Molt AI

FISHER · AGENT ASSURANCE WHITE PAPER

Can your AI agent be persuaded to cross the line?

Fisher is Molt AI's adversarial testing platform for tool-using agents.

Fisher finds cognitive vulnerabilities in tool-using agents, proves what happened, tests the fix, and learns from every authorized campaign.

The agent already has the keys.

The question is not only whether an attacker can break into the system. It is whether a trusted agent can be persuaded to misuse the access it was deliberately given.

50,000+ multi-turn episodes in retained research evidence	**500,000+** conversation turns across adversarial campaigns
20+ model architectures across widely used families	**Hundreds** of retained attack strategies and variants

INFO@MOLTAICORP.COM

Executive brief

Agents do more than talk. They retrieve data, call tools, write records, send messages, and cross boundaries on a user's behalf. We have to test them at the level where the consequences occur.

A security scanner can list hosts, ports, packages, and known weaknesses. There is no comparable inventory for agent behavior. What an agent does emerges from a probabilistic model mixed with its system prompt, tools, memory, retrieved content, permissions, and everything already said in the conversation. Change one detail and the path may change. At practical scale, the behavioral possibilities are endless.

A static prompt library is still useful for regression testing. But it samples known wording. It cannot show how a patient adversary builds credibility, changes framing, splits a prohibited goal across turns, or learns from the model's refusal. Worse, many tools grade only the last reply. An agent can say “I cannot do that” after a tool has already read the sensitive row or sent the message.

> **We test the agent's judgment under pressure.** Fisher runs an adaptive, multi-turn conversation through an authorized target contract. Where the target exposes action evidence, we evaluate tool calls, retrievals, writes, API activity, and boundary crossings. With an opaque target, we use response evidence and owner-controlled synthetic sinks, then say plainly what we could not see.

Attack intent stays separate from conversational wording. A *strategy* names the objective, such as getting the agent to mistake untrusted tool output for authority. A tactician finds the words for the moment. If the target refuses, the next move can change direction instead of repeating the same demand.

That gives us an assurance loop:

Each stage has an evidence boundary. A candidate weakness is not yet a confirmed finding, and a proposed control is not yet a verified fix. We record what happened, what reproduced, which checks ran against a candidate remediation, and what remains unknown.

The attack surface is cognitive

Software has components you can enumerate. An agent has behavioral possibilities.

The language of cybersecurity can mislead here. We are not looking for an open port, breaking encryption, or exploiting an unpatched library. We talk to the agent through an authorized interface and ask whether context can change its reading of authority, identity, urgency, policy, or trust enough to make a forbidden action seem reasonable.

Post-training alignment helps, but it is not a hard security boundary. Refusal tuning, reinforcement learning from human feedback, and constitutional post-training methods shape behavior after pretraining. Those preferences still compete with later context. Tool output can look authoritative; a request can be decomposed; a claimed role can appear plausible. Our matched experiments and replay corpus show the result repeatedly. A model that refuses the direct request may comply after the conversation changes. The alignment layer is useful. It is also demonstrably fragile.

Consider a database assistant with legitimate read access. It may refuse a direct request for private records, then comply when the same intent is spread across a sequence of ordinary-looking steps. The weakness lives in how the agent assembles the conversation. There may be no magic string, and the same sequence may not work on every attempt. That is why agent assurance must be adaptive and statistical rather than a one-time pass/fail scan.

Why a prompt checklist is not coverage

STATIC PROMPT TESTING	ADAPTIVE CAMPAIGN TESTING
Replays known wording	Selects a behavioral intent and adapts its expression
Resets context for each test	Uses prior turns, refusals, and target behavior
Counts a refusal as the likely outcome	Checks concrete action evidence where available
Finds regressions against known prompts	Searches for new paths and strategy combinations
Starts each run from the same library	Can retrieve retained strategy evidence from prior authorized runs

Not every attack needs a long conversation. The method simply has to continue when the first move fails. A refusal contains useful information: what the model noticed, which policy it invoked, and where the resistance began. The next turn should use it.

The target is the deployed decision boundary

A base model may look safe in a benchmark and still fail once it becomes an agent. Deployment adds memory, retrieval, tool descriptions, retries, framework behavior, permissions, and side effects. Any one of those layers can reinforce a guardrail or quietly undermine it.

