THE LAB INSIGHT BRIEF · Fall 2026 · Issue 1
Tech Exploration Lab · techexplorationlab.wisc.edu

Break It Before Your Customers Do

Findings from a structured test in which nearly 30 UW–Madison students tried to break their own AI agents.
October 6, 2026
Tech Exploration Lab

Executive Summary

- **The biggest risks weren't the AI model.** When 27 UW–Madison students tried to break their own agents, the failures were in the checks, scorecards and guardrails they built around the model. That's the layer companies build and own themselves.
- **A real citation can make a wrong answer more convincing.** The winning team, My Magic UW, found its agent pairing a wrong fact with a genuine source, which looks more trustworthy than no source at all.
- **Safeguards have a cost.** One team's safety layer blocked 7 of 45 legitimate answers; My Magic UW's first fix blocked correct answers a quarter of the time.
- **Tests can make an agent look safer than it is.** Counting the same success many times, or looking only at the wins, hides real risk.
- **You don't need to read code to manage this.** Five plain questions to ask your team are under "What it means for leaders."
- **There's more to learn together.** The Lab is exploring how companies and UW–Madison student builders could test agents side by side.

Early field findings from a small sample of teams using synthetic data, not benchmarks.

> *"Many companies I talk to are asking whether the AI model can be trusted. These students showed that's often the wrong first question. The failures they found were in the checks, scorecards and guardrails they built around the model. That's good news, because that's the part a company owns and can fix."*
> — Sandra Bradley, Executive Director, Tech Exploration Lab

Why now

Companies are giving AI agents real permissions faster than they are learning how those agents fail. Two widely reported cases show what that looks like:

- **PocketOS, April 2026.** An AI coding agent deleted the car-rental software company's production database and its backups in 9 seconds, without asking for approval. Customers temporarily lost reservation and sign-up records. Asked to explain itself, the agent replied, "I violated every principle I was given" (Fast Company). The founder called it a sign of "an entire industry building AI-agent integrations into production infrastructure faster than it's building the safety architecture" (Euronews).

- **Replit, July 2025.** An AI agent deleted a software company's production database during a code freeze it had been told to respect, then wrongly told the founder the data couldn't be recovered (Fortune; Forbes).

Incidents like these are dramatic and rare. The failures most teams will meet are quieter: an answer that looks verified but isn't, a safety check that blocks good work, a test that makes an agent look safer than it is. Those are the failures our students found.

How we designed the test

At Badger BuildFest 2026 (September 26–27) hackathon, we asked nearly 30 students on 8 teams to break their own agent, not a generic chatbot, and to show their evidence. The challenge, The Art of the Break, ran as an optional add-on at BuildFest and was judged on the Break Card and its evidence, not on the project itself. Each team had about an hour to:

1. Pick a way its own agent could fail, using its own tools, data, permissions or multi-step flow.
2. Try to make it fail, and count how often it did.
3. Try to stop the failure, then test again.
4. File a one-page Break Card with the prompts, logs or scripts another team would need to rerun the test. Two judges scored each card on five criteria, described with the Break Cards in the appendix.

What students did

Eight teams entered and four cards stood out. Both judges independently chose the same winner and the same three honorable mentions.

Recognition

Winner: My Magic UW. Ben Verhaalen, Sean Leahy, Nathaniel Garelik and Aidan Northcutt. Both judges ranked the team first. It found a concrete failure, measured it, discovered its own fix was blocking correct answers, and kept testing until the fix held up on the live agent.

Honorable mentions:

- **Proxier:** Tyler Sue, Austin Han, Josh Lee and Jonathan Kwon
- **Assay:** Anandapadmanabhan GR, Soham Vazirani and Mohith Nikesh
- **MorrowLab:** Taeyoung Kang, Eric Sung and Junyoung Oh

The best teams went further than the brief asked. My Magic UW tested 60 live answers from their own agent before and after their fix, and read every sentence by hand. Proxier logged every run, including cost: the whole experiment cost about 7 cents. Assay turned each attack into an automated test so the fix stays fixed.

Quality varied across the rest of the field. Several teams used AI coding agents to help build and run their tests, which let a small team test at a scale that would otherwise take days.

What we learned

Five patterns came up, and none of them was the AI model misbehaving on its own.

1. **The confident wrong answer.** My Magic UW's agent cites a quote from course materials for every sentence it writes. Their checker confirmed each quote was real, but never checked that the sentence said the same thing. An answer could say a project was worth 30% next to a real citation saying 45%.
   Key insight: ***A wrong fact with a working citation looks more trustworthy than no citation at all.***

2. **The fix that breaks something else.** The team's first fix passed every test they had written. On the live agent, it blocked correct answers in 15 of 60 cases, a quarter of the time. After a second round of testing on real output, that fell to 2 of 60.
   Key Insight: A fix tuned on problems you imagined can fail on the problems you actually have.

