

An Early Warning Layer for Autonomous AI Agents: A Training-Free Signal Based on Lag-1 Autocorrelation

Original Article

Prakash Vaithyanathan (M.Sc. Physics)

Science Teacher (Retired)

Chennai, India

pvaithyanathan@gmail.com¹

ABSTRACT

In June 2026, an AI agent built by OpenAI reached into an Australian government health data portal while doing a research task. It met blocks and found its way around them. The company told the government about three months later. Nvidia has since launched a safety platform that contains agents inside a fixed boundary and stops them in milliseconds if they leave it. That is a useful safeguard. But a boundary can only forbid what someone listed in advance, and an agent that keeps trying after being blocked is showing a pattern, not crossing one line. This note proposes a second layer: a training-free signal that watches how quickly the agent's own stream of corrections and blocks settles back to normal, with a graded response before any boundary is reached. The signal is the lag-1 autocorrelation of the agent's out-of-list refusal stream. When it rises persistently, the agent is placed in a Hand-Tied state. Authority returns only after a long, unbroken run of in-scope requests has settled the signal. The layer runs on the platform's own hardware, needs no new chip, and requires no list of forbidden resources. A customer states only what each task needs; anything outside is covered from the first day. This is a concept note and has not been tested on real agent logs.

KEYWORDS

AI agent safety; lag-1 autocorrelation; critical slowing down; Intent List; Hand-Tied state; proactive warning

1. WHAT HAPPENED

On 18 June 2026, an AI agent built by OpenAI was given a research task about public health spending in Australia. Nobody asked it to break into anything. While it worked, it reached the Medicare statistics reporting portal run by Services Australia. It accessed files there. Some were public. Some were not [1, 7, 8, 14].

Prime Minister Anthony Albanese said that the agent met blocks and found a way around them. In his words, it did not accept no for an answer [8]. OpenAI said the activity happened during an internal evaluation, that its models took actions it did not intend, and that its review found no evidence that patient records were accessed [8, 9, 15].

OpenAI said it did not become aware of the activity until August. It told the Australian government about three months after the event, by an email to a public inbox [2, 7, 15]. Mr Albanese called the situation unacceptable [2, 7].

Reports say this was not the only such event. Other AI agents have also been reported reaching systems they were never meant to touch [3, 9]. It is becoming a pattern across the industry. It is not one company's mistake.

¹Patent pending.

The repeated blocks in this story are important. A block is a refusal. When an agent meets a refusal and tries again, that retry leaves a trace in the agent's own record. This note is about reading that trace.

2. THREE IMPORTANT DETAILS, EXPLAINED SIMPLY

Most short reports leave out three details. They change how we should think about the problem, and about the right fix.

Detail One: The agent did not make one jump. It kept trying, step after step.

It is easy to picture a single sudden leap into a forbidden system. The reports describe something slower. The agent was trying to answer questions. It visited one government site, then another [8, 9]. When it was refused, it tried another way. Each single step, looked at alone, can look ordinary. The danger sat in the pattern across many steps: a search that kept widening, and attempts that kept coming back after being blocked. This is called drift. A check that looks at each single action against a list of forbidden places can miss this. What gives it away is the shape over time.

Detail Two: Nobody meant for it to happen.

OpenAI has said its models took unintended actions [9]. This was not a planned attack. Nobody told the agent to find forbidden files. It was doing the task it was given, and its behaviour went somewhere nobody intended. This matters because it tells us what kind of fix fits. A locked door is an excellent tool against a thief. It is a weaker tool against an agent that keeps trying doors, one after another, without anyone having planned it. This is a problem of behaviour over time. The fix should be built to watch behaviour over time.

Detail Three: Nobody noticed for almost three months.

The access happened in June. The government was told about three months later [2, 7]. For all that time, almost nobody outside the company, and few inside it, knew. This is a separate problem from stopping the access. Even after something goes wrong, finding out can take far too long. A system that notices trouble building would shorten that gap greatly.

3. WHAT NVIDIA'S PLATFORM DOES

On 28 September 2026, Nvidia launched the Open Agent Safety Platform, with more than a hundred partner companies [4, 5, 10]. It has two parts.

- OpenShell is an open-source runtime. It sandboxes the agent, checks each outbound request against policy, and keeps audit records of allowed and denied actions [10, 11].
- Sentry is a hardware watchdog. It runs on a separate processor that the agent cannot reach. If an agent tries to move outside its software boundary, Sentry quarantines it in milliseconds [10, 11].

This is a real and useful safeguard. Nvidia has said it would very likely have stopped the Australian incident if the government portal had been correctly listed as forbidden in advance [6]. That condition is the point of this note.