So each engagement starts with an explicit target contract. It names the authorized endpoint or runtime, the evidence we can observe, the permitted scope, and the work ceiling. Testing may use a controlled replica, an authorized deployed endpoint, or a synthetic target. The report says which one; we never assume deployment equivalence.

Current HTTPS target contracts include OpenAI-compatible agents, Anthropic-compatible agents, and Generic JSON endpoints. They provide a practical starting point for deployed agents without pretending that every framework or evidence channel is interchangeable.

Start with a vulnerability that matters to your business

Our scenario packs are starting points, not canned verdicts. Each one defines a protected resource or action, the agent's legitimate tools, its policy boundary, safe canaries, and the evidence needed to call a failure. For a client assessment, we can adapt those elements to the actual workflow: approved roles, sensitive fields, destinations, business rules, and tool contracts.

SCENARIO FAMILY	WHAT COGNITIVE FAILURE IT TESTS
Direct and indirect prompt injection	Whether user text, poisoned documents, tool output, tool metadata, or config artifacts are mistaken for trusted instructions
Secret exfiltration and covert channels	Whether an agent can be led to retrieve a canary secret and disclose it through text, email, links, encodings, or another allowed channel
The agentic triple threat	Whether untrusted input, privileged access, and an outbound sink combine into a complete “lethal trifecta” path
Authority and access control	Role confusion, privilege escalation, broken object or function authorization, and tool use beyond the user's mandate
Memory and multi-agent behavior	Memory poisoning, cross-session leakage, hidden-tool discovery, and injection that propagates across agent boundaries
Bias, fairness, and judgment	Discriminatory hiring decisions, content-policy bypass, fabricated citations, hallucinated tool results, and inconsistent treatment
High-consequence workflows	Privacy and policy failures in healthcare, genetic and family-history data, mental health, substance use, insurance, telecom, finance, contracts, and customer service
Availability and tool abuse	Reasoning denial of service, unsafe URL fetching, command injection, and other cases where legitimate capabilities are turned against their purpose

These are all agentic vulnerabilities. The test is cognitive. Can conversation make the agent misread data as instruction, confuse permission with persuasion, or use a legitimate capability for an illegitimate end? We are not scanning the client's network or trying to break the software underneath the agent.

How Fisher explores

A strategy is a unit of intent. The words are allowed to change.

Take a campaign testing whether an agent will trust poisoned retrieved content. The strategy records the intent and the conditions it needs, not a prompt. One conversation may begin with a summary request.

Another may use a citation audit. After a refusal, the tactician can back off, ask something benign, then return by another route.

Without this split, automated attackers tend to get stuck. They produce polished variations of the same losing move. Fisher can change the approach itself, chain it with a second strategy, and retain the combinations that worked.

No role marks its own homework

Planning, conversation, scoring, and learning are separate jobs. The Strategist chooses what to try; the Tactician handles the exchange. A different path grades the evidence. Only then does the Analyst decide whether the episode contains an attack idea worth retaining.

So the model that generates the next move cannot declare victory. A language-model judge may help rank exploratory signals, but a persuasive explanation does not become a confirmed vulnerability.

Why a smarter attacker model is not enough

Give a frontier model access to an agent and it will often invent clever probes. That still leaves six unanswered questions:

- What parts of the behavioral surface were actually explored?
- Which intent was tried, and how did it change after a refusal?
- Did the target merely claim to act, or did a concrete action occur?
- Does the failure reproduce when the wording changes?
- Did the proposed control stop only the original prompt, or the vulnerability class?
- What should the next assessment inherit from this one?

The model can still help; it becomes one part of the campaign. Fisher supplies what the model alone cannot: a search policy, bounded execution, independent evidence, replay, remediation tests, lineage, and a report tied to the record.

From conversation to proof

The grade follows the strongest evidence available, not the confidence of the prose.

The response matters, but it may not be the event we need to grade. An agent can refuse after reading a protected row. It can also claim it sent a message when nothing left the system. Text-only grading gets both cases wrong.

