Authority Is Not Ambient: A Mediated Control Plane for MCP-Style Tool Use and Local Computer-Use Agents

Alex Price (StellarRequiem) | security@xclusivexo.com | https://xclusivexo.com
2026-08-03 | Working paper v1.0 | claim-safe | open artifacts

Abstract

Agent systems that call tools-especially Model Context Protocol (MCP) servers and computer-use interfaces-create a new trust boundary: the model *proposes* actions that execute with operator privileges. We describe a mediated control plane that treats tool calls as gated requests rather than ambient authority. The design separates (1) a deny-by-default runtime assurance layer for MCP-style calls (mcp-assure), (2) local loopback leashes for Chrome and macOS desktop input under arm/allowlist/human high-blast gates (browser-leash, desktop-leash), (3) a single assured host that routes plane tools through AdaptiveGate (agent-control), and (4) an agent-plane collector/detector/responder for receipt-shaped abuse (agent-soc), including abhorrent tool-shape lockdown via FREEZE.

We emphasize what the system *does not* claim: it is not an enterprise SOC, not unlimited computer use, not a guarantee against all agent attacks, and not automatic gating of a host runtime's native shell unless that runtime is configured to route through the host. Evaluation is framed as re-runnable hold-tests, purple fixtures, and smoke suites rather than unmeasured detection rates. Public package surface: mcp-assure (PyPI/GitHub). Local open-source control-plane siblings: browser-leash, desktop-leash, agent-control, agent-soc.

Keywords: MCP security, agent tool use, deny-by-default, computer use, runtime authorization, agent-plane SOC

1. Introduction

1.1 Problem

LLM agents increasingly:

* invoke tools (file systems, browsers, shells, SaaS APIs) via MCP or host-native tool APIs;
* operate computer-use loops (observe -> act -> verify) over GUI surfaces;
* publish or mutate external state (social posts, PR comments, mail) with high blast radius.

The failure mode is systematic: capability is confused with authority. Once a tool is "available," models and orchestrators tend to treat it as ambient power-including high-blast actions (publish, quit apps, shell execution) that operators would never grant continuously.

Concurrently, MCP security discourse has matured around OAuth for remote servers, tool integrity, and gateway RBAC [CSA Agentic MCP guidance; NSA MCP security design considerations, 2026]. Those layers matter, but they do not fully answer a local, multi-plane question:

*When an agent runs *on the operator's machine* with Chrome, desktop accessibility, and a high-trust shell, who decides which mediated action may execute *this turn*-and can we prove refuse paths?*

1.2 Contributions

We contribute a working architecture and open implementations:

1. mcp-assure - plug-in deny-by-default assurance for MCP-style tool calls: policy packs, velocity/blast limits, forbidden args, hash-chained receipts, campaign/AdaptiveGate, purple fixtures, FastMCP middleware integration.
2. browser-leash - local Chrome control plane (loopback bridge + extension): sticky ARM, host allowlist, tiered actions, human post confirm, ambient multi-tab surface, X article read/search under arm.
3. desktop-leash - local macOS desktop control plane: arm, app allow/deny lists, layout-first window control, D4 dual-control for high-blast keys/quit, budgeted CUA depth.

4. agent-control - AssuredPlaneHost: single choke point (AdaptiveGate -> handlers -> leashes), session CUA, gated shell subset, lifecycle (available / session / lockdown), launchd always-available *infrastructure* without always-on *authority*.
5. agent-soc - agent-plane collect/detect/respond on control receipts; abhorrent-shape rules; FREEZE ± optional disarm; watch that auto-freezes only on abhorrent class (not every "high").

1.3 Design thesis

Authority is not ambient.
Capability can rise under leashes; authority stays deny-by-default, session-armed, and human-gated on high blast.

