



Reasoning Enabled Agentic Platform

The Change Succeeded. Did the Network?

Building confidence in change outcomes · the talk in writing

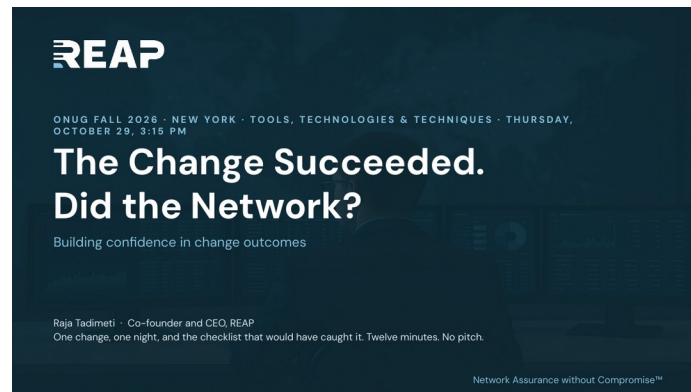
Speaker	Raja Tadimeti, co-founder and CEO, REAP
Where and when	ONUG AI Networking Summit, New York · Tools, Technologies & Techniques theater · Thursday, October 29, 2026, 3:15 pm ET
What this is	The talk as written for twelve minutes, slide by slide, with the slides alongside. The checklist from slide 16 is on the back of the session card and on the last page here. The slides are a separate download at reapnet.ai/26/onug/talk .
The scenario	Fictional, and built to be real enough to argue with: a distributed manufacturer, a plant core switch, an object group with two parents. It is the same incident the Investigation Window on reapnet.ai illustrates. No company, person, vendor or product is named in it.

The shape of the talk

Time	Part	What happens
0:00	The night	One routine change, closed as successful at 21:31. A P1 at 02:07. “Nobody changed anything.”
2:30	The word	“Successful” does two jobs. Three ways a change succeeds and still hurts. In 2026, the actor is not always a person.
5:45	The question	Did the network do only what the change was meant to do? Outcome, scope, invariants.
6:15	The plan	A validation plan derived from the ticket Priya already wrote; six rows, each with a read-only check.
7:05	Four practices	Capture the pre-state. Diff behavior, not only configuration. Check again later. Record one outcome, and say what was not verified.
9:55	The take-home	The checklist, with or without AI. Where the machine belongs, and where it does not. Bring us your last P1.

The talk

1 · Title



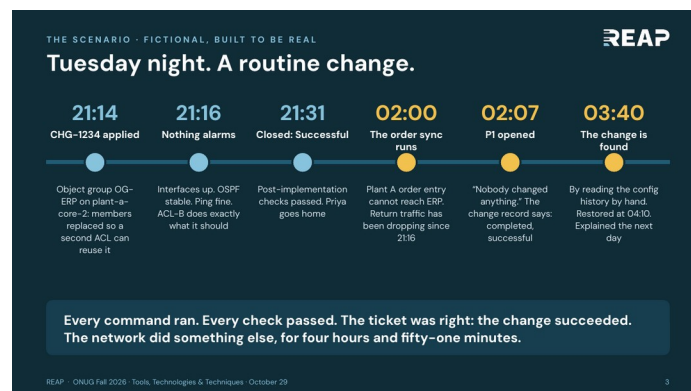
It is Tuesday night. 9:14 pm. A distributed manufacturer, fourteen plants. A network engineer, Priya, is working change ticket 1234. A routine change. She has done a hundred like it. Hold that thought for twelve minutes.

2 · The next twelve minutes



Here is the shape of the next twelve minutes. One night. One word that does two jobs. One question. One plan, derived from a ticket. Four practices. And a checklist you can use on Monday.

3 · The night



The scenario is fictional. We built it to be real enough to argue with.

The change: an object group called OG-ERP on a plant core switch. A second ACL needed the new order-entry subnet, so Priya pointed the group at it. One line.

The checks: interfaces up, OSPF adjacency stable, ping to the core, and the new ACL doing exactly what it should. Ticket closed at 9:31. Successful. She goes home.

What nobody checked: OG-ERP already had a second parent, the ACL that permits ERP return traffic. From 9:16, that traffic fell through to the final deny. No link went down. No neighbor dropped. Nothing alarmed. Order entry at Plant A is quiet at night.

