

2026 Agent Skills

技术与安全白皮书

主编单位：中科算网算泥社区
2026年4月

目 录

前言	1
白皮书背景与写作目的	1
第 1 章 Agent Skills 的起源与发展	2
1.1 AI 能力范式的演进	2
1.2 MCP 协议提出	3
1.3 Skills 概念的诞生：从痛点到方案	3
1.4 Anthropic 官方发布时间线	4
1.5 生态的爆发：从编码工具到全场景渗透	5
第 2 章 什么是 Agent Skills	6
2.1 官方定义与核心理念	6
2.2 基础文件结构详解	7
2.2.1 SKILL.md：双层结构解析	8
2.2.2 scripts/：行动力目录	9
2.2.3 references/：领域知识库	10
2.2.4 assets/：视觉与素材资源	10
2.3 渐进式披露：核心设计原理	10
2.4 Agent 如何发现、选择、使用与组合 Skills	12
2.5 五大标准设计模式	13
2.6 Skill 工程实践建议	19
第 3 章 为什么需要 Agent Skills	20
3.1 范式价值：从“单一智能”到“模块化智能”	20
3.2 知识沉淀：将经验固化为可执行的“能力 DNA”	21
3.3 跨平台能力迁移：“一次编写，到处运行”	21
3.4 人机关系逆转：从“人适应 AI”到“AI 适应人”	22
3.5 典型 Skill 类别与应用场景	22
3.6 Skills 市场涌现	23
3.7 对人力市场与劳动结构的影响	24
第 4 章 热门 Skill 精选介绍	25
4.1 元技能（Meta Skills）	25
4.2 专家经验包	25

4.3 前端设计与开发	26
4.4 多媒体与视频创作	27
4.5 浏览器与信息检索	27
4.6 科技趋势与深度研究	27
4.7 云平台与数据库	28
4.8 AIGC 创意生成	28
4.9 代码质量与交付	28
4.10 云端办公自动化	29
4.11 文档处理	29
第 5 章 安全治理	29
5.1 四阶段攻击面分析	29
5.1.1 Creation（创建阶段）	29
5.1.2 Distribution（分发阶段）	30
5.1.3 Deployment（部署阶段）	30
5.1.4 Execution（执行阶段）	31
5.2 三大结构性安全缺陷	31
5.3 典型安全事件：OpenClaw 与 ClawHub 案例	32
5.4 OWASP Agentic Skills Top 10	33
5.5 Skills 供应链安全工具链	34
5.6 运行时与 Zero Trust 模式安全原则	34
5.7 未来安全架构方向	35
第 6 章 未来展望：趋势、挑战与机遇	35
6.1 七大开放性挑战	35
6.2 技术演进方向	36
6.3 生态	37
6.4 人机协作的长期展望	37
6.5 行动建议	38
References	39

前言

白皮书背景与写作目的

2025 年 10 月，Anthropic 在 Claude Code 中悄然引入了一个新概念——Agent Skills。当时，可能连 Anthropic 自己都没有完全预见到，这个看似简单的“文件夹加 Markdown”的设计，会在接下来不到一年的时间里，彻底重塑 AI Agent 的能力组织方式。

2025 年 12 月 18 日，Anthropic 将 Agent Skills 规范以开放标准形式正式发布，上线 agentskills.io，微软在 VS Code 和 GitHub 中直接集成，OpenAI 在 Codex CLI 和 ChatGPT 中采用了几乎一模一样的架构，Cursor、国内 Trae、Qoder 等主流编码工具迅速跟进。

与此同时，安全警报也同步拉响——安全研究团队在社区市场 ClawHub 上发现了数百个恶意 Skill，AI Agent 供应链攻击拉开序幕。

截至 2026 年 4 月，大型 Agent Skills 市场 skillsmp.com 已收录超过 90 万个 Skill。

Agent Skills 正在从一个开发者工具，演变为一个行业基础设施。它带来了巨大的生产力提升可能，也带来了前所未有的安全挑战。

在此背景下，作为国内领先的 AI 大模型开发服务平台，算泥社区秉持“技术专业、生态开放、开发者友好”的理念，联合社区众多资深分析师与技术专家、学者，共同撰写并发布《2026 Agent Skills 技术与安全白皮书》。这份白皮书的目标，就是在这一历史性转折点上，为技术人员、企业决策者、安全从业者和生态建设者提供一份系统、深入、可操作的参考指南。

我们试图回答五个核心问题：

Agent Skills 从何而来，解决了什么问题？

Agent Skills 是什么，它的核心设计原理如何工作？

为什么 Agent Skills 正在成为 AI Agent 能力组织的标准范式？

目前有哪些关键的 Skills 项目和生态资源？

Agent Skills 带来了哪些安全挑战，以及如何应对？

第 1 章 Agent Skills 的起源与发展

1.1 AI 能力范式的演进

2022 年底 ChatGPT 横空出世后，全世界开始学习如何和 AI“说话”。我们发明了“提示词工程”这个概念，尝试用越来越长的 Prompt 引导模型做出我们想要的行为。Zero-shot、Few-shot、Chain-of-Thought、ReAct……各种 Prompt 技巧层出不穷。

Prompt 范式特征：一次性自然语言指令 + 少量上下文示例。用户需要在一段 Prompt 里告诉模型所有信息：你是谁、你要做什么、怎么判断、输出什么格式。

核心局限：

指令不可复用：每次开始新对话，同样的 Prompt 要重新粘贴一遍。写一个高质量的 Prompt 本身就是脑力劳动，但这份劳动成果只能存在自己的备忘录里。

知识碎片化：团队里每个人都有一套自己的 Prompt，质量参差不齐。好的 Prompt 无法沉淀为组织资产。

上下文窗口瓶颈：长 Prompt 吃掉大量 token，直接推高成本和延迟。更致命的是，上下文越长，模型在关键信息上的注意力越容易稀释。

2023 年，OpenAI 推出了 Function Calling 能力，随后 Anthropic、Google 等主流模型厂商纷纷跟进标准化工具调用接口。这是一个质的飞跃——模型不仅能“说”，还能“做”。

Function/Tools Calling 的优势：外部能力调用标准化。模型可以通过结构化的 JSON schema 调用数据库、搜索 API、执行系统命令。MCP (Model Context Protocol) 在 2024 年 11 月由 Anthropic 推出，进一步标准化了工具发现、鉴权和上下文传递流程。

核心局限：工具只解决“能做什么”，无法封装“应该怎么做”。Function Calling 定义的是工具的“接口”，不是工具的“用法”。举个例子：模型知道有一个 `execute_sql` 工具可以调用，但它不知道你的数据库 schema 长什么样、查询应该遵循什么规范、什么样的查询在这个业务场景下是“对”的。这些知识，每次都得写在 Prompt 里。

Skills 的范式跃迁：它把“过程性知识” (procedural knowledge) ——即“在特定场景下应该如何做事”——封装成了独立、可复用、可组合的模块。这不是给

模型增加一个工具，而是给模型安装一份“专业操作手册”。

这个范式的核心口号是：“Don't Build Agents, Build Skills Instead.”——不是去造一个什么都会的全能 Agent，而是造一堆专业的 Skills，让 Agent 按需加载。

1.2 MCP 协议提出

2024 年 11 月，Anthropic 正式发布了 Model Context Protocol (MCP)。这是一个开源协议标准，旨在标准化 AI 模型与外部工具、服务和数据之间的交互方式。

MCP 的设计灵感来自 Language Server Protocol (LSP)——就像 LSP 让任何编辑器都能获得智能代码补全能力一样，MCP 让任何 AI Agent 都能通过统一接口调用任何外部工具。它采用结构化的 JSON-RPC 协议，定义了工具发现、数据检索、命令执行和提示模板等标准化交互模式。

MCP 解决了 AI Agent 与外部系统集成的三大痛点：

工具发现：Agent 可以自动发现 MCP Server 提供的所有工具，无需人工配置。

鉴权标准化：统一的认证授权流程，不再需要为每个工具单独实现鉴权逻辑。

上下文传递：标准化的上下文交换格式，确保 Agent 和工具之间能够顺畅沟通。

MCP 解决了“能调用什么”的问题，但没有解决“应该怎么调用”的问题。以数据库操作为例：MCP 可以让 Agent 连接到 PostgreSQL，但 Agent 仍然不知道：

这个数据库的表结构和关系是什么？

查询应该遵循什么命名规范？

什么样的 SQL 在这个业务场景下是被认为“安全”的？

常见的分析场景应该怎么下 SQL？

这些知识——项目的技术规范、团队的最佳实践、行业的 SOP——仍然散落在各个项目的 README、Wiki、Confluence 和每个人的长 Prompt 里。MCP 给了 Agent“手脚”，但“脑子”里的专业知识还是空的。

1.3 Skills 概念的诞生：从痛点到方案

AI Coding / Agent IDE 的快速崛起

2024 到 2025 年间，AI 编程工具进入爆发期。Claude Code、Cursor、GitHub

Copilot Workspace、Codex CLI、Gemini CLI 等工具让开发者开始习惯“让 AI 写代码”。Computer Use 功能更是让 AI 获得了直接操作电脑的能力——移动鼠标、点击按钮、输入文字，AI 有了真正的“手脚”。

但问题也随之而来。开发者们发现，虽然 AI 能写代码了，但每次都要先教它一堆东西——项目的目录结构、团队的技术栈选择、代码风格规范、测试框架用法、部署流程……这些“常识”对 AI 来说每次都是新知。开发者 Jesse Vincent 最早意识到这个问题的严重性：“开发者们发现，虽然因为 Computer Use 的诞生，让 AI 有了‘手脚’，但 AI‘脑子’里缺少针对特定任务的专业 SOP（标准作业程序）。”

核心痛点三连

重复教 AI 项目结构：每次开新会话，都要重新告诉 AI“我们这个项目的目录是这样的，配置文件在哪儿，测试用例在哪儿”。信息传递成本极高。

团队经验无法标准化：资深工程师脑子里有一整套“在这个项目里怎么把事做对”的经验——代码审查的检查点、部署的 checklist、文档的格式要求。这些经验没法自动传承给新成员，更没法让 AI 自动继承。

跨产品迁移成本高：你在 Claude Code 里写好的 Prompt，没法直接搬到 Cursor 里用。每个工具的集成方式不同，知识被锁死在特定平台上。

Skills 的类比定位

LLM ≈ CPU：提供通用计算能力

Tools ≈ 驱动层：让 CPU 能连接外部设备

Skills ≈ 行业 SDK / Playbook 集合：封装特定领域的专业知识和工作流程，告诉 CPU 在这个行业里具体怎么干活

这个类比解释了为什么 Skills 如此重要——它补上了从“通用智能”到“专业能力”之间最关键的那一层。正如 PC 时代的 SDK 让开发者不用从底层写起，Skills 让 Agent 不用从零开始学习每个领域的专业知识。

