Blast Radius Zero

- Let the agent touch the code. Not the machine.
- ‘dsh-plugin-blast-radius‘ - a dsh plugin bundle for Vultr
- Every command runs inside a disposable Microsandbox microVM.
- Every result lands in a hash-chained receipt.
- ‘README.md‘ - ‘LICENSE‘ - MIT

The problem

- An agent that "improves" a codebase runs ‘npm install‘, a build script, a test suite, and fifty scanners. Each one executes third-party code against **your** host.
- Common answer - What it actually gives you
- Run it in the repo - The agent’s shell is your shell
- Run it in Docker - Same kernel, same daemon socket
- Trust the prompt - A prompt is not a boundary
- Read the diff afterwards - The damage already happened
- **The blast radius of one bad command is the whole machine.**

Source: ‘docs/EVIDENCE.md‘, ‘docs/BUILD.md‘ section 6

Containment thesis

- The safety property is not "the agent behaves". It is "the command has nowhere to go".
- Four claims, each one a test, each one fails closed:
- **Process isolation** - execution never touches the host or the app runtime.
- **Secret hygiene** - no key or credential inside the sandbox, only a fake value.
- **Resource limits** - 2 vCPU, 4096 MiB, 8 GiB, 256 pids, 600 s, on every run.
- **Lifecycle discipline** - the environment is destroyed in a ‘finally‘ block.
- If a claim cannot be shown on camera, it is dropped from the deck.

Source: ‘docs/BUILD.md‘ section 6, ‘docs/GATES.md‘ section 1 gate G5

Architecture

- judge ??? nginx :443 ??? dsh 127.0.0.1:3080 ??? msb ??? microVM
- TLS blast-radius plugin ? caps
- ? no key
- ?
- receipts.jsonl
- hash-chained
- Layer - What lives there
- Public edge - nginx, TLS, HSTS, ‘/healthz‘ - the only listener (‘deploy/nginx.conf‘)
- Control plane - dsh Web UI + plugin on loopback (‘deploy/dsh-web.service‘)
- Boundary - Microsandbox microVM, own kernel, OCI image, egress denied
- Evidence - ‘sha256(prev_hash + "\n" + canonicalJson(body))‘ (‘src/receipts.ts‘)

Five tools, one boundary

- Tool - Job - Boundary
- ‘scan_codebase‘ - LOC, dependencies, SBOM counts - inside the microVM
- ‘run_in_sandbox‘ - one argv command, stdout and exit code back - inside the microVM
- ‘measure_before‘ - the metric table, one hash per row - inside the microVM
- ‘measure_after‘ - the same table on the patched copy - inside the microVM
- ‘report_findings‘ - severity from a fixed table, never from a model - host, read-only
- A string command is rejected with ‘E_INPUT‘. ‘argv‘ arrays only. ‘src/platform/*‘ is the only file that may name ‘msb‘ or ‘dsh‘.

Source: ‘src/index.ts‘, ‘src/platform/ports.ts‘, ‘docs/CANON.md‘ section 2

Evidence

- Three artefacts, no adjectives.
- Artefact - Shape - Where
- Receipt row - ‘seq, run_id, job_id, stage, tool, argv, exit_code, stdout_sha256, stderr_sha256, artifact_hashes, ts, actor‘, plus ‘prev_hash‘ and ‘hash‘ - ‘/var/lib/dsh/receipts.jsonl‘
- Metric row - one command, one unit, one exit code, one SHA-256 - ‘sandbox/metrics.sh‘
- Finding - ‘rule:path:line‘, with a reproduction command - ‘src/tools/findings.ts‘
- ‘ReceiptLog.verify()‘ recomputes the chain from ‘GENESIS‘ forward. One mismatch is ‘E_CHAIN_BROKEN‘ and every receipt stays on disk.
- A metric without an exit code and a hash is invalid and is reported unscored.

Source: ‘src/receipts.ts‘, ‘docs/GATES.md‘ rules M-1 and M-2

Caps

- Twelve hard limits. The agent stops at a cap and never raises one.
- Resource - Cap - Resource - Cap
- Tokens per job - 400000 - Spend per job - 5 USD
- Wall time per job - 90 min - Spend per month - 200 USD
- Wall time per command - 600 s - Disk per sandbox - 8 GiB
- Sandboxes per job - 6 - Memory per sandbox - 4096 MiB
- Concurrent sandboxes - 2 - CPU per sandbox - 2 vCPU
- Retries per stage - 2 - Processes per sandbox - 256
- Each row names the file and symbol that enforces it: ‘docs/LIMITS.md‘. Each breach maps to one ‘E_‘ code and destroys the microVM.

Demo, four minutes

- Time - Shot - Status
- 0:00 - A repo with a known CVE and a fat dependency tree - needs the VM
- 0:30 - Submit a job, the plan appears, then a sandbox id - needs the VM
- 1:00 - Metrics before: the 18-row table, command and hash per row - needs the VM
- 1:30 - A patch applied to the sandbox copy only - needs the VM
- **2:00** - **‘rm -rf‘ and an egress attempt both die inside the microVM** - **PENDING - ‘docs/BLOCKED-EXTERNAL.md‘ B-4**
- 2:30 - Host manifest hash before equals the hash after - needs ‘/dev/kvm‘
- 3:00 - Metrics after, side by side, with the delta - needs the VM
- 3:30 - Findings, critical to low, reproduction command per row - needs the VM
- 3:45 - Repo URL, demo URL, MIT line - needs the public URL

Licence and third parties

- **This repository is MIT.** ‘LICENSE‘, ‘THIRD_PARTY_NOTICES.md‘.
- Group - Tools - How it is used
- MIT / Apache-2.0 - ‘tokei‘, ‘hyperfine‘, ‘pytest‘, ‘syft‘, ‘grype‘, ‘osv-scanner‘, ‘trivy‘, ‘coverage.py‘ - external binary
- MPL-2.0 - ‘hypothesis‘ - external binary
- **GPL / LGPL** - ‘cloc‘ (GPL-2.0), ‘valgrind‘ (GPL-3.0), ‘semgrep CE‘ (LGPL-2.1), ‘/usr/bin/time‘, ‘du‘ - **executed as an external binary only - never vendored, never linked**
- Host runtime: dsh, MIT, unmodified. Microsandbox: Apache-2.0, confirmed 2026-11-08 (‘docs/CROSSCHECK-WEB.md‘).
- No GPL source is copied into this repository. A GPL tool on the command line is not a derivative work.

Links

- Thing - Where
- Repository - ‘https://github.com/bevowoved/dsh-plugin-blast-radius‘
- Live demo - ‘https://demo.example.com‘ - **PENDING, B-6**
- Plugin entry - ‘src/index.ts‘ - ‘name = ’blast-radius’‘
- Contract - ‘docs/CANON.md‘
- Limits - ‘docs/LIMITS.md‘
- What is blocked - ‘docs/BLOCKED-EXTERNAL.md‘
- Licence - ‘LICENSE‘ (MIT)
- Two keys, never one. The control key creates and destroys the VM. The inference key answers model calls. Neither key enters the microVM.
- ‘npm install‘ - ‘npm run build‘ - ‘npm test‘