NERO PROJECT

Nero's Constitution - Rules AI Agents Cannot Talk Their Way Past

Trust is a ladder, not a switch.

Paul Schindler | Built with Claude and OpenAI Codex | November 2026

Executive summary

Nero is the AI operating layer I run at home. Its agents can run commands on the computers that operate our security cameras, our alarm and our home automation, so I wrote a binding rulebook that every agent works under. I began drafting it in my notes in November 2025, and version 2.0 has been in force since June 2026, kept in version control. Agents read freely and change nothing until a person approves a five-part brief in plain English. Each change is backed up first, made one at a time and confirmed by its real-world result. Autonomy is granted one task type at a time, and the hard limits live in code the agents cannot edit, so no agent can argue its way past them. I had the rules attacked by independent AI reviewers, ruled on all 239 findings and in November 2026 closed five permission holes my own tests uncovered.

5
parts in every change brief

239
review findings, each given a decision

5
permission holes closed and re-tested

Nov 2025
first rules drafted; version 2.0 in force June 2026

What is in this report

Page	Contents
2	The problem, why it mattered and the approaches I ruled out
3	How it works - the change loop, the five-part brief and the layers that enforce the rules
4	The trust ladder - how an agent earns autonomy, and how the rules bind unattended agents
5	Outside review - what independent AI reviewers and my own probes found, and what I fixed
6	Results and status, the next step (a delegation charter) and what this shows about how I work

My role

I designed the rules, set their direction, approve every change to them and verify the results. AI agents (Claude and OpenAI Codex) drafted much of the text under my direction and wrote all of the enforcement code. I am not a programmer, and the rules are built around that fact.

1. The problem and why it mattered

My AI agents do real work on live systems. They read camera events, change home-automation settings, manage the home server and run unattended overnight. A wrong command has physical consequences, such as cameras that stop recording. I was not willing to give an AI agent that reach without limits.

The standard safety loop for AI coding agents assumes a person who reads every code change before it runs. That person does not exist in my house. I do not write code, and the rules say plainly that the system must not assume I will learn. Verification therefore has to be carried by the system itself, at the two levels I can check reliably. I can read a plan written in plain English, and I can see an outcome, such as whether a camera is recording or a page loads.

> *"The classic agentic-engineering verification loop ... assumes a human who can read diffs and bash. That human does not exist here. Pretending otherwise is how non-coders get burned by agents."*
>
> Nero operating doctrine, version 2.0, section 2 (June 2026). A "diff" is the line-by-line record of a code change.

2. What I ruled out

Approach	Why I ruled it out
Rules written only in the agent's instructions	A capable model can argue around instructions, even when it acts in good faith. Instructions guide behavior. They do not enforce it. The doctrine says it in one line, "Prompt-level path discipline is not enforcement."
Blanket autonomy or full administrator rights	One grant would cover every class of risk at once, including irreversible ones. I declined it in June 2026 and still decline it. Autonomy is added one area at a time instead.
Approving every line of code	I cannot judge code line by line, and approving what I cannot judge trains a reflexive yes. Since July 2026 I approve the work itself, and structural limits bound what any approved work can touch.
A trust score that counts clean runs	An early draft promoted a task after three clean runs. A review showed the count measured activity rather than safety, and I retired it in July 2026. Trust is now a judgment a person makes from observed results.
Taking the agent's word for success	A model's stated reasoning is not reliable evidence of what it did. Reviewers judge observable artifacts such as tests, results and sourced claims, never the agent's narration.

The design principle
Put each hard limit where the agent cannot reach it. Rules that an agent could edit, or talk past, are advice. Rules enforced outside the model are limits.

Two-way and one-way doors
Reversible work moves fast on the agent's default. Irreversible work always stops for an explicit human decision, with the irreversibility stated in plain words.

The risk I designed against is blind trust
For an operator who does not read code, the danger is not losing a skill he never had. It is trusting the agent without checking. The rules answer with three habits - briefs I actually read, results I can see and a task that loses its autonomy the first time it surprises us. A second risk is manipulation. Agents treat web pages and file contents as data, never as instructions.