2. Threat model

2.1 In scope (mediated plane)

```
Adversary / failure | Example
Model tool spray | Unknown tools, catalog probing
High-blast churn | Repeated post/quit/Return without confirm
Arg shape abuse | Path traversal, template injection smells, credential-shaped args
Campaign / swarm shape | Velocity + multi-tool deny patterns
Unarmed session use | Actions while DISARMed
Allowlist escape | Host or app not on list
Operator confusion | Treating draft as publish; treating freeze as "all shell stopped"
```

2.2 Out of scope (explicit)

```
Surface | Why
Enterprise SIEM / EDR / network SOC | Wrong product class
Unmediated native host shell | Bypasses FREEZE unless runtime routes through host
Formal proof of all agent misbehavior | No holdout detection-rate study claimed
Remote MCP OAuth AS implementation | We check bindings when provided; we are not an authorization server
Stopping data-plane / K8s incidents end-to-end | Different surfaces
```

2.3 Assets

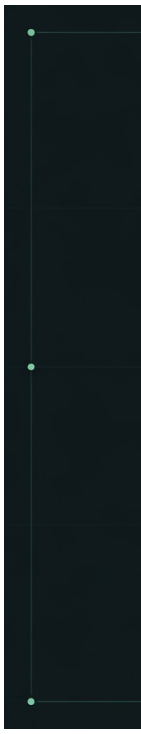
* Operator machine integrity and privacy (screenshots may contain PII-local only).
* External accounts (X, GitHub) under human publish gates.
* Integrity of decision receipts and freeze state.

2.4 Trust assumptions

* Operator controls arm/disarm and Soft Reload of the browser extension.
* Loopback bridges are not exposed off-host.
* Policy packs and freeze allowlists are operator-controlled.

3. Architecture

Figure 1. Mediated control plane (AssuredPlaneHost, leashes, agent-soc).



3.1 Layered stack

```
                  ????????????????????????????????
                  ?  Operator / Agent (Grok, ...) ?
                  ????????????????????????????????
                                 ? prefer mediated path
                  ????????????????????????????????
                  ?     AssuredPlaneHost        ?
                  ?  AdaptiveGate + Dispatcher  ?
                  ?  pack: local_planes.json    ?
                  ?  FREEZE . receipts          ?
                  ????????????????????????????????
     browser.*    ?          ? desktop.*?   shell.* / cua.* / plane.*
????????????????????  ????????????????????????
? browser-leash    ?  ? desktop-leash      ?
? :8756 + extension?  ? :8757 AX/input     ?
? ARM . hosts . T0-T4? ? ARM . apps . D4  ?
????????????????????  ????????????????????????
                  ?
                  ?
agent-soc: collect receipts -> detect -> FREEZE / disarm
```

3.2 Planes

```
Plane | Role | Authority controls
**Tool (MCP-style)** | Authorize tool name + args before handler | Deny-by-default pack, velocity, freeze, arg smells
**Browser** | Chrome session actions | Sticky ARM, host allowlist, tiers, post confirm
**Desktop** | Non-Chrome GUI | Sticky ARM, allow/deny apps, D4 confirm, layout-first
**Shell subset** | Named/path-confined commands | Pack allowlist; **not** ambient bash
**Agent-plane SOC** | Receipt analytics + lockdown | FREEZE files; optional DISARM
```

3.3 Always available ? always armed

Infrastructure (bridges, optional watch) may run at login via launchd KeepAlive. ARM remains intentional. This distinguishes *availability* from *authority*-a recurring operational failure mode when agents stay "computer-use ready" 24/7.

3.4 Dual human gates (high blast)

Examples:

```
Action | Host | Leash
`browser.x_post` | `operator_confirm=true` this turn | ARM + post confirm
`desktop.quit` / Return | `operator_confirm=true` | D4 session + confirm queue
```

Agents must not invent operator_confirm=true.

4. Components

4.1 mcp-assure (public package)

Role: The model proposes; the gate decides.

Properties (claim-safe):

* Deny-by-default tool catalog.
* Velocity / blast / lab flags / freeze / forbidden args.
* Hash-chained decision receipts (tamper-detecting).
* Middleware does not execute on DENY/DRY_RUN.
* Campaign watch + AdaptiveGate (escalate/freeze on spray/swarm *shape*).
* Proactive arg-smell blocks (path/template/packer class markers).
* Purple fixtures and mcp-assure check CLI for re-runnable CI.
* FastMCP on_call_tool middleware path.

Public: https://github.com/StellarRequiem/mcp-assure . PyPI mcp-assure
Not claimed: full SOC, CVE scanner replacement, "stops all MCP attacks," production scale.

