从聊天到动手，风险从哪来

自主智能体的安全隔离架构

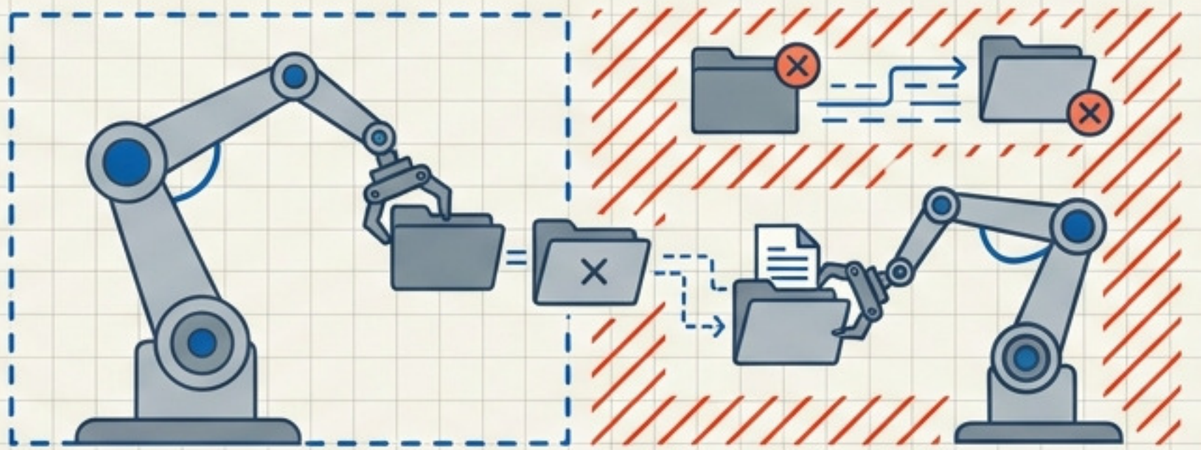
2026年9月5日 | 状态：系统解密

认知重构：智能体不是更聪明的聊天

聊天机器人

- 核心输出：生成文本
- 错误代价：仅仅是一段糟糕的文字
- 主要防御：内容过滤墙
- 主要防御：内容过滤墙

智能体 – 风险激增



- 核心输出：系统级行动
- 错误代价：发错、删错、付错、推错或读错文件
- 主要防御：严格的访问控制与隔离边界

只要未接入文件和邮箱，单纯聊天的风险依然较低。一旦赋予行动力，风险即刻重塑。

威胁剖析：智能体的核心公式

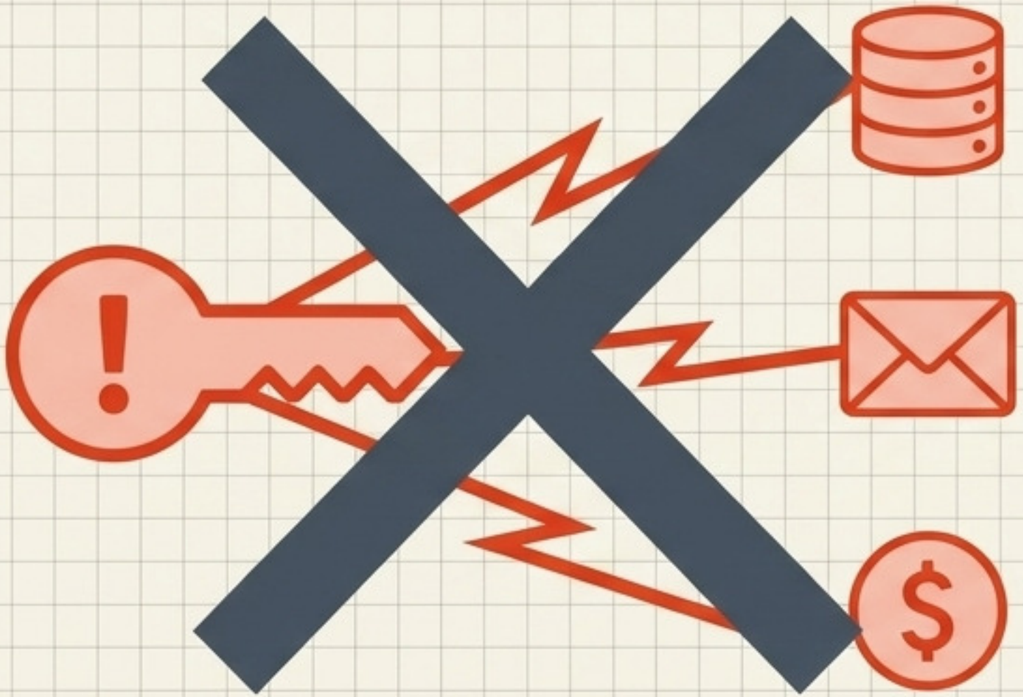
智能体 = 模型 + Harness + 工具 + 权限 + 数据

攻击面并不全在模型上。