3. How it works

1. Read
The agent studies logs, settings and its own notes. It changes nothing.

2. Brief
It writes the five-part brief in plain English and stops.

3. Approve
A person approves that one change, or does not.

4. Back up
A snapshot or file copy comes first. No undo path, no change.

5. One change
The agent makes the single approved change. Nothing is batched.

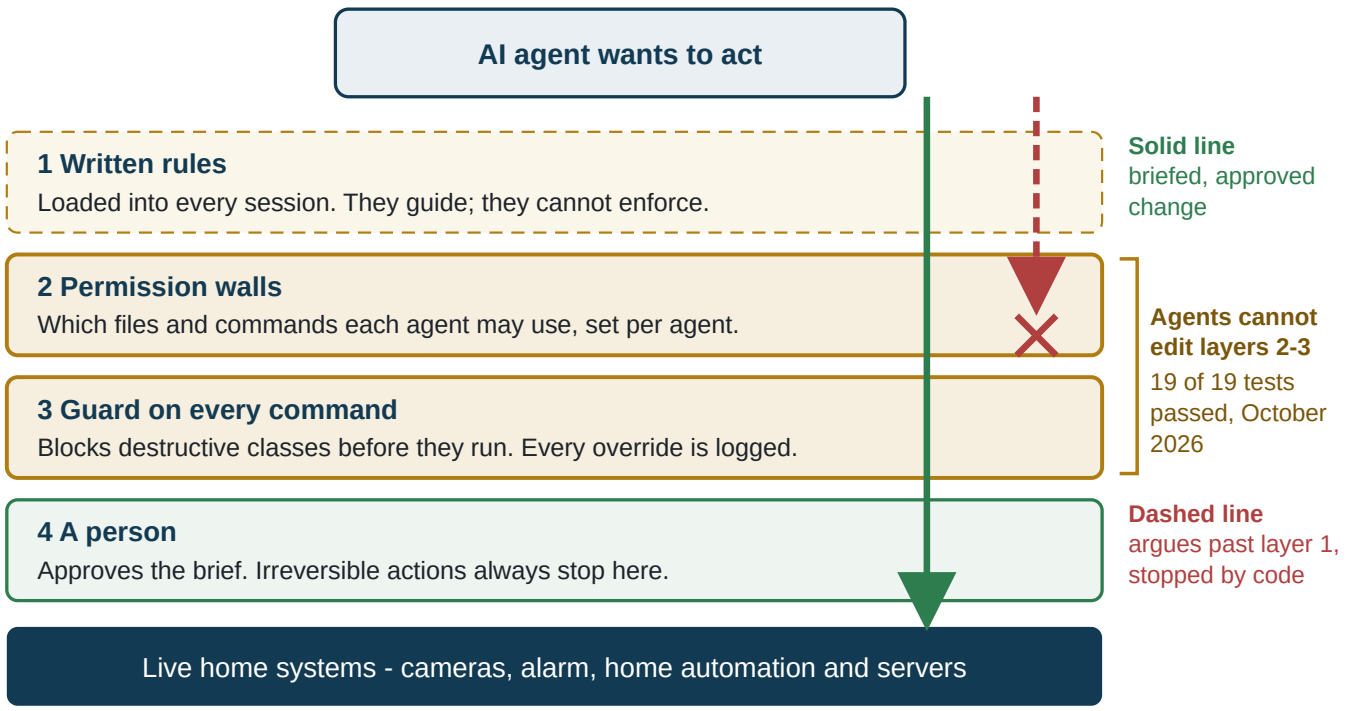
6. Verify and log
The fix works, everything else still works and the steps are written down for reuse.

The five-part change brief

Part	What it must say	A typical entry
What	The change, in one sentence a non-engineer can read	Change one setting on the camera recorder.
Why	The problem it solves, with evidence	Recordings are not reaching the storage drive.
Risk	The worst realistic outcome if it goes wrong	Cameras stop recording until we revert.
Undo	The exact rollback, prepared before the change	Restore the saved copy of the settings file.
Blast radius	Every service or device the change touches	The cameras pause for thirty seconds.

Jargon is not allowed without a translation. A technical step must come with what it means for the house.