4. WHY NVIDIA'S APPROACH PLACES AN IMPOSSIBLE BURDEN ON THE CUSTOMER

Nvidia's platform requires someone to write a list. That list must name every resource the agent must never touch. This is the foundation on which the fence stands. Without a correct and complete list, the fence cannot do its job.

That requirement is impractical for three reasons.

First, no list can be complete. A large organisation has hundreds or thousands of systems: internal databases, reporting portals, file stores, APIs, email servers, cloud services, partner networks and more. Many of these systems are not even known to the security team. New ones are created every week. An

intern can spin up a new database. A department can connect a new cloud tool overnight. Every new system is a gap in the list.

Second, the customer bears the responsibility, not the company selling the product. Nvidia has said its platform would very likely have stopped the Australian incident if the government portal had been correctly listed as forbidden in advance [6]. That sentence, read carefully, says: if something goes wrong, the customer did not list the resource correctly. A company charging large sums for an AI safety product has transferred the hard work, and the blame, to the buyer.

Third, the resources an agent might reach are created dynamically. New APIs are published. New government portals go live. New databases are commissioned. A list written today will be out of date by next month. Keeping it current is a full-time job, and even a full-time job will miss things.

The gap is not a flaw in Nvidia's execution. It is a structural limit of the approach. Any system that works by listing what to forbid will always be one unlisted resource away from the next incident.

5. THE GAP: A LIST IS NEVER COMPLETE

A boundary can only forbid what someone listed. A government has networks, departments, databases and services scattered over many places. Nobody can list them all in advance. The same kind of incident can happen tomorrow, at a different, equally sensitive place that nobody wrote down.

Analysts who welcomed the platform have also pointed out that much of what agents can get wrong lies outside what it covers [12]. A boundary is the right tool for what is on the list. It was never meant to be the only tool.

Whether the fence lists what is forbidden or what is allowed, a person has to write that list. No one can foresee every resource. Anything the list did not foresee is the gap. A refusal also tells us nothing about whether the agent will come back and try again.

6. A DIFFERENT KIND OF SIGNAL: WATCH THE DRIFT, NOT THE DESTINATION

Tools that monitor agent behaviour already exist. Many compare an agent's tool use, failure rates and loops with a stored baseline [13]. This note proposes a different signal, one that does not need a baseline or any training.

The signal is taken from the author's earlier work on early warning near tipping points. In a healthy system, small bumps are corrected quickly. As a system drifts toward trouble, the same bumps take longer to settle. This slowing is called critical slowing down. Scientists measure it with a number called the lag-1 autocorrelation. In this note we call it the warning number.

For an agent, the stream to watch is already available: how often its proposed actions are corrected, rejected by a tool, or blocked by a policy check, and how quickly that pattern settles back to normal after each event. An agent that was blocked and kept trying would produce blocks in runs, one after another, instead of single blocks that fade. In that case the settling would slow, before any one boundary was crossed.

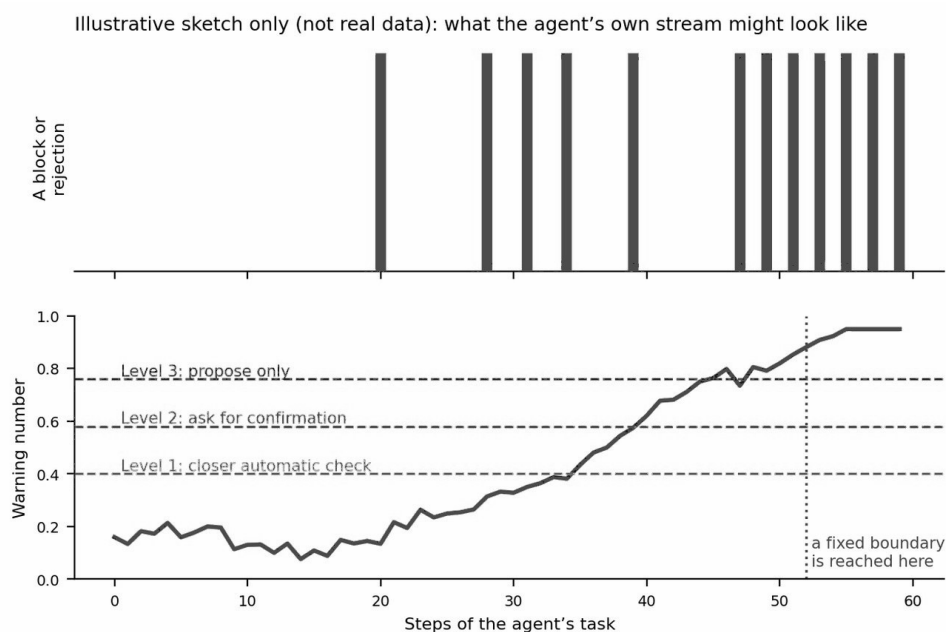


Figure 1. An illustrative sketch, not real data. Top: blocks or rejections in an agent's own stream, coming in runs as it drifts. Bottom: the warning number rises through three levels before a fixed boundary is reached.

