# AI AGENTS Checklist

Complete Security Checklist for AI Agent Deployments • 2026 Edition

Prepared by Bithost | ZHOST Consulting Private Limited | sales@bithost.in

*v1.0 • February 2026 • 18 Sections • 200+ Controls • Confidential*

### ⚠ Why This Matters

AI agents routinely receive credentials via environment variables, system prompts, tool configurations, and retrieved documents — then log, cache, or transmit them without any human ever noticing. Unlike traditional apps, the attack surface includes the context window, vector databases, agent memory, inter-agent messages, and LLM-generated code. This checklist covers all of it.

## PRIORITY KEY

| | |
|---|---|
| CRITICAL | Immediate risk of credential exposure. Block deployment until resolved. |
| HIGH | Must be in place before going live. Direct risk of breach. |
| MEDIUM | Address within 30 days. Significant risk if left unresolved. |
| LOW | Best practice. Resolve within 90 days on next review cycle. |

| ✓ | Checklist Item | What to Check / Notes | Priority |
|---|---|---|---|
| **1 \| AI AGENT ARCHITECTURE & THREAT SURFACE** | | | |
| **Design-Time Security Foundations** | | | |
| □ | **Map Every Credential the Agent Touches** | List all API keys, tokens, DB passwords, OAuth secrets used at design time. | CRITICAL |
| □ | **Document the Full Agent Execution Graph** | Which tools, APIs, and services each agent can call. No undocumented paths. | HIGH |
| □ | **Apply Least Privilege to Every Agent Role** | Agents should only hold credentials for actions they actively perform. | CRITICAL |
| □ | **Separate Agent Roles by Function** | Retrieval agent, action agent, planner — different credentials, different permissions. | HIGH |
| □ | **Identify All Credential Injection Points** | Env vars, prompt context, tool configs, system prompts — all catalogued. | HIGH |
| □ | **Design with Credential-Free Flows First** | Use managed identities / service accounts before falling back to API keys. | HIGH |
| □ | **Threat Model the Agent Before Building** | STRIDE or equivalent. Where can credentials leak? Map attack paths explicitly. | HIGH |

| ✓ | Checklist Item | What to Check / Notes | Priority |
|---|---|---|---|
| □ | **Do Not Build Monolithic All-Knowing Agents** | One agent with access to everything is one breach away from everything. | CRITICAL |
| **Runtime Architecture Controls** | | | |
| □ | **Run Agents in Isolated Execution Environments** | Separate containers, VMs, or sandboxes per agent. No shared memory. | HIGH |
| □ | **Apply Network Egress Restrictions to Agent Processes** | Agents should only reach the specific endpoints they need. Block everything else. | HIGH |
| □ | **Enforce Read-Only Filesystem for Agent Containers** | Prevents an agent from writing credentials to disk or installing persistence. | HIGH |
| □ | **Use Ephemeral Environments for Agent Execution** | Spin up, run, tear down. Credentials exist only for the duration of the task. | HIGH |
| □ | **Validate Agent Output Before Acting on It** | Never let an agent's output directly trigger privileged actions without validation. | HIGH |

| ✓ | Checklist Item | What to Check / Notes | Priority |
|---|---|---|---|
| **2 \| CREDENTIAL STORAGE & LIFECYCLE** | | | |
| **Where Credentials Live** | | | |
| □ | **Never Store Credentials in Agent Code or Repos** | Zero hardcoded keys in Python, JS, YAML, Dockerfiles, or notebooks. | CRITICAL |
| □ | **Never Store Credentials in Agent Prompts or System Prompts** | System prompts are readable — treat them as public. | CRITICAL |
| □ | **Use a Dedicated Secrets Manager** | AWS Secrets Manager, HashiCorp Vault, Azure Key Vault. Not env files. | CRITICAL |
| □ | **Never Store Credentials in Vector Databases** | Embeddings and retrieved chunks frequently contain secrets developers forgot. | CRITICAL |
| □ | **Never Store Credentials in Agent Memory / Logs** | Conversation histories, session logs, and traces commonly expose credentials. | HIGH |
| □ | **Audit Every File the Agent Can Read or Write** | Agents with filesystem access can read .env, config.json, ~/.aws/credentials. | HIGH |
| □ | **Enforce Encryption at Rest for All Agent State Storage** | Session state, memory stores, and caches must be encrypted. | HIGH |
| **Credential Lifecycle Management** | | | |
| □ | **Rotate All Agent Credentials on a Fixed Schedule** | Maximum 90 days for API keys. 30 days for high-privilege tokens. | HIGH |
| □ | **Rotate Immediately After Any Suspected Exposure** | Treat 'maybe exposed' the same as 'definitely exposed'. Rotate first, investigate second. | CRITICAL |
| □ | **Set Short Expiry on All Agent Tokens** | OAuth tokens: 1 hour max. Session tokens: task-scoped. Never open-ended. | HIGH |