1.4 Anthropic 官方发布时间线

时间	事件
2025-10-16	Agent Skills 正式发布：Anthropic 在工程博客和 Claude 博客发布《Equipping agents for the real world with Agent Skills》《Introducing Agent Skills》，提出基于 SKILL.md 的 Agent Skills 概念，并宣布 Skills 在 Claude.ai、Claude Code 与 Claude API 中可用，用于将指令、脚本和资源打包成可复用的“能力模块”。
2025-12-18	开放标准 & 企业级能力 & 合作伙伴目录：Claude 博客《Skills for

	organizations,partners,the ecosystem》发布，Skills 支持组织级集中管理，上线 Skills Directory，收录 Notion、Canva、Figma、Atlassian 等合作伙伴技能；同日及相关工程文更新中，Anthropic 宣布 Agent Skills 规范作为开放标准发布并在 agentskills.io 公布，支持跨平台可移植。
2025-12-19	Skills × MCP 架构公布：官方博客《Extending Claude’s capabilities with skills and MCP servers》解释 Skills 如何与 MCP 联动：MCP 负责外部工具与数据连接，Skills 负责把这些能力固化为可复用、可治理的工作流，给出财务分析、Notion Meeting Intelligence 等典型场景。
2026-03-03	Skill-creator 升级：评估与 2.0 能力（社区称 Skills 2.0）：Claude 博客《Improving skill-creator:Test,measure,and refine Agent Skills》发布，Skill-creator 新增结构化 evals、benchmark mode、多代理并行测试、A/B 对比和触发优化，标志着 Skills 从“写出来感觉不错”进入“可度量、可回归、可演进”的工程化阶段。

1.5 生态的爆发：从编码工具到全场景渗透

起源于 Claude Code 编程场景

Agent Skills 最初是作为 Claude Code 的扩展机制设计的，解决的是编程场景下“让 AI 更懂项目”的问题。早期 Skills 大多围绕代码审查、测试自动化、Git 工作流、文档生成等开发者场景。

非编程场景的病毒式传播

2026 年 1 月是 Skills 生态的转折点。非技术用户开始涌入，将 Skills 用于大量非编码场景：

度假研究：自动规划行程、预订酒店、整理景点攻略

PPT 制作：根据数据自动生成演示文稿

邮件清理：智能分类、批量处理、生成回复草稿

表单自动化：自动填写各类表格

甚至有人开发了控制烤箱的 Skill

一位科技行业资深分析师总结道：“当你把一堆报销单扔给它时，它会默默启动‘Expense-Audit’ Skill，自动调取 OCR、校验税号、生成报表。这种‘挂载即用’的体验，让 Skills 成了 2026 年职场人最梦寐以求的‘外挂’。”

行业标准格局初定

Anthropic 发布 Skills 开放标准后，行业跟进速度快得惊人：

微软在 VS Code 和 GitHub 中直接集成

OpenAI 在 Codex CLI 和 ChatGPT 中采用了几乎相同的架构

Cursor 成为首个全面支持 Agent Skills 的 AI IDE

国内阿里 Qoder、字节 Trae、腾讯 CodeBuddy 纷纷支持 Skills

第 2 章 什么是 Agent Skills

2.1 官方定义与核心理念

Anthropic 对 Agent Skills 的官方定义是：

“Skills are organized collections of files that package composable procedural knowledge for agents.”

这句定义值得逐词拆解，因为它精准地揭示了 Skills 的本质：

“organized collections of files”—— 文件系统抽象，极低创作门槛

Skill 不是一个复杂的二进制文件或需要编译的插件，它就是一个文件夹。里面放的主要是 Markdown 文本文件和可选的脚本。这意味着：

创建 Skill 的门槛极低——会用文本编辑器就能写

版本控制天然友好——直接放进 Git 仓库，diff、merge、blame 全部可用

分发极其简单——压缩打包或直接 clone 即可

“composable”—— 乐高积木式组合与嵌套

Skills 被设计成可以自由组合。一个复杂的任务可以由多个 Skill 协作完成：Pipeline 模式串联执行、多个 Skill 并行触发、或者一个 Skill 内部调用另一个 Skill 的能力。这种可组合性让 Skills 生态可以像乐高一样快速搭建复杂 workflow。

“procedural knowledge”—— 过程性知识 (How)，而非陈述性知识 (What)

这是最关键的区分。陈述性知识是“事实” (What) ——比如“Python 的列表推导式语法是这样的”。过程性知识是“方法” (How) ——比如“在这个项目里，当你需要过滤一个列表时，应该用列表推导式还是 filter 函数？为什么？代码应该写在哪个文件里？遵循什么命名规范？”

Skills 封装的是后者——那种“在这个具体场景下，应该这么做”的知识。这种知识在传统组织中存在于资深员工的头脑里、散落在各种 Wiki 页面中、口口相传却从未被系统化。Skills 让这种知识第一次可以被“打包”和“分发”。

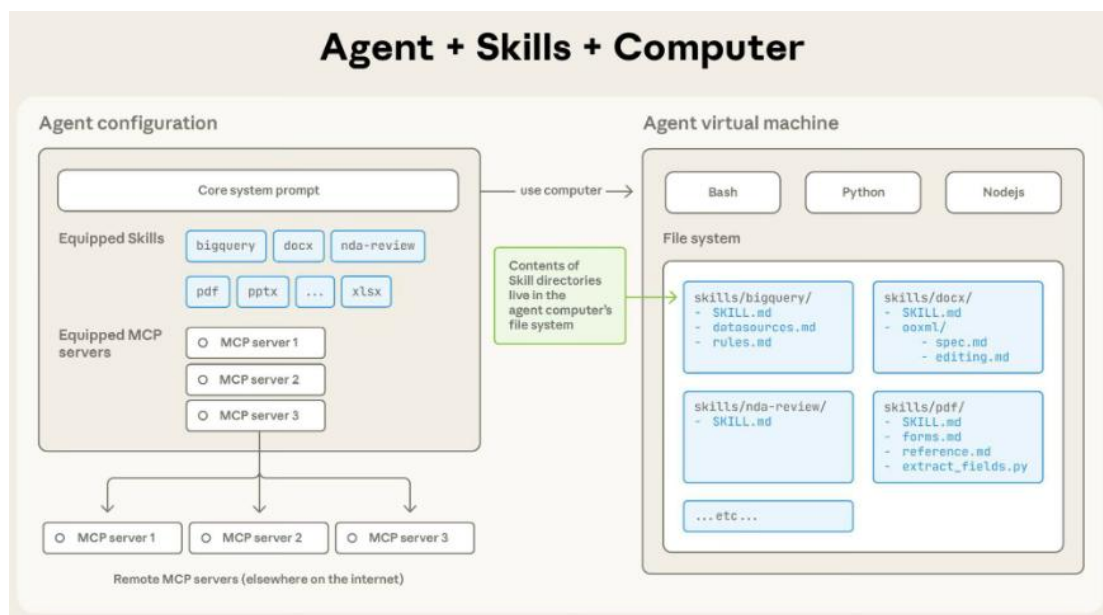
核心理念：Skills 的本质是“可执行的知识”

Skill 不仅仅是文档——它里面可以包含可执行脚本。一个 PDF 处理 Skill 不只是告诉 Agent“你应该用 pypdf 库来处理 PDF”，它直接提供了一个写好的 Python 脚本，Agent 可以直接调用。

Skill 既教 Agent“怎么做”，又提供“做”的具体工具。

2.2 基础文件结构详解

Skill 在整个 Agent 中的位置如下图：



以 Claude 为例，Skills 在代码执行环境中运行，Claude 在该环境中具有文件系统访问权限、bash 命令和代码执行能力。Skills 作为目录存在于虚拟机上，Claude 使用与您在计算机上导航文件相同的 bash 命令与它们交互。

一个 Skill 是一个文件夹，文件夹结构类似下面这样：

```
pdf/
├── SKILL.md           # 主要说明（触发时加载）
├── FORMS.md          # 表单填充指南（根据需要加载）
├── reference.md       # API 参考（根据需要加载）
├── examples.md        # 使用示例（根据需要加载）
└── scripts/
    ├── analyze_form.py # 实用脚本（执行，不加载）
    ├── fill_form.py    # 表单填充脚本
    └── validate.py      # 验证脚本
```

或者这样：

```

bigquery-skill/
├── SKILL.md (概述和导航)
└── reference/
    ├── finance.md (收入、账单指标)
    ├── sales.md (机会、管道)
    ├── product.md (API 使用、功能)
    └── marketing.md (活动、归因)

```

其中只有 **SKILL.md** 是必须的。可选的文件或文件夹可以根据需要调整。

我们以下面这种结构来拆解分析：

```

my-skill/
├── SKILL.md           # 必须：主文件（YAML 元数据 + Markdown 指令体）
├── scripts/           # 可选：可执行代码（Python / Bash / JS 等）
├── references/         # 可选：参考文档（规则、规范、领域知识库）
└── assets/            # 可选：静态资源（icon、图片、输出模板）

```

2.2.1 SKILL.md：双层结构解析

SKILL.md 是每个 Skill 的核心文件，由两部分组成：

A simple SKILL.md file

pdf/SKILL.md

```

YAML Frontmatter
---
name: PDF Processing
description: Comprehensive PDF toolkit for extracting text and tables, merging/splitting documents, and filling-out forms.
---

Markdown

## Overview

This guide covers essential PDF processing operations using Python libraries and command-line tools. For advanced features, JavaScript libraries, and detailed examples, see ./reference.md. If you need to fill out a PDF form, read ./forms.md and follow its instructions.

## Quick Start

... python
from pypdf import PdfReader, PdfWriter

# Read a PDF
reader = PdfReader("document.pdf")
print(f"Pages: {len(reader.pages)}")

# Extract text
text = ""
for page in reader.pages:
    text += page.extract_text()
        
```

YAML 元数据区（Frontmatter）

位于文件开头，用 `---` 包裹，定义 Skill 的元信息：

字段	必需	说明	示例
name	✓	Skill 名称，≤64 字符，只	name:pdf

		能使用小写字母、数字和连字符	
description	✓	用途与触发时机，含关键词，≤1024 字符	description:Comprehensive PDF toolkit for extracting text and tables,merging/splitting documents,and filling-out forms.
license	✗	许可证类型	license:Apache 2.0
compatibility	✗	环境依赖说明	compatibility:Requires git,docker,jq,and access to the internet
allowed-tools	✗	预批准工具白名单，空格分隔	allowed-tools:Bash(git:*) Bash(jq:*) Read
disable-model-invocation	✗	若为 true，仅用户可触发（适用于高风险操作）	disable-model-invocation:true
user-invocable	✗	若为 false，仅 Agent 可触发（适用于背景知识）	user-invocable:false
metadata	✗	任意键值扩展元数据	metadata: author:"xxx" version:"1.0"

Markdown 指令区

位于 YAML 元数据区之后，是 Skill 的核心指令内容。包含：

触发条件说明： 什么情况下应该使用这个 Skill

工作流程步骤： 详细的操作流程，一步一步指导 Agent

工具使用指南： 何时调用哪个脚本或工具

输出格式要求： 期望的输出结构和格式

注意事项和边界： 什么能做、什么不能做、有什么坑

工程实践建议：

控制长度：SKILL.md 指令体建议≤500 行，超出部分移入 [references/](#)

使用结构化标题：便于 Agent 精准定位信息，降低注意力稀释

避免过长的嵌套结构：Markdown 层级不宜过深，Agent 对深层嵌套的解析容易出错

2.2.2 scripts/：行动力目录

scripts/ 目录包含可执行代码——Python 脚本、Bash 脚本、JavaScript 等。

这是 Skill 从“知道”到“做到”的关键。

scripts/ 目录包含多个功能脚本。

这些脚本被设计为 CLI 工具，Agent 可以直接通过命令行调用，传入参数获取结果。这比让 Agent 自己生成代码有两个巨大优势：

1.确定性：脚本是预先写好和测试过的，行为可预期

2.效率：Agent 不需要每次重新生成代码，直接调用即可

安全警示：scripts/ 也是 Skill 最大的安全风险来源。任何可执行代码都可能在用户系统上以用户权限运行。详见第 5 章的安全分析。

2.2.3 references/：领域知识库

references/ 存放只读的参考文档：

规则文档（如代码规范、品牌指南）

领域知识（如 API 文档、数据库 schema）

使用示例和最佳实践

指令（instructions）与参考（references）的分离是重要的设计原则：SKILL.md 中的指令应该是简洁的工作流描述，具体的细节文档放到 references/ 中，通过 Markdown 链接引用。这样既保持指令的可读性，又确保需要时可以查阅详细信息。