4.2 browser-leash (local Chrome plane)

Loopback HTTP bridge + Chrome extension (poll/hello). Sticky ARM (toggle; optional TTL). Host allowlist (professional surfaces: e.g. x.com, github.com, xclusivexo.com, ...). Action tiers T0-T4 (status -> observe -> interact -> compose -> publish).

Ambient Chrome surface (v0.5.x era): tabs list/create/close/activate, navigate, wait, find, links, snapshot, screenshot, click/type/press/scroll, history back/forward/reload, X article read/search. Fail-fast EXTENSION_DISCONNECTED when extension stops helloing.

Not claimed: free browsing, auto-post, parity with proprietary vendor native messaging hosts as a product clone.

4.3 desktop-leash (local macOS plane)

Observe (apps/windows/screenshot/AX), allowlisted focus/input, hard denylist (password managers, Mail, ...), layout-first window-local clicks, D4 dual-control for high-blast, CUA observe bundle, profiles (strict / cua).

Not claimed: unlimited human-equivalent CUA, unattended OS takeover.

4.4 agent-control (assured host)

Single operator/agent CLI:

* session - one JSON surface: available, arms, freeze, lockdown summary, claim ceiling, SOP.
* available - infrastructure readiness without requiring ARM.
* up / down / stack - lifecycle.
* call <tool> - all plane tools through AdaptiveGate.
* smoke - cannot-bypass suite (unknown tools, high-blast without confirm, 1Password deny, shell path confinement, ...).
* lockdown - proxy to agent-soc.

CUA: budgeted session (default work budget 40 steps / 30 minutes), layout included in observe, layout-first before GUI click steps.

4.5 agent-soc (agent-plane SOC)

Collects local receipt streams (leashes + plane-host). Detects deny spikes, high-blast churn, unknown-tool attempts, risky action shapes, and abhorrent classes (secret/path/template/shell-bypass/tool-spray). Respond: write FREEZE (agent-control + mcp-assure), optional disarm leashes, open incident JSON. Watch: continuous ticks; auto-FREEZE only when abhorrent class recommends engage (not every generic high).

Not claimed: enterprise SIEM/EDR, network monitoring, stop-all-attacks.

5. Abhorrent lockdown (Phase C)

5.1 Philosophy

Abhorrent *tool-plane* signals are shapes on mediated calls, not a content-moderation product for all model text. Examples:

* credential-shaped arguments;
* ambient shell_exec probes;
* catalog spray (many UNKNOWN_TOOL);
* path/template injection smells;
* mass high-blast without operator confirm.

5.2 Lockdown levers

1. FREEZE - host pack only allows freeze-allow tools (status/route/roots).
2. Optional DISARM - browser/desktop arm off (env AGENT_SOC_AUTO_DISARM or explicit flag).
3. Incident record - append-only under agent-soc/incidents/.

5.3 Recovery

lockdown clear -> inspect incidents -> intentional re-ARM -> resume mediated path only.

6. Verification methodology

We refuse detection-rate percentages without a holdout study. Instead we use re-runnable evidence:

```
Class | Example
Unit / property tests | mcp-assure P-suite; agent-soc detector thresholds; shell path confinement
Purple fixtures | mcp-assure purple; agent-soc `purple.py` abhorrent shapes
Cannot-bypass smoke | `agent-control` smoke: unknown tool DENY, x_post/quit/Return without confirm, 1Password deny
Operator hold-tests | Unarmed refuse; confirm-reject; post-disabled refuse
Live integration | Multi-tab ambient browser; FREEZE blocks navigate; status still ALLOW; clear restores
```

6.1 Known residual gaps (honest)

```
Gap | Implication
Native Grok/runtime shell outside host | FREEZE does not stop it; SOP: prefer gated `shell.*`
Extension SW disconnect / hang history | Mitigated by timeouts + EXTENSION_DISCONNECTED; still operationally sensitive
Semantic intent inside *allowed* tools | Only partially covered by arg smells / host lists
No third-party formal red team report | Hold-tests ? formal RT
```

7. Related work

MCP security guidance. CSA agentic MCP practices; NSA MCP security design considerations; industry gateway/RBAC products. These often emphasize remote server auth, identity, and audit. Our focus is local

mediation of tool proposals and GUI planes under explicit claim ceilings.