| ✓ | Checklist Item | What to Check / Notes | Priority |
|---|---|---|---|
| □ | **Use Dynamic Secrets Where Possible** | Vault-generated just-in-time credentials that expire after agent task completes. | HIGH |
| □ | **Revoke Credentials of Decommissioned Agents Immediately** | Old agent credentials frequently persist long after agents are retired. | HIGH |
| □ | **Maintain a Credential Inventory for All Agents** | Which agent holds which credential, when it was last rotated, who owns it. | HIGH |

| ✓ | Checklist Item | What to Check / Notes | Priority |
|---|---|---|---|
| **3 \| PROMPT INJECTION & CONTEXT WINDOW SECURITY** | | | |
| **Prompt Injection Defense** | | | |
| □ | **Treat All External Content as Untrusted Input** | Web pages, documents, emails, database records — all can carry injections. | CRITICAL |
| □ | **Never Concatenate User Input Directly Into System Prompts** | Parameterize prompt construction. User content goes in user turn, not system. | CRITICAL |
| □ | **Implement Prompt Injection Detection Layer** | Scan retrieved content for injection patterns before passing to LLM. | HIGH |
| □ | **Validate Agent Instructions Against Allowed Action Set** | Agent should reject instructions that fall outside its defined capabilities. | HIGH |
| □ | **Use Separate LLM Calls for Trusted vs Untrusted Content** | Retrieval augmentation uses a sandboxed LLM that cannot access tools. | HIGH |
| □ | **Test for Indirect Prompt Injection in All RAG Pipelines** | Injections hidden in documents are the most common real-world vector. | CRITICAL |
| □ | **Apply Input Length Limits to Prevent Context Flooding** | Flooding the context window is a technique for burying injections. | MEDIUM |
| □ | **Log All Prompts Sent to LLM (With Masking)** | Log the structure; mask credential values. Needed for forensic investigation. | HIGH |
| **Context Window Hygiene** | | | |
| □ | **Never Place Raw Credentials in the Context Window** | If an agent needs a credential, it fetches it from Secrets Manager at call time. | CRITICAL |
| □ | **Clear Context Between Independent Agent Tasks** | Don't carry conversation history containing sensitive data into unrelated tasks. | HIGH |
| □ | **Audit What Goes Into the Context Window at Each Step** | Log the full context (masked) for debugging. Review regularly for credential leakage. | HIGH |
| □ | **Strip Credentials from Retrieved Documents Before Injection** | Pre-process retrieved content to redact patterns matching secrets. | HIGH |
| □ | **Limit Context Window Sharing Between Agent Roles** | A retrieval agent's context should not be visible to a code-execution agent. | HIGH |

| ✓ | Checklist Item | What to Check / Notes | Priority |
|---|---|---|---|
| **4 \| TOOL USE & FUNCTION CALL SECURITY** | | | |
| **Tool Definition & Permissions** | | | |
| ☐ | **Define an Explicit Allowlist of Tools Per Agent** | No catch-all tool access. Each agent gets only the tools its role requires. | CRITICAL |
| ☐ | **Never Expose Credential-Bearing Tools to Untrusted Agents** | If a tool uses an API key internally, the agent calling it should not see that key. | CRITICAL |
| ☐ | **Validate Tool Input Parameters Before Execution** | LLM-generated tool arguments must be validated against a schema before use. | HIGH |
| ☐ | **Implement Human-in-the-Loop for Irreversible Tool Actions** | Delete, send email, transfer funds — require human confirmation. | HIGH |
| ☐ | **Rate-Limit Agent Tool Calls** | Unbounded tool calls enable exfiltration and resource abuse. | HIGH |
| ☐ | **Log Every Tool Call with Input and Output** | Full audit trail. Tool calls are where credential leaks most often appear. | HIGH |
| ☐ | **Sign Tool Definitions to Prevent Tampering** | Ensure the tool schema an agent receives matches what was approved. | MEDIUM |
| **External API & Service Calls** | | | |
| ☐ | **Use Service Accounts for Agent External API Calls** | Not developer personal tokens. Dedicated service accounts with scoped permissions. | HIGH |
| ☐ | **Never Pass Credentials as Tool Arguments** | Credentials go in the tool implementation, not in the LLM-visible parameter schema. | CRITICAL |
| ☐ | **Validate All URLs Before Agent Makes HTTP Requests** | Prevent SSRF attacks where agent is tricked into calling internal services. | HIGH |
| ☐ | **Enforce Allowlist of Domains Agent Can Call** | Agent should not be able to reach arbitrary external URLs. | HIGH |
| ☐ | **Redact API Keys from Tool Response Logs** | Tool responses sometimes echo back the key used. Mask before logging. | HIGH |
| ☐ | **Use Short-Lived OAuth Tokens for External Service Calls** | Token exchange per session. Never reuse tokens across agent runs. | HIGH |