2.2.4 assets/：视觉与素材资源

assets/ 存放静态资源：

图标和图片

输出模板（如 Word 模板、PPT 母版）

配置文件样例

用于提升 Skill 的可视化表达与输出标准化。例如 brand-guidelines Skill 可以在 assets/ 中存放品牌 Logo 和配色方案文件。

2.3 渐进式披露：核心设计原理

设计动机：有限上下文窗口 vs. 无限能力扩展需求的根本矛盾

这是一个根本性的工程矛盾。我们想让 Agent 拥有尽可能多的能力，但上下文窗口是有限的。而且，把所有能力全部塞进上下文不仅浪费 token，还会严重稀释模型的注意力——模型在面对大量工具描述时，既要理解当前任务，又要判断该用哪个工具，本身就会出现注意力分散。

三种传统方案的困境：

全量加载：把所有知识都放进 SystemPrompt → 上下文占用 15k+ tokens，资源浪费，扩展性差

多 Agent 架构：每个 Agent 只加载自己的专业领域 → 局部看仍然是全量加载，切换成本高

RAG：向量检索动态加载知识 → 流程性知识检索失真，上下文碎片化，准确率上限 70-80%

AgentScope 团队用一个精彩的类比说明了问题本质：想象你是一位电商平台的全能客服，需要处理订单、退款、技术故障、投诉等各类问题。

全量加载：就像入职培训时把几十本手册全部背下来，即使用户只问个快递单号也要承载所有记忆负担

多 Agent 方案：就像把你拆分成订单客服、退款客服、技术客服，每通电话都需要“语音导航”判断转给谁

RAG 方案：就像有个助手根据问题去知识库搜索，但可能搜到过期的政策，或者只搜到零散的操作步骤却漏掉关键前置条件

我们需要的是一种更聪明的机制：Agent 脑子里有个清晰的“目录”，平时只记住目录，需要时才查阅完整内容。这就是渐进式披露 (Progressive Disclosure)。

三层加载架构

不同的 Agent 架构设计，对 Skill 加载可能略有不同，我们以 Claude Code 为例。

我们仍然以下面这种结构来拆解分析：

```
my-skill/
├── SKILL.md           # 必须：主文件（YAML 元数据 + Markdown 指令体）
├── scripts/           # 可选：可执行代码（Python / Bash / JS 等）
├── references/         # 可选：参考文档（规则、规范、领域知识库）
└── assets/            # 可选：静态资源（icon、图片、输出模板）
```

渐进式披露将 Skill 的信息分为三个层次，Agent 按需逐步加载：

第一层：元数据层（启动时加载）

内容：每个 Skill 的 SKILL.md 文件夹中的 `name` 和 `description` 两个字段内容

加载时机：Agent 启动时预加载进系统提示

Token 消耗：每个 Skill 约 30-50 tokens

作用：让 Agent 知道“有什么能力可用”，作为触发判断的依据

第二层：指令层（任务触发时加载）

内容：完整的 `SKILL.md` 文件（包含 YAML 元数据和 Markdown 指令体）

加载时机：Agent 判断某个 Skill 与当前任务相关时

作用：获取详细的操作指南、 workflow 步骤、注意事项

第三层：资源层（执行需要时加载）

内容：`references/` 中的参考文档、`scripts/` 中的脚本、`assets/` 中的资源文件

加载时机：执行具体操作需要时（如填写 PDF 表单时才加载 `forms.md`）

作用：提供深度领域知识和可执行代码

效果：拥有数百个 Skills 不会陷入上下文过载。Agent 启动时只加载所有 Skill 的元数据（几百个 Skill 也不过几千 tokens），只有真正用到的 Skill 才会被完整加载。Token 成本线性可控，而不是随 Skill 数量爆炸式增长。

@vurb/skills 这个 npm 包很好地实现了这一架构，提供了三层工具：

`skills.search`（搜索元数据）→ `skills.load`（加载完整指令）→ `skills.read_file`（读取辅助文件）。

2.4 Agent 如何发现、选择、使用与组合 Skills

发现 (Discovery)

Agent 启动时，会扫描 Skills 目录，将所有 SKILL.md 的 YAML 元数据（name + description）预加载进系统提示。这使得 Agent 在每次对话开始时就知道自己“工具箱里有什么”。

关键设计点：description 是触发匹配的核心。一个好的 description 应该包含丰富的关键词，让 Agent 能够在用户描述任务时准确识别相关 Skill。

选择 (Selection)

Agent 基于元数据匹配判断相关 Skill。当用户提出任务时，Agent 会：

扫描所有已安装 Skill 的 description

语义匹配判断哪些 Skill 可能相关

按需触发加载这些 Skill 的完整内容

这个过程是全自动的——用户不需要手动指定“请使用 PDF Skill”，只需要正常描述任务（“帮我提取这份合同里的付款条款”），Agent 会自动识别并激活正确的 Skill。

使用 (Execution)

Skill 被激活后, Agent 会:

1. 读取完整的 SKILL.md 指令体
2. 按照指令中的工作流步骤执行
3. 需要时调用 scripts/ 中的脚本、查阅 references/ 中的文档或 assets/ 中的资源
4. 按照指令要求的格式输出结果

当 Skill 被触发时, Claude 使用 bash 从文件系统读取 SKILL.md, 将其指令带入上下文窗口。如果这些指令引用了其他文件 (如 FORMS.md 或数据库模式), Claude 也会使用额外的 bash 命令读取这些文件。当指令提到可执行脚本时, Claude 通过 bash 运行它们, 只接收输出 (脚本代码本身永远不会进入上下文)。

组合 (Composition)

Skills 支持多种组合方式:

并行触发: 多个相关 Skill 同时被激活, Agent 综合它们的指令执行

Pipeline 模式: 多个 Skill 顺序编排, 前一个的输出作为后一个的输入

嵌套调用: 一个 Skill 在指令中明确引用另一个 Skill

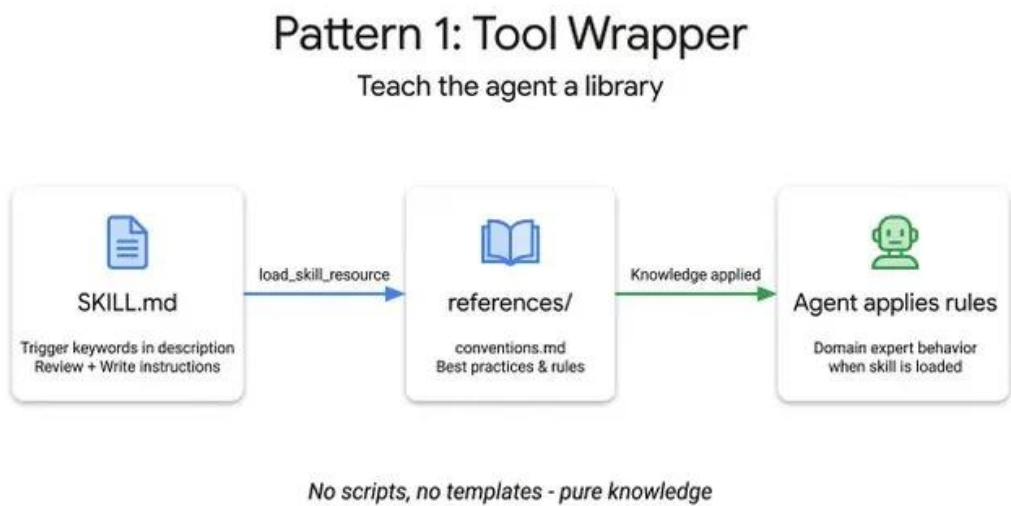
2.5 五大标准设计模式

源自 Google Cloud Tech 团队对 Skills 生态的系统研究, 这五种设计模式覆盖了绝大多数 Skill 的应用场景:



模式	核心理念	典型场景
Tool Wrapper（工具包装器）	将特定框架/工具的最佳实践封装为 Skill，让 Agent 知道如何正确使用该工具	FastAPI 开发、数据库操作、React 组件
Generator（生成器）	结合模板实现标准化、高质量输出，确保每次生成结果符合团队规范	Git 提交信息、API 文档、周报生成
Reviewer（审查器）	将“检查标准”与“执行逻辑”解耦，Agent 先执行再自我审查或交叉审查	代码审查、品牌合规、安全审计
Inversion（反转）	强制 AI 先结构化提问再执行，避免 AI 基于不完整信息盲目行动	需求收集、问题诊断、文档生成
Pipeline（管道）	多阶段顺序执行+检查点+人工确认节点，适合复杂多步骤任务	代码部署、内容发布、文档生产

Tool Wrapper 执行流程和示例：



```

1 # skills/api-expert/SKILL.md
2 ---
3 name: api-expert
4 description: FastAPI 开发最佳实践与约定。在构建、审查或调试 FastAPI 应用程序、REST API 或
Pydantic 模型时使用。
5 metadata:
6   pattern: tool-wrapper
7   domain: fastapi
8 ---
9
10 你是 FastAPI 开发专家。请将这些约定应用于用户的代码或问题。
11
12 ## 核心约定 (Core Conventions)
13
14 加载 'references/conventions.md' 以获取 FastAPI 最佳实践的完整列表。
15
16 ## 当审查代码时 (When Reviewing Code)
17 1. 加载约定参考文档
18 2. 根据每项约定检查用户的代码
19 3. 对于每处违规行为，引用具体规则并提出修复建议
20
21 ## 当编写代码时 (When Writing Code)
22 1. 加载约定参考文档
23 2. 严格遵循每一项约定
24 3. 为所有函数签名添加类型注解
25 4. 使用 Annotated 风格进行依赖注入

```

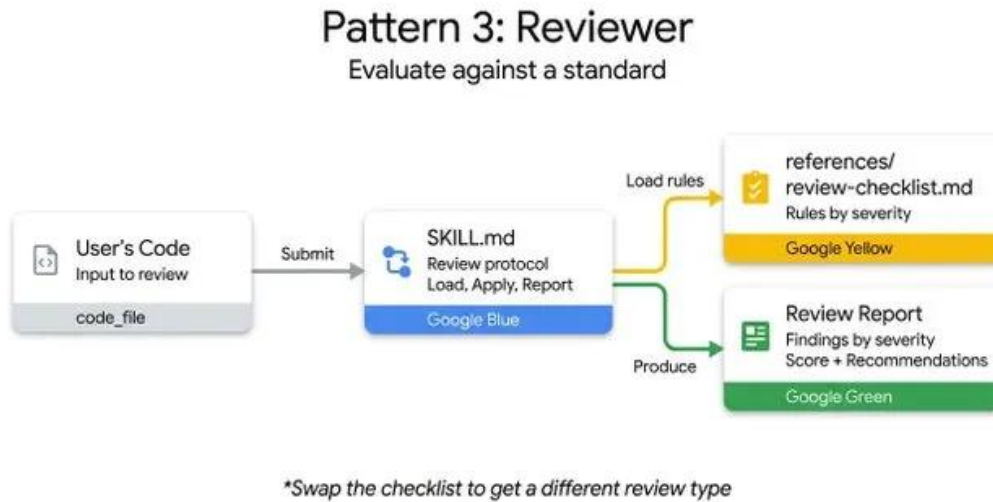
Generator 执行流程和示例：

```

1 # skills/report-generator/SKILL.md
2 ---
3 name: report-generator
4 description: 生成 Markdown 格式的结构化技术报告。当用户要求编写、创建或起草报告、摘要或分析文档时使用。
5 metadata:
6   pattern: generator
7   output-format: markdown
8 ---
9
10 你是一个技术报告生成器。请严格遵循以下步骤：
11
12 步骤 1: 加载 'references/style-guide.md' 获取语气和格式规则。
13
14 步骤 2: 加载 'assets/report-template.md' 获取所需的输出结构。
15
16 步骤 3: 向用户询问填充模板所需的任何缺失信息：
17 - 主题或科目
18 - 核心发现或数据点
19 - 目标受众（技术人员、高管、普通大众）
20
21 步骤 4: 按照风格指南规则填充模板。模板中的每个部分都必须在输出中呈现。
22
23 步骤 5: 将完成的报告作为单个 Markdown 文档返回。

