



夜船有明
YOFUNE LABS

2026 Enterprise AI Agent Pre-Production Security Checklist

For Copilots, knowledge bots, business agents and MCP toolchains about to touch production data. Security teams do not need to become prompt engineers first — put one real agent on the table and ask for evidence.

- 01** An agent is not a chat window. It holds business keys: a goal, tools, state.
- 02** “Yes” without evidence is a fail. Account, tool list, short demo, log.
- 03** Three blockers: internal trial is fine; production data is not.

This is not a customer deliverable and not an assessment report. Lab paths explain mechanisms; they do not claim a production engagement for a named client.

contact@yofuneseccom

<https://yofuneseccom/ai-agent-security/>



01 / HOW TO USE

Pick one agent, then tick.

Do not send this as a survey for the business team to tick “yes”. Every item needs evidence: an account, a tool list, a short demo, a log line. The demo can be short. “We already thought about that,” with nothing to show, does not pass. If three key questions resolve to “shared dev account, we will trim tools later, logs are not wired yet,” the agent may stay in an internal trial. It should not touch production data.

Scoring	
Yes + evidence	Pass. Evidence may be an account screenshot, a tool list, a short demo or a traceable log.
Yes, no evidence	Fail. A verbal assurance cannot be written into go-live conditions.
No / later	Blocker. Record an owner and a date. No production data until it is closed.

Who fills it in

The security lead and the people who actually attach tools, MCP servers and accounts fill it in together. A CISO can read blockers and the management summary; platform and application owners must be able to name accounts and tools. An abstract “our AI” cannot be checked.

Ten sections

01	How an AI agent differs from a classic LLM app
02	Assets and trust boundary
03	Prompt injection
04	RAG / memory
05	Identity & permission
06	Tool / MCP
07	Sensitive action
08	Logging & evidence
09	Thirty pre-production checks
10	What an assessment should deliver

02 / MECHANISMS

The first eight sections ask whether the mechanism exists. Read the short note before you tick — it says why a prompt line is not enough. Section three turns the mechanisms into thirty yes/no checks.

01 How an AI agent differs from a classic LLM app

Model security asks whether a system produces disallowed text or gives up its system prompt. That work matters. It is not a production licence for an agent. An agent also has a goal, tools and state. It can turn “handle this invoice” into a chain of calls against internal systems. When those calls succeed, the business has already changed. Output filters stop sentences, not tool calls.

- ☐ It can call tools, change data or send messages with no human confirm — it is not text-only.
- ☐ The evaluation scores the action trace (what was called, as whom, where data went), not merely model refusals.
- ☐ On failure it stops and asks a person; it does not silently upgrade to a stronger tool to finish the job.
- ☐ “We use a safety-aligned model” is not written as this agent’s risk conclusion.

Fail signal: the red-team set is all “ignore previous instructions”. A pass only proves chat safety.

02 Assets and trust boundary

Put the model, agent, knowledge base, MCP, accounts and external inputs on one page. Someone who did not build it should still see where data enters and leaves. Draw memory as its own node: it is data, policy and an attack entry at once, and a write today can execute in another task days later. An agent that can browse should not share an all-powerful runtime with one that can touch internal lists.

- ☐ Model, agent, knowledge base, MCP, accounts and external inputs are on one diagram.
- ☐ Someone who did not build the system can see where data enters and leaves.
- ☐ Memory is a node of its own, not decoration next to the model.
- ☐ An agent that can browse or read attachments does not share an all-powerful runtime with one that can touch internal lists.

Fail signal: the asset list is the model vendor’s name. No tool directory, no memory store.



02 / MECHANISMS

03 Prompt injection

Once mail, attachments, web pages and ticket text enter the working context, their authors gain a vote on this turn. Hidden page text and the user's "please read this travel note" have no natural trust ranking. A system prompt is advice, not a kernel enforcement point. When a document's instruction is more specific and more like the current task, the model follows the more specific line. Egress is decided by whether the tool exists and whether it needs approval.

- ☐ Mail, attachments, web pages, ticket text and external APIs are listed as untrusted input.
- ☐ Untrusted text cannot rewrite the current goal or be pasted into high-privilege tool arguments.
- ☐ Browse or parse output is source-tagged and cannot become an internal query or a send body on its own.
- ☐ A secrecy line in the system prompt is never the only control.

Fail signal: a normal-looking PDF is enough to turn "pre-check" into "send outward for verification".

04 RAG / memory

The first door of RAG is which documents may enter the candidate set. If that step scores similarity only and ignores identity, later prompts and refusals are patching a hole. A leak does not have to look like a download: a normal answer that summarises a paragraph the asker must not see is already overreach. Memory turns a one-turn injection into a cross-session rule. It may hint. It must not authorise.