| ✓ | Checklist Item | What to Check / Notes | Priority |
|---|---|---|---|
| **5 \| RAG PIPELINE & VECTOR DATABASE SECURITY** | | | |
| **Data Ingestion Security** | | | |
| ☐ | **Scan All Documents for Credentials Before Ingestion** | Run secret detection (truffleHog, Gitleaks) on all files before they enter the pipeline. | CRITICAL |
| ☐ | **Strip or Redact Credentials from Source Documents** | Regex + ML-based PII/secret detection at ingestion time. | CRITICAL |
| ☐ | **Restrict Who Can Add Documents to the Knowledge Base** | Unauthorized document injection is a primary indirect injection vector. | HIGH |

| ✓ | Checklist Item | What to Check / Notes | Priority |
|---|---|---|---|
| □ | **Validate Document Provenance Before Ingestion** | Only ingest from trusted, verified sources. Sign source documents. | HIGH |
| □ | **Scan Ingested Content for Prompt Injection Patterns** | Documents designed to manipulate the agent's behavior must be caught here. | HIGH |
| □ | **Version-Control All Knowledge Base Updates** | Know exactly when a document was added, by whom, and what changed. | MEDIUM |
| **Vector Store & Retrieval Security** | | | |
| □ | **Encrypt the Vector Database at Rest and in Transit** | Embeddings can be reconstructed to reveal original text including credentials. | HIGH |
| □ | **Apply Row-Level Access Control to Vector Stores** | Users and agents should only retrieve chunks they're authorized to see. | HIGH |
| □ | **Audit All Embedding Queries and Retrieved Chunks** | Log which chunks were retrieved for each agent query. Review for anomalies. | HIGH |
| □ | **Re-Scan the Knowledge Base Periodically for New Credential Leaks** | New CVE patterns and secret formats emerge. Rescan monthly. | MEDIUM |
| □ | **Namespace Vector Stores by Trust Level** | Public knowledge, internal knowledge, and confidential data in separate namespaces. | HIGH |
| □ | **Do Not Store Raw Source Text in Vector DB Alongside Embeddings** | If raw text is stored, it can be directly retrieved — including any secrets in it. | HIGH |

| ✓ | Checklist Item | What to Check / Notes | Priority |
|---|---|---|---|
| **6 \| AGENT MEMORY & STATE MANAGEMENT** | | | |
| **Short-Term Memory (In-Session)** | | | |
| □ | **Clear Working Memory Between Agent Invocations** | Session state should not carry over credential references between tasks. | HIGH |
| □ | **Never Store Credentials in Agent Working Memory** | Working memory is frequently logged, serialized, or shared across calls. | CRITICAL |
| □ | **Mask Sensitive Values in Memory Traces** | Debugging tools that dump memory state must redact credential patterns. | HIGH |
| □ | **Limit Conversation History Retained in Context** | Older turns containing sensitive data should be truncated or summarized. | HIGH |
| □ | **Encrypt In-Memory State for Long-Running Agents** | Agents running for extended periods hold more sensitive data; encrypt it. | MEDIUM |
| **Long-Term Memory & Persistence** | | | |
| □ | **Scan Long-Term Memory Stores for Credential Leakage** | Agent memory databases accumulate secrets over time. Scan regularly. | CRITICAL |
| □ | **Apply Retention Limits to All Agent Memory Stores** | Auto-expire memories older than your retention policy. Don't keep forever. | HIGH |
| □ | **Encrypt All Persistent Agent Memory at Rest** | AES-256 minimum. KMS-managed keys with access logging. | HIGH |

| ✓ | Checklist Item | What to Check / Notes | Priority |
|---|---|---|---|
| □ | **Apply Access Control to Agent Memory Read/Write** | Not every agent should read every other agent's memory. | HIGH |
| □ | **Log All Reads and Writes to Long-Term Memory** | Memory access is a data exfiltration path. Audit it. | HIGH |
| □ | **Test for Credential Extraction via Memory Queries** | Red-team: can a crafted query pull credential-containing memories? | HIGH |

| ✓ | Checklist Item | What to Check / Notes | Priority |
|---|---|---|---|

## 7 | MULTI-AGENT & AGENT-TO-AGENT COMMUNICATION

### Agent Identity & Authentication

| ✓ | Checklist Item | What to Check / Notes | Priority |
|---|---|---|---|
| □ | **Authenticate Agent-to-Agent Communications** | One agent calling another must prove its identity. No implicit trust. | CRITICAL |
| □ | **Use Short-Lived Signed Tokens for Inter-Agent Calls** | JWT or similar with 5-15 minute expiry. Agents cannot reuse each other's tokens. | HIGH |
| □ | **Define and Enforce Trust Levels Between Agents** | Orchestrator agents do not automatically inherit all sub-agent permissions. | CRITICAL |
| □ | **Never Propagate Credentials Across Agent Boundaries** | Each agent fetches its own credentials. Credentials are not passed in messages. | CRITICAL |
| □ | **Log All Inter-Agent API Calls with Source Identity** | Full audit trail of which agent asked which agent to do what. | HIGH |
| □ | **Validate Agent Instructions Regardless of Source** | Even instructions from a 'trusted' orchestrator agent must be validated. | HIGH |

