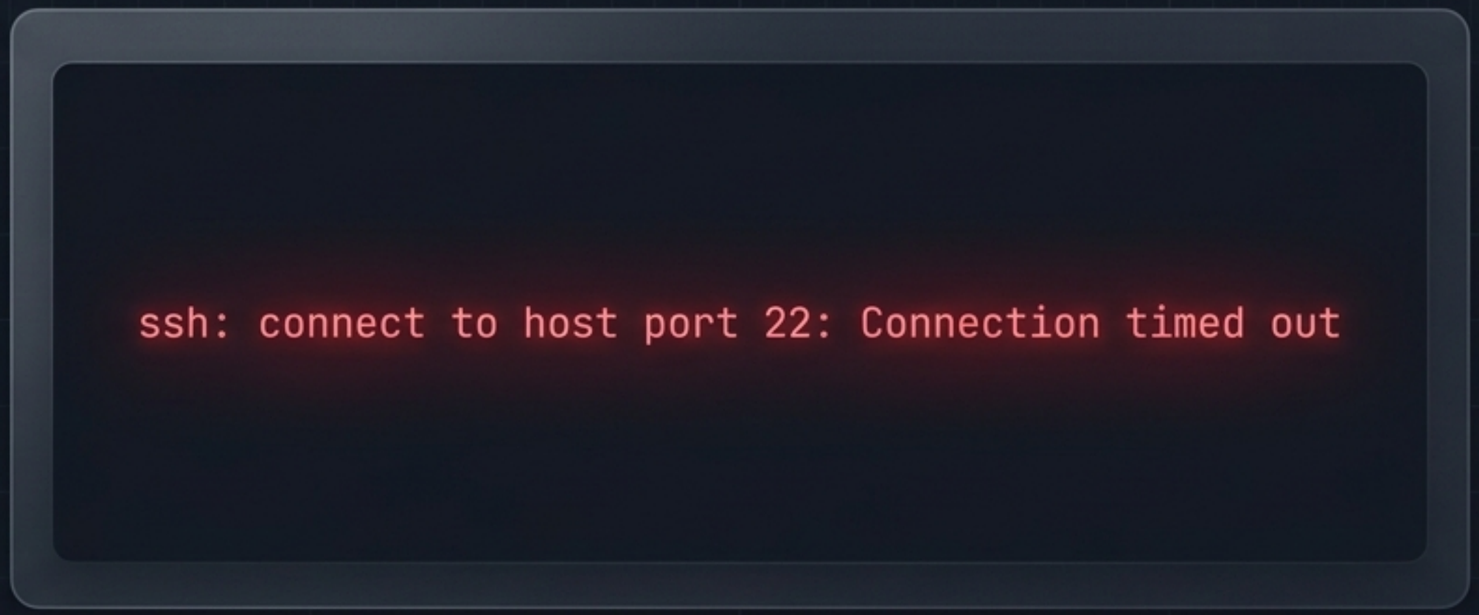


凌晨八点，我的开发机死了两次

一次关于 VM 运维纵深防御的完整勘探记录

[Status: Offline]

[Target: GCP c3-standard-4]

[OS: Ubuntu 24.04]