Host-side enforcement wrappers. Academic and industry "MCP-secure" / least-privilege wrappers regulate tool use at the host. mcp-assure is in this class, with AdaptiveGate campaign shape response and zero core runtime deps as a packaging constraint.

Computer use (vendor). Proprietary Chrome native hosts and OS computer-use stacks (Claude/Codex-class). We do not claim feature parity; we claim a deny-by-default control philosophy with open local leashes.

Runtime application self-protection / policy engines. Conceptual relatives (decide before side effect). Difference: agent *proposals* are untrusted text-shaped intents that must not authorize themselves.

8. Operational model

8.1 Day loop

```
launchd: bridges (+ freeze-only watch) always available
operator: Soft Reload extension if version mismatch
operator: ARM browser + desktop for work session
agent:    cli.py session -> mediated browser/desktop/CUA/shell.*
high-blast: human confirm only
end:      DISARM (leave bridges up)
incident: lockdown engage/clear as needed
```

8.2 Claim ladder

Public language follows Lead -> Candidate -> Verified -> Published, always weaker than evidence. mcp-assure is the primary public package claim. Leashes and host are open-source code with local verification narratives unless separately productized.

9. Reproducibility (public repos)

```
Component | Repository (StellarRequiem) | Notes
mcp-assure | github.com/StellarRequiem/mcp-assure | PyPI; CI `mcp-assure check`
browser-leash | github.com/StellarRequiem/browser-leash | Loopback + extension
desktop-leash | github.com/StellarRequiem/desktop-leash | macOS local plane
agent-control | github.com/StellarRequiem/agent-control | AssuredPlaneHost
agent-soc | github.com/StellarRequiem/agent-soc | Collect/detect/lockdown
```

Site / contact: https://xclusivexo.com . security@xclusivexo.com

Representative commands:

```
pip install mcp-assure && mcp-assure check
python3 ~/agent-control/cli.py smoke
python3 ~/agent-control/cli.py session
python3 ~/agent-soc/purple.py
python3 ~/agent-control/cli.py lockdown status
```

10. Conclusion

Mediated agent control is a runtime authority problem, not only an authentication or model-alignment problem. By splitting tool gate, browser leash, desktop leash, assured host, and agent-plane lockdown-and by refusing ambient authority and inflated SOC claims-we obtain a stack that is usable for real operator work while keeping refuse paths real and re-runnable.

Future work: formal third-party hold-test report; optional runtime product integration to gate native shell; expanded semantic policies on allowed tools; broader purple corpora; optional public packaging of leashes with install polish.

Acknowledgments

Built under an operator protocol that prioritizes facts over narrative, claim ceilings, and VERIFIED closeouts (tested / results / live-proof / gaps).

Appendix A - Claim ceiling (copy for abstracts)

Allowed one-liner:

A local mediated control plane for MCP-style tool calls and arm-gated browser/desktop computer use: deny-by-default packs, AdaptiveGate, human high-blast gates, and receipt-driven FREEZE lockdown-not a full SOC and not unlimited ambient OS control.

Forbidden one-liners: "stops all agent attacks"; "full computer use like a human"; "enterprise SOC"; "every Grok tool gated"; unmeasured win-rates.

Appendix B - Artifact map

```
Concern | Path
Host architecture | agent-control/ARCHITECTURE.md
Always available vs armed | agent-control/docs/ALWAYS_AVAILABLE.md
Abhorrent lockdown | agent-control/docs/ABHORRENT_LOCKDOWN.md
Shell gap | agent-control/docs/SHELL_CANNOT_BYPASS.md
mcp-assure claims | mcp-assure/CLAIMS.md
Stack claims | desktop-leash/docs/CLAIMS.md
Roadmap | ops/AMBIENT_TO_LOCKDOWN_ROADMAP.md
```

Appendix C - Document history

```
Ver | Date | Note
1.0 | 2026-08-03 | Initial working paper from shipped stack (Phases A-C + harden)
```