### Message & Payload Security

| ✓ | Checklist Item | What to Check / Notes | Priority |
|---|---|---|---|
| □ | **Encrypt All Inter-Agent Message Payloads** | TLS for transport. Consider payload-level encryption for sensitive content. | HIGH |
| □ | **Validate Schema of All Inter-Agent Messages** | Structured message contracts. Reject anything that doesn't match the schema. | HIGH |
| □ | **Sanitize Agent Outputs Before Passing as Input to Next Agent** | Output from LLM becomes input to the next step — treat it as untrusted. | HIGH |
| □ | **Implement Circuit Breakers for Agent Cascades** | A compromised agent in a chain should not be able to call downstream freely. | MEDIUM |
| □ | **Test for Credential Leakage in Agent-to-Agent Messages** | Red-team: can agent A be tricked into leaking a credential to agent B? | HIGH |

| ✓ | Checklist Item | What to Check / Notes | Priority |
|---|---|---|---|

## 8 | CODE-GENERATING & CODE-EXECUTING AGENTS

### Code Generation Security

| ✓ | Checklist Item | What to Check / Notes | Priority |
|---|---|---|---|
| ☐ | **Never Inject Credentials Into LLM-Generated Code** | Generated code that contains real credentials is a guaranteed leak. | CRITICAL |
| ☐ | **Review All LLM-Generated Code Before Execution** | Human or automated review. LLMs routinely generate insecure credential patterns. | CRITICAL |
| ☐ | **Use Placeholder Variables in Generated Code** | Generated code references ${DB_PASSWORD} — real value injected at runtime separately. | HIGH |
| ☐ | **Scan LLM-Generated Code for Secret Patterns** | Run truffleHog or similar on all generated code before it runs. | HIGH |
| ☐ | **Never Log Generated Code That Contains Secrets** | Generation logs frequently contain the code that was produced, including secrets. | HIGH |
| ☐ | **Apply Static Analysis to LLM-Generated Code** | SAST tools on generated code before execution in any environment. | HIGH |
|  | **Code Execution Sandboxing** |  |  |
| ☐ | **Execute LLM-Generated Code in Isolated Sandboxes** | No network, no filesystem, no credential access in the sandbox by default. | CRITICAL |
| ☐ | **Block Filesystem Access in Code Execution Environments** | Agents cannot read host files, ~/.aws, /etc/passwd, environment files. | CRITICAL |
| ☐ | **Block Network Access by Default in Code Sandboxes** | Explicitly allowlist only required outbound endpoints. | HIGH |
| ☐ | **Restrict Environment Variables Visible to Executed Code** | Filter env vars passed to sandbox. Real credentials must not be visible. | CRITICAL |
| ☐ | **Set Strict Resource Limits on Code Execution** | CPU, memory, time limits. Unbounded execution enables data exfiltration. | HIGH |
| ☐ | **Audit All System Calls Made During Code Execution** | Network calls, file I/O, subprocess spawning — all logged and reviewed. | HIGH |
| ☐ | **Destroy Sandbox Environment After Each Execution** | No state persistence between runs. Fresh sandbox every time. | HIGH |

| ✓ | Checklist Item | What to Check / Notes | Priority |
|---|---|---|---|
|  | **9 \| LLM PROVIDER & API SECURITY** |  |  |
|  | **API Key Management** |  |  |
| ☐ | **Use Separate API Keys per Agent or per Environment** | Not one shared key across all agents. Breach of one key = breach of one agent only. | CRITICAL |
| ☐ | **Store LLM API Keys in Secrets Manager, Never in Code** | Not in .env files, not in Docker config, not in CI environment variables. | CRITICAL |
| ☐ | **Set Spending Limits on All LLM API Keys** | Compromised keys used for prompt-farming will exceed normal usage. Detect it. | HIGH |

| ✓ | Checklist Item | What to Check / Notes | Priority |
|---|---|---|---|
| □ | **Monitor LLM API Usage for Anomalies** | Unusual token consumption, off-hours requests, or new request patterns. | HIGH |
| □ | **Rotate LLM API Keys Every 90 Days** | Standard credential hygiene applies to LLM provider keys too. | HIGH |
| □ | **Revoke and Rotate Immediately on Provider Data Breach** | LLM providers get breached too. Have a rotation runbook ready. | HIGH |
| □ | **Use Provider-Level IP Allowlisting Where Available** | Restrict which IPs can use your API key at the provider level. | MEDIUM |
| **Data Sent to LLM Providers** | | | |
| □ | **Audit What Data Is Sent in LLM Prompts** | Real credentials in prompts go to the provider's servers. Audit every call. | CRITICAL |
| □ | **Mask Sensitive Data Before Sending to External LLM APIs** | If using external providers, PII and credentials must be masked pre-call. | CRITICAL |
| □ | **Use On-Premises or Private LLMs for Sensitive Workloads** | Data that cannot leave your perimeter should use self-hosted models. | HIGH |
| □ | **Review LLM Provider Data Retention Policies** | Know how long your prompts are retained and who can access them. | HIGH |
| □ | **Enable Provider-Level Prompt Logging Controls** | Opt out of training data inclusion. Enable zero-data-retention where available. | HIGH |
| □ | **Confirm Provider Compliance Certifications** | SOC 2, ISO 27001, HIPAA BAA if applicable — verify before production use. | HIGH |