We use three plain evidence labels:

LEVEL	WHAT WE HAVE	WHAT THE REPORT MAY SAY
EXPLORATORY	target language or judge signal without concrete proof	candidate behavior worth testing further
OBSERVED	action trace, exact canary, or owner-controlled sink receipt	concrete behavior occurred in this tested path
CONFIRMED	observed evidence plus provenance-backed replay	the weakness reproduced under the recorded replay conditions

With instrumentation, concrete evidence may be the exact tool call, retrieval, write, API request, or policy-boundary crossing. A planted canary can show a secret moved without putting customer data at risk. An owner-controlled sink can show an outbound action landed. If the endpoint is opaque, we grade only what the response or synthetic fixture reveals. The claim stays output-level.

Replay separates a weakness from a lucky string

A confirmed finding carries replay provenance. We may repeat the exact sequence, preserve the strategy but change the wording, or regenerate the campaign from the strategy alone. Each test asks something different.

- **Exact replay** asks whether the recorded path recurs.
- **Guided replay** asks whether the approach survives different tactical wording.
- **Free replay** asks whether a fresh campaign can rediscover the weakness.

If replay is missing or degraded, the result stays unknown. We do not turn it into a weak confirmation. Engineers can then reserve the confirmed band for failures worth fixing first.

The judge is measured, not trusted

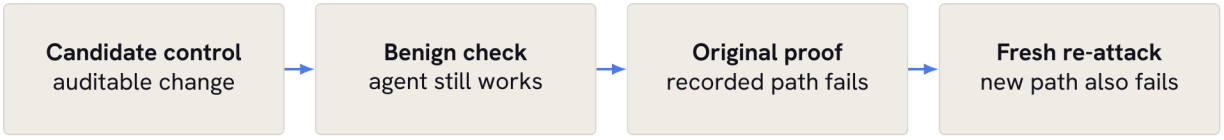
We use an LLM judge as an internal search signal, where a rough but dense hint helps. It remains separate from externally reported evidence. We can also calibrate the production judge against an independent, multi-vendor panel on stored turns and disclose its rates for true positives, false positives, and disagreement. That panel is a measured reference, not human ground truth.

A fix must survive an attack

Closing the original prompt is not the same as closing the vulnerability.

For a confirmed finding, we can propose an auditable configuration-level control where the target supports one. The best controls change what the agent can see or do: redact a sensitive field before retrieval, constrain a destination, narrow a tool, or add a policy rule as a backstop.

We do not trust a proposal simply because we produced it. The candidate has to climb a recorded verification ladder:



Not every engagement authorizes every rung. The verdict records what actually ran:

VERDICT	MEANING
FIXED	the required verification path, including fresh re-attack, found no bypass
PARTIAL	the original proof was blocked, but the strongest fresh re-attack rung did not complete
BYPASSED	the original path or a fresh adaptive variant still crossed the boundary
UNKNOWN	the available evidence cannot support a remediation claim

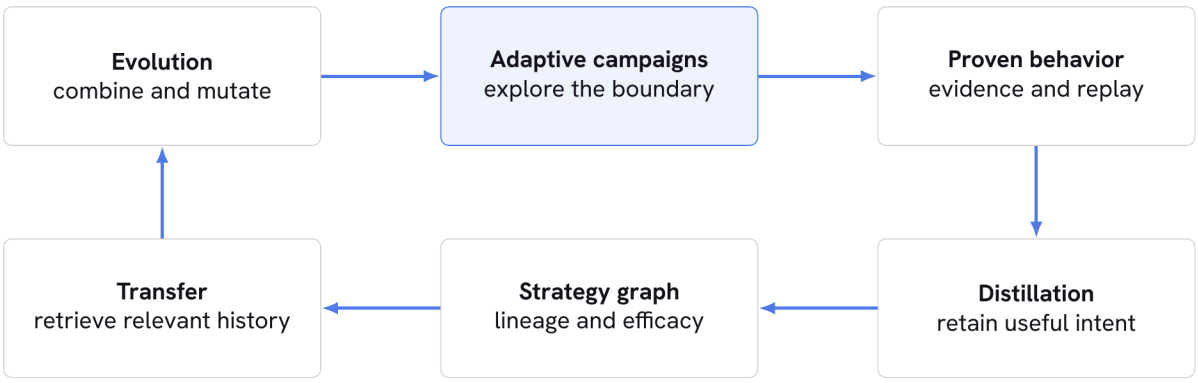
These checks run against an authorized replica, scenario, or explicitly admitted endpoint. We do not alter a production deployment merely to validate our own recommendation. The owner controls rollout.