诊断第一步：TCP 存活，用户态冻结

SSH 握手卡在 banner exchange 阶段，意味着内核网络栈依然存活，但 sshd 进程已无法响应。

User Space

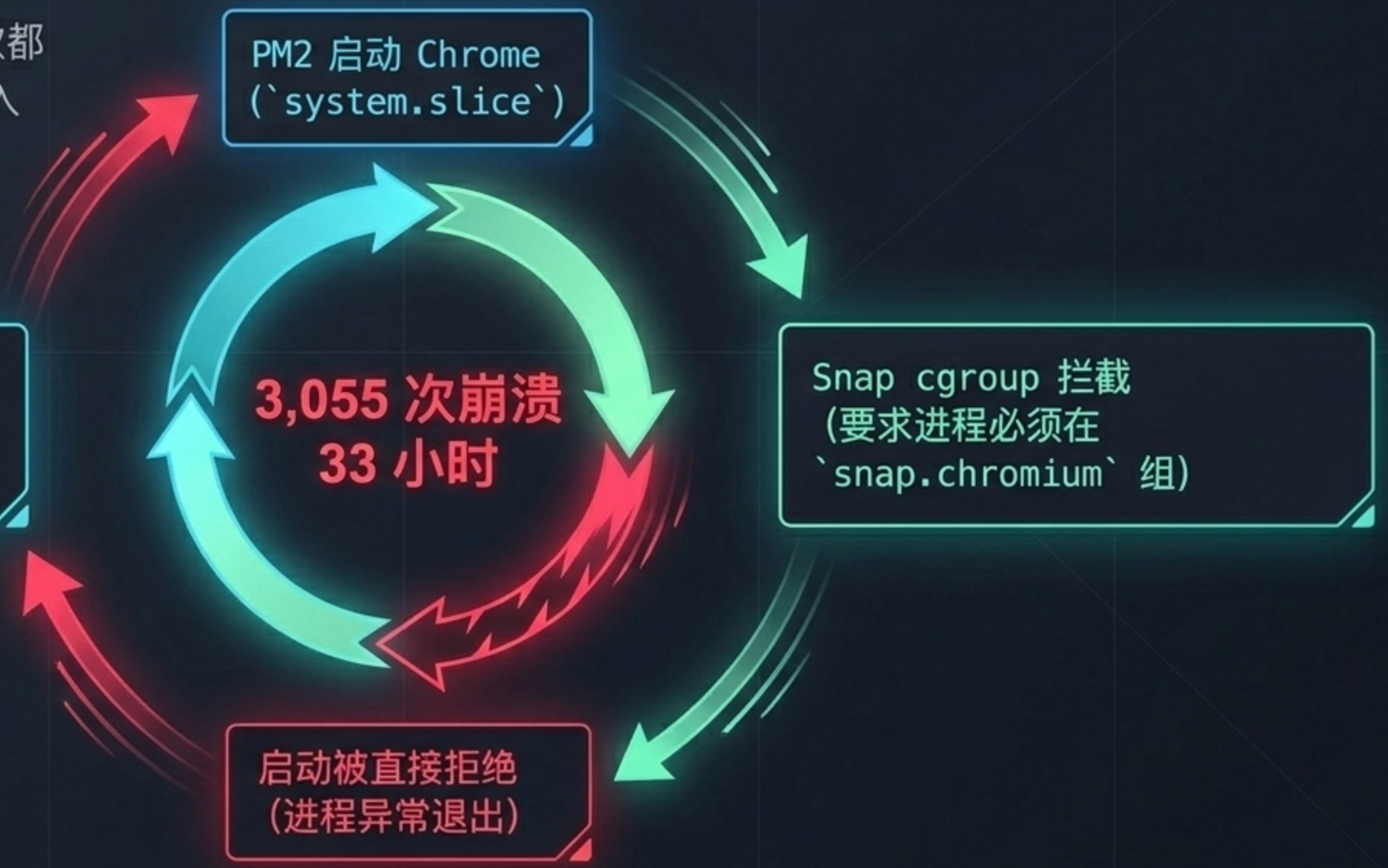
```
....
....
....
rsyslogd omfile action suspend/resume
....
....
```