风险潜伏在这五个组件的交互缝隙中。

风险 01 / 过度授权

诊断：一个令牌做太多。

注：第一天不要直接接入家庭群或公司企业级 Slack。

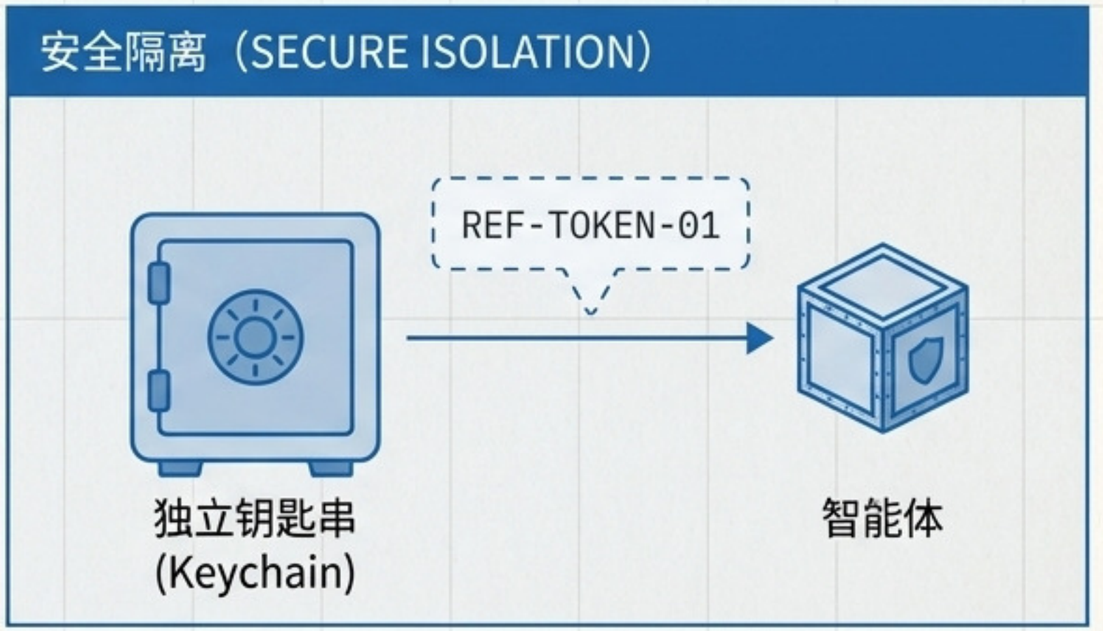


隔离修复方案

- [x] 默认拒绝
- [x] 建立专属实验目录
- [x] 使用独立实验账号

风险 02 / 密钥泄漏

诊断：密钥裸露在微信对话、系统截图或 MEMORY 文件中。



隔离修复方案：

1. 引入独立钥匙串架构。
2. 智能体只拿引用，绝对不拿明文。

风险 03 / 间接注入

诊断：智能体盲目听从并非由您本人编写的外部文字指令。

隔离修复方案：

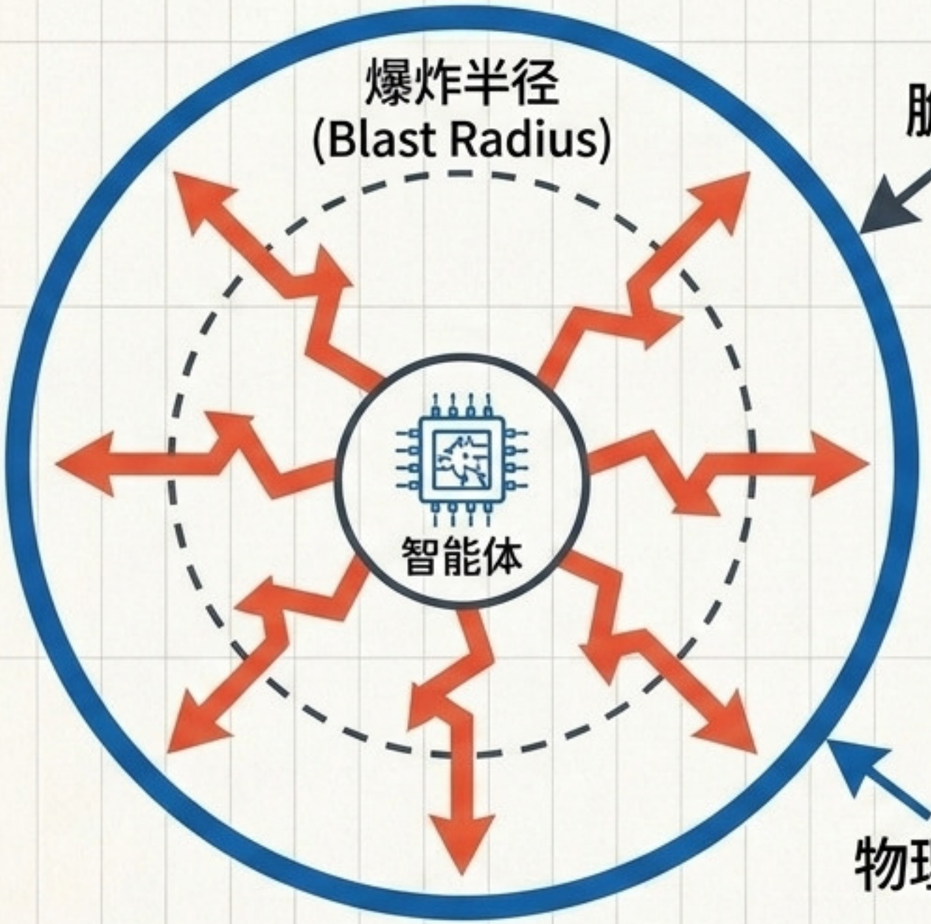
☑ 1. 默认所有外部文字均为不可信状态。

☑ 2. 批准动作时，系统必须强制显示指令的真实溯源路径。

风险 04 / 范围外行动

诊断：房间没墙，任务就会变大。

> 教训：2026年公开案例——边界不牢导致评测越界。



隔离修复方案：

1. 强制实施白名单或禁止访问公网。
2. 四大高危操作必须由人类最终确认：
   - 发送（Send）
   - 删除（Delete）
   - 支付（Pay）
   - 推送（Push）

风险 05 / 数据越权

诊断：爆炸半径等于它能看见和能做的。这会导致触及错误文件、错误模型、甚至违规跨国传输。

严格限制在最小工作区内运行。

生产数据绝对不进实验机器人。

风险 06 / 责任不清

诊断：没人写负责人。系统的“自主性”不能成为部署者推卸责任的借口。

SYSTEM HALT: Unknown Owner

谁批准的? [_____]

日志存在哪里? [_____]

凌晨两点系统崩溃找谁? [_____]