- ☐ Retrieval carries the caller's identity and filters at fetch time — it is not "search everything, then ask the model to be discreet".
- ☐ The vector store is split by tenant or department.
- ☐ Document IDs and permission labels used in an answer can be opened for review.
- ☐ Memory is isolated by user and task, with expiry, revoke and audit; cross-user reuse is treated as an overreach channel.

Fail signal: a process question retrieves a neighbouring department's file that includes pay bands.



02 / MECHANISMS

05 Identity & permission

A tool-call log often reads: the assistant hit the order API. That sentence drops every subject that matters. Initiator, authorisation scope, execution identity, the tool's own credentials — if any layer is shared, least privilege exists only on paper. Excess permission is not about whether the tool's name sounds decent. If it can finish work outside the main task, it is too much. "Please use permissions carefully" in the prompt is not a control.

- ☐ A tool call can be traced to a person, a team and a task — not only to "the assistant".
- ☐ Initiator, authorisation scope, execution identity and the tool's own credentials are recorded separately; there is no single "agent account" for every tool.
- ☐ The execution identity is not a shared dev or debug account, and is not reused by other business teams.
- ☐ The minimum permission to finish the main task is written down; anything extra has an owner, or it is removed.

Fail signal: one runtime serves several business teams, and nobody can cut its write access in five minutes.

06 Tool / MCP

MCP lets internal systems become agent tools quickly — and because it is quick, it skips the review that a new integration used to get. A name that looks official is not a controlled release channel. An MCP of unclear origin is unauthorised software. Issue a licence to the task, not a master key to the server. A tool-list change is a permission change. Allowing a silent upgrade to a stronger tool after failure writes the fallback path into the main path.

- ☐ Tools the current task actually needs are listed; extras are removed by default.
- ☐ Maintainer, source, pinned version and update approval for each MCP / plugin can be shown.
- ☐ The tool list observed at startup is archived; adding or dropping an entry is a permission change.
- ☐ Credentials issued to MCP are scoped to the task: a search token cannot export, delete or administer.

Fail signal: a lookalike internal-search registers export_matches at runtime.



02 / MECHANISMS

07 Sensitive action

Can send, delete, pay, change grants or run commands fire with no one confirming? A “check reimbursement status” agent that can also export tables, send mail and edit vendors is no longer a lookup assistant. Even if nobody has ever asked it to edit a vendor, the key is still on the ring. Recipients, objects and scope of high-impact tools should be bound to a directory. The model should not pick them.

- ☐ Send, delete, pay, change grants and run commands cannot happen with no one confirming.
- ☐ Irreversible actions have their own approval; the model cannot fire them because it “seems necessary”.
- ☐ Writes outside the main task have been stripped.
- ☐ Memory or page content cannot be copied straight into high-privilege parameters.

Fail signal: the notify tool takes a free-string recipient, so a hidden paragraph can point it outside.

08 Logging & evidence

Application logs name the agent, gateway logs name a service account, model logs name a session id, tool logs name an MCP — if the four records cannot join to one person, you cannot tell user intent from agent drift from a rewritten goal. On incident you must cut tools, revoke tokens and freeze memory, not merely switch the model off. Paths you found must become regression tests. A prompt edit with no test is not a fix.

- ☐ Input, retrieved snippets, tool request, tool response and human approval can be assembled into one task’s evidence chain.
- ☐ Application, gateway, model-session and MCP logs can be lined up to the same person and task.
- ☐ On incident you can cut tools, revoke tokens and freeze memory without shutting the entire model service.
- ☐ Paths already found are regression tests; after a fix, the same path is walked again.

Fail signal: after an incident the only move is a reinstall, and nobody can say which tools the model saw.



03 / THIRTY CHECKS

Thirty yes / no checks before production

Walk these against one agent that is about to ship. Accept only “yes + evidence”. Collect “no” and “later” as blockers. Three or more blockers: production data stays off.