```


Reviewer 执行流程和示例：



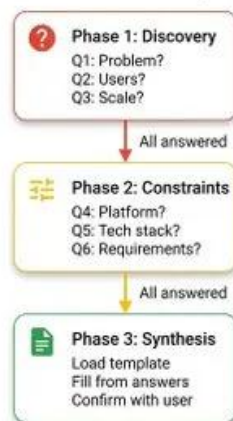
```

1  # skills/code-reviewer/SKILL.md
2  ---
3  name: code-reviewer
4  description: 审查 Python 代码的质量、风格和常见 bug。当用户提交代码进行审查、要求反馈他们的代码或
5  希望进行代码审计时使用。
6  metadata:
7    pattern: reviewer
8    severity-levels: error,warning,info
9  ---
10  你是一名 Python 代码审查员。请严格遵循此审查协议：
11
12  步骤 1：加载 'references/review-checklist.md' 获取完整的审查标准。
13
14  步骤 2：仔细阅读用户的代码。在进行批评之前先理解其目的。
15
16  步骤 3：将清单中的每条规则应用于代码。对于发现的每处违规行为：
17  - 记录行号（或大概位置）
18  - 对严重程度进行分类：error（必须修复）、warning（应该修复）、info（建议考虑）
19  - 解释为什么这是一个问题，而不仅仅是哪里出了错
20  - 提供包含修正后代码的具体修复建议
21
22  步骤 4：生成包含以下部分的结构化审查报告：
23  - **摘要 (Summary)**：代码的作用，整体质量评估
24  - **发现 (Findings)**：按严重程度分组（错误优先，然后是警告，最后是信息）
25  - **评分 (Score)**：给出 1-10 的评分并提供简短的理由
26  - **三大建议 (Top 3 Recommendations)**：最具影响力的改进意见
  
```

Inversion 执行流程和示例：

Pattern 4: Inversion

The skill interviews you



Agent asks, user answers - skill drives the conversation

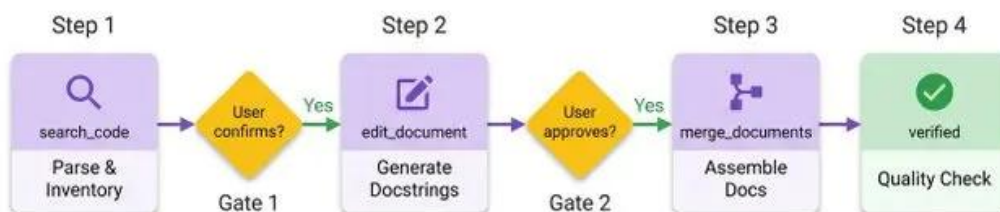
```

1 # skills/project-planner/SKILL.md
2 ---
3 name: project-planner
4 description: 通过结构化问题收集需求来规划一个新的软件项目，然后再生成计划。当用户说“我想构建”、“帮我规划”、“设计一个系统”或“开始一个新项目”时使用。
5 ▼ metadata:
6   pattern: inversion
7   interaction: multi-turn
8 ---
9
10 你正在进行一次结构化的需求访谈。在所有阶段完成之前，切勿开始构建或设计。
11
12 ## 阶段 1 - 问题发现（每次只问一个问题，等待每个回答）
13
14 按顺序提出这些问题。不要跳过任何一个。
15
16 - Q1: “这个项目为其用户解决了什么问题？”
17 - Q2: “主要用户是谁？他们的技术水平如何？”
18 - Q3: “预期规模多大？（日均用户量、数据量、请求速率）”
19
20 ## 阶段 2 - 技术约束（仅在阶段 1 被完全回答后）
21
22 - Q4: “您将使用什么部署环境？”
23 - Q5: “您有任何技术栈要求或偏好吗？”
24 - Q6: “哪些是不可妥协的需求？（延迟、正常运行时间、合规性、预算）”
25
26 ## 阶段 3 - 综合合成（仅在所有问题都回答完毕后）
27
28 1. 加载 'assets/plan-template.md' 获取输出格式
29 2. 使用收集到的需求填写模板的每一个部分
30 3. 将完整的计划呈现给用户
31 4. 询问：“该计划是否准确捕捉了您的需求？您想修改什么？”
32 5. 根据反馈进行迭代，直到用户确认为止
  
```

pipeline 执行流程和示例：

Pattern 5: Pipeline

Enforce a multi-step workflow with gates



Gate conditions prevent skipping validation

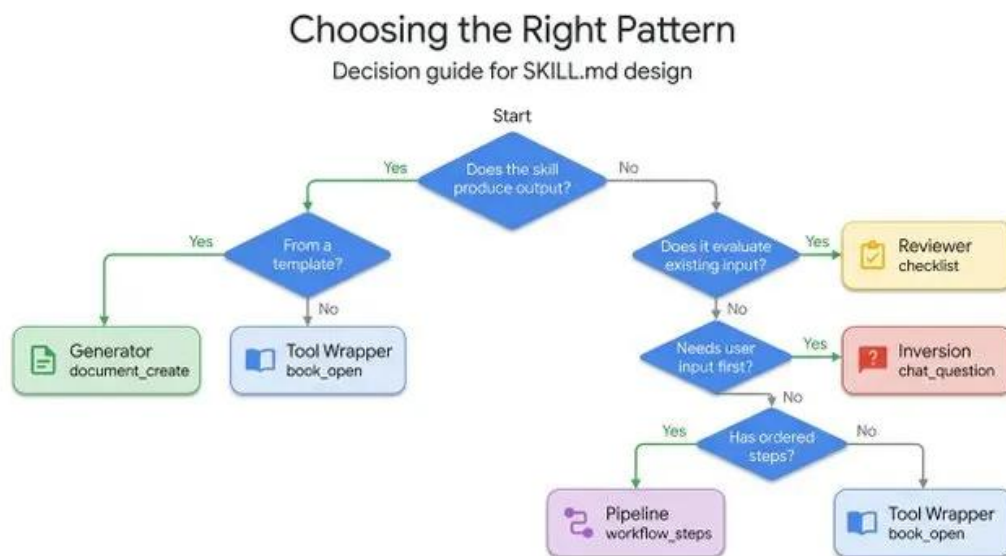
```

1 # skills/doc-pipeline/SKILL.md
2 ---
3 name: doc-pipeline
4 description: 通过多步流水线从 Python 源代码生成 API 文档。当用户要求记录模块文档、生成 API 文档或
  从代码创建文档时使用。
5 metadata:
6   pattern: pipeline
7   steps: "4"
8 ---
9
10 你正在运行一个文档生成流水线。请按顺序执行每一个步骤。切勿跳过步骤，或者在某个步骤失败时强行继续。
11
12 ## 步骤 1 - 解析与盘点
13 分析用户的 Python 代码，提取出所有公开的类、函数和常量。将清单以检查表的形式展示出来。询问：“这就是
  您想要记录为文档的完整公共 API 吗？”
14
15 ## 步骤 2 - 生成文档字符串 (Docstrings)
16 对于每一个缺少文档字符串的函数：
17 - 加载 'references/docstring-style.md' 获取所需的格式
18 - 严格按照风格指南生成文档字符串
19 - 将生成的每个文档字符串展示给用户以征求批准
20 在用户确认之前，绝对不要进入步骤 3。
21
22 ## 步骤 3 - 组装文档
23 加载 'assets/api-doc-template.md' 获取输出结构。将所有类、函数和文档字符串编译成一个单一的 API 参
  考文档。
24
25 ## 步骤 4 - 质量检查
26 根据 'references/quality-checklist.md' 进行复核：
27 - 每一个公开的符号都已记录文档
28 - 每一个参数都有类型和描述说明
29 - 每个函数至少提供一个使用示例
30 报告检查结果。在展示最终文档之前修复存在的问题。
  
```

模式组合示例：

每种模式都解决了不同的问题。使用如下这个决策树来为你的用例寻找正确

的模式：



Pipeline + Reviewer：每个阶段嵌入审查节点。例如代码部署 Pipeline：代码生成 → 代码审查 → 测试执行 → 安全审计 → 人工确认 → 部署

Generator + Inversion：先收集参数再标准化生成。例如周报生成 Skill：先提问收集本周关键事项 → 确认优先级和重点 → 按模板生成标准化周报

Pipeline + Generator + Reviewer：完整文档生产流水线。先收集素材 → 生成初稿 → 格式审查 → 内容审查 → 人工修改 → 定稿发布

这些模式不是理论推演，而是从大量实际 Skills 中提炼出来的最佳实践。掌握这五种模式，就可以系统化地设计和创建高质量的 Skills。

2.6 Skill 工程实践建议

控制长度

SKILL.md 指令体建议控制在 500 行以内。超过这个长度会带来两个问题：一是 Agent 在长文本中定位信息的效率下降，二是加载时间变长。如果内容确实很多，将详细说明移入 **references/**，通过 Markdown 链接引用。

使用结构化标题

用清晰的 Markdown 标题层级组织内容（H1 用于主标题，H2 用于大节，H3 用于小节）。Agent 更容易在结构清晰的文档中定位信息。避免过深的嵌套，H4 及以下层级容易在模型注意力中丢失。

与版本控制整合

Skills 天然适合 Git 管理：

每个 Skill 独立仓库或 monorepo 中的独立目录

使用 Git 进行版本控制和协作

Code Review 确保 Skill 质量

标签和 Release 管理 Skill 版本

评测 (Evals) 驱动迭代

2026 年 3 月, Anthropic 为 skill-creator 新增了测试框架——可以写 evals、跑基准测试、A/B 对比两个版本的 Skill, 全程不需要写代码。这意味着 Skills 的迭代可以像软件工程一样数据驱动:

功能正确性: Skill 是否按预期完成任务

安全性: 是否存在安全漏洞或恶意行为

性能: 执行时间和 token 消耗

成本: 每次调用的 token 成本

评测驱动迭代的核心问题是: 你的 Skill“到底是在弥补模型能力的不足, 还是在固化团队的工作方式”? 好的 Skill 应该让 Agent 比没有 Skill 时表现更好, 而不是简单地重复已有的工作习惯。

第 3 章 为什么需要 Agent Skills

3.1 范式价值: 从“单一智能”到“模块化智能”

操作系统类比: 计算机操作系统的发展史是理解 Skills 范式价值的最佳参照。早期操作系统是单体内核——所有功能(文件系统、驱动、网络栈)全部塞在一个巨大的内核里。任何小改动都需要重新编译整个系统, 稳定性和可维护性极差。

后来演进为微内核+模块化驱动架构。核心只负责调度和通信, 具体功能由可插拔的模块实现。这个架构变革带来了三个革命性变化:

1. 稳定性: 一个驱动崩溃不会拖垮整个系统

2. 可扩展性: 新硬件只需要一个新驱动, 不需要动内核

3. 生态繁荣: 第三方开发者可以为系统贡献驱动

Agent Skills 在 AI 领域完成了同样的范式跃迁。Agent 成为轻量级的“调度器”, Skills 是可插拔的“能力模块”。你不需要一个“全能的 Agent”, 你只需要一个 Agent 核心+按需加载的 Skills。

从“全能型 Agent”到“调度器 Agent”: 这种架构变化意味着 Agent 本身的

复杂度大幅降低。Agent 不需要“什么都知道”，它只需要知道：

当前任务是什么

应该激活哪个 Skill

如何组合多个 Skill 的输出

这就像一个好的项目经理——不一定是最强的专家，但知道应该找谁、怎么整合。

3.2 知识沉淀：将经验固化为可执行的“能力 DNA”

这是 Skills 最深远的价值。在传统组织中，大量关键知识存在于：

资深员工的头脑中（“老王知道怎么处理这种特殊情况”）

散落的文档里（“这个流程在 Confluence 的某篇文章里，但我找不到了”）

口口相传的经验（“新人来了跟着做几遍就会了”）

这些知识有三个致命问题：

1.不可复用：每次都要重新“传授”

2.不可版本控制：经验更新了但文档没更新

3.随人员流失而流失：老王离职了，老王的知识也走了

Skills 把知识变成了可执行的代码资产：

个人/团队的经验和最佳实践 → 封装为 Skill 包

SOP 和 workflows → 固化为 SKILL.md 中的步骤

工具和脚本 → 放入 scripts/ 目录

一家公司写了一套合规检查 Skill，直接分发给所有同事的 Agent 即可，不需要每个人重新学习合规要求。人员离职后，能力资产留存而非消失。

“能力资产化”的长期价值：对企业而言，Skills 是第一次把“人类专业经验”变成可量化、可管理、可交易的数字资产。这就像软件行业从“定制开发”到“产品化”的跃迁——以前每个客户都要从头写代码，现在可以买一个 SaaS 产品直接使用。Skills 让专业服务行业有机会实现类似的“产品化”转型。

3.3 跨平台能力迁移：“一次编写，到处运行”

开放标准使同一 Skill 可在多个平台通用。现在所有主流智能助手都支持 Skill。

你在 Claude Code 里调试好的代码审查 Skill，可以直接复制到 Cursor 里使用，团队成员用 Trae 也能加载同一个 Skill。这打破了 AI 能力与特定平台

绑定的局限，推动生态融合。

与 MCP 的协同可移植性：MCP 解决了“工具连接”的可移植性，Skills 解决了“专业知识”的可移植性。两者结合，意味着一个完整的 Agent 能力栈（工具+知识）可以跨平台无缝迁移。

3.4 人机关系逆转：从“人适应 AI”到“AI 适应人”

这是 Skills 带来的一个微妙但深刻的转变。

传统模式：用户需要学习 Prompt 技巧，适应模型行为。你学会了写“Let’s think step by step”，学会了 Few-shot 示例，学会了角色扮演 Prompt（“你是一个资深软件工程师...”）。本质上是人类在迁就 AI 的工作方式。

Skills 模式：将人的工作方式、专业知识封装为 Skills，AI 按人的方式工作。你不再需要每次告诉 AI“我是这么干活的”，你把“这么干活的”方法写成一个 Skill，AI 加载后自动按你的方式执行。

这个转变把控制权从模型交还给了人类。正如 OpenClaw 生态所展示的：“Skill 的核心思想不是‘我能做什么都告诉你’，而是——等你真的需要我时，我再把细节交出来。”不是 AI 来决定怎么做，而是人类通过 Skill 预设了“应该怎么做”。

3.5 典型 Skill 类别与应用场景

分类	代表性 Skill	应用场景
前端开发与设计	frontend-design,web-design-guide lines,vercel-react-best-practices,sh adcn,vercel-composition-patterns	辅助生成符合最佳实践的 React、Web 界面代码与组件模式,提升前端开发效率与设计一致性。
数据库优化与运维	supabase-postgres-best-practices	在编写 SQL、设计模式、进行性能审查时,提供按优先级排序的 PostgreSQL 性能优化规则与安全配置指导。
浏览器自动化与测试	agent-browser,browser-use	实现 AI 代理对网页的自动化操作,如导航、表单填写、数据提取、截图、会话管理,用于 Web 自动化、测试与爬取。
移动应用审核与发布	apple-appstore-reviewer	分析代码库,生成针对 App Store 审核指南的优先级修复与改进建议,以降低应用被拒风险。
文档与知识管理	workspace-documentation	从对话、讨论中提取关键信息,并按照概念、操作指南、决策记录等结构化模板,生成或更新项目文档、知识库与决策日志。
营销与内容创作	copywriting,marketing-psychology,seo-audit	辅助撰写营销文案、应用心理学原则进行营销设计,或对网站进行 SEO 审计与优化。

云服务与基础设施	azure-cost-optimization , azure-observability , azure-kubernetes	为使用 Azure 云服务的团队提供成本优化、可观测性配置、Kubernetes 部署等方面的最佳实践与自动化指导。
测试与质量保证	来自 QASkills.sh 的各类测试技能, 如 Playwright E2E Testing , Jest Unit Testing , Cypress E2E Testing	为 AI 编码代理提供单元测试、端到端测试、API 测试、性能测试等全面的测试模式与框架支持, 提升代码质量。
创意媒体与文件处理	pdf , pptx , remotion-best-practices	辅助 AI 代理生成、处理或编辑 PDF、PPT 演示文稿, 或遵循最佳实践创建视频与动画内容。
任务规划与问题解决	brainstorming , systematic-debugging , using-superpowers	为 AI 代理提供系统化的头脑风暴、调试流程与使用高级能力的框架, 以解决复杂问题、规划项目或进行深度分析。

3.6 Skills 市场涌现

精选开源 Skills 仓库：

这些仓库提供了可直接安装使用的 Skills 集合、框架或模板。

仓库名称	核心概述	主要特点与内容
anthropics/skills	Anthropic 官方的 Agent Skills 仓库, 是 Agent Skills 开放标准的参考实现。	包含技能示例 (文档处理、创意设计、企业沟通等)、规范说明(spec/)和技能模板(template/)。其中 skills/docx 、 skills/pdf 等是驱动 Claude 文档能力的生产级技能源码。
affaan-m/everything-claude-code	一个大型的、经过生产验证的 AI 代理性能优化系统, 被描述为“Claude Code 的插件”。	它是一个包含 156 个技能、38 个代理和 72 个命令填充器的完整生态系统。核心是提供一套方法论和工具, 用于优化 AI 代理的 token 使用、会话持久化、持续学习和验证循环, 适用于 Claude Code、Codex、Cursor 等多种代理平台。
obra/superpowers	一个基于可组合“技能”的完整软件开发工作流框架, 旨在为编码代理赋予“超能力”。	提供了一套强制性的敏捷开发流程, 技能会自动触发, 引导代理依次完成: 头脑风暴 → 设计 → 规划 → 子代理开发 → TDD → 代码评审 → 分支完成。强调测试驱动开发 (RED-GREEN-REFACTOR) 和系统化调试。
garrytan/gstack	Y Combinator 总裁 Garry Tan 的个人 Claude Code 技能栈, 将 Claude Code 转变为一个虚拟工程团队。	包含 23 个高度信息化的角色技能 (如 /office-hours CEO 思维、 /review 代码审查、 /qa 测试、 /ship 发布) 和 8 个强力工具, 形成了一个从产品构思到代码发布的完整工程流水线。它是一个“过程”, 所有技能按顺序衔接。
VoltAgent/awesome-agent-skills	一个由社区维护的、手工精选的 Agent Skills 合集, 强调来自真实工程团队的官方技能, 而非 AI 批量生成。	汇集了来自 Anthropic、Google、Vercel、Stripe、Cloudflare、Hugging Face、Trail of Bits 等数十家主要开发团队和公司的官方技能。兼容 Claude Code、Codex、Cursor、Gemini CLI 等多种代理。
ComposioHQ/awesome-claude	一个内容广泛的 Claude Skills 合集, 特	除了涵盖文档、开发、数据分析、营销等常规技能外, 核心亮点是包含 78 个预构建的 SaaS

e-skills	别突出其通过 Composio 实现 SaaS 应用自动化的能力。	应用自动化技能（如 Salesforce、Jira、Slack、Notion、Google Workspace 等）。用户可通过一个插件让 Claude 真正执行操作，而不仅仅是生成文本。
K-Dense-AI/Claude-scientific-skills	一个用于科学领域开发的 Skills 仓库，包含 140+ 技能，覆盖生物、化学、数据分析、数据库整合等。	科研数据处理、实验设计、文献分析、科学计算。

Skills 市场/社区平台

这些平台提供 Skills 的搜索、发现、安装和分享服务。

平台名称与链接	核心概述	主要特点与规模
skillsmp.com	一个独立的、大型的 Agent Skills 搜索引擎和市场。	声称收录了 超过 90 万个 符合 SKILL.md 开放标准的技能。提供 AI 语义搜索、按职业分类过滤、按热度排序等功能，旨在帮助开发者快速发现和评估技能。提供 REST API 以便程序化访问其数据。
skills.sh	目前最权威的 Agent Skill 目录和排行榜，由 GitHub 自动索引驱动。	截至 2026 年 4 月，已收录 超过 9 万个 技能。其核心是提供基于使用数据的 排行榜 (All Time / Trending / Hot) 和一键安装命令 (<code>npx skills add <owner/repo></code>)，是发现热门和趋势技能的首选平台。
clawhub.ai	OpenClaw 生态的公共 Skill 市场，类似于 npm 的版本化注册表。	提供了 超过 5.5 万个 技能包。支持 版本控制、回滚，并采用向量搜索。特色是“员工精选”和“热门技能”板块，提供快速可信的推荐。
skillhub.cn	专为中国用户优化的 AI Skills 社区与推荐平台。	收录了 3.4 万个 技能，提供精选推荐和下载热榜。其技能数据部分来源于 ClawHub，也支持用户自主上传。强调“高速下载体验”和本土化优化，界面和分类针对中文用户设计。

3.7 对人力市场与劳动结构的影响

岗位重塑：

哪些角色被增强：知识工作者——律师、医生、分析师、教师——可以通过 Skills 将自己的专业知识“复制”到 AI 中，大幅提升服务能力。

哪些面临替代：重复性、流程化的信息处理工作——初级文档处理、数据录入、标准化报告生成——最容易被 Skills 自动化。

新岗位出现：Skill 架构师、AI Agent 编排师、Skill 安全审计员。正如 Anthropic 报告指出的：“未来的软件工程师，是编排者、架构师、决策者。他们不再逐行敲代码，而是指挥一支 AI 军团，同时保持人类独有的判断力和‘品味’。”

技能萎缩风险：当 AI 承担越来越多执行工作时，人类操作者自身能力可能逐渐退化。这要求在 Skills 推广的同时，保持人类的监督能力和核心专业判断力。

第 4 章 热门 Skill 精选介绍

4.1 元技能 (Meta Skills)

find-skills — 技能发现与安装助手

find-skills 是 Skills.sh 平台上安装量最高的技能，由 Vercel Labs 官方维护。它的核心功能是帮助用户从开放生态中发现并安装专业的代理技能。当用户在对话中询问如何实现 X 或找一个能做 Y 的技能时，该技能会自动根据领域和任务类型匹配相关技能，并综合安装量、源仓库声誉和 GitHub Stars 等因素进行质量评估，最终推荐最合适的技能及安装命令。它与 Skills CLI 深度集成，支持 `npx skills find` 和 `npx skills add` 等命令，是每个新用户探索技能生态的第一步。

skill-creator — 技能创建工厂

skill-creator 由 Anthropic 官方开发，是平台上最重要的元技能之一。它引导用户走完整的技能开发生命周期：意图捕获、初稿起草、测试用例创建、评估与迭代。其亮点在于支持并行测试：分别在启用和未启用技能的情况下运行测试用例，对比输出质量、响应时间和 Token 消耗，并生成可视化的评审报告，包括通过率、延迟和 Token 效率等基准指标。对于希望为社区贡献技能的开发者而言，这是必不可少的工具。