Evidence Box

结论：连日志服务都无法将数据写入磁盘，系统资源已彻底耗尽，发生严重死锁。只能硬重启。

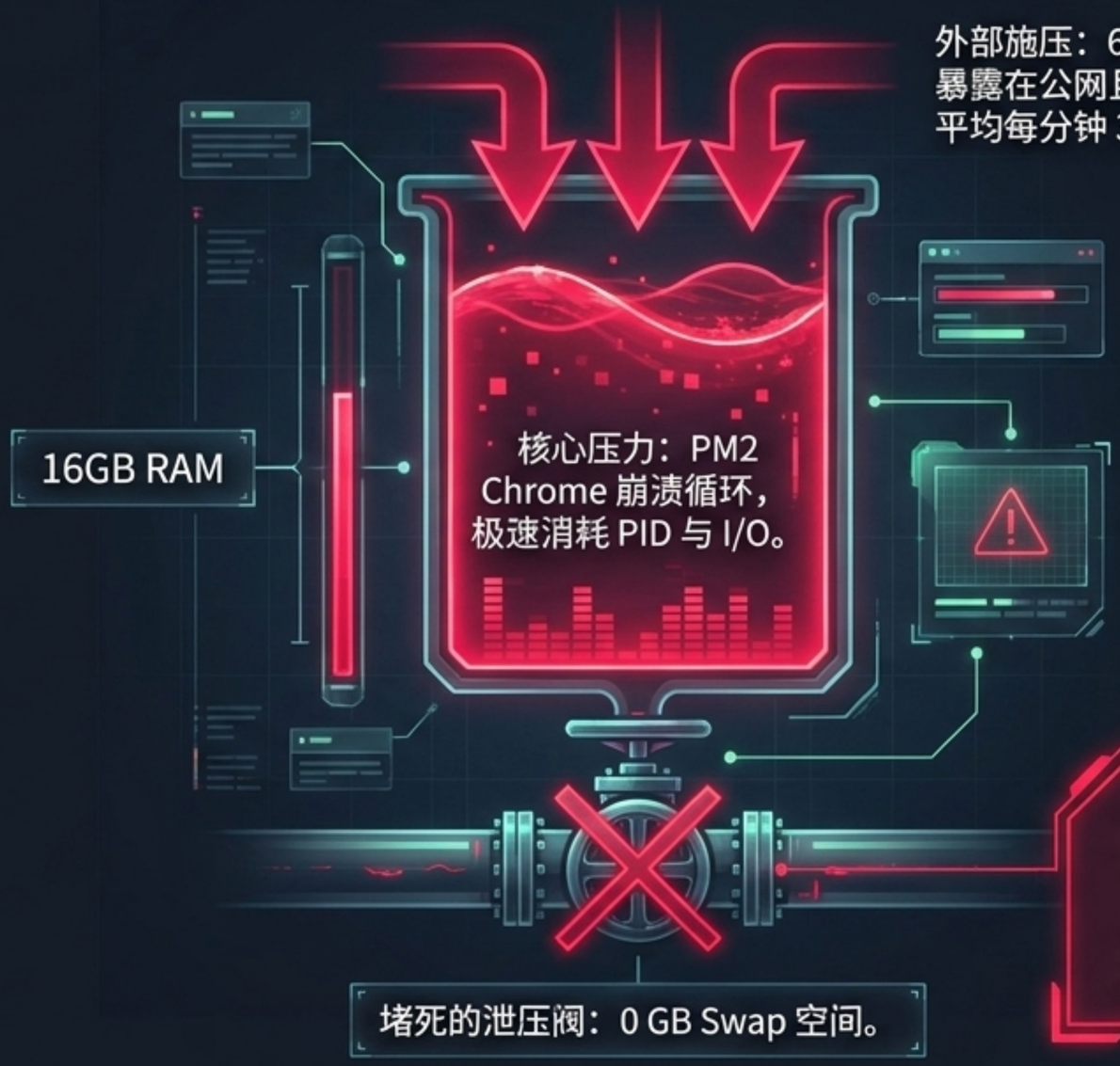
致命的进程隔离冲突：Fork 炸弹是如何形成的

每秒钟数百次的无效重启。每一次都在不断 fork 进程、分配内存、写入错误日志，最终耗尽所有 PID 和磁盘 I/O。



雪上加霜的系统高压锅

外部施压：6,066 次 SSH 暴力破解尝试。
暴露在公网且无 fail2ban。
平均每分钟 3 次额外的 sshd 子进程 fork。



没有 Swap 意味着系统从“高压”到“死亡”之间没有任何缓冲。内存一旦触顶，OOM killer 来不及回收，直接导致内核死锁。

第一幕修复：五道战术止血防线

1. 替换 Snap Chrome： 卸载 snap 包，换用 .deb 原生包，彻底消除 cgroup 隔离冲突。

2. PM2 熔断机制： 强制增加 `max_restarts` 和 `restart_delay`，阻止崩溃循环演变为 Fork 炸弹。

3. 封锁爆破攻击： 部署 `fail2ban`，十分钟内封禁首个恶意 IP。

4. SSH TCP 限流： 配置 `MaxStartups 5:50:10`。在 TCP 层面（认证发生前）直接按概率丢弃超量连接。

5. 开启缓冲窗口： 挂载 4GB Swap 空间（内存的 25%），给 OOM killer 留出操作余地。

