the unofficial

AI Security Baseline Checklist

⚒ AI Craftspeople Guild • Vendor-neutral, drawn from the open standards • What "secure AI-assisted development" means in practice

The foundation

Establish these first. Without them you can't even tell whether you're secure — you're just hoping.

- [ ] You know **where AI generates code** and **where agents run** across your stack
- [ ] A human **owns accountability** for security — not the model, not "the tool"
- [ ] You can name **which standards apply** to your architecture (and which don't)
- [ ] Security is verified **continuously**, not once before launch and never again
- [ ] Verification is **independent** of whoever sells your tooling

The code layer — OWASP LLM Top 10

Securing what AI generates. Most flaws here are semantic & behavioral — classic SAST misses them.

- [ ] **Prompt-injection exposure**
  - [ ] Untrusted input can't redirect model behavior
  - [ ] No direct path from user input to privileged action
- [ ] **Insecure output handling**
  - [ ] Model output is validated, not trusted by default
  - [ ] Generated code reviewed before it touches production
- [ ] **AI-generated vulnerabilities**
  - [ ] Insecure defaults from outdated training examples caught
  - [ ] Input validation present, not assumed
  - [ ] Secrets never hard-coded by the assistant
- [ ] **Supply-chain & dependencies**
  - [ ] AI-suggested packages verified to exist & be maintained
  - [ ] No known-vulnerability or typo-squatted dependencies
  - [ ] An AI Bill of Materials (AIBOM) is maintained

The agent layer — OWASP Agentic Top 10 + SAFE-MCP

The hardest, newest risks. Agents decide, chain tools & mutate systems over time. Static checks don't reach here.

- [ ] **Identity & privilege**
  - [ ] Each agent has its **own identity** — not a borrowed human session
  - [ ] Permissions are task-scoped & time-bound (least privilege)
  - [ ] No long-lived secrets riding on inherited admin access
- [ ] **Goal & behavior integrity**
  - [ ] Goal-hijacking paths assessed & bounded
  - [ ] Memory / context poisoning considered
  - [ ] Runaway autonomy & cascading-failure risks bounded
- [ ] **Tool & action control**
  - [ ] Tool misuse paths reviewed (what can it actually invoke?)
  - [ ] High-consequence actions require authorization
  - [ ] Agent actions in production are auditable after the fact
- [ ] **MCP / protocol layer** *(if using MCP)*
  - [ ] Model misbinding & context spoofing assessed
  - [ ] MCP servers meet the SAFE-MCP baseline
  - [ ] Tool-chain integrity verified end to end

The practice

Security isn't a state you reach — it's a discipline you keep. These make it continuous.

- [ ] **Cadence matches your work** — re-reviewed each AI-assisted sprint, not annually
- [ ] **Fast-changing surfaces watched** — agent permissions, new tools, new deps, prompt changes
- [ ] **Tooling is verified, not trusted** — your scanner is checked for blind spots, not assumed correct
- [ ] **Findings are recorded** with owners & remediation priority
- [ ] **Baseline evolves** with the standards — you're assured against today's canon, not last year's
- [ ] **Regulatory exposure mapped** where relevant (e.g. EU AI Act high-risk obligations)

Positive indicators

Signs of a genuinely secure AI-assisted practice — culture, not just controls.

- [ ] The team can **explain its threat model**, not just list its tools
- [ ] "The tool didn't flag it" is never accepted as a security answer
- [ ] Someone independent has **actually verified** the posture recently
- [ ] New agents are scoped tightly **by default**, loosened only on purpose
- [ ] Security keeps pace with delivery speed — not lagging behind it

The Guild position

- [ ] Security you cannot **independently verify** is not security — it's a vendor's promise about its own product. Buy the tooling. Then have someone with nothing to sell confirm it reads true.

SAST = Static Application Security Testing • **MCP** = Model Context Protocol • **AIBOM** = AI Bill of Materials • **Least privilege** = grant only the access a task needs • **OWASP / NIST** = open, vendor-neutral standards bodies

This checklist is a **discussion tool, not a compliance scorecard.** Drawn from OWASP Agentic Top 10, OWASP LLM Top 10, SAFE-MCP & the NIST AI Agent Standards Initiative. Inspired in form by Henrik Kniberg's "unofficial Scrum Checklist."

⚒ *AI Craftspeople Guild*

AI Security Baseline Checklist — How to use it

What is this? Who is it for?
A vendor-neutral map of what "secure AI-assisted development" actually means — assembled from the open standards (OWASP, SAFE-MCP, NIST), owned by no company. AI-generated code is vulnerable often enough, and agents change their own behavior fast enough, that buying a tool and running one audit isn't security — it's a snapshot that expires. **These aren't rules. They are guidelines.** A system with no embedded agents skips the entire agent layer; a project not using MCP skips that section. Fine — as long as you skipped it *on purpose*, knowing it didn't apply, rather than because nobody looked. That judgment is the whole point.

How do I use it?

Use it as a **discussion tool** — ideally as a recurring review, because the agent layer especially changes fast.

> **Maya:** "We bought a security platform and it's green across the board. Are we good?"
>
> **Dev:** "On the code layer, mostly. But our agent runs on my admin token — it inherits everything I can do."
>
> **Maya:** "That's 'identity & privilege' in the agent layer. The scanner won't flag it — it's an architecture choice, not a code bug. Least privilege says that agent needs its own scoped identity."
>
> **Dev:** "And nobody outside our team has actually checked any of this."
>
> **Maya:** "That's the foundation row: independent verification. A green dashboard from the tool we bought isn't the same as someone with nothing to sell confirming it."

That's the right use: finding the risks your tooling structurally can't see, and deciding what to do about them — deliberately.

Why a tool & an audit aren't enough

General-purpose security built for static, human-paced software is insufficient when autonomous agents make decisions, chain tools, and mutate systems over time. Many AI flaws are semantic and behavioral, not structural — so classic static analysis and one-off audits miss them. And a tool can't impartially grade itself. The baseline exists to cover what the tooling leaves out.

How do I *NOT* use it?

Don't let it become a procurement checkbox or a vendor's marketing prop.

> **Big Boss:** "Our vendor's dashboard is 100% green, so report us as fully secure to the board."
>
> **Dev:** "The dashboard only covers the code layer. The whole agent layer is outside what that product even looks at."
>
> **Big Boss:** "The tool's the industry leader. If it's green, we're green. Sign it off."
>
> **Maya:** "A tool reporting on its own coverage isn't independent verification — that's exactly the gap this baseline names. Green means 'what we measure is fine,' not 'we're secure.'"
>
> **Big Boss:** "It says green, so we're done. No exceptions."

A vendor's dashboard answers "is my product happy." The baseline answers "am I actually secure against the standards that apply to me." Those are not the same question — and only one of them is the point.

Is this an official standard?

No — and deliberately so. It doesn't compete with OWASP or NIST; it *assembles* them into one practitioner-grade reference, owned by no vendor. Argue with it, adapt it to your stack, experiment. If it surfaces the risk your green dashboard couldn't see, it's done its job.

The one principle

Security you cannot independently verify is not security. It is a vendor's promise about its own product.

Standards referenced: OWASP Top 10 for Agentic Applications, OWASP Top 10 for LLM Applications, SAFE-MCP (Linux Foundation), NIST AI Agent Standards Initiative. Inspired in form by Henrik Kniberg's "the unofficial Scrum Checklist."