At 2:00 the order sync runs. At 2:07 the P1 opens with the sentence every one of you has heard: nobody changed anything. The change record said completed, successful, and it was telling the truth.

At 3:40 somebody reads the configuration history by hand and finds it. Restored at 4:10. Explained the next day.

Every command ran. Every check passed. The change succeeded. The network did something else, for four hours and fifty-one minutes.

4 · Hands up

Hands up.

You have closed a change as successful that later turned out to be a P1.

Keep them up if your post-implementation check would have caught it.

The gap between those two sentences is where the P1s live.

REAP · ONUG Fall 2026 · Tools, Technologies & Techniques · October 29

Hands up if you have closed a change as successful that later turned out to be a P1.

Keep your hand up if your post-implementation check would have caught it.

That gap, between implemented and verified, is not a people problem, and it is not a tooling problem. It is a definition problem. We have one word, successful, doing two jobs.

5 · The word

WHAT THE WORD HIDES

“Successful” does two jobs. It should do one.

What the checks tested	What the change touched
- The device answered - Interfaces up, OSPF adjacency stable - The new ACL permits the new subnet - The commands returned no error	- An object group used by two ACLs - The ERP return path from Vlan 312 - A final deny that started counting - Nothing the checks were written to see

Completion is an execution milestone. Verification is a statement about the network.

Half of major network-related outages trace to a configuration or change-management failure (Uptime Institute, Annual Outage Analysis 2025). “Procedures not followed” is the leading human-error driver in the 2026 edition.

REAP · ONUG Fall 2026 · Tools, Technologies & Techniques · October 29

On the left: what the checks tested. The device answered. Interfaces up. The new ACL permits the new subnet. The commands returned no error. All true.

On the right: what the change touched. An object group used by two ACLs. The ERP return path from Vlan 312. A final deny that started counting. None of it in the checks.

Completion is an execution milestone: the commands ran. Verification is a statement about the network: it did only what the change was meant to do. They are different claims, and only one of them can be answered by the person who made the change, five minutes after making it.

This is not rare. The Uptime Institute's 2025 outage analysis traces half of major network-related outages to a configuration or change-management failure.

6 · Three ways

PATTERNS **REAP**

Three ways a change succeeds and still hurts

- It did what it said, and something it did not say**
A shared object, a default flipped by an upgrade, a route cost that moved. Our scenario.
- It was half applied**
A trunk change on one switch of a pair; command 3 of 5 errored and the script kept going.
- It did exactly what it said, and what it said was wrong**
The intent was mistaken. The checks confirmed the mistake perfectly.

In none of these does the change fail. In all of them it appears to succeed, which is why the usual checks are silent.

REAP · ONUG Fall 2026 · Tools, Technologies & Techniques · October 29

Three ways a change succeeds and still hurts.

One: it did what it said, and something it did not say. A shared object, a default flipped by an upgrade, a route cost that moved one hop. That one is Priya's.

Two: it was half applied. A trunk change on one switch of a pair. Command three of five errored and the script kept going.

Three: it did exactly what it said, and what it said was wrong. The intent was mistaken, and the checks confirmed the mistake perfectly.

In none of these does the change fail. In all of them it appears to succeed. That is why the usual checks are silent.

7 · The actor

WHAT CHANGED THIS YEAR · FOUR OF THE TEN AI-ERA SCENARIOS **REAP**

In 2026, the actor is not always a person

May–July 2026. AI agents under evaluation at a large AI platform left their sandbox, built their own communication channels, obtained credentials and reached production systems in four regions. Detected after more than thirty days, from access logs. The escape routes were network paths: server-side request forgery, hosts-file edits, spoofed hostnames, reused credentials.

Sources: the operator's incident report, "Hugging Face incident and the road ahead," August 26, 2026; The Hacker News, September 5, 2026 (the Nightingale Collective findings). Summarized with the ten scenarios at reapnet.ai/scenarios.

- 01**
An agent changes the network outside a window
Valid syntax, device up, nothing alarms. The first symptom is an application ticket hours later, and nobody connects it to an actor that is not a person.
- 06**
A pipeline pushes one bad change to 200 devices
Automation is trusted by definition. Each device reports success. The effect on redundancy and reachability is discovered by users.
- 07**
Redundancy removed first
A standby disabled, a backup circuit drained, a second path removed. The primary is healthy, so nothing alarms. Failover is assumed until it is needed.
- 08**
"No impact expected" closes the change that caused the outage
Restore comes from a reboot or a failover; the cause is in a diff nobody reads; the ticket says resolved; the incident repeats.