几个小时后，它又死了。

System Stable (Uptime: 4 hours)

第一轮修复治了表面症状（崩溃循环），
却没有改变系统的羼弱体质（资源无上限）。

这一次，我们将系统开膛破肚。

深入解剖：无上限的内存黑洞

致命缺陷： Docker 容器与 Node 进程均未设置内存上限，它们可以无限膨胀，直到吃光宿主机最后一字节。

基础设施错配：为不需要的性能付溢价

开发机的工作负载主要是“等待 API 响应”和“空闲 IDE”，极少消耗 CPU。切换至通用型高内存实例，不仅内存翻倍，月费反而下降 $21。

指标	c3-standard-4（当前）	e2-highmem-4（推荐）
定位	计算优化型（Intel Sapphire Rapids，HPC/游戏）	通用型（共享核心，适合突发负载）
vCPU	4 核	4 核
内存	16 GB	**32 GB（翻倍）**
月费	~$152	~$131

第二幕根治（上）：系统内核与生存优先级

彻底肃清余毒：`snap remove chromium --purge`。斩草除根，消除底层 AppArmor 潜在冲突。

Sysctl 内存调优

1. `vm.swappiness=10`：极力避免使用 Swap，优先回收缓存。
2. `vm.overcommit_memory=0`：严格模式，拒绝进程申请超出实际可用的内存。