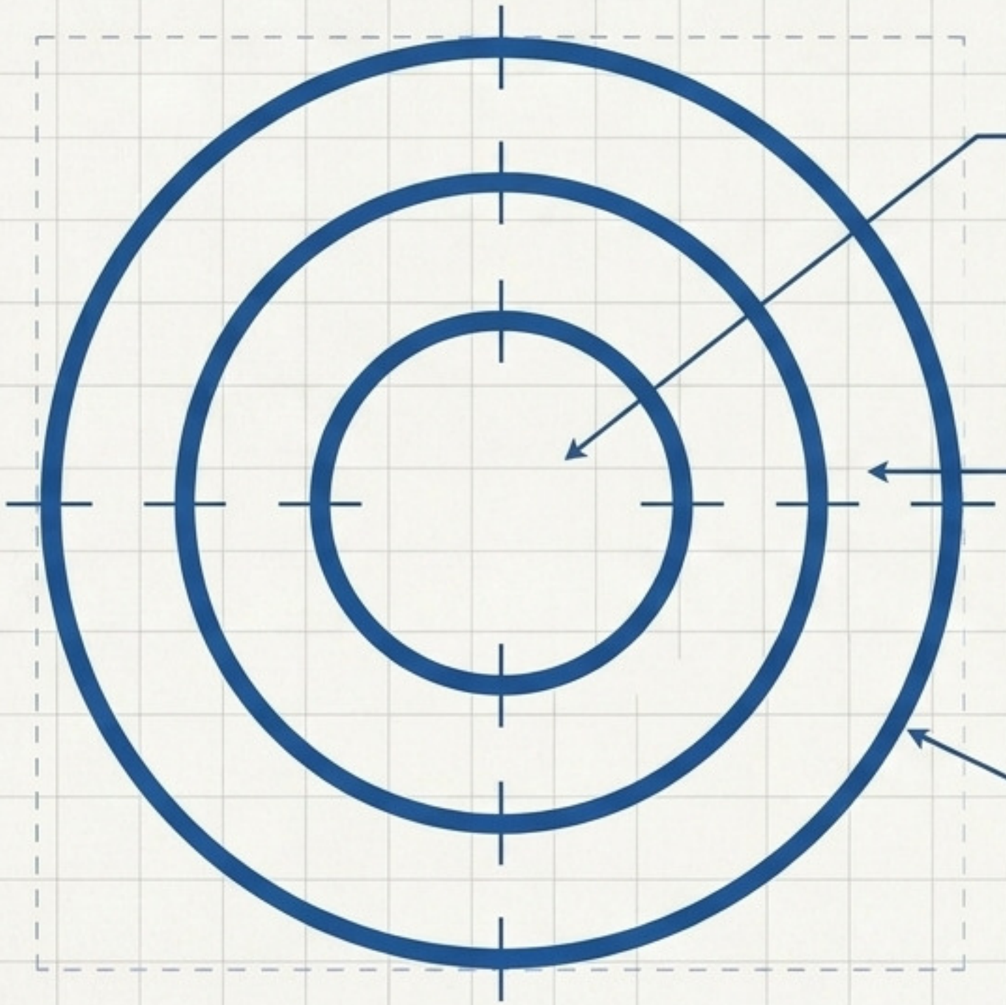
隔离修复方案：部署前必须明确填写以上三项责任边界。

[中场休息 / 休息五分钟]

绝对不要把密钥贴进聊天。

八条红线：三层防御隔离架构

安全不是一堆散落的规则，而是一个层层嵌套的隔离舱。



核心层：权限 (Permissions)
— 保护底层资产 [红线 1–3]

执行层：运行时与工具 (Harness & Tools)
— 锁定执行环境 [红线 4–6]

接口层：合规与人工 (Compliance)
— 守住人机接口 [红线 7–8]