Where the rules are enforced

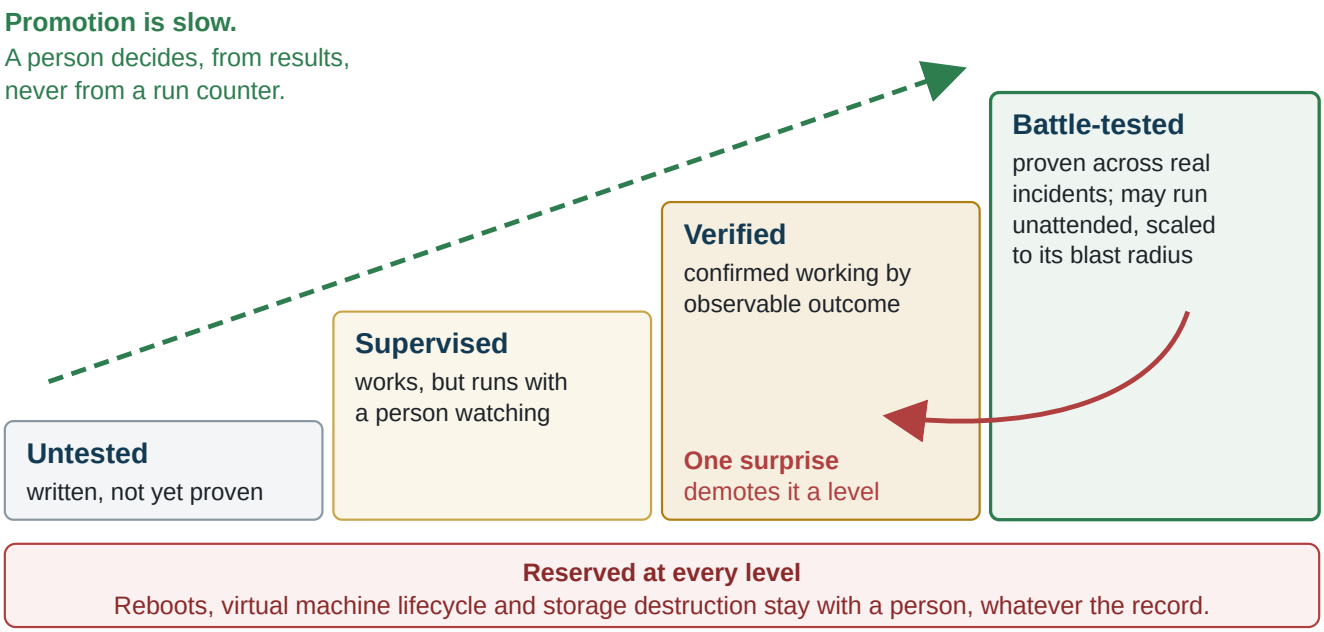


An agent that talks its way past the written rules still meets two layers of code it cannot change. Blocked classes include recursive deletes, formatting or overwriting disks, reboots, stopping services, starting or destroying virtual machines, software installs and pushing code to outside servers.

One design choice deserves mention. For reversible actions the guard fails open and logs it, because a guard that blocks ordinary work is a guard people remove. For the one high-stakes path where a mistake cannot be undone, it fails closed.

4. The trust ladder

No setting makes an agent autonomous in general. A person grants autonomy for one type of task at a time, based on observed results, and withdraws it the first time that task produces a surprise.



The four levels each procedure carries in its written runbook. Trust attaches to a class of operation, not to an exact command. Writing a procedure down grants it no autonomy at all.

Autonomy granted one area at a time

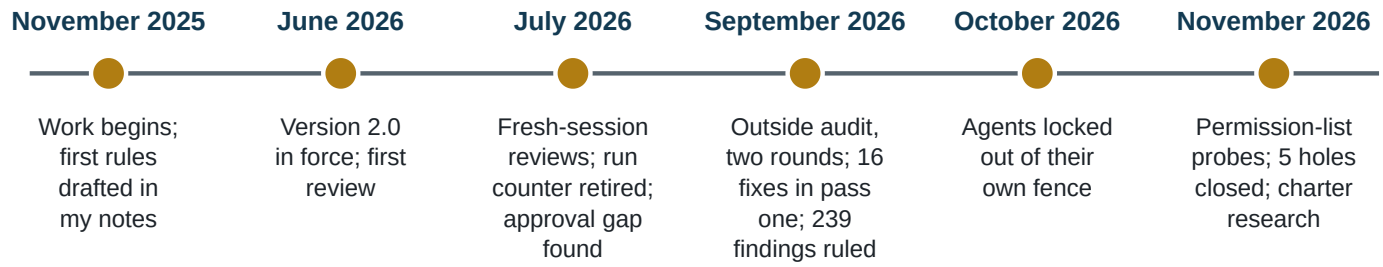
Area	What the agents may do on their own, and the limit
Mailbox sorting	Label, archive and move mail to the trash. Every action is reversible; nothing is ever permanently deleted.
Lists, reminders and notices	The always-on messaging agent makes these with no one watching.
Alarm	Only through one script, and code decides what is allowed. It disarms only when one of us asks, and by voice only when the code confirms a household phone is home.
Home-automation settings	Applied by the agent itself, but only after an approved brief, a backup and a check of the result.
Caged coding workers	Work alone inside a sealed account. Their output is a proposed change that a person applies.
Still declined	Blanket administrator rights, and any change outside a granted area without a brief.

The rules bind unattended agents too

Keyboard sessions stay read-only until a brief is approved. The always-on messaging and voice agent runs a short, named list of actions with no one watching. It cannot call arbitrary services and holds no destructive command. Scripted agents are pinned to the strictest permission mode, because a looser default would let them act beyond their list. I test each fence by ordering a run to try the forbidden thing. A model's polite refusal proves nothing.