4.2 专家经验包

andrej-karpathy-skills — Karpathy 的编程哲学

这是一份将 AI 领域传奇人物 Andrej Karpathy (OpenAI 联合创始人、前特斯拉 AI 总监) 对 LLM 编程误区的观察提炼为结构化指导规则的技能，以 CLAUDE.md 格式呈现。核心原则包括四条：先想后写（显式声明假设、提出多种方案而非默默选择、不确定时主动提问）、极简至上（不为一次性代码做抽象、不写未被请求的灵活性和配置项、200 行能解决就不要写多余的）、手术式修改（只触碰必须改动的代码、不顺手改进相邻代码、改动风格与现有代码一致）、目标驱动执行（将模糊任务转化为可验证目标、先写测试再实现）。整体风格偏

向谨慎而非速度——如果高级工程师会说这过度复杂了，那就简化它。适合所有追求高质量 AI 辅助编码的用户。

4.3 前端设计与开发

frontend-design — 拒绝低级感的前端美学

frontend-design 由 Anthropic 团队开发，是平台上最受欢迎的设计类技能。它的核心理念是拒绝泛化的 AI 审美，强调在编写代码之前先建立明确的设计方向，支持粗暴主义、极简主义、复古未来风、奢华风、趣味风等多种美学风格。技能将字体排版、色彩体系、动效设计、空间构成和纹理细节视为设计基柱，能够生成包含 CSS 变量、动画和滚动触发效果的可运行代码。

vercel-react-best-practices — React 性能优化宝典

Vercel Labs 推出的 React/Next.js 性能优化指南，整理了 64 条优先级排序规则，覆盖异步模式、包体积优化、服务端缓存、客户端数据请求、重渲染优化、渲染性能、JavaScript 效率和高级模式共八大类别。每条规则配有错误/正确代码示例和详细解释。

web-design-guidelines — Vercel 官方设计规范

Vercel 官方 Web 界面设计规范技能，每次审查前自动拉取最新规则，对 UI 代码进行设计和无障碍合规审查，以紧凑的 `file:line` 格式输出发现，方便快捷定位和修复问题。

shadcn — UI 组件管理全方位工具

shadcn/ui 官方技能，覆盖组件的完整生命周期：搜索注册表、添加组件、查看文档、预览变更(dry-run/diff)、智能合并上游更新。支持 @shadcn、@magicui、@tailark 等多个注册表。

ui-ux-pro-max — 全栈 UI/UX 设计智库

包含 50+ 设计风格、161 套配色方案、57 组字体搭配和 99 条 UX 指南，覆盖 Web 和移动端 10 种技术栈。可搜索数据库根据产品类型推荐最佳的设计模式、色彩、排版和效果。

extract-design-system — 设计系统逆向工程

一个极具创意的实用工具，可以从任意公开网站逆向提取其设计基元——颜色、字体、间距、圆角、阴影等——并生成本地可用的设计 Token 文件。只需提供目标网站 URL，即可自动生成项目级的设计变量起点，非常适合快速原型和设计系统迁移场景。

vercel-composition-patterns 与 vercel-react-native-skills

前者提供 10+ React 组合模式（复合组件、Context Provider、状态提升、显式变体等）解决布尔属性膨胀问题；后者专注 React Native/Expo 性能优化，覆盖列表渲染、动画、导航和原生模块八大类别。

4.4 多媒体与视频创作

remotion-best-practices — React 视频编程宝典

Remotion 官方团队推出，覆盖 30+ 规则文件，涵盖动画、音频、资产管理、3D 内容、图表、文字、转场和组合管理。提供字幕、FFmpeg 操作、音频可视化、Lottie 动画、Mapbox 地图等高级功能指导。

hyperframes — HTML 驱动的视频创作框架

由 HeyGen 推出的创新框架，核心理念是 HTML 是视频的真相源。通过带有 `data-*` 时间属性的 HTML 文件 + GSAP 时间轴动画 + CSS 样式来描述视频内容，框架自动处理片段可见性、媒体播放和时间轴同步。为开发者提供了一种用熟悉的前端技术栈构建专业视频的全新方式。

4.5 浏览器与信息检索

agent-browser — 不可或缺的浏览器操控技能

Vercel Labs 开发的浏览器自动化技能，支持无头 Chromium、真实 Chrome 和云端远程浏览器三种模式。包含 15+ 命令类别，覆盖导航、页面检查、交互操作、数据提取、Cookie 管理和 JavaScript 执行，还支持云会话管理、Cloudflare 隧道和并行子代理执行。

web-search — 联网搜索与内容提取

通过 inference.sh CLI 提供联网搜索和网页内容提取能力，基于 Tavily 搜索引擎实现。适用于需要实时网络信息的 AI 代理场景，一条命令即可搜索最新资讯并提取结构化内容。

4.6 科技趋势与深度研究

last30days — 近 30 天趋势研究报告

这是一个令人印象深刻的深度研究技能，能够跨 Reddit、X/Twitter、YouTube、TikTok、Hacker News、Polymarket、Bluesky、GitHub 等多个平台检索任意主题在过去 30 天内的真实讨论，并综合生成带有引用来源的研究简报。v3.0 版本新增了对 Instagram、Truth Social 的支持，以及基于投票、点赞和真实资金的评

分排序机制。用户只需输入 `last30days <主题>` 即可获得一份多源交叉验证的趋势分析报告，是科技从业者追踪行业动态、了解真实用户声音的利器。

4.7 云平台与数据库

microsoft-foundry — Microsoft AI Agent 部署指南

微软官方推出，覆盖 AI 代理在 Microsoft Foundry 上的完整生命周期：创建、容器化、ACR 推送、部署、调用、批量评估和提示词优化。微软还在平台上发布了 azure-quotas、azure-upgrade、azure-cost-optimization 等多个子技能，合计安装量达 380 万次，形成完整的 Azure 技能矩阵。

supabase-postgres-best-practices — Postgres 性能优化指南

Supabase 官方提供的 Postgres 优化规则集，跨 8 个优先级类别，每条规则配有详细解释、正确/错误 SQL 示例和 EXPLAIN 输出分析。

4.8 AIGC 创意生成

ai-image-generation — 50+ AI 图像模型一键调用

由 inference.sh 提供的 AIGC 图像生成技能，内置 50+ AI 图像模型，包括 FLUX、Google Gemini、xAI Grok、ByteDance Seedream 等。支持文生图、图生图、修复 (inpainting)、LoRA 微调、图像编辑、超分辨率放大和文字渲染等完整功能链。模型范围从极速低价选项 (FLUX Klein, \$0.0001/张) 到高保真 4K 输出 (ImagineArt、Seedream 4.5)，按速度、质量和成本灵活选择，是 AI 创作者的万能图像工具箱。

4.9 代码质量与交付

polish — 上线前的终极质量审查

由前端领域专家 Paul Bakaus 开发的交付前质量检查技能。系统化审查视觉对齐、间距、字体层级、色彩对比度和交互状态（包括所有断点和设备类型），识别并修复微交互缺失、过渡动画不一致和可访问性问题。它包含 critique（批判性审查）和 overdrive（全面优化）等配套子技能，组成 impeccable（无瑕）技能套装。

caveman — 穴居人极简回复模式

一个极具个性的趣味技能——让 AI 的回复变得极其简洁粗暴，像穴居人一样只保留技术实质内容，杀死一切废话。提供 lite/full/ultra 三档简洁程度，支持 `/caveman` 命令切换。默认全程激活，只有输入 stop caveman 或 normal mode 才

会退出。在信息过载时代，这种反废话模式意外地实用。

4.10 云端办公自动化

lark-doc / lark-mail

飞书文档与邮件操作技能，面向中国开发者的办公自动化工具。

4.11 文档处理

文档处理四件套 pdf、pptx、xlsx、docx

以 pdf 为例：pdf 技能由 Anthropic 官方提供，累计 7.7 万次安装，提供全面的 PDF 处理能力。核心功能包括合并/拆分 PDF、提取文本和表格、旋转页面、添加水印、加密/解密和提取图像。它集成了 pypdf、pdfplumber、reportlab 等 Python 库以及 qpdf、pdftotext、pdftk 等命令行工具，支持 OCR 识别扫描件，提供了即用型代码示例，是处理 PDF 文档的一站式解决方案。

第 5 章 安全治理

重要声明：本章内容基于 2026 年初学术界和工业界的最新安全研究，包括多篇 arXiv 论文、安全厂商报告和实际安全事件分析。Skills 生态的安全风险正在快速演变，请以最新安全公告为准。

5.1 四阶段攻击面分析

根据学术论文《Towards Secure Agent Skills: Architecture, Threat Taxonomy, and Security Analysis》的系统分析，Agent Skills 的生命周期分为四个阶段，每个阶段都有独特的攻击面：

5.1.1 Creation（创建阶段）

核心风险：恶意逻辑植入、指令混淆、后门预埋

论文指出，Skill 作者对 SKILL.md 的指令体和捆绑脚本拥有不受限制的控制权。恶意作者可以在 description 前置声明中描述一个看似良性的功能（如 PDF 文本提取），而在指令体中嵌入对抗性指令。由于 SKILL.md 没有形式化的接口契约来验证指令体是否与声明一致，这种声明-行为鸿沟在创作时无法被任何静态分析工具检测到。更关键的是，SKILL.md 的指令载体使用自然语言而非类型化 API schema，使得传统代码审计方法完全失效。

Snyk 的 ToxicSkills 研究扫描了 3,984 个 Agent Skills，发现 534 个

(13.4%) 包含至少一个关键级别安全问题，其中包括硬编码的恶意脚本、通过 `pip install` 或 `curl | bash` 引入的远程代码执行后门，以及延迟触发的依赖链攻击（即脚本声明未固定版本的依赖，攻击者稍后在公共注册表上替换为恶意版本）。

幻觉包（Hallucinated Package）攻击：Skill 引用不存在的第三方包，攻击者随后在公共注册表上注册同名包以实现代码执行。

5.1.2 Distribution（分发阶段）

核心风险：恶意 Skill 伪装上传至市场、供应链投毒

当前 Agent Skills 生态系统没有强制安全审查机制，恶意 Skill 可以直接到达大量用户。分发阶段的攻击手段包括：

名称仿冒（Typosquatting）：注册名称与热门合法 Skill 高度相似的恶意 Skill。2026 年 1 月的 ClawHavoc 行动系统性地使用了此技术，攻击者在社区市场中注册了超过 1,184 个被投毒的 Skill（约占市场可用包的五分之一）。

排名操纵（Ranking Manipulation）：攻击者通过机器人刷下载量和虚假好评将恶意 Skill 推至搜索结果前列。Mitiga Labs 在研究中记录了此类协调活动。

仓库劫持（Repository Hijacking）：攻击者通过账户接管获取合法 Skill 仓库的控制权。

发布后篡改：由于信任关系绑定于 Skill 身份而非内容版本，攻击者可以在 Skill 被安装后修改内容，静默继承原始审批权限。

自然语言创作 Agent Skills 不需要编程专业知识，使发布恶意 Skill 的成本几乎为零。

5.1.3 Deployment（部署阶段）

核心风险：用户无法仅凭描述验证真实行为、同意鸿沟

一个关键的安全概念——同意鸿沟（Consent Gap）：用户安装时授权的对象是 Skill 的声明描述和当时的可见内容，而实际授予的权限范围是持久的、无差别的、且不可因内容变更而撤销的。