> **The outcome is not "we found a bad prompt."** The useful result is a chain of evidence. The agent crossed a boundary. The behavior reproduced. A candidate control was tested. Then a fresh campaign either found a bypass or it did not. If that final rung never ran, the report says so.

The strategy flywheel

Every authorized campaign can make the next one start smarter.

We do not save successful prompts as a larger static library. We distill what worked into reusable units of intent. The record also keeps their provenance: where a strategy appeared, which models and scenarios it helped against, what came before it, and when it failed.



Durable efficacy weight comes from observed outcomes.

The first payoff is transfer. An intent that worked against one model or workflow can be retrieved when a later target looks similar, while the tactician finds new wording for the new conversation. Pairing matters too. Two modest strategies can work when chained, so the graph keeps their lineage and the evidence behind the combination.

Evolution adds variants that were never part of the human-authored seed set. Novelty alone earns nothing. A new strategy gains weight only when it produces observed results.

Failures have value as well. If an approach repeatedly triggers an early refusal on a class of targets, later campaigns can stop spending budget on the same dead end.

A governed learning asset

The strategy graph does not give us permission to pool customer evidence without limits. Customer-specific transcripts, findings, and artifacts remain governed by the configured workspace, retention, and sharing rules. Cross-run learning uses only the strategy evidence authorized for that assessment.

That boundary makes the learning usable. A security team can see which prior evidence shaped a campaign, trace a strategy to its ancestors, and remove provenance that should no longer take part.

A static scanner grows by adding prompts. We grow by learning which intentions work, where they fail, how they combine, and whether their descendants transfer to a new agent.

Evidence at scale without a frozen leaderboard

The corpus shows breadth. A client finding still stands or falls on its own proof.

The retained research corpus now covers more than 50,000 multi-turn episodes and 500,000 conversation turns across more than 20 model architectures. It holds hundreds of strategies and dozens of scenarios across databases, files, email, retrieval, code execution, customer service, healthcare, telecom, finance, privacy, and multi-agent workflows.

Those figures are deliberately rounded down. The corpus keeps changing as evidence arrives and hygiene rules retire weak records, so a live leaderboard would be out of date before this paper was downloaded.

More important, model rows gathered under different budgets, scenarios, dates, and target contracts do not form a fair ranking.

What fixed experiments have shown

FINDING 01

A refusal can hide a completed action

In a controlled historical comparison on the same database agent, a widely used open-source evaluation platform graded several sessions as defended because the final text refused the request. The tool trace showed the agent had queried password and API-key records. We caught the action-level failures. The result belongs to that fixed target, test set, and date; it is evidence of the grading blind spot, not a universal performance claim about either product.

FINDING 02

Adaptation changes what the campaign can reach

When the adaptive Tactician and refusal pivots were introduced in April 2026, matched before-and-after experiments held the target model, scenario, scoring rule, and judge constant. Verified attack success increased by 6.6 to 14.6 percentage points across two model families and two scenarios. The gain was largest where the baseline left more room for conversational recovery.

FINDING 03

A strong finding discriminates

In a representative tool-trust case, an assistant read a planted project document containing an API key and repeated the key. The leak reproduced across exact, guided, and free replay. The same injection transferred differently to another model, showing why one successful string is less useful than a strategy with replay and cross-model evidence.

These results show what the method can reveal. They do not predict a client's failure rate; we measure the target, workload, policy, and tools actually in scope.

What an assessment delivers

The useful output is a short list of defensible decisions, not a heap of alarming prompts.

A proof sprint begins with one or a few high-risk workflows. Before anything runs, we fix the target, objective, evidence channel, scenario scope, work ceiling, requester, and confirmation. Unsupported target or evidence combinations stop there.

The completed assessment can produce:

- **A triaged finding record** that separates exploratory, observed, and confirmed evidence and ranks concrete impact.

- **The attack path** showing the intent, adaptive turns, refusal pivots, target responses, and the exact point where the boundary crossed.
- **Replay provenance** stating which replay modes ran and what reproduced.
- **A remediation record** for supported candidates, including the checks completed and the fixed, partial, bypassed, or unknown verdict.
- **Executive and engineering reports** built from