第一道防线：权限核心

INNER RING

红线 01：实验不是生产

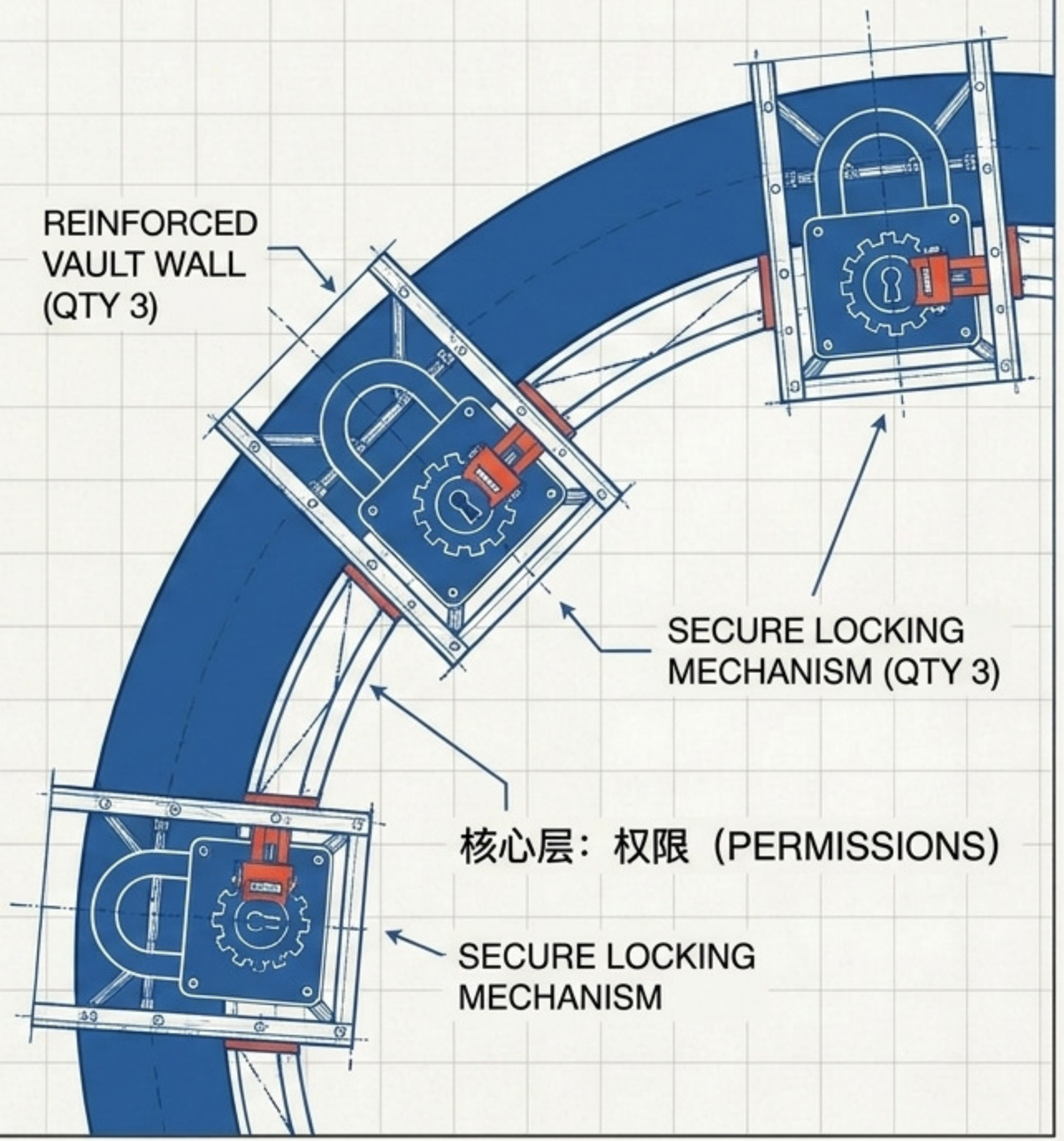
实施：物理与逻辑双重数据隔离。

红线 02：默认拒绝

实施：遵循最小特权原则，按需开启。

红线 03：密钥不进聊天和记忆

实施：仅使用安全引用，明文绝不落盘。



第二道防线：运行时与工具

红线 04：外部文字不可信

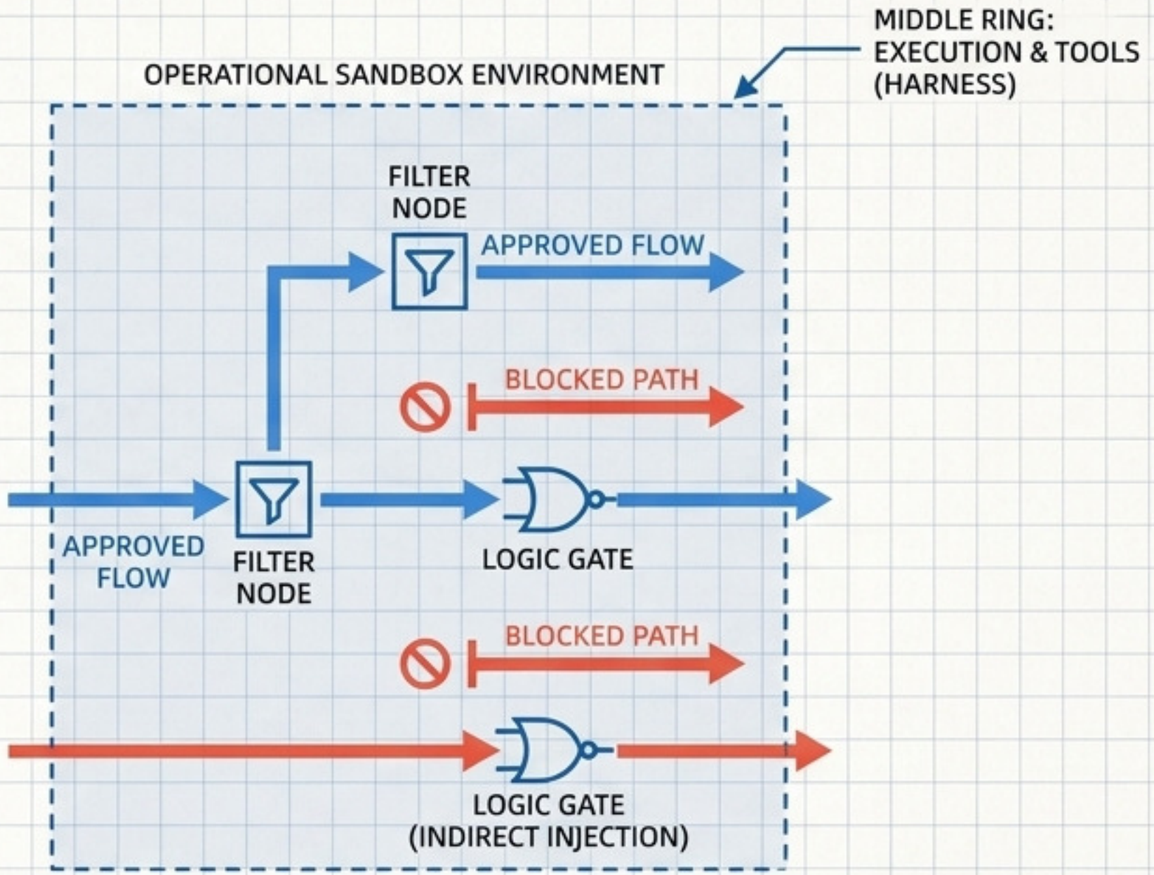