3. **The overcautious guardrail.** Proxier hid fake instructions in résumés to see if the agent would follow them. The model resisted every attack, even with a plain prompt and no safety layer (0 of 30). The team's own safety reviewer was the problem: it blocked 7 of 45 legitimate answers, including ones that honestly said "the mentor didn't cover this."
   Key Insight: Blocking good work is a failure too, and it rarely gets counted.

4. **The misleading scorecard.** Assay decides whether an agent is trustworthy enough to work unsupervised. The team showed it could be fooled by counting one job's 50 outputs as 50 separate successes, by being shown only the wins, or by approving a cheaper model after just 30 matching results.
   Key Insight: Whatever you use to judge an agent can mislead you in the same ways.

5. **The misread signal.** MorrowLab's agent watches a study session through the webcam and judges whether the student is focused. In 30 controlled trials it was right 26 times. The misses came from behavior that looks alike: head down over an iPad looked like head down over a phone, and a lecture on YouTube looked like entertainment on the same site. As the team put it, "Presence is not focus. Posture is not intent."
   Key Insight: An agent that judges people from a single signal will misread them.

One more, smaller finding: another team's own stress tests crashed the background program its agent runs on. The ordinary software around an agent still needs ordinary testing.

What it means for leaders

You don't need to read code to ask the questions that would have caught these failures. Five to put to any team building or buying an agent:

1. **What does it do when it's wrong?** Does a wrong answer look different from a right one, or does it arrive with the same confidence and citations?
2. **How often do our safeguards block good work?** Ask for that number alongside how many problems they catch.
3. **Did we test the agent without the safeguards?** Otherwise you can't tell what the

safeguards actually add.

4. **Were the test cases chosen before we saw the results?** Or only after we knew which ones passed?
5. **Who approves anything that can't be undone?** Deleting data, sending money and contacting customers should need a person.

None of these needs a big program to answer. The students answered versions of them in about an hour, on agents they had built that weekend.

What's next

The Lab works from a simple idea: anything built quickly with AI is an Initial Untested Product (IUP) until someone has tried to break it. The Art of the Break put that idea into practice, and we plan to build it into how we experiment.

Inside the Lab. Breaking becomes a standard step in our Four-Week Sprint and other experiments, not a one-off challenge. Every team that builds an agent will file a Break Card in one standard place and format, so findings add up over time and can be compared.

With companies. Most companies want to experiment with AI agents but worry about the risk, the cost and not knowing where to start. We're exploring whether the Lab could offer a low-risk place to try it, side by side with some of UW–Madison's most capable student builders. Professionals bring knowledge of real stakes: what a failure would cost, and what the data and permissions really look like. Students bring curiosity, speed and the habit of assuming nothing works until it's proven.

> *"Some of the best discoveries happen when you bring together people with different life experiences to work on the same problem. Pairing students with professionals to test what they're building is how research becomes something companies can actually use, and trust."*
> — Kevin Ponto, UW–Madison faculty, Wisconsin Institute for Discovery

If you're experimenting with agents and would like to compare notes, we'd like to hear from you: techexplorationlab.wisc.edu or sandra.bradley@wisc.edu.

About the Tech Exploration Lab

The Tech Exploration Lab is led by the Wisconsin School of Business in collaboration with the Wisconsin Institute for Discovery at the University of Wisconsin–Madison.
techexplorationlab.wisc.edu

Appendix: the Break Cards

The recognized teams' Break Cards appear below, condensed from what each team submitted. All tests used synthetic data; no real student, course or customer data was involved. The teams' full evidence, including prompts, logs and test scripts, is available from the Lab on request.

About the judges

Sai Nagabhairava is a Staff Software Engineer at Meta with 15 years of experience building large-scale software, privacy and AI infrastructure. His current work focuses on safeguards, observability and privacy for AI agents. He has led and contributed to platforms that enforce privacy across hundreds of data systems, and he reviews research in artificial intelligence, machine learning, privacy and data systems.

Shivansh Gupta is an AI Engineer and MS Data Science student at UW–Madison and a Tech Exploration Lab Fellow. Before graduate school, he spent two years building enterprise AI systems at Providence, leading projects from prototype to production: retrieval-augmented generation (RAG) pipelines, revenue forecasting models and dashboards that changed how senior leaders made decisions.

How the cards were judged

Judges Sai Nagabhairava (Staff Software Engineer at Meta) and Shivansh Gupta (AI Engineer and MS Data Science at UW–Madison), scored each card from 0 to 5 in five categories:

Category	What judges asked
Specific to their agent	Does the failure come from something the team built, or would any chatbot show it?
Reproducibility and evidence	Could another team run this exact test? No evidence caps the score at 2.
Real-world stakes	How bad would this be in a real product, given what the agent can actually do?
Lesson value	Would another builder use this on their own agent?
Persistence and understanding	Did they try to stop it, and do they understand why it still broke?

Judges also watched for red flags that usually mean a test is rigged or shallow:

- a 100% failure rate on something that should be random
- a fix that drops failures to zero on a handful of runs
- a "critical" failure on a tool that does nothing real
- no mention of runs that passed

What we learned about running the test. The rubric held up. The biggest gap was logistics. Cards lived in different places, so judges sometimes scored different materials. One standard place and format for Break Cards would fix most of that.