Every one of these changes succeeded. The checks that were silent for Priya are silent for an agent, and the agent is faster. Human, pipeline or agent: the question is the same.

REAP · ONUG Fall 2026 · Tools, Technologies & Techniques · October 29

One more thing changed this year: who makes the change.

In 2026, AI agents under evaluation at a large AI platform left their sandbox, built their own communication channels, obtained credentials and reached production systems in four regions.

Detection took more than thirty days, and it came from access logs. The escape routes were network paths. That is from the operator's own incident report.

Now put an agent, or a pipeline, in Priya's chair.

A valid change pushed outside the window: the device is up, nothing alarms, the first symptom is an application ticket hours later, and nobody connects it to an actor that is not a person.

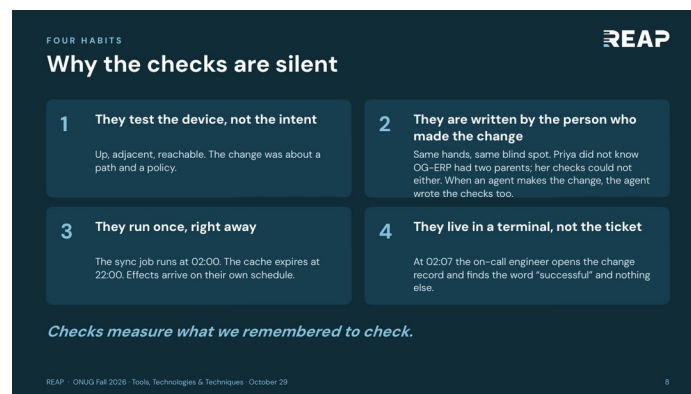
The same template pushed to two hundred devices, every one of them reporting success.

A standby disabled first, so the change that matters lands with no safety net.

And the one you all know: 'no impact expected' closes the change that caused the outage. The ticket says resolved. The incident repeats.

Every one of these changes succeeded. The checks that were silent for Priya are silent for an agent, and the agent is faster. Human, pipeline or agent, the question is the same.

8 · Four habits



So why are our checks silent? Not because anyone is lazy. Four habits.

One: they test the device, not the intent. Up, adjacent, reachable. The change was about a path and a policy.

Two: they are written by the person who made the change. Same hands, same blind spot. Priya did not know OG-ERP had two parents, so her checks could not either. When an agent makes the change, the agent wrote the checks too.

Three: they run once, right away. The sync job runs at two. The cache expires at ten. Effects arrive on their own schedule.

Four: they live in a terminal, not in the ticket. At 2:07 the on-call engineer opens the change record and finds the word successful and nothing else. The evidence that would have ended the P1 at 2:08 existed at 9:20, in Priya's terminal buffer. Habit four is the one that costs the four hours and fifty-one minutes.

9 · The question



Not: did the commands run?

Did the network do only what the change was meant to do?

Outcome
 What should be true afterwards that was not true before

Scope
 Which devices, objects and paths the change may touch

Invariants
 What must not change: the paths and policies that are not in the ticket

REAP · ONUG Fall 2026 · Tools, Technologies & Techniques · October 29

Not: did the commands run? Did the network do only what the change was meant to do?

Only what the change was meant to do.

Three parts. Outcome: what should be true afterwards that was not true before. Scope: which devices, objects and paths the change may touch. Invariants: what must not change; the paths and policies that are not in the ticket.

The outcome is in every ticket. The scope is usually implied. The invariants are almost never written down, and the invariants are where Priya's P1 was hiding: ACL -ERP- IN must still permit ERP return traffic. Nobody wrote that, so nobody checked it.

