5 things you need to know before using OpenClaw

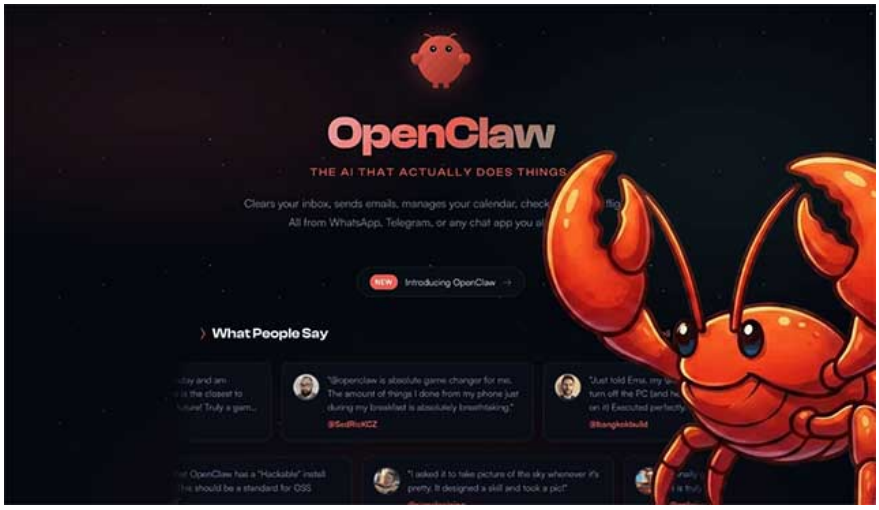
OpenClaw is a powerful AI agent framework, but it also carries security risks. Learn these 5 important things before deploying OpenClaw securely.

OpenClaw is currently one of the most powerful open-source agent frameworks available. But it's important to understand that it's not just an upgraded chatbot.

OpenClaw operates as a true system, with its own gateway, the ability to configure executable skills, connect external tools, and even perform actions directly on the system or messaging platform.

Therefore, when using OpenClaw, you are no longer 'playing with AI,' but operating a system with real risks. Without careful setup, you could encounter security issues, data breaches, or even lose control of the system.

Here are five important things you need to understand before implementation.



OpenClaw is not an app, it's a server.

As soon as you run OpenClaw, you start a gateway – the place that connects the model, tools, and communication channels. If you open it to the network, you are essentially running a server that can be attacked.

This means you need to approach OpenClaw as a true backend system, not just a simple application.

In the initial phase, it's best to keep the system running locally, check logs regularly for unusual behavior, and run security audits after each configuration change. These steps may sound 'overkill,' but they are actually the minimum standard when working with an agent that has execute privileges.

Skills are not plugins – they are executable code.

A common misconception is that 'skills' in OpenClaw are the same as harmless plugins. The reality is quite the opposite.

Skills are code that can execute commands, access files, trigger workflows, and interact directly with the system. This makes them extremely powerful, but also opens up risks related to the 'supply chain'.

There have been instances of malicious skills being uploaded to repositories like ClawHub, exploiting social engineering to trick users into executing dangerous commands. Therefore, the installation of these skills needs to be strictly controlled.

The safest approach is to only install skills you truly need, carefully read the documentation before use, and check for security warnings. If a skill requires running long, confusing shell commands, that's a sign you should stop immediately.

The model you use determines the level of safety.

Unlike typical chatbots, the model in OpenClaw not only 'responds', but also **makes decisions and triggers actions** .

Using a weak or unsuitable model can lead to problems such as calling the wrong tool, executing incorrect commands, or misinterpreting the context when multiple tools are running simultaneously.

Therefore, model selection is no longer about 'answer quality', but about system security.

In the context of 2026, models such as Claude Opus 4.6, GPT-5.3-Codex, or GLM-5 are highly regarded for workflow agents.

More importantly, you should clearly control which models are allowed to use the tool and which are only for text generation, avoiding inadvertently granting excessively high privileges to untrustworthy models.

Protecting your secrets is more important than you think.

The biggest risk when using OpenClaw doesn't lie in the model or the skill, but in **the sensitive information** .

Because OpenClaw often runs near critical resources such as API keys, tokens, SSH keys, or login sessions, if this data is exposed, an attacker doesn't need to hack the system – they can simply reuse your information.

Therefore, the most important principle is never to store secrets as plain text in config or skill files. Instead, use environment variables or system-based secret management.

Additionally, the OpenClaw workspace should be kept as limited as possible, avoiding mounting entire personal directories. For critical systems, it's best to run them in separate containers or virtual machines to isolate risks.

Voice calls are a real power… and a real risk.

One of OpenClaw's coolest features is its ability to make voice calls. But it's also its riskiest feature.

When voice calls are enabled, agents can interact not only through text, but also by making phone calls and communicating directly with people. This opens up many practical applications, but also entails operational risks and costs.

Without proper control, agents can make wrong calls, spam, or even create awkward communication situations.

Therefore, voice-related actions should be considered 'high-permission,' equivalent to payment or admin privileges. It is necessary to clearly define who can call, when they can call, and whether human approval is required.

Conclude

OpenClaw is not just an AI tool, but a platform with real execution capabilities. It can connect systems, automate workflows, and operate across multiple channels.

But that very power also demands a more serious approach.

If you treat OpenClaw as part of an infrastructure that controls skills, selects appropriate models, protects secrets, and limits sensitive actions, it can become an extremely powerful platform for building automated AI systems.

The future of AI agents lies not only in intelligence, but also in the ability to execute securely and reliably. And with OpenClaw, you hold both that opportunity and that responsibility.