | Free | GitHub security scanning | | Snyk Code | Commercial | AI-powered SAST |

Static Analysis - Python/ML | Tool | Type | Purpose | |—|—|—|—| | Bandit | Open Source | Python security | | Safety | Open Source | Python dependency check | | PyRight | Open Source | Type checking |

AI-Specific Implementation Review - LangChain security patterns - Prompt handling review - RAG implementation review

---

ST - Security Testing

LLM Security Testing | Tool | Type | Purpose | |—|—|—|—| | Garak | Open Source | LLM vulnerability scanner | | PyRIT | Open Source | Microsoft red team tool | | Rebuff | Open Source | Prompt injection detection | | LLM Guard | Open Source | Input/output validation | | Vigil | Open Source | LLM security scanner |

AI Security Platforms | Tool | Type | Purpose | |—|—|—|—| | HiddenLayer | Commercial | AI security platform | | Robust Intelligence | Commercial | AI validation | | Protect AI | Commercial | ML security | | Calypso AI | Commercial | AI security testing |

General Security Testing | Tool | Type | Purpose | |—|—|—|—| | OWASP ZAP | Open Source | Web app testing | | Burp Suite | Commercial | Web security testing | | Nuclei | Open Source | Vulnerability scanner |

Red Team Frameworks - MITRE ATLAS Navigator - OWASP LLM Testing Guide - AI Red Team Playbook (Microsoft)

Operations Practices

IM - Issue Management

Issue Tracking | Tool | Type | Purpose | |—|—|—|—| | Jira | Commercial | Issue tracking | | Linear | Commercial | Modern tracking | | GitHub Issues | Mixed | Developer-focused | | ServiceNow | Commercial | Enterprise ITSM |

Vulnerability Management | Tool | Type | Purpose | |—|—|—|—| | DefectDojo | Open Source | Vuln management | | Tenable.io | Commercial | Vulnerability platform | | Qualys | Commercial | Vuln management |

Incident Response | Tool | Type | Purpose | |—|—|—|—| | PagerDuty | Commercial | Incident management | | Opsgenie | Commercial | Alerting | | TheHive | Open Source | Security IR platform |

EH - Environment Hardening

Container Security | Tool | Type | Purpose | |—|—|—|—| | gVisor | Open Source | Container sandboxing | | Firecracker | Open Source | MicroVM isolation | | Falco | Open Source | Runtime security | | Trivy | Open Source | Container scanning |

Infrastructure Security | Tool | Type | Purpose | |—|—|—|—| | Checkov | Open Source | IaC scanning | | Terraform Sentinel | Commercial | Policy as code | | AWS Config | Commercial | Configuration compliance | | Azure Policy | Commercial | Azure compliance |

Secret Management | Tool | Type | Purpose | |—|—|—|—| | HashiCorp Vault | Mixed | Secret management | | AWS Secrets Manager | Commercial | AWS secrets | | Azure Key Vault | Commercial | Azure secrets |

AI Execution Sandboxes | Tool | Type | Purpose | |—|—|—|—| | E2B | Commercial | Code execution sandbox | | Modal | Commercial | Serverless AI | | Fly.io | Commercial | App sandboxing |

ML - Monitoring & Logging

Logging & SIEM | Tool | Type | Purpose | |—|—|—|—| | Elasticsearch/ELK | Mixed | Log management | | Splunk | Commercial | Enterprise SIEM | | Datadog | Commercial | Observability | | Grafana Loki | Open Source | Log aggregation |

AI-Specific Monitoring | Tool | Type | Purpose | |—|—|—|—| | Weights & Biases | Mixed | ML experiment tracking | | MLflow | Open Source | ML lifecycle | | Arize | Commercial | ML observability | | WhyLabs | Commercial | AI observability |

Anomaly Detection | Tool | Type | Purpose | |—|—|—|—| | Apache Spark MLlib | Open Source | ML for anomaly detection | | Amazon SageMaker | Commercial | ML platform | | Azure Anomaly Detector | Commercial | Anomaly API |

Supply Chain Security

Dependency Scanning | Tool | Type | Purpose | |—|—|—|—| | Syft | Open Source | SBOM generation | | Grype | Open Source | Vulnerability scanning | | Snyk | Commercial | Dependency security | | Socket.dev | Commercial | Supply chain security |

Code Signing | Tool | Type | Purpose | |—|—|—|—| | Sigstore | Open Source | Keyless signing | | Cosign | Open Source | Container signing | | In-toto | Open Source | Supply chain integrity |

Model Registries | Tool | Type | Purpose | |—|—|—|—| | MLflow Model Registry | Open Source | Model versioning | | AWS SageMaker Registry | Commercial | AWS model registry | | Azure ML Registry | Commercial | Azure model registry | | Hugging Face | Mixed | Model hub |

Key References

OWASP Resources

- OWASP AI Exchange
- OWASP Top 10 for LLMs

- OWASP Top 10 for Agentic Applications
- OWASP ML Security Top 10

Industry Frameworks

- NIST AI Risk Management Framework
- MITRE ATLAS
- ISO/IEC 42001 - AI Management Systems
- IEEE AI Ethics

Research & Publications

- Anthropic AI Safety Research
- OpenAI Safety Research
- Google AI Safety
- Microsoft Responsible AI

Tool Selection Guide

By Organization Size

Startups (< 50 employees) - Focus: Open source tools, cloud-native - Recommended: Garak, Semgrep, Grafana, GitHub Security

Mid-Size (50-500 employees) - Focus: Balance of open source and commercial - Recommended: Snyk, Datadog, HiddenLayer, PagerDuty

Enterprise (500+ employees) - Focus: Enterprise-grade with support - Recommended: Splunk, ServiceNow, Robust Intelligence, CrowdStrike

By Budget

Low Budget (< $10K/year) - All open source stack - Garak, Semgrep, DefectDojo, ELK, Prometheus

Medium Budget ($10K-$100K/year) - Mixed open source + commercial - Snyk, Datadog, one AI security platform

High Budget (> $100K/year) - Enterprise stack - Full commercial suite with enterprise support

---

Document Information

Field	Value
Document	Tools & Resources
HAIAMM Version	2.2
Last Updated	January 2026

Related Documents: - First 30 Days - Implementation guide - Assessment Checklist - Self-assessment - Maturity Roadmap - Level progression

Back to Index