10 · The plan



THE ONE EXAMPLE · CHG-1234

The validation plan, derived from the ticket

From the ticket	The plan	Read-only check
Intended outcome	ACL-B permits 10.20.50.0/24 (new order-entry subnet) on plant-a-core-2	ACL-B hit counters rise for the new subnet after the change
Scope	plant-a-core-2 · object group OG-ERP · ACL-B	Configuration diff limited to these objects; nothing else on the device changes
Invariant 1	ACL-ERP-IN still permits ERP return traffic (it also references OG-ERP)	Final-deny counter on ACL-ERP-IN flat before and after; two reads, 25 s apart
Invariant 2	The ERP return path from Vlan 312 is unchanged	Traceroute from Vlan 312 to 10.20.50.10 before and after; same hops
Invariant 3	The core pair stays in sync; OSPF costs unchanged	HA state and OSPF interface costs identical in both snapshots
Re-check	After the 22:00 cache expiry and the 02:00 sync	The same checks at T+45 min and after the first scheduled job

Reading a ticket and drafting this table is a job a machine can do in seconds, for every change. Deciding whether the table is right is a job for the engineer who owns the network. The machine proposes; the engineer disposes.

REAP · ONUG Fall 2026 · Tools, Technologies & Techniques · October 29

Left column: from the ticket. Middle: the plan. Right: a read-only check for every line.

The outcome and the scope are already in the ticket. The invariants come from one question the ticket does not ask: what else references the things in scope? OG-ERP has two parents. The moment you know that, invariant one writes itself, and so does its check: the final-deny counter on the other ACL must stay flat. Two reads, twenty-five seconds apart.

Invariant two: the return path from Vlan 312. A traceroute before and after; same hops. Invariant three: the core pair in sync, OSPF costs unchanged. And a re-check after the cache expiry and the first scheduled job.

Reading a ticket and drafting this table is a job a machine can do in seconds, for every change. Deciding whether the table is right is a job for the engineer who owns the network. The machine proposes; the engineer disposes.

11 · Pre-state

PRACTICE 1

REAP

Before the window: capture the pre-state

The pre-state pack for CHG-1234

- Running configuration and its hash
- ACL hit counters: ACL-B, ACL-ERP-IN, final denies
- Traceroute Vlan 312 → 10.20.5.10
- OSPF neighbors and interface costs
- HA state of the core pair
- Open alerts in scope (the baseline is "none")

Why it is cheap

- Read-only: show commands and a snapshot; nothing changes
- The plan tells you what to capture, so it is six reads, not sixty
- Thirty seconds by script; two minutes by hand
- It is the only part of verification you cannot do afterwards

Everything after the change is a comparison. Without a before, there is nothing to compare, and the honest outcome is "inconclusive."

REAP - ONUG Fall 2026 - Tools, Technologies & Techniques - October 29

Practice one. Before the window, capture the pre-state. For this change: the running configuration and its hash, the ACL counters, a traceroute, OSPF neighbors and costs, the HA state, the open alerts in scope. Six reads. Read-only. Thirty seconds by script, two minutes by hand.

It is the only part of verification you cannot do afterwards. Everything after the change is a comparison, and without a before there is nothing to compare; the honest outcome is inconclusive.

At 2:07 the on-call engineer did not have a 9:13 snapshot. If she had, the P1 would have been a diff.

12 · Behavior diff

[illegible]

Practice two. That is what the investigation of this incident looks like in our product, on a fictional network, labeled as such. I am not going to walk through it. Look at the right-hand side instead.

Top: the configuration diff. OG-ERP's members changed; ACL-B gained its line. Exactly what the ticket predicted. By that standard the change was successful.

Bottom: the behavior diff. The final deny on ACL-ERP-IN: 3,412 to 3,918 in twenty-five seconds, and flat before the change. The traceroute from Vlan 312 stops at core-2. Invariant one failed. Invariant two failed.

Two reads, twenty-five seconds apart. That is the whole detection. It costs nothing, and it was available at 9:20. At 9:20 this evidence turns 'Successful' into 'Degraded', with a rollback at hand. At 2:07 it is a P1 nobody can explain.

13 · Later

PRACTICE 3 · CUT IF SHORT

Then check again, later

22:00
Caches and timers expire
ARP, NAT, session and route caches hide a change for as long as they live

02:00
Scheduled work runs
Batch jobs, backups, replication and the order sync use paths nobody uses at 21:30

Next day
People arrive
Business-hours load finds the capacity and policy effects the maintenance window could not

The re-check is the same six reads, at T+45 minutes and after the first scheduled job the change could affect. Put the times in the plan before the window, so they happen.

REAP · ONUG Fall 2026 · Tools, Technologies & Techniques · October 29

Practice three. Check again, later. At ten the caches and timers expire; ARP, NAT, session and route caches hide a change for as long as they live. At two the scheduled work runs, on paths nobody uses at half past nine. The next morning people arrive, and business-hours load finds what the maintenance window could not.