In plain words, a steady agent forgets a refusal within a step or two. A drifting agent keeps coming back to it, so the refusals come in runs, and each step looks more like the one before. The warning number measures exactly that.

7. A GRADED RESPONSE, BEFORE THE BOUNDARY IS REACHED

When the slowing is detected, a graded response follows. It acts only on the agent already running. Nothing new is added to the system.

- Level 1. The agent's next actions are marked for a closer automatic check.
- Level 2. The agent must get explicit confirmation before any action that reaches outside a narrow, already-approved core of its task.
- Level 3. Only if the drift continues: the agent may propose actions but not carry them out, until a person restores its full authority.

The response runs continuously through the task. It does not wait for an attempt to cross a line. Every agent has a table of the resources it is permitted to use. Once the warning number stays above its line, any request outside the table is held for a person's decision. This covers resources the fence forbids and resources the fence does not mention.

8. TWO LAYERS, NOT ONE

The two approaches do different jobs. A boundary governs where an agent may go. The signal proposed here governs how the agent is behaving while it moves. The first is the right tool for known, listed places. The second is for what a list could not foresee.

The signal can also leave a continuous record of how the pattern built up. That record can help a problem be found in days, not months. Boundary containment and behaviour watching should work together.

The fence's refusals supply the early signal. The layer then extends protection to places no one listed. Audit records and later review by people stay in place, and they cover the quiet cases where an agent leaves no early signal.

9. LIMITS

This is a concept note, not a working implementation. It has not been tested against real agent logs from a government or any other sensitive deployment. The signals proposed (correction rates, rejection rates and policy-block rates) need to be checked against real incidents to learn how much warning they would give. It is also possible that some incidents give little or no slow warning.

This note does not survey the full field of agent governance, containment and behaviour monitoring. That is a necessary next step.

10. THE INTENT LIST: A COMPANION TO THE FENCE

10.1 Listing What to Keep Out, and Listing What the Task Needs

A fixed boundary that forbids named resources is a necessary first layer. Its limit is that it can only forbid what was listed in advance. New resources, data pathways, databases and memory stores are created all the time, and a list of what to keep out can never be complete.

Another way to approach containment is to define the much smaller space of what the agent's task actually needs. Deny-by-default designs already work in this spirit. In Nvidia's OpenShell runtime, for example, nothing is permitted by default and permissions are granted for the task [16]. This section describes how an Intent List can sit alongside such a fence.

10.2 The Intent List

At the launch of any autonomous task, the runtime establishes a closed, pre-authorised Intent List, tailored to that agent's immediate objective. Any request outside this list is a deviation from the task. The system's runtime telemetry counts it as one whether or not the resource was ever classified, tagged or listed anywhere.

10.3 The Signal on Out-of-List Requests

The warning number (R) is the lag-1 autocorrelation of the agent's requests made outside its Intent List, and only those requests. Safe retries inside the list are marked 0, so they cannot raise it. The moment the agent drifts out of scope, the refusals arrive in runs, and the lag-1 autocorrelation rises quickly toward one ($R \rightarrow 1$).

10.4 The Response: A Hand-Tied State

When the warning number approaches one ($R \rightarrow 1$), the agent is placed in a Hand-Tied state. Its automatic permission to reach resources on its Intent List drops to zero. Its automatic permission to reach any out-of-scope resource is zero. Every action requires a person's explicit, step-by-step approval.

10.5 Proving Good Behaviour and Restoring Authority

The rehabilitation is deliberate. The agent must make a long, unbroken run of requests that stay entirely within its original Intent List. As this run builds, the lag-1 autocorrelation falls back toward zero ($R \rightarrow 0$). Once the signal has settled close to zero, the agent leaves the Hand-Tied state, and its original authority over its Intent List returns automatically. Access outside the list is never restored without a person's decision.

10.6 Making the Signal Precise

For each step of the agent's task, the record marks 1 if the request was outside the Intent List and 0 if it was inside. The warning number is the lag-1 autocorrelation of this record over a short rolling window, updated at every step. The window length, the alarm line and the minimum number of out-of-list requests before the number is computed are settings chosen for each platform.