| ✓ | Checklist Item | What to Check / Notes | Priority |
|---|---|---|---|
| **10 \| LOGGING, OBSERVABILITY & AUDIT** | | | |
| **What to Log** | | | |
| □ | **Log Every Agent Action with Timestamp and Identity** | Who did what, when, to what system. The foundation of incident response. | CRITICAL |
| □ | **Log All Tool Calls with Arguments (Masked)** | Tool inputs are where credential leaks appear most. Log structure, mask values. | CRITICAL |
| □ | **Log All Credentials Accessed (Not Their Values)** | Log the secret name/ARN accessed, not the credential value. | HIGH |
| □ | **Log All LLM Prompts and Completions (Masked)** | Prompt logging is controversial but essential for forensics. Mask secrets. | HIGH |
| □ | **Log All Context Window Contents at Each Agent Step** | Trace mode logging for debugging. Ensure secret values are redacted. | HIGH |
| □ | **Log Inter-Agent Calls with Full Call Chain** | Track cascading calls. An injected prompt may travel through 4 agents. | HIGH |
| □ | **Log All Secrets Manager Access by Agent Identities** | GetSecretValue calls with agent identity, timestamp, and outcome. | HIGH |
| **Log Security & Retention** | | | |

| ✓ | Checklist Item | What to Check / Notes | Priority |
|---|---|---|---|
| ☐ | **Send Agent Logs to Centralized SIEM Immediately** | Local logs can be tampered with by a compromised agent. Centralize. | CRITICAL |
| ☐ | **Apply Write-Once Protection to Agent Audit Logs** | Agents must not be able to modify or delete their own logs. | CRITICAL |
| ☐ | **Retain Agent Logs for Minimum 12 Months** | Credential leaks are often discovered weeks after they occur. | HIGH |
| ☐ | **Apply Secret Detection Regex to All Outbound Log Streams** | Auto-detect and mask credentials appearing in logs in real time. | HIGH |
| ☐ | **Alert on Any Credential Pattern Appearing in Logs** | AWS key format, JWT tokens, bearer credentials — alert immediately. | CRITICAL |
| ☐ | **Encrypt Agent Logs at Rest with Separate KMS Key** | Log encryption key managed separately from application keys. | HIGH |
| ☐ | **Test Log Integrity Monthly** | Verify logs have not been tampered with since collection. | MEDIUM |

| ✓ | Checklist Item | What to Check / Notes | Priority |
|---|---|---|---|
| **11 \| SECRET DETECTION & DLP FOR AI SYSTEMS** | | | |
| **Automated Secret Scanning** | | | |
| ☐ | **Deploy Secret Detection in CI/CD for All Agent Code** | Block commits containing credentials. Git pre-commit hooks + CI gate. | CRITICAL |
| ☐ | **Scan All Agent Configuration Files for Credentials** | YAML, TOML, JSON config files are frequently overlooked. | HIGH |
| ☐ | **Scan Jupyter Notebooks for Embedded Credentials** | Notebooks are one of the most common places AI team credentials appear. | CRITICAL |
| ☐ | **Scan Agent Training Data and Fine-Tuning Datasets** | Datasets scraped from the internet frequently contain valid credentials. | CRITICAL |
| ☐ | **Scan Model Weights and Serialized Checkpoints** | Credentials can be embedded in fine-tuned model artifacts. | HIGH |
| ☐ | **Scan All Prompt Templates and Few-Shot Examples** | Developers often use real keys in examples. These go into production. | HIGH |
| ☐ | **Run Real-Time DLP on Agent Output Streams** | Intercept credential patterns before they reach end users or external APIs. | CRITICAL |
| **DLP Policy Configuration** | | | |
| ☐ | **Define Credential Pattern Library for Your Tech Stack** | AWS keys, GCP service accounts, OpenAI keys, Slack tokens — all regex-defined. | HIGH |
| ☐ | **Apply DLP to Agent Memory Store Writes** | Credentials written to memory stores must be caught before persistence. | HIGH |
| ☐ | **Apply DLP to Retrieval Outputs Before Injecting to Context** | Retrieved documents go through DLP before entering the LLM context. | HIGH |
| ☐ | **Set Up DLP Alerts with <5 Minute Detection SLA** | Credential leaks via AI agents can exfiltrate data within seconds. | HIGH |

| ✓ | Checklist Item | What to Check / Notes | Priority |
|---|---|---|---|
| ☐ | **Configure DLP to Mask, Not Just Alert** | Real-time masking stops the leak. Alerts alone are too slow. | HIGH |
| ☐ | **Test DLP Rules Monthly with Synthetic Credentials** | Verify detection still works as patterns evolve. | MEDIUM |