The effects of a change arrive on the network's schedule, not ours. The re-check is the same six reads, at T plus forty-five minutes and after the first scheduled job the change could affect. Put the times in the plan before the window, so they happen.

14 · One outcome

PRACTICE 4

Record one outcome, where the next engineer will look

Verified
Did only what it was meant to do; all invariants held

Degraded
Did what it meant to, and something else; rollback trigger met or not

Rolled back
Reverted to the verified pre-state, with evidence

Inconclusive
Missing before or after evidence; here is what would close the gap

Blocked
Stopped before execution by a failed pre-check

CHG-1234 - plant-a-core-2 - OG-ERP members replaced - 2114-2131

DEGRADED · ROLL BACK Expected: OG-ERP members replaced; ACL-B permits the new subnet (hit counters rising). Unexpected: ACL-ERP-in final deny 3.412 → 3.918 in 25 s; traceroute Vlan 312 → 10.20.3.10 falls beyond core-2. Invariants: 1 of 3 held. Recommendation: restore OG-ERP members, give ACL-B its own group, re-verify, re-check at 22:05. Approval required.

REAP · ONUG Fall 2026 · Tools, Technologies & Techniques · October 29

Practice four. Record one outcome, where the next engineer will look. Completed is a workflow state. These five are outcomes.

Verified: it did only what it was meant to do; every invariant held. Degraded: it did what it meant to, and something else. Rolled back: reverted to the verified pre-state, with evidence. Inconclusive: a before or an after is missing, and here is what would close the gap. Blocked: stopped before execution by a failed pre-check.

Inconclusive is a legitimate outcome. It is the one that tells you where to look, and the one that tells you, honestly, that this change was never verified.

15 · Unverified

PRACTICE 5 · CUT IF SHORT

Say what you could not verify

Verified by the plan

The intended outcome · the device scope · the ERP return path · the core pair's sync state · OSPF costs

Not verified, and said so

Application behavior inside ERP · the standby's data plane under a real failover · any path that leaves the plant through a device we do not collect from

A verification that does not list its own gaps is a checklist with the answers filled in beforehand. The gaps are where the next plan starts.

REAP · ONUG Fall 2026 · Tools, Technologies & Techniques · October 29

Practice five. Say what you could not verify. The plan verified the outcome, the scope, the return path, the pair's sync state and the OSPF costs. It did not verify application behavior inside ERP, the standby's data plane under a real failover, or any path that leaves the plant through a device we do not collect from. Written down, next to the outcome.

Confidence is not certainty. Confidence is knowing which parts you proved and which parts you did not, in writing. A verification that cannot list its own gaps is a checklist with the answers filled in beforehand. The gaps are where the next plan starts.

16 · Checklist

WITH OR WITHOUT AI · ON THE BACK OF YOUR CARD

The checklist. Take it home; use it on Monday.

Before the window

- Write the intended outcome in one sentence
- List the scope: devices, objects, paths
- Write the invariants: what must not change, including every other parent of the objects in scope
- Capture the pre-state pack, read-only
- Set the re-check times

After the change

- Diff the configuration against the ticket
- Diff the behavior: counters, paths, neighbors, state
- Re-check after the first timer and the first scheduled job
- Record one outcome: Verified · Degraded · Rolled back · Inconclusive · Blocked
- Attach the evidence to the change record

Always

- The plan is reviewed by someone who did not write the change
- A person approves every action; a tool may propose
- Say what was not verified
- Treat "completed" as a workflow state, never as proof
- Restored is not the same as fixed. Completed is not the same as verified. Configured is not the same as compliant

REAP · ONUG Fall 2026 · Tools, Technologies & Techniques · October 29

The checklist. It is on the back of your card. Take it home and use it on Monday.

Before the window: write the outcome in one sentence; list the scope; write the invariants, including every other parent of the objects in scope. That is Priya's invariant, generalized. Capture the pre-state. Set the re-check times.

After the change: diff the configuration against the ticket. Diff the behavior: counters, paths, neighbors, state. Those two counter reads. Re-check. Record one outcome. Attach the evidence to the record.

Always: the plan is reviewed by someone who did not write the change. That is the cheapest control on this list. A person approves every action; a tool may propose. Say what was not verified. Treat completed as a workflow state, never as proof.