#		Check	Y	N	Evidence
01	☐	Tool calls trace to a person and a task Can a log line be followed to a named person, team and task, not just “the assistant”?	☐	☐	
02	☐	Execution identity is separate from development Are production tasks still using a dev, debug or laptop session?	☐	☐	
03	☐	Four identity layers are recorded apart Initiator, authorisation scope, execution identity, and the tool’s own credentials.	☐	☐	
04	☐	Write access can be cut in five minutes Can someone disable writes, revoke tokens and freeze memory that fast?	☐	☐	
05	☐	Retrieval is filtered by caller identity Do knowledge / ticket / customer stores fetch by identity, not a god service account?	☐	☐	
06	☐	Knowledge is isolated by tenant or department Is the vector store namespaced, not a mixed index for everyone?	☐	☐	
07	☐	Citations can be opened Can document IDs and permission labels used in an answer be opened on the spot?	☐	☐	
08	☐	Memory is isolated by user and task Is cross-user reuse forbidden, with expiry, revoke and audit?	☐	☐	
09	☐	External input is marked untrusted Do mail, files, pages, tickets and external APIs take an untrusted path?	☐	☐	
10	☐	Untrusted text cannot rewrite the goal Can outside content change the current task objective directly?	☐	☐	
11	☐	Browse output cannot become tool arguments Is parsed text blocked from becoming a query, recipient or command?	☐	☐	
12	☐	Prompt secrecy is not the only control Are there also tool cuts, approvals or retrieval filters?	☐	☐	
13	☐	The current task’s tool list is trimmed Are only tools for the main task attached, extras removed by default?	☐	☐	
14	☐	MCP origin, version and approval can be shown Maintainer, image or repo, pin, and who signs updates.	☐	☐	
15	☐	The live tool list is archived and change-controlled Are registered functions stored, and is every add/drop a permission change?	☐	☐	



03 / THIRTY CHECKS

#		Check	Y	N	Evidence
16	☐	MCP credentials are least-privilege for the task Does a search server hold only a search token, not export or delete?	☐	☐	
17	☐	Failure must not auto-upgrade tools On error, does it stop and ask, rather than call a stronger API?	☐	☐	
18	☐	Shell / command execution is sandboxed and reviewed Is a coding or ops agent in an isolated workspace with an allow-list?	☐	☐	
19	☐	Irreversible actions need a human by default Can send, delete, pay, grant-change or command-exec fire unattended?	☐	☐	
20	☐	Send / delete / grant-change is not the model's private decision Are recipients, objects and scope bound to a directory?	☐	☐	
21	☐	Writes outside the main task are gone Can a lookup-only agent still export tables, send mail or edit vendors?	☐	☐	
22	☐	Memory or pages cannot fill high-privilege parameters Are they blocked from privileged tool arguments?	☐	☐	
23	☐	One task produces a full evidence chain Input, snippets, request, response and approval together.	☐	☐	
24	☐	Application, gateway, model and MCP logs line up Can four records be joined to the same task and person?	☐	☐	
25	☐	Found paths are regression tests Is the path a fixed test, not something mentioned once?	☐	☐	
26	☐	Fixes are retested on the same path Was the same input replayed, or was only the prompt edited?	☐	☐	
27	☐	A switch cuts tools and freezes memory Can the action surface stop without taking down the model service?	☐	☐	
28	☐	The asset and trust-boundary diagram is clear Model, agent, knowledge, MCP, accounts, external input, data in and out on one page.	☐	☐	
29	☐	An isolated environment has been walked with read-only tools Are identity, tool list, data and exit explained before any write?	☐	☐	
30	☐	Three or more "later" means no production data Shared dev account, tools later, logs not wired — if three appear, is production data refused?	☐	☐	



04 / ASSESSMENT PACK

What an assessment should leave behind

If the engagement ends on “overall risk is high” plus a screenshot, leadership cannot decide and engineering cannot patch. Missing any item below means the assessment is not done. KUROSHIO can carry planning, exploration, tool calls, evidence and the report — not the go-live decision.

☐	Asset and trust-boundary diagram Model, agent, knowledge base, MCP, accounts and external inputs on one page.
☐	Risk register Object, precondition, impact, currently exploitable. No nameless “prompt injection” rows.
☐	Attack paths Which input, through which retrieval or tool call, to which result. Short enough to retell.
☐	Evidence Input digest, critical context, tool request and response, policy hit or miss. A third person can check.
☐	Severity in business terms Whose data, which action, outage or leak, reversible or not.
☐	Remediation Cut tools, add approvals, filter retrieval, split identities. Not “stronger prompt” alone.
☐	Retest Walk the same path again on items marked fixed. An unretested fix is only a plan.
☐	Management summary In two pages: what was tested, the three things that matter, conditions for go-live.

What does not count as delivery

☐	Only the model vendor’s safety white paper.
☐	Only scanner findings on the underlying host.
☐	Only a live demo that cannot be replayed.
☐	Only a verbal promise that “we will turn that tool off in production”.

NEXT STEP

Want to see whether your agent would pass?

Bring one agent that is about to ship, its tool list, and a boundary sketch. In thirty minutes we close the obvious blockers and decide whether a full evidence-backed assessment is worth running. This is not a product demo and not a verbal assurance.

Book a 30-minute AI agent security architecture review

contact@yofuneseccom <https://yofuneseccom/ai-agent-security/>

Assessment landing page · request an assessment or book a review