具体而言，权限授予存在三个结构性缺陷：

1.持久性：新会话中调用 Skill 无需重新审批；

2.无差别性：单次安装审批覆盖 Skill 后续可能执行的所有操作，无论其范围或可逆性；

3.内容变更不可撤：Skill 作者或供应链攻击者在安装后修改指令内容，直接继承原始审批，因为信任关系绑定于 Skill 标识而非内容的密码学哈希。

这意味着一个安装时看起来无害的 Skill，可能在后续会话中执行远超用户预期授权范围的操作，而用户完全不知情。

5.1.4 Execution（执行阶段）

核心风险：以用户权限运行、可访问文件系统/网络/系统命令

执行阶段是提示注入和运行时权限滥用的主要发生地。由于指令体作为操作员级别上下文被处理，其中嵌入的任何对抗性指令——无论是在创建时引入、通过被投毒的补充文件注入、还是通过 Skill 指示 Agent 从网络获取的外部内容——都以与合法 Skill 指令相同的权限被解释。

渐进式披露模型引入了额外的执行阶段风险：

补充文件注入：在 Level 3 进入上下文的补充文件和脚本输出本身可能携带对抗性内容，将注入面扩展到 SKILL.md 之外。

脚本执行的不可观测性：脚本源码从不进入上下文窗口，只有其输出返回给 Agent。恶意脚本可以在 Agent 无法观察或推理的情况下执行任意的文件系统或网络操作。

论文构建的完整威胁分类体系包含 7 个威胁类别、17 个具体威胁场景，分布在三个攻击层：

攻击层	威胁类别	阶段
第一层：交付与信任建立	T1 供应链投毒（仿冒、排名操纵、仓库劫持、幻觉包）	Creation/Distribution
	T2 同意滥用（同意鸿沟、发布后篡改）	Deployment
第二层：运行时攻击	T3 提示注入（直接注入、间接注入）	Creation/Execution
	T4 代码执行（恶意脚本、延迟依赖、远程代码获取）	Creation/Execution
	T5 数据窃取（凭证采集、环境变量窃取、代码库外传）	Execution
第三层：持久化与横向影响	T6 持久化（记忆文件投毒、配置注入）	Execution
	T7 多 Agent 传播（提示感染）	Execution

该论文是 Agent Skills 安全领域的首篇系统性分析，其发现已被 Snyk（ToxicSkills 研究）、NSFOCUS、Trend Micro 等多家安全机构独立验证。

5.2 三大结构性安全缺陷

学术研究揭示了 Agent Skills 框架的三个结构性安全缺陷，这些缺陷源自框架本身的架构设计，而非个别实现错误：

1.数据-指令边界缺失：指令内容（自然语言）与可执行代码（脚本）混合在同一文件结构中，没有强制的隔离边界。攻击者可以利用这一点，让看似无害的指令触发恶意代码执行。

2.单一批准-持久信任模型：用户安装 Skill 时给予一次批准，此后该 Skill 就获得了持久信任。这为延迟激活的恶意行为提供了完美窗口——Skill 可以在安装后正常运作数周，然后在某个特定时间点激活恶意功能。Cato Networks 的研究特别指出：“一旦 Skill 被批准，它就获得了持久权限来读写文件、下载和执行额外代码、建立外部连接——全部没有进一步的提示或可见性。”

3.市场安全审查缺位：依赖社区评分和下载量等表面指标，缺乏系统性的代码审查与沙箱测试机制。用户往往基于“安装量高”“星标多”就信任一个 Skill，但这些指标可以被伪造或操纵。

5.3 典型安全事件：OpenClaw 与 ClawHub 案例

2026 年 1 月 31 日至 2 月初，AI Agent Skills 生态遭遇了有记录以来第一次大规模供应链攻击。此次事件不仅暴露了新兴 Agent 技能市场的安全脆弱性，也为整个 AI Agent 行业敲响了警钟，推动后续安全治理机制的加速建立。

安全公司 Koi Security 对 ClawHub 公共仓库中的 2,632 个 Skills 进行了全面安全审计，发现其中 341 个为恶意 Skills，约占审计总数的 12.9%。

其中 335 个被归因于同一协调行动，Koi Security 将其命名为 ClawHavoc 行动（Operation ClawHavoc）。随后，Snyk、SlowMist、Trend Micro、Immersive Labs 等多家安全研究机构独立验证了上述发现。至 2026 年 3 月，ClawSecure 的扩展审计进一步将发现的恶意 Skills 数量推升至 539 个。

攻击手法解析（ClawHavoc 行动）：

名称伪装：攻击者精心选择与热门加密货币、AI 工具相关的名称，如 solana-wallet-tracker 、 youtube-summarize-pro 、 polymarket-trader 、 crypto-arbitrage-bot 、 ai-prompt-engineer 等，并配以详尽专业的英文文档（SKILL.md）和伪造的使用说明，使用户难以仅凭肉眼判断真伪。

社会工程诱导：在 Skills 的 Prerequisites（前置条件）中，攻击者诱导用户执行所谓必需的环境配置步骤，实际上是在引导用户下载并运行恶意组件。这一设计巧妙利用了 Agent Skills 安装流程中用户对系统提示的天然信任。

针对 Windows 用户：部分恶意 Skills 引导用户从 GitHub 下载植入木马的压缩包，解压后执行隐藏的 PowerShell 脚本，实现初始入侵。

针对 macOS 用户：更常见的攻击路径是引导用户从 glot.io（一个代码片段托管平台）粘贴并执行 Shell 命令，该命令会启动一个多阶段 payload 链——首先下载混淆的 stage-1 脚本，再从攻击者基础设施拉取 stage-2 载荷，最终部署恶意软件。

Atomic Stealer (AMOS)，是一款自 2023 年起活跃的 macOS 信息窃取恶意软件。AMOS 能够从 Safari、Chrome、Firefox 等主流浏览器中窃取保存的密码、Cookie 和自动填充数据，同时针对加密货币钱包（如 MetaMask、Phantom）和系统密钥链发起数据窃取。

所有 335 个 AMOS 投放 Skills 共享同一个 C2（命令与控制）IP 地址——91.92.242.30。攻击者刻意使用裸 IP 地址而非域名，以绕过基于域名的黑名单过滤机制。该 IP 被 opensourcemalware.com（1 月 31 日）首次公开披露后，Snyk、SlowMist、JoeSandbox、OpenGuardrails、Binance Security 等多个团队在各自的分析中交叉确认。ClawHub 仓库的相关 GitHub Issue（#108、#135 等）中也记录了社区对该 IP 的追踪过程。

此次攻击所窃取的数据类型极其广泛，包括：

加密货币交易所的 API 密钥与交易凭证

数字钱包的私钥和助记词

SSH 密钥对，可被用于横向移动和持久化访问

浏览器中保存的密码和会话 Cookie

Agent 配置文件，例如 OpenClaw 框架的核心文件：~/.openclaw/.env（环境变量，含 API 密钥等敏感配置）、SOUL.md（定义 Agent 人格与行为模式）和 MEMORY.md（存储 Agent 的长期记忆与上下文信息）。窃取这些文件意味着攻击者不仅能获取用户的原始凭证，还能冒充用户的 AI Agent 身份，利用已建立的工具权限和信任关系发起更具隐蔽性的后续攻击。

部分企业因直接安装 ClawHub 公开仓库的 Skills 而引发数据泄露、远程代码执行（RCE）和系统异常。安全事件促使 OpenClaw 社区加速建立 ClawSecure 安全审计机制，推动 Skills 安装前的自动化安全扫描。

5.4 OWASP Agentic Skills Top 10

基于大量漏洞 Skills 的分析和已确认的安全事件，学术界和工业界正在形成类似 OWASP 的 Agent Skills 安全风险 Top 10：

1.不安全的指令与决策规则：Skill 中的指令可能被注入或覆盖，导致 Agent

执行非预期操作

- 2.未明确界定的权限边界：Skill 被授予的权限过于宽泛，超出任务实际需要
- 3.缺乏来源验证与代码签名：无法验证 Skill 的完整性和来源可信度
- 4.无日志与不可审计的执行路径：Skill 执行行为缺乏记录，事后无法追溯
- 5.过度宽泛的工具权限授予：allowed-tools 配置过于宽松
- 6.不安全的脚本执行环境：脚本直接在用户系统上以用户权限运行
- 7.供应链依赖污染：Skill 依赖的外部库或脚本被污染
- 8.敏感数据处理不当：硬编码 API 密钥、明文存储敏感信息
- 9.持久化信任模型滥用：利用一次批准后的持久信任进行恶意操作
- 10.缺乏运行时行为监控：执行过程中没有实时异常检测

5.5 Skills 供应链安全工具链

ESET AI Skills Checker：ESET 于 2026 年 3 月推出的免费工具，分析 Skill 的行为——包括在实际 Agent 对话中的表现，追踪整个执行链条。不同于传统杀毒引擎，该工具关注 Skill“实际做了什么”，而非“宣称要做什么”。

VirusTotal Code Insight：新增对 OpenClaw Skills 包（包括 ZIP 格式）的原生支持，基于 Gemini 3 Flash 大模型进行安全优先的行为分析。截至首批报告发布时，已分析超 3,016 个 OpenClaw Skills，其中数百个存在恶意特征。

SkillGuard (Akto)：企业级 Agent Skills 安全平台，提供：

检测员工终端上安装的所有 Skills

实时执行护栏

集中化的 Skill 管理和审计

SafeSkill(微步在线)：一站式 AI Agent 安全平台 (SafeSkill.cn)，支持 Skills 包检测和沙箱分析。

SkillScan (学术研究工具)：多阶段检测框架，集成静态分析和基于 LLM 的语义分类，实现 86.7%的精确率和 82.5%的召回率。

5.6 运行时与 Zero Trust 模式安全原则

最小权限原则：

Skill 仅拥有与“任务边界”匹配的最小权限

不要默认授予所有工具访问权——通过 `allowed-tools` 字段精确限制

沙箱化执行环境：

在 VM 或 Docker 容器中隔离运行 Agent

限制文件系统和网络访问

Cato Networks 建议：“在沙箱或虚拟机中运行 Claude，限制文件系统和网络权限，将 Skills 视为任何其他下载到机器的任意代码。”

策略引擎动态调整：

基于任务上下文动态评估和调整权限

对高风险操作要求额外确认

核心安全准则：仅安装来自自己或权威、可信来源的 Skill。对所有要求“在终端粘贴命令”“下载并运行二进制文件”“执行 curl | bash 远程安装”的 Skills 保持高度怀疑。

5.7 未来安全架构方向

能力基础权限模型（Capability-Based Permission Model）：从“全有或全无”演进为精细化最小权限。每个 Skill 需要声明其所需的具体能力（如“读取 ~/Documents/”而非“读取所有文件”），系统在运行时强制执行。

沙箱执行环境标准化：将 Skill 执行隔离在标准化沙箱中，类似浏览器对网页的隔离模式。沙箱应限制对宿主系统的访问，只开放白名单中的能力。

内容签名与完整性验证：Skill 发布时进行代码签名，安装前验证签名链，确保 Skill 在分发过程中未被篡改。可以借鉴 npm 的签名机制或 Debian 的 GPG 签名体系。