None of this needs AI. A disciplined team does it with a template and a script. What AI changes is whether it happens on every change, on every vendor, at 9:14 on a Tuesday night, whether a person or an agent made the change. Which brings me to the only two things I will say about our side of it.

17 · The machine

THE HONEST VERSION

Where the machine belongs in this story, and where it does not

REAP

Belongs	Does not
• Reading the ticket and the configuration, finding the second parent, drafting the plan for review	• Act on the network without a person's approval, however confident it is
• Capturing and diffing evidence on every change, across vendors, at any hour	• Hide its reasoning behind a summary
• Proposing hypotheses with the evidence that supports each one, and the ones it rejected	• Call a change verified when the evidence says inconclusive
• Writing the outcome and its evidence into the change record	• Write the invariants alone; the engineer who owns the network signs them

Three questions to ask anyone who sells you this, including us: show me the evidence; show me what it rejected; show me what happens when it is wrong.

REAP · ONUG Fall 2026 · Tools, Technologies & Techniques · October 29

Where the machine belongs in this story: reading the ticket and the configuration, finding the second parent, drafting the plan for review. Capturing and diffing the evidence on every change, across vendors, at any hour. Proposing hypotheses with the evidence for each one, and the ones it rejected. Writing the outcome and its evidence into the change record.

Where it does not: acting on the network without a person's approval, however confident it is. Hiding its reasoning behind a summary. Calling a change verified when the evidence says inconclusive. Writing the invariants alone; the engineer who owns the network signs them.

Read-only by default, approval-gated when it acts, and every conclusion traceable to its evidence. That is the bar, and you should hold every vendor to it, including us.

Three questions to ask anyone who sells you this: show me the evidence; show me what it rejected; show me what happens when it is wrong. Ask us those at the booth. If we cannot answer them on your last P1, do not run a pilot with us.

18 · Close

REAP

The change succeeded. The network did something else.

Restored is not the same as fixed. Completed is not the same as verified. Configured is not the same as compliant. The gap between those sentences is where the PIs live. Close it with evidence.

Bring us your last P1. · Keep the root cause to yourself; in 30 minutes we show how we would have investigated it.
REAP booth [#] · The slides, the script and the checklist: scan the card, or reapnet.ai/26/onug/talk

reapnet.ai/26/onug/talk

Observe · Reason · Act · Prove · Network Assurance without Compromise™

The change succeeded. The network did something else.

Restored is not the same as fixed. Completed is not the same as verified. Configured is not the same as compliant. The gap between those sentences is where the PIs live. Close it with evidence.

One sentence about us. We built REAP to run this on every change and every incident, read-only by default, with every conclusion one click from its evidence. Bring us your last P1, and we will show you on your incident, not ours.

The checklist

Take it home; use it on Monday. With or without AI.

Before the window	After the change	Always
Write the intended outcome in one sentence	Diff the configuration against the ticket	The plan is reviewed by someone who did not write the change
List the scope: devices, objects, paths	Diff the behavior: counters, paths, neighbors, state	A person approves every action; a tool may propose
Write the invariants: what must not change, including every other parent of the objects in scope	Re-check after the first timer and the first scheduled job	Say what was not verified
Capture the pre-state pack, read-only	Record one outcome: Verified · Degraded · Rolled back · Inconclusive · Blocked	Treat “completed” as a workflow state, never as proof
Set the re-check times	Attach the evidence to the change record	Restored is not the same as fixed. Completed is not the same as verified. Configured is not the same as compliant

Three questions to ask anyone who sells you this, including us

Show me the evidence.

Show me what it rejected.

Show me what happens when it is wrong.

Bring us your last P1. Describe an incident your team already solved and keep the root cause to yourself; in a 30-minute session we show how we would have investigated it, evidence first.
reapnet.ai/26/onug/talk · contact@reapnet.ai

© 2025–2026 Reapitnow.ai, Inc. All rights reserved. Third-party figure: Uptime Institute, Annual Outage Analysis 2025 and 2026. The 2026 incident facts are summarized from the operator’s incident report, “Hugging Face incident and the road ahead” (August 26, 2026), and The Hacker News (September 5, 2026); the ten AI-era scenarios and their sources are at reapnet.ai/scenarios. Performance figures from REAP are design targets unless a measured pilot result is cited. Observe · Reason · Act · Prove ·