## 12 | SUPPLY CHAIN SECURITY FOR AI AGENTS

| ✓ | Checklist Item | What to Check / Notes | Priority |
|---|---|---|---|
| | **Model & Library Dependencies** | | |
| ☐ | **Verify Integrity of All Pre-Trained Models Before Use** | Hash verification of model weights. Untrusted models can exfiltrate data. | CRITICAL |
| ☐ | **Scan All Python/Node Dependencies for Known Vulnerabilities** | AI stacks pull dozens of transitive dependencies. Scan all of them. | HIGH |
| ☐ | **Pin All Dependency Versions in Agent Requirements** | Floating versions allow malicious package updates to reach production. | HIGH |
| ☐ | **Review LangChain, LlamaIndex, and Framework Updates Carefully** | Agent frameworks have had security vulnerabilities. Review changelogs. | HIGH |
| ☐ | **Use a Private Package Mirror for Agent Dependencies** | Dependency confusion attacks target public registry name collisions. | HIGH |
| ☐ | **Sign and Verify Container Images for Agent Deployments** | Use Cosign or AWS Signer. Verify signature before deploying. | HIGH |
| | **Third-Party Plugins & Tools** | | |
| ☐ | **Audit Every Third-Party Tool/Plugin an Agent Can Call** | Third-party tools have their own credential handling and logging. | HIGH |
| ☐ | **Review Plugin Source Code Before Enabling for Agent Use** | MCP servers and agent plugins can exfiltrate credentials via tool use. | CRITICAL |
| ☐ | **Do Not Allow Agents to Install Their Own Tools or Plugins** | Dynamic tool installation is a major supply chain risk. | CRITICAL |
| ☐ | **Restrict Tool Registry Access to Vetted Sources Only** | No unapproved plugins from public registries at runtime. | HIGH |
| ☐ | **Monitor Third-Party Tool API Calls from Agent Context** | Log all calls made by external plugins on behalf of your agent. | HIGH |
| ☐ | **Require Security Review for All New Tools Added to Agent Arsenal** | New tool = new attack surface. Security sign-off before deployment. | HIGH |

| ✓ | Checklist Item | What to Check / Notes | Priority |
|---|---|---|---|

## 13 | MODEL CONTEXT PROTOCOL (MCP) SECURITY

### MCP Server Security

| ✓ | Checklist Item | What to Check / Notes | Priority |
|---|---|---|---|
| □ | **Run MCP Servers in Isolated, Least-Privilege Environments** | MCP servers are high-value targets. They broker tool access for agents. | CRITICAL |
| □ | **Authenticate All MCP Client-to-Server Connections** | No unauthenticated MCP connections in any environment. | CRITICAL |
| □ | **Apply TLS to All MCP Transport Connections** | MCP over stdio or HTTP must use encrypted transport. | HIGH |
| □ | **Restrict MCP Server Tool Exposure to Necessary Tools Only** | The MCP server should not expose every tool to every client. | HIGH |
| □ | **Log All MCP Tool Calls with Client Identity and Arguments** | MCP calls are a primary vector for credential exfiltration. | HIGH |
| □ | **Rotate MCP Server API Keys and Tokens Frequently** | MCP servers hold credentials for the tools they proxy. Rotate them. | HIGH |
| □ | **Scan MCP Server Configuration for Hardcoded Credentials** | MCP server configs commonly contain API keys. Scan before deployment. | CRITICAL |
| **MCP Client Security** | | | |
| □ | **Validate MCP Server Identity Before Connecting** | Prevent MITM attacks where a fake MCP server intercepts tool calls. | HIGH |
| □ | **Do Not Pass Credentials in MCP Tool Arguments** | Credential injection via MCP tool params is a known leak pattern. | CRITICAL |
| □ | **Apply Tool Call Allowlists in MCP Client Configuration** | Clients should only be able to call pre-approved tools. | HIGH |
| □ | **Alert on Unexpected MCP Tool Calls** | Deviations from expected tool call patterns may indicate injection. | HIGH |
| □ | **Review All MCP Server Packages Before Deployment** | Third-party MCP packages can contain credential-harvesting code. | CRITICAL |

| ✓ | Checklist Item | What to Check / Notes | Priority |
|---|---|---|---|
| **14 \| DATA EXFILTRATION PREVENTION** | | | |
| **Outbound Data Controls** | | | |
| □ | **Apply Network Egress Filtering to All Agent Processes** | Agents should only reach allowlisted destinations. Block everything else. | CRITICAL |
| □ | **Inspect Agent HTTP Requests for Credential Patterns** | A compromised agent may POST credentials to external URLs. | CRITICAL |
| □ | **Block Agent Access to Pastebin, File-Sharing, and Webhook Sites** | Common exfiltration destinations. Block by category, not just domain. | HIGH |
| □ | **Rate-Limit Outbound Agent Requests** | Exfiltration at scale requires many requests. Rate limits slow the attack. | HIGH |
| □ | **Monitor Outbound Data Volume per Agent** | Unusual spikes in outbound data are a key exfiltration signal. | HIGH |
| □ | **Apply DLP to All Outbound Agent API Responses** | Data leaving the agent boundary must pass through DLP inspection. | HIGH |