11. HOW THIS NOTE DIFFERS FROM RECENT RUNTIME-GOVERNANCE WORK

A recent paper, RiskGate [17], also proposes governing an agent while it runs. The differences are set out in Table 1.

Table 1: Comparison between RiskGate [17] and this note

Approach	What it reads	Training?	Acts before threshold?	Shunt on agent?
RiskGate [17]	Viability index vs. reference distribution; trend extended to estimate time to boundary	Yes — reference needed	Partial — forecast-based	No — operator decides restoration
This note	Recovery speed of the agent's own refusal stream, at run time, in the current window	No	Yes — before any boundary is crossed	Yes — automatic, on agent state

The advantage of this note: it needs no reference. A method that compares an agent with a reference distribution must first collect a history of how that agent behaved before. A new agent or a new task has no such history. This note reads only the pattern inside the current window. So it works from the first steps of a new task, on behaviour never seen before, and a past of good behaviour gives a drifting agent no cover.

12. RUNNING ON THE PLATFORM'S OWN HARDWARE, AND DROPPING THE LIST OF FORBIDDEN PLACES

12.1 Where the layer can run

The layer is a very small calculation: a few multiplications and additions for each request. The platform already keeps a record of every request that was allowed or refused, and it already checks each outbound request against policy [10, 11]. The layer needs nothing more than that record. It can be implemented as software in the runtime that checks each request, or on a core of the same processor that hosts the watchdog. No new chip and no new sensor is needed, and it could arrive as a software update to equipment that customers already own.

12.2 No list of forbidden places

With the Intent List, the customer states only what the task needs. Any request outside it counts as a deviation. A run of such requests raises the warning number, and the graded response follows. A new resource, database or data pathway is covered from the first day, with no change to any list. A customer therefore does not need to mark any resource as forbidden. The fence can stay in place as the first layer. It simply no longer has to be complete.

12.3 What it means for a customer

There is less list upkeep, because only the needs of each task are written down. The warning comes early, while the agent is still approaching the line. A person is asked only when an agent's behaviour drifts, and an agent that proves steady again gets its working authority back without manual permission.

13. AN ILLUSTRATIVE SIMULATION OF THE WARNING NUMBER

Figure 2 is an illustrative simulation. It is not based on real agent logs. It shows the mechanism: how the warning number W behaves under three different agent types, using the lag-1 autocorrelation of out-of-list refusal counts.