OOM Killer 处决优先级

`sshd`：`oom_score_adj = -1000`（最高免死金牌，确保永远可通过 SSH 登录救场）。

`Chrome`：`oom_score_adj = +500`（优先牺牲品，内存紧张时率先被杀）。

第二幕根治（下）：给所有进程戴上紧箍咒

Docker 硬件隔离

在 docker-compose 中硬性设置内存上限（Gateway 1.5GB，LiteLLM 512MB）。

PM2 自动重启红线

配置 max_memory_restart，Chrome 超出 400MB、其他服务超出 200MB 自动重启单一进程，防止同归于尽。

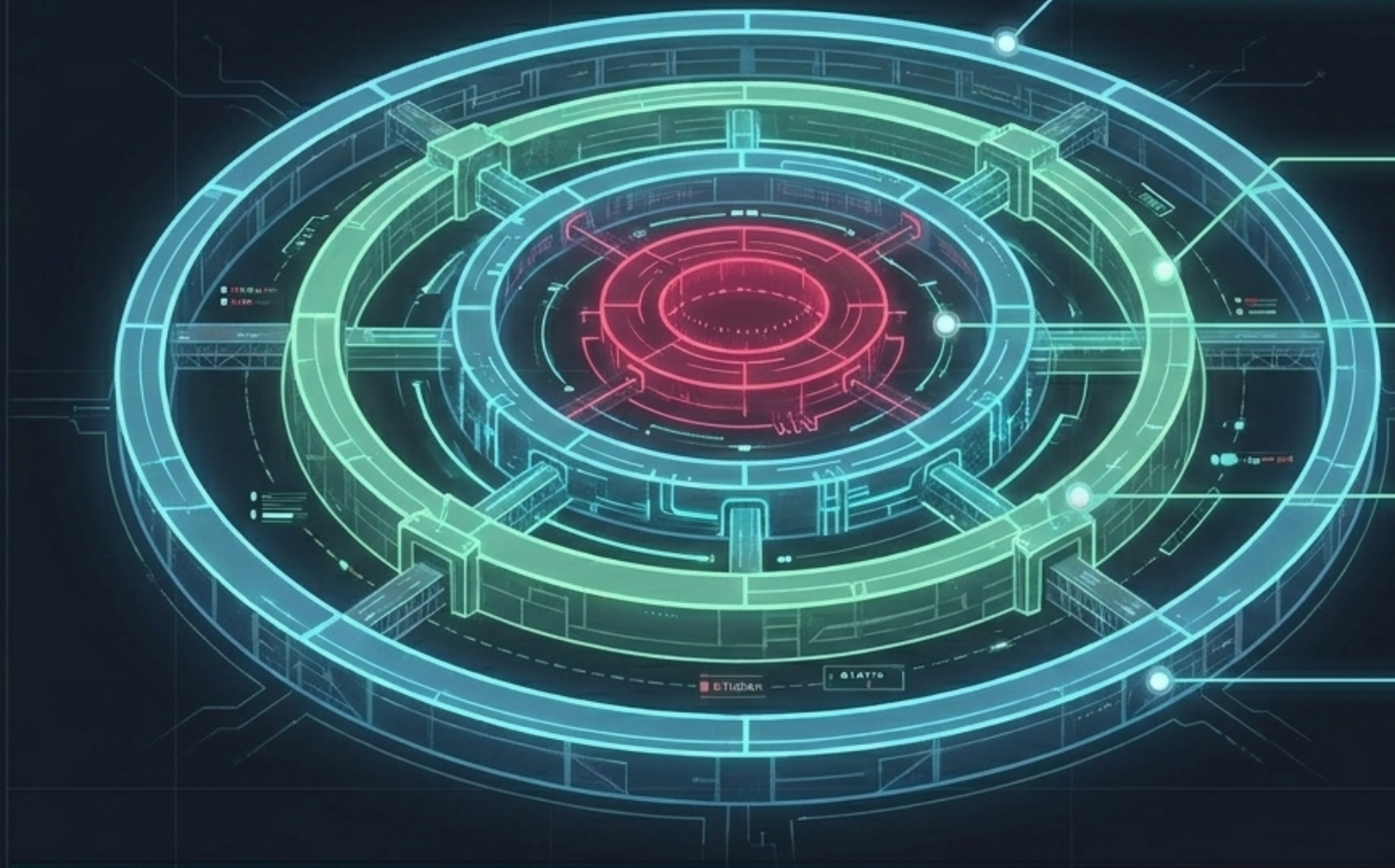
内存看门狗（Watchdog）

编写 Cron 脚本每分钟巡检。使用率超 85% 自动斩杀 Chrome，超 95% 杀高耗能 Node 进程并记录日志。

GCP Ops Agent 告警

部署云端原生监控，当内存使用率持续 2 分钟突破 85% 时，触发手机告警推送。

纵深防御：12项加固构建的系统壁垒



Layer 1: 基础设施层 (Infrastructure)
e2-highmem 充足内存（硬件底座）

Layer 2: 网络防线 (Network)
fail2ban 拦截爆破，SSH MaxStartups 限流

Layer 3: 监控与告警 (Observability)
Ops Agent 实时监控，Cron Watchdog

Layer 4: 内核管控 (Kernel)
Swap 缓冲，Sysctl 调优，OOM 免死金牌

Layer 1: 基础设施层 (Infrastructure)
e2-highmem 充足内存（硬件底座）

任何单点故障都不应致命。一个坏进程搞垮整台机器，是因为所有防线同时失守。

运维的终极拷问

治标 (Fixing the Surface)

消除当前的报错日志。修复了崩溃循环，但系统依旧脆弱。

治本 (Fixing the Root)

建立资源配额与监控边界。假设所有的进程随时都会失控。

不要在第一次修复后就觉得安全了。
永远问自己一个问题：如果这个修复生效了，系统还有什么方式可以死？
—— 保持敬畏，防线永不嫌多。