| ✓ | Checklist Item | What to Check / Notes | Priority |
|---|---|---|---|
| □ | **Block DNS-Based Exfiltration from Agent Environments** | DNS tunneling is used to exfiltrate data in low-and-slow attacks. | HIGH |
| **Inbound Credential Injection Prevention** | | | |
| □ | **Validate All External Data Before Processing by Agent** | Data arriving from external sources can carry payloads designed to exfiltrate. | HIGH |
| □ | **Apply Content Security Policy to Agent Web Interfaces** | Prevent injected JavaScript from exfiltrating credentials via the browser. | HIGH |
| □ | **Never Reflect User Input in Agent System Prompts** | User-controlled content in system prompt enables credential exfiltration. | CRITICAL |
| □ | **Sanitize Email and Webhook Payloads Processed by Agents** | Agents processing emails or webhooks are exposed to injection via content. | HIGH |

| ✓ | Checklist Item | What to Check / Notes | Priority |
|---|---|---|---|
| **15 \| HUMAN OVERSIGHT & ACCESS CONTROLS** | | | |
| **Agent Authorization Gates** | | | |
| □ | **Require Human Approval for High-Privilege Agent Actions** | Any action that creates, modifies, or deletes credentials requires approval. | CRITICAL |
| □ | **Implement Breakglass Procedure to Pause All Agent Actions** | One command to suspend all running agents during an active incident. | HIGH |
| □ | **Define Maximum Agent Autonomy Levels per Sensitivity Tier** | Low-risk tasks: autonomous. High-risk tasks: supervised. Critical: human-only. | HIGH |
| □ | **Require MFA for Granting New Permissions to Agents** | No one should be able to expand an agent's access without MFA verification. | HIGH |
| □ | **Implement Time-Boxed Permissions for Agent Tasks** | Elevated permissions expire after the task completes, automatically. | HIGH |
| □ | **Review Agent Permission Changes in Change Management System** | Every agent permission change tracked, reviewed, and approved. | MEDIUM |
| **Developer & Operator Controls** | | | |
| □ | **Require Code Review for All Agent System Prompt Changes** | System prompts define agent behavior. Treat them like production code. | HIGH |
| □ | **Restrict Who Can Deploy or Update Agent Configurations** | Production agent config changes require elevated approval. | HIGH |
| □ | **Apply Separation of Duties for Agent Credential Management** | The person who builds the agent should not be the one who grants it credentials. | HIGH |
| □ | **Train Developers on AI-Specific Credential Risks** | LLM prompt leakage, context window exposure, indirect injection. | HIGH |
| □ | **Conduct Quarterly Security Reviews of All Agent Architectures** | Agents evolve. Security reviews need to keep pace. | MEDIUM |
| □ | **Maintain Agent Architecture Documentation Up to Date** | Outdated docs lead to unknown credential paths and unreviewed exposures. | MEDIUM |

| ✓ | Checklist Item | What to Check / Notes | Priority |
|---|---|---|---|
| **16 \| TESTING & RED TEAMING FOR AI AGENTS** | | | |
| **Security Testing Cadence** | | | |
| ☐ | **Test for Prompt Injection in Every Sprint** | Not just at launch. Injection vulnerabilities are introduced with every change. | HIGH |
| ☐ | **Run Automated Injection Tests Against All Agent Endpoints** | Payload library of known injection patterns. Run in CI. | HIGH |
| ☐ | **Conduct Annual Third-Party AI Security Assessment** | External specialists in LLM security. Not general penetration testing. | HIGH |
| ☐ | **Red-Team Each New Agent Before Production Deployment** | Internal team attempts credential extraction before real attackers do. | HIGH |
| ☐ | **Test All RAG Pipeline Ingestion Sources for Injection** | Can an attacker plant a document that exfiltrates credentials? | CRITICAL |
| ☐ | **Simulate Credential Leakage Scenarios in Pre-Prod** | Deploy synthetic credentials and verify detection triggers correctly. | HIGH |
| ☐ | **Test Agent Behavior with Adversarial Inputs Quarterly** | Jailbreak attempts, role-play attacks, context confusion attacks. | HIGH |
| **Specific Attack Scenarios to Test** | | | |
| ☐ | **Test: Can Agent Be Tricked Into Printing Its Credentials?** | Direct ask, role-play, code generation — all vectors. | CRITICAL |
| ☐ | **Test: Can Injected Document Exfiltrate Secrets via Tool Call?** | Most dangerous real-world attack. Test thoroughly. | CRITICAL |
| ☐ | **Test: Does Agent Leak Credentials in Error Messages?** | Exception handling often exposes the full context including credentials. | HIGH |
| ☐ | **Test: Can Agent Be Instructed to Ignore Security Policies?** | 'Ignore previous instructions' attacks on system prompt constraints. | HIGH |
| ☐ | **Test: Does Agent Carry Credentials Across Session Boundaries?** | Re-use of session state or memory can persist credentials. | HIGH |
| ☐ | **Test: Can Agent Be Used to Enumerate Other Agents' Permissions?** | Information disclosure as a precursor to privilege escalation. | HIGH |
| ☐ | **Test: What Happens When Agent Receives a 401 Response?** | Agents sometimes retry with elevated credentials on auth failure. | MEDIUM |