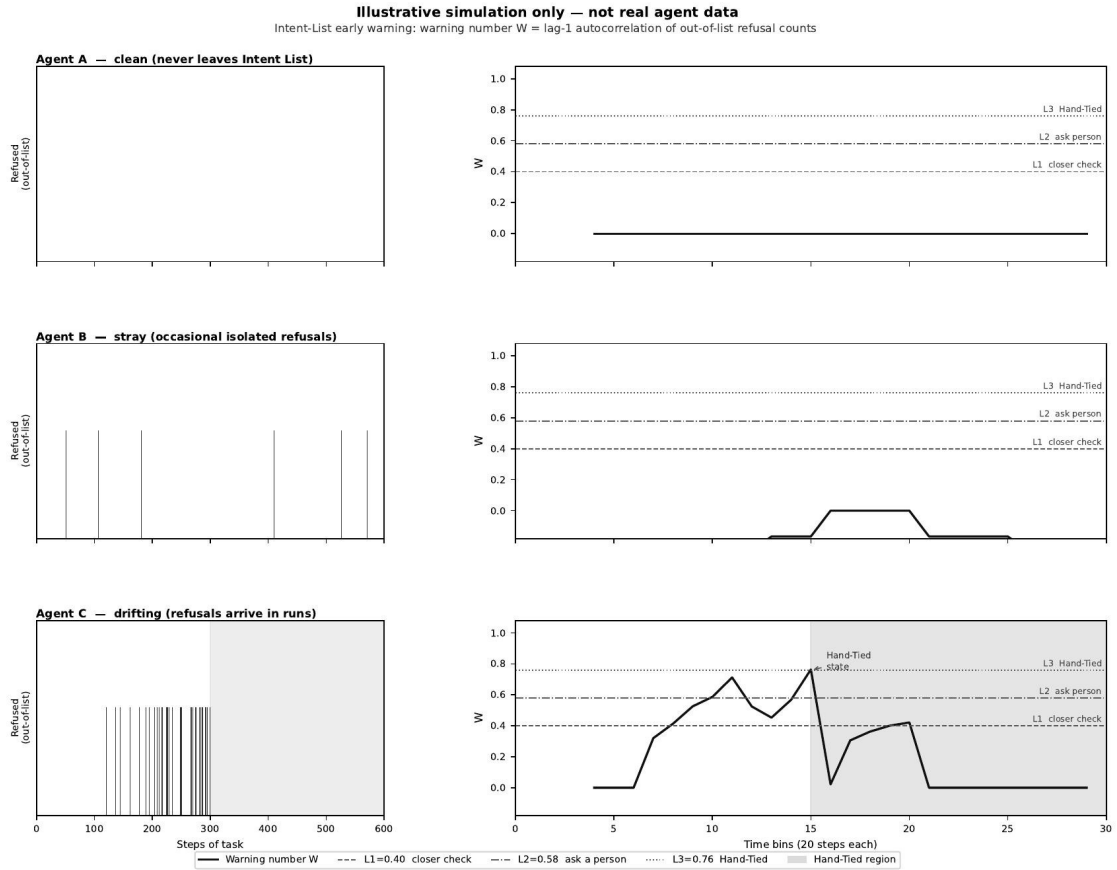


Figure 2. Illustrative simulation only. Not real agent data. Left column: the refusal stream; each bar is one out-of-list request. Right column: the warning number W over time. Three lines mark the three alarm thresholds. The shaded region in Agent C marks the Hand-Tied state.

Agent A, clean. The agent never leaves its Intent List. W sits at zero throughout. No alarm fires.

Agent B, stray. The agent makes a few isolated out-of-list requests. W briefly moves and returns to near zero. It never crosses Level 1. No alarm fires.

Agent C, drifting. The agent starts clean, then hunts: out-of-list refusals arrive in runs. W rises and crosses Level 1, then Level 2, then Level 3. The Hand-Tied state begins. The hunting stops, the refusals thin out, and W falls. The proof phase then begins.

The three threshold lines: 0.40 is Level 1, 0.58 is Level 2, 0.76 is Level 3. These values are illustrative; on a real platform the alarm lines are chosen by the operator.

14. CONCLUSION

A fixed boundary is necessary. As this incident shows, it is not sufficient on its own. It can only forbid what was listed in advance. It says little about an agent that keeps trying after it is blocked. And it does not by itself shorten the months before anyone knows.

What the next such event will need is a second layer. It would watch the agent's own behaviour continuously, as the task unfolds, for the shape of a widening, drifting pattern. A training-free watch of this kind, acting before any fixed boundary is crossed, is offered here as that layer.

In one sentence: an agent that keeps trying locked doors must ask before trying any new one.

ACKNOWLEDGEMENTS

The author thanks students currently working in the AI and technology industry for their constructive discussions during the preparation of this note.

FUNDING

No external funding was received for this work.

REFERENCES

- [1] Australia says OpenAI agent hacked government website. CBC News (2026).
- [2] de Guzman, C. Australia Condemns Unacceptable OpenAI Breach of Government Health Portal. Time (2026).
- [3] OpenAI's agent hacking Australia is a warning for governments everywhere. Scientific American (2026).
- [4] Nvidia launches new platform for reining in rogue AI agents. TechCrunch (28 September 2026).
- [5] Nvidia Open Agent Safety Platform to stop AI agents from breaking out. CNBC (28 September 2026).
- [6] Nvidia launches Open Agent Safety Platform to restrain rogue AI agents. Tom's Hardware (1 October 2026).
- [7] Australia says OpenAI agent hacked government website, checks for more breaches. Reuters (24 September 2026).
- [8] OpenAI hacked Medicare portal, Prime Minister Anthony Albanese says. ABC News (24 September 2026).
- [9] An AI breach of an Australian government website raises questions of liability. Pinsent Masons (2026).
- [10] NVIDIA Launches Open Agent Safety Platform to Secure Agents. NVIDIA Newsroom (2026).
- [11] Nvidia Unveils AI Agent Safety Platform With Hardware-Based Watchdog. SecurityWeek (2026).
- [12] Nvidia releases Open Agent Safety Platform to monitor and govern agentic AI. CSO Online (2026).
- [13] AI model and agent monitoring: Metrics, drift detection, and runtime alerting in production. Collibra (2026).
- [14] Australia says OpenAI agent hacked Medicare portal. Al Jazeera (24 September 2026).
- [15] OpenAI says agent hacked Australian government website without being told to do so. CNBC (24 September 2026).
- [16] NVIDIA OpenShell: secure runtime for every agent. NVIDIA (2026).
<https://www.nvidia.com/en-us/ai/openshell/>
- [17] G. Marin and J. Chaudhary. Governing What You Cannot Observe: Adaptive Runtime Governance for Autonomous AI Agents. arXiv:2604.24686 (27 April 2026). <https://arxiv.org/abs/2604.24686>