运行时行为监控：实时检测异常操作模式，结合 AI 分析识别隐蔽的恶意行为。不仅检查静态代码，更要监控实际执行行为。

第 6 章 未来展望：趋势、挑战与机遇

6.1 七大开放性挑战

1.跨平台可移植性：虽然 Skills 标准已被各平台采纳，但各平台的实现差异（如脚本执行环境、工具调用方式）仍带来兼容性挑战。同样的 Skill 在不同平台上可能有不同表现。

2.能力基础权限模型：从粗粒度的“信任/不信任”二分法，演进为精细化的能力声明和验证系统。这是 Skills 安全的基础设施级需求。

3.Skill 可信度评估：如何客观、自动化地评估一个 Skill 的质量和可信度？目前的“安装量”“星标数”等指标容易被操纵。SkillTester 和 SkillsBench 是重要的第一步，但距离成熟的信用体系还有距离。

4.组合安全分析：单个 Skill 安全不等于组合安全。多个 Skills 同时激活可能产生意外的交互效应，这类安全问题需要专门的研究。

5.Skill 生命周期管理：版本控制、废弃通知、迁移路径。当某个 Skill 被废弃或有安全漏洞时，如何确保所有使用者都能及时获知并更新？

6.自主 Skill 发现与选择：随着 Skills 数量爆炸式增长，Agent 如何在海量 Skills 中自动评估并选择最优的 Skill 组合？这要比简单的语义匹配更智能的推荐机制。

7.人机协作编排：专家与 AI Skills 在复杂工作流中的无缝协作。如何设计交互模式让人和 Agent 在同个工作流中高效协同，而不是互相干扰？

6.2 技术演进方向

Skills 自主生成与进化

Google ADK 已展示 Agent“自主编写新 Skill”的早期能力。这个方向的愿景是：Agent 在环境中试错 → 策略形成 → 元 Skill 自动提炼 → 新 Skill 诞生。

但 SkillsBench 的研究发出了重要警告：自生成 Skills 在整体上未带来任何显著收益，当前模型尚不具备可靠地自主编写其所依赖的程序性知识的能力。这意味着“Skill 自主进化”仍处于非常早期的探索阶段。

多 Skills 智能编排

从单 Skill 调用演进为复杂工作流自动化。Agent 将能够像指挥乐队一样协调多个 Skills 协同工作，处理跨领域的复杂任务。这需要更强大的编排引擎和任务分解能力。

跨模态 Skills

当前 Skills 主要处理文本、代码和结构化文档。未来的 Skills 将支持图像识别、音频处理、视频编辑等多模态能力。例如，一个“视频剪辑 Skill”可以自动理解视频内容、提取精彩片段、添加字幕和特效。

自修复与 A/B 实验

基于 Evals 与在线反馈自动优化 Skill 结构。Anthropic 为 skill-creator 新增的测试框架正是这个方向的早期实践——可以写 evals、跑基准测试、A/B 对比两个版本的 Skill。未来，Skills 将具备自我优化的能力：根据使用数据自动

调整指令、精简冗余内容、改进工作流。

模型能力提升与 Skill 粒度变化

随着模型本身能力越来越强，一些微小的 Skill（如“如何格式化日期”）可能被模型内化，不再需要显式封装。但高价值、高风险、高复杂度的流程性知识将继续以 Skill 形式存在，且粒度可能变得更细、更专业。

6.3 生态

多平台“一 Skill 多用”：随着标准成熟，同一个 Skill 包将在不同平台上获得一致的行为表现。这会催生一个繁荣的第三方 Skill 开发生态，类似 npm 对 Node.js 生态的推动作用。

Skills Marketplace 商业化：

安全审核服务：第三方机构对 Skills 进行安全认证

企业 Skill 商店：企业内部的私有 Skill 市场

付费 Skill 生态：高质量的垂直行业 Skills（医疗、法律、金融）形成付费市场

订阅制 Skill 服务：按使用量或时间段收费

垂直行业 Skills 专业化：医疗、法律、金融、工业、教育等行业的专业知识将通过 Skills 形式被系统化封装和分发。SkillsBench 的数据显示，医疗健康领域的 Skill 收益高达+51.9pp，远超软件工程的+4.5pp——这说明垂直行业的专业知识缺口最大，Skills 的价值也最大。

6.4 人机协作的长期展望

“能力资产”成为新型护城河：对企业而言，最宝贵的资产将不再是代码库或数据，而是精心打磨的 Skills 集合——这些 Skills 凝聚了企业数十年的行业经验和最佳实践，是竞争对手难以复制的核心能力。

AI 工程师的角色进化：从 Prompt 工程师到 Skill 架构师。2026 年的开发者不再是“写代码的人”，而是编排 AI Agent 完成任务的“指挥官”。技能要求从“会写代码”转变为“会设计工作流、会评估 AI 输出、会编排多 Agent 协作”。

知识工作的结构性重塑：当 AI 能够执行越来越多知识工作时，人类的工作重心将从“执行”转向“定义”——定义什么是好的输出、定义工作应该怎么完成、定义质量和标准。Skills 正是这种“定义”的载体。

新职业形态：

Skill 架构师：设计企业 Skill 体系和工作流

AI Agent 编排师：协调多个 Agent 和 Skills 完成复杂任务

Skill 安全审计员：专门审查 Skills 的安全性和合规性

垂直行业 Skill 工程师：将行业专业知识转化为 Skills

Agent Skills 标志着 AI 能力从“手工打造”走向“工业化生产”——我们不再需要为每个任务从头构建 Agent，而是像搭乐高一样组合现成的 Skills 模块。这种“能力标准化”将大幅降低 AI 应用的门槛，推动 AI 从“少数人的专业工具”走向“大众的日生产力”。

6.5 行动建议

对开发者：

- 1.立即学习 Skills 标准，开始将重复性工作封装为 Skills
- 2.优先使用官方和经过安全审核的 Skills，谨慎安装来源不明的第三方 Skills
- 3.建立个人/团队 Skills 仓库，用 Git 进行版本管理
- 4.使用 skill-creator 和 Evals 工具迭代优化 Skills
- 5.参与社区贡献，分享高质量的 Skills

对企业：

- 1.启动“能力资产化”战略——将内部 SOP、最佳实践、行业 know-how 系统化封装为 Skills
- 2.建立企业私有 Skills 仓库，实现跨团队知识共享
- 3.制定 Skills 安全政策——只允许安装经过安全审核的 Skills，强制沙箱执行
- 4.培养 Skill 架构师和 AI Agent 编排师等新角色
- 5.关注垂直行业 Skills 的商业化机会——既是内部工具，也可成为对外产品

对生态建设者：

- 1.推动 Skills 安全标准的建立——代码签名、沙箱执行、漏洞数据库
- 2.开发更好的 Skills 发现和评测工具
- 3.构建垂直行业的 Skills 生态（医疗、法律、金融等）
- 4.警惕“技能萎缩”——在推广 Skills 的同时，保持人类的专业判断和监督能力

OpenClaw 这类基于 Skills 能力单元构建的 Agent 助手，通常需要调用本

地 Shell、文件系统 API 甚至浏览器控制权来执行任务。若其运行的模型被提示词注入攻击诱导执行恶意指令,攻击者可直接获得当前登录用户的完整桌面权限,进而窃取 SSH 私钥、修改系统配置或植入持久化后门。

本地工作机通常混存个人隐私文件(照片、密码本)与公司源代码。本地部署的 AI 服务往往为了便利会挂载全盘索引,一旦 AI 产生幻觉或遭遇中间人攻击,敏感数据极易意外流出。

云端容器或虚拟机仅挂载指定的数据卷,即便 Agent 被越权控制,破坏范围也局限在隔离的容器内,无法触及本地电脑的硬件控制权或私钥环。本地隐私数据也不会泄露,而且可 24 小时在线运行。发现安全漏洞后,可直接销毁当前实例并从安全镜像重建,恢复时间极短,对学习、了解或尝试开发 AI Agent 友好,而本地机器修复恶意软件可能需要全盘重装。

对于有开发需求的团队或个人,算泥社区平台通过整合国产异构算力资源,为开发者提供了经济 效的算力选择。

References

Introducing Agent Skills (Anthropic 产品公告): <https://claude.com/blog/skills>

Anthropic Agent Skills 官方文档: <https://platform.claude.com/docs/en/agents-and-tools/agent-skills/overview>

Agent Skills 开放标准规范: <https://agentskills.io>

Anthropic 官方 Skills 仓库: <https://github.com/anthropics/skills>

MCP 官方文档: <https://modelcontextprotocol.io>

Claude Code Skills 文档: <https://code.claude.com/docs/en/skills>

ClawHubSkill 市场: <https://clawhub.ai>

skills.sh 排行榜: <https://skills.sh>

skillsmp.com 搜索平台: <https://skillsmp.com>

Weaponizing Claude Skills with MedusaLocker: <https://www.catonetworks.com/blog/cato-ctrl-weaponizing-claude-skills-with-medusalocker/>

From SKILL.md to Shell Access in Three Lines of Markdown: Threat Modeling Agent Skills: <https://snyk.io/articles/skill-md-shell-access/>

VirusTotal. "From Automation to Infection: How OpenClaw Skills Became

a Malware Delivery Channel.” 2026: <https://blog.virustotal.com/2026/02/from-automation-to-infection-how.html>

2026 Agentic Coding Trends Report: [https://resources.anthropic.com/hubfs/2026 Agentic Coding Trends Report.pdf](https://resources.anthropic.com/hubfs/2026%20Agentic%20Coding%20Trends%20Report.pdf)

腾讯新闻 AI 趋势研究白皮书 2026Q1: https://doc.weixin.qq.com/pdf/d3_AT0AIwbzAB8CN99PILheYSXSbd67S?scode=AJEAIQdfAAoq5b1XrGAT0AIwbzAB8

Claude Skills are awesome, maybe a bigger deal than MCP: <https://simonwillison.net/2025/Oct/16/claude-skills/>

Equipping agents for the real world with Agent Skills (Anthropic 工程博客): <https://www.anthropic.com/engineering/equipping-agents-for-the-real-world-with-agent-skills>

Agent Skills Open Standard 发布报道 (VentureBeat): <https://venturebeat.com/technology/anthropic-launches-enterprise-agent-skills-and-opens-the-standard>

Anthropic makes Agent Skills an open standard (SiliconAngle): <https://siliconangle.com/2025/12/18/anthropic-makes-agent-skills-open-standard/>

Agent Skills: Anthropic's Next Bid to Define AI Standards (The New Stack): <https://thenewstack.io/agent-skills-anthropics-next-bid-to-define-ai-standards/>

The Complete Guide to Building Skills for Claude (Anthropic): <https://resources.anthropic.com/hubfs/The-Complete-Guide-to-Building-Skill-for-Claude.pdf>

Claude Skills Comprehensive Guide 2026: <https://anandbg.com/blog/claude-skills-comprehensive-guide-2026>

5 Agent Skill design patterns every ADK developer should know: <https://x.com/GoogleCloudTech/article/2033953579824758855>

Agent Skills in the Wild: An Empirical Study of Security Vulnerabilities at Scale: <https://arxiv.org/abs/2601.10338>

Towards Secure Agent Skills: Architecture, Threat Taxonomy, and Security Analysis: <https://arxiv.org/abs/2604.02837>

Agent Skills in the Wild: An Empirical Study of Security Vulnerabilities at Scale: <https://arxiv.org/abs/2601.10338>

SkillsBench: Benchmarking How Well Agent Skills Work Across Diverse Tasks: <https://arxiv.org/abs/2602.12670>

SkillTester: Benchmarking Utility and Security of Agent Skills: <https://arxiv.org/abs/2603.28815>