| ✓ | Checklist Item | What to Check / Notes | Priority |
|---|---|---|---|
| **17 \| INCIDENT RESPONSE FOR AI CREDENTIAL LEAKS** | | | |
| **Preparation** | | | |
| ☐ | **Write AI-Specific Incident Response Playbooks** | Generic IR playbooks miss the nuances of agent-driven credential leaks. | HIGH |

| ✓ | Checklist Item | What to Check / Notes | Priority |
|---|---|---|---|
| □ | **Define What Constitutes a Credential Leak Incident for AI Systems** | Prompt exposure, memory leak, tool call exfiltration — all defined. | HIGH |
| □ | **Pre-Establish 'Kill Switch' for Each Agent in Production** | One action that stops an agent and revokes its credentials. | HIGH |
| □ | **Designate an AI Security Incident Response Team** | Who handles an agent-driven credential incident at 2 AM? | HIGH |
| □ | **Maintain a Credential Revocation Runbook for Each Agent** | Which credentials to rotate, in what order, how to verify rotation worked. | HIGH |
| □ | **Store IR Playbooks Outside the AI System Environment** | An agent cannot be allowed to read or modify its own incident response plan. | HIGH |
| **Detection & Response** | | | |
| □ | **Alert on Any Credential Pattern in Agent Output Streams** | Real-time detection. Sub-5-minute alert-to-action SLA. | CRITICAL |
| □ | **Automatically Suspend Agent on Credential Leak Detection** | Auto-response: pause agent, alert team, begin containment. Don't wait. | CRITICAL |
| □ | **Rotate All Agent Credentials Immediately on Suspected Compromise** | Rotate first. Investigate while the rotated credential is already invalid. | CRITICAL |
| □ | **Capture Agent State Snapshot for Forensic Analysis** | Context window, memory, tool call history — snapshot before rotation. | HIGH |
| □ | **Trace the Full Injection-to-Exfiltration Path** | Reconstruct exactly how credentials were exposed. Fix the path, not just the leak. | HIGH |
| □ | **Notify Affected Downstream Systems After Rotation** | Any system that used the leaked credential must be audited for misuse. | HIGH |
| □ | **Conduct Post-Incident Review Within 48 Hours** | What detection fired? What didn't? What would have prevented it? | HIGH |

| ✓ | Checklist Item | What to Check / Notes | Priority |
|---|---|---|---|
| **18 \| COMPLIANCE, GOVERNANCE & AI SECURITY POLICY** | | | |
| **Policy & Standards** | | | |
| □ | **Publish an AI Agent Security Policy** | Covers credential handling, approved tools, logging requirements, review cadence. | HIGH |
| □ | **Define AI-Specific Data Classification for Agent Access** | What data levels can each agent class access? Documented and enforced. | HIGH |
| □ | **Include AI Agents in Your Existing Security Policy Framework** | Agent systems are not exempt from your credential management standards. | HIGH |
| □ | **Establish AI Agent Change Management Process** | Credential changes, tool changes, prompt changes — all require approval. | HIGH |
| □ | **Map AI Agent Controls to Compliance Frameworks** | NIST AI RMF, OWASP Top 10 for LLMs, ISO 42001 — document the mapping. | MEDIUM |

| ✓ | Checklist Item | What to Check / Notes | Priority |
|---|---|---|---|
| □ | **Include AI Security in Annual Security Awareness Training** | Developers building agents need to understand prompt injection and credential hygiene. | MEDIUM |
| **Audit & Continuous Improvement** | | | |
| □ | **Conduct Quarterly AI Security Reviews** | Architecture, credential hygiene, new attack patterns, tool changes. | HIGH |
| □ | **Track All AI Security Findings as Work Items** | Untracked findings get forgotten. Put them in your project backlog. | HIGH |
| □ | **Benchmark Against OWASP Top 10 for LLMs Annually** | The list evolves. Reverify your controls against the current version. | MEDIUM |
| □ | **Publish Internal AI Security Scorecard** | Visibility drives accountability. Teams should know their agent's security score. | MEDIUM |
| □ | **Participate in AI Security Community Disclosures** | Stay current on new LLM attack patterns and emerging vulnerabilities. | LOW |
| □ | **Review and Update This Checklist Quarterly** | The AI threat landscape moves fast. This checklist must move with it. | HIGH |

*This checklist is aligned with OWASP Top 10 for Large Language Model Applications, NIST AI Risk Management Framework (AI RMF 1.0), MITRE ATLAS, and AWS/Azure AI security guidance as of February 2026. Review quarterly — the AI threat landscape evolves rapidly.*

For AI security audits, SIEM/DLP integration, or implementation support: **sales@bithost.in | +91 911-336-6525**