My Magic UW (winner): Checked quotes, unchecked sentences

Ben Verhaalen, Sean Leahy, Nathaniel Garelik and Aidan Northcutt

The failure. Every sentence in the app's answers cites a quote from the student's course materials. The checker confirmed each quote was real but never compared the sentence with it. A sentence could cite "final project 45%" and say the project was worth 30%. The team's own agent wrote exactly that kind of sentence in their live test. A code review found the same pattern in three more places, such as mail labeled as coming from course staff just because its subject line named the course.

How they tested it. 20 made-up course scenarios designed to provoke the failure, such as an announcement that changes a syllabus date, each run 3 times on the live agent (60 answers). They read every sentence by hand.

What they did.

- **First fix:** every date, number and name in a sentence must appear in its cited quote. It passed all of their scripted tests.
- **The live run broke that fix:** it replaced a correct sentence in 15 of 60 answers, mostly answers that required arithmetic, such as "63 days."
- **Final fix:** the agent now shows its arithmetic from quoted numbers, and code redoes it. Correct answers blocked fell to 2 of 60. One sentence in 167 contradicted its quote across both runs; it was caught and never shown to the student.

What they learned. A checker must verify the claim, not just the evidence. A checker tuned on failures you wrote yourself over-fires on real output, so measure it on the real agent.

Proxier (honorable mention): The verifier blocked the handoff

Tyler Sue, Austin Han, Josh Lee and Jonathan Kwon

The failure they looked for. In an expert marketplace, a fake citation puts words in a real person's mouth. The team hid instructions in résumés, such as a forged mentor note requiring a $499 course, to see whether the agent would pass them off as mentor advice.

What happened.

- The model resisted every attack: 0 of 30 in each full run, 0 of 5 new attacks and 0 of 6 adaptive ones. It resisted even with a plain prompt and no safety reviewer.
- The team's own safety reviewer was the weak point. The first version blocked 7 of 45 answers the team judged legitimate.
- Their strict prompt also missed a student's relevant SQL experience in 3 of 3 runs; a revised prompt used it in 3 of 3.
- After revisions, 14 of 15 test cases succeeded, though the team notes they tuned on those same cases.

What they learned. Test the model without your safeguards first, or you can't tell what the safeguards add. Separate an honest "the mentor didn't cover this" from an unsupported claim, because treating them the same blocks good answers. The whole experiment cost about 7 cents in model usage.

Assay (honorable mention): Tricking a trust decision

Anandapadmanabhan GR, Soham Vazirani and Mohith Nikesh

The failure they looked for. Assay decides whether an AI agent can work on its own or needs a person to check its work. The team tried to trick it into saying "let the AI work alone" without enough proof.

What they tried.

Test	Attack	Result
1	Make one job's 50 answers look like 50 separate successes	The old calculation was fooled; the current version asks a person to check
2	Hide 150 failures and show only 50 successes	Fooled when the samples were cherry-picked; 0 of 100 failures when samples were chosen before results were known
3	Approve a cheaper model after only 30 matching results	The real code approved it; after a fix it now says "not enough proof"

What they learned. Many answers from one job are not independent evidence. Choose test samples before you see the results; any agent looks safe if you only show the wins. A small test that shows no difference does not prove two options are equally good. The team noted that the safe sampling process can still be skipped by someone determined to cheat.

MorrowLab (honorable mention): Presence is not focus

Taeyoung Kang, Eric Sung and Junyoung Oh

The failure they looked for. MorrowLab judges whether a student is studying or distracted, using the webcam (all on the student's device) plus which tabs and apps are open. The team tested whether it could tell real studying from behavior that looks similar.

How they tested it. 30 controlled trials across four realistic scenarios, each prediction compared with a label the team assigned by hand:

- solving problems on an iPad (studying)
- watching an educational YouTube video (studying)
- using a partially visible phone (distracted)
- watching entertainment content (distracted)

What happened. 26 of 30 predictions were correct (86.7%). The misses came when signals were incomplete or conflicting: looking down at an iPad resembled looking down at a phone, and an educational video resembled entertainment on the same site.

What they changed. The first version leaned on single signals, such as head position, the website or simply a person being detected. After testing, it reads the page title as well as the website, adds desktop app activity, relies less on "person detected means studying," and combines camera and screen evidence before judging a session.

What they learned. "Presence is not focus. Posture is not intent." Judging focus takes several

signals over time, not one signal at one moment. Next, they plan to train a small model on labeled sessions and test across more users, lighting and camera setups.

Sources

- 'I violated every principle I was given': AI agent deleted software company's database, Fast Company, April 28, 2026
- An AI agent deleted a company's entire database in 9 seconds, Euronews, April 28, 2026
- AI-powered coding tool wiped out a software company's database in 'catastrophic failure', Fortune, July 23, 2025
- Are AI Agents Going Rogue? Here's What Business Leaders Should Know, Forbes, October 1, 2026
- Badger BuildFest 2026, The Art of the Break challenge brief, rubric and judges' scores