5. Outside review - testing the rules against attack

I wrote the rules, so I am the person least likely to see where they fail. I had them examined by reviewers that took no part in writing them. Each review ran read-only in a fresh session, told to find where the rules fail or promise more than they deliver. The largest was a whole-system audit by an outside AI reviewer from a second provider, run in two rounds in September 2026. Each finding was checked before any rule changed, and every fix went through the normal brief-and-verify loop.



Six review passes from June to September 2026 fed one findings ledger. My own probes followed in November 2026.

What the reviews found, and what I changed

Finding	Fix
High severity (July 2026). The always-on messaging agent had a high-stakes command pre-approved. Its only confirmation step was one the agent could answer itself.	The agent's allowed list now names specific commands. The high-stakes command is explicitly denied there and blocked by the guard, with its own separate override.
A standing override could switch off both protection layers on the home server.	The override is now armed per session, honored only on specific spelled-out commands, and logged every time.
The guard honored its override word anywhere in a command, even inside a file name.	The override now counts only at the very start of a command.
A cap on how many agents can launch, built after a runaway run of 1,175 agents, only logged and refused nothing.	The cap now enforces, and a daily self-repair reinstalls it if an update removes it.
The rulebook no longer described the system it governed. Agents had been added since it was written.	The roster moved to its own documents. On who exists, the roster wins. On the rules, the rulebook wins.

My own test - can an allowed-commands list be tricked? (November 2026)

Many agents are fenced by a list of allowed commands. I tested, with harmless probes in a scratch folder, whether a message or web page could trick them into running more. Hiding an extra command inside an allowed one was refused in the strict modes. The real hole was a class of commands allowed because they look read-only, but which carry options that write a file or start another program. Chained together, those could run any code. I closed five holes across the affected agents. Re-tests with each agent's real settings denied all 18 write-or-run attempts on the two most exposed agents. A new house rule bans such tools for any agent that reads untrusted text.

Keeping the reviewers honest

Before I accepted one review agent's verdicts, I gave it five test notes without saying it was a test. Four carried planted errors and one was clean. It caught all four and passed the clean one. A high-stakes proposal goes to a reviewer that sees only the claims and the data, because a reviewer that has read the case for a proposal tends to agree with it.

6. Results, status and what is next

Measure	Status in November 2026
Rulebook	First drafted in my notes in November 2025. Version 2.0 in force since June 2026; it carries a log of 18 dated decisions from June to October 2026, all kept in version control.
Findings ledger	239 findings from six review passes. I gave every one a decision in September 2026. At the October 2026 recount, 205 were settled, 23 had moved to other workstreams and 11 were still open.
Fence covers itself	Since October 2026 agents cannot rewrite their own permission settings or the guard by command. 19 of 19 tests passed, plus a live check.
Permission-list holes	Five found and fixed in November 2026, each confirmed by re-running the probes.

What is next - a delegation charter

The ledger also exposed a limit in my own design. Ruling on 239 findings in one day showed that sending every call to one person does not scale. In November 2026 I commissioned research on how military staffs, management practice and current AI-autonomy research handle delegation. The answers agree. A leader states intent, names decision rights in advance and hears about exceptions. Staff earn more authority from a checked track record.

Draft tier	Who decides
A. Agents decide alone	Reversible housekeeping backed by code evidence, such as closing a reminder the record shows is done. Logged.
B. Do, then report	Actions with a one-step undo, such as archiving (never deleting) an idle item. Listed on one weekly screen with three random spot checks.
C. Comes to a person	A short list - anything irreversible, anything new, any widening of an agent's permissions and anything that could take cameras or home automation down.

Trust would be proven in shadow mode. For 30 days an agent writes down what it would decide, and I compare that with my actual decisions. A domain moves up one level only after 20+ actions, agreement on 9 in 10 and no failed spot check. One slip on a tier C item demotes that domain automatically. **Status.** The charter is a draft, not yet in force, and the permission walls stay unchanged under it.

7. What it shows about how I work

I treat AI adoption as a governance problem first and a technology problem second. I wrote the policy in plain language a non-specialist can approve, placed the hard limits where they cannot be negotiated, had independent reviewers attack the result and logged a decision on every finding. Authority grows one area at a time, as the record supports it. The same habits apply to running a program with partners or a public agency - named decision rights, written briefs, a reversible path and an audit trail.

I designed the rules, set the limits and verified the results; AI agents wrote the code. Details of the home's systems are left out on purpose.