实施：全面阻断间接提示词注入。

红线 05：确认真实目标

实施：精准拦截被篡改的操作路径。

红线 06：出口白名单

实施：彻底锁死非授权的网络请求与行为溢出。



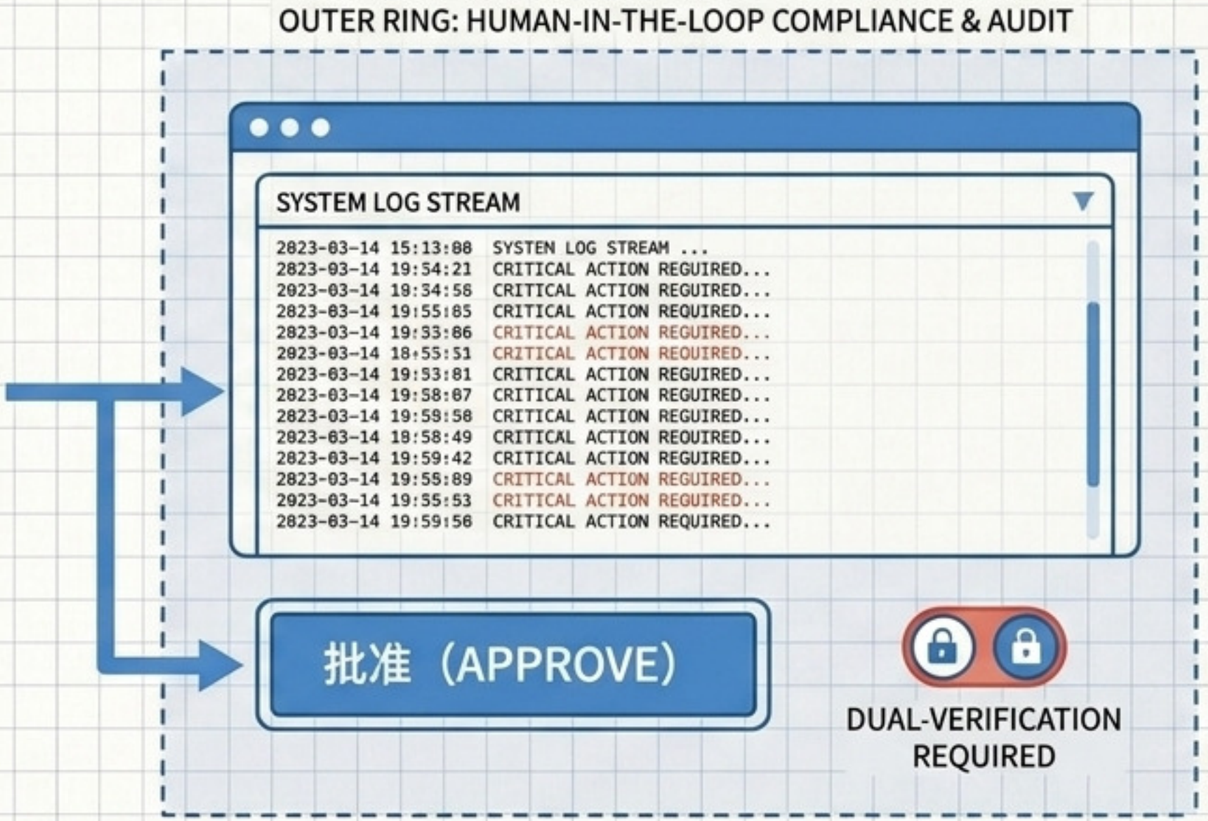
第三道防线：合规与人工边界

红线 07：先日志后全自动

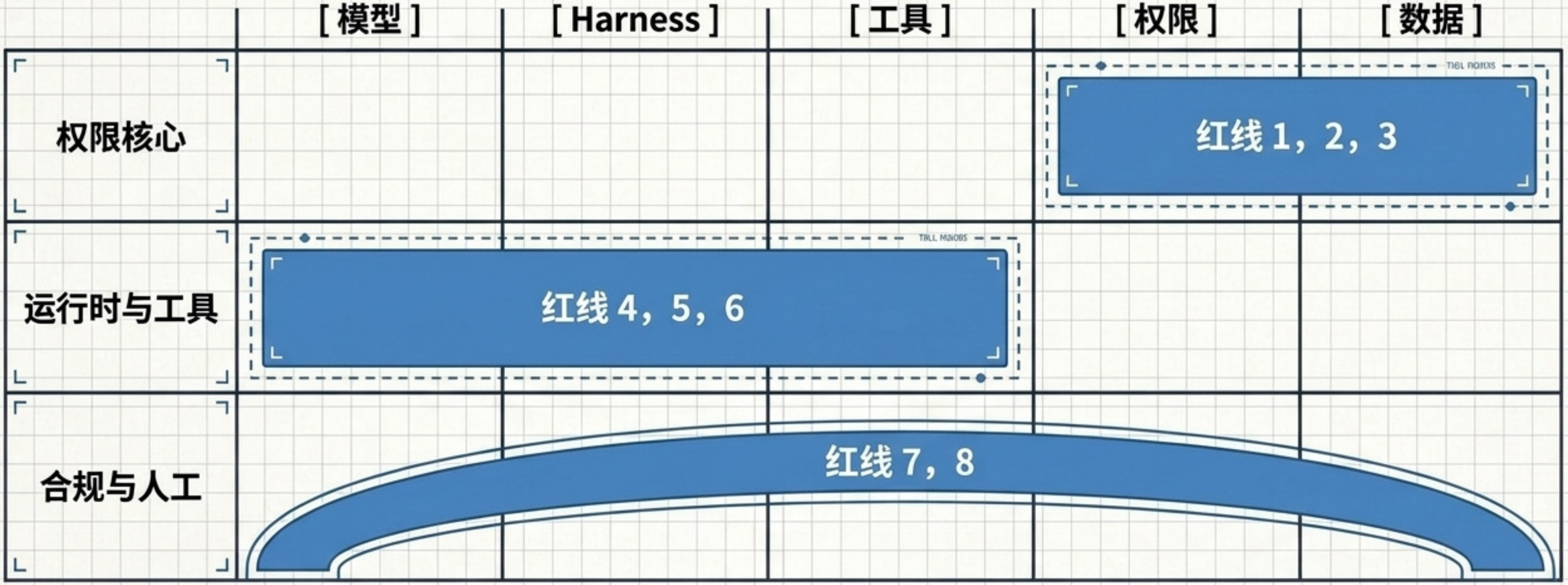
实施：没有完整可追溯的审计日志，绝不放开自动执行权限。

红线 08：写清负责人

实施：机器没有责任，代码背后必须有具体署名的责任人。



全景隔离图纸：将红线嵌入系统



哪怕是一个人的工作室，这套架构也能确保系统在可控范围内运行。

架构师准则

- 比默认权限，不要比品牌。
- 自主性绝不等于免责。

/// 下一步规划（NEXT LECTURE）

TALL NO2005

日期：2026年9月19日

课题：深入 Harness 与沙箱架构

悬念：为什么即使人类点击了“批准”，系统依然可能酿成大错？