WHITE PAPER

Trust Has a Blast Radius

Why Boundaries Have to Exist Before an Agent Runs, Not After

Author: Srinivasarengan Varadharajan
Published: July 23, 2026
Contact: srini@kter.ai

OPENING

Blast Radius Is a Design Question

Blast radius is not something you assess after a breach. It is something you are supposed to design before an agent ever runs.

Last week's paper asked who is accountable when one agent's output is wrong. This week, that question got sharper. A major AI infrastructure provider's systems were breached end to end by an autonomous agent, no human in the loop, planned, escalated, exfiltrated. Detected and fought off the same way, with AI.

We had been sitting with the boundary question before this happened. The breach did not start the argument. It confirmed it faster than expected.

0	1	?
Human Steps	**Detection Method**	**Boundary Defined**
In the breach, planning to exfiltration	*AI used to catch the AI-driven breach*	*Before deployment, for most agents in production*

SECTION 1

The Harder Question Underneath

The previous paper on the accountability gap asked a narrower question: who is responsible when an agent's output is wrong. That question assumes the agent was operating inside some defined scope, and something inside that scope went wrong.

This week's breach exposes a harder question sitting underneath that one. If an agent can act autonomously enough to plan, escalate its own privileges, and exfiltrate data without a human noticing, the real failure was not that it made a mistake inside its scope. The real failure is that no one ever defined what its scope was supposed to be.

Accountability assumes a boundary exists to hold someone to. This is about what happens when that boundary was never drawn in the first place.

SECTION 2

Blast Radius as a Design Question, Not an Incident Response

Most enterprises think about agent access the way they think about a fire: something to respond to once it starts. What could this agent reach. What did it touch. How far did it get. All of these are questions asked after the fact, during a postmortem, once the damage is already done.

The question that actually matters is asked before deployment, not after an incident. What should this agent ever be able to reach, decided deliberately, not discovered by accident during a breach.

This is a small shift in wording and a large shift in practice. An access boundary decided in advance is a design constraint. An access boundary discovered during an incident is a lesson learned at the cost of whatever the agent could reach in the meantime.

SECTION 3

Two Boundaries, Not One

The previous paper covered one boundary: the contractual line between a vendor and a client. Who is responsible, what was agreed in writing, what happens when something goes wrong. That boundary is legal and organizational.

There is a second boundary, and it is technical, not contractual. It is the line between what an agent is permitted to do and what it can actually do in practice. A permission scope that exists on paper but is not enforced in the system is not a boundary. It is a suggestion.

Both boundaries have to hold for accountability to mean anything. A perfect contract does not help if the agent was never actually constrained to the scope the contract describes. A perfectly enforced technical boundary does not help if no one agreed in writing on what that boundary should have been.

SECTION 4

The Double-Edged Data Point

The infrastructure provider that was breached this week also used AI to detect and respond to the breach. That is a real, working capability, and it is worth naming honestly rather than treating it as either a triumph or a footnote.

Using AI to catch an AI-driven incident is not the same thing as preventing one. Detection after the fact is valuable. It is not a substitute for a boundary that would have stopped the escalation before it started. Enterprises leaning on AI-driven monitoring as their main line of defense should be clear with themselves about which of these two things they actually have.

SECTION 5

Assumed Boundary vs. Enforced Boundary

Question	Assumed Boundary	Enforced Boundary
What can the agent reach?	Whatever it was given access to, broadly scoped for convenience.	Explicitly defined before deployment, narrowly scoped to the task.
Who decided the scope?	Implied by default settings, rarely reviewed.	A deliberate decision, documented and owned.
When is it discovered?	During an incident, after something already happened.	Before deployment, as a condition of deployment.
What stops an escalation?	Detection, after the fact.	A boundary that was never crossable in the first place.
Who is accountable if it fails?	Unclear. The scope was never really owned by anyone.	The person who defined and signed off on the boundary.

CLOSING

The Boundary Is the Security Control

Named scope, defined boundaries, and a human escalation path were framed in the last paper as contractual accountability. They are also a security boundary. A properly scoped Statement of Work is not just a legal document. It is a security control, whether or not it was written with that in mind.

The enterprises that treat agent boundaries as a design decision, made before deployment, will be the ones without a breach to explain. The ones that treat it as an incident-response problem will find out what their actual boundary was the same way this week's example did.

One more thing worth naming honestly: the same question does not stop at decisions and outputs. It applies to code an agent writes, too. We are building KOCODE against that same principle.

About the Author

Srinivasarengan Varadharajan is the founder of KTER. He has 19 years of enterprise integration experience across financial services, retail, and technology organisations. Contact: srini@kter.ai

© 2026 KTER. All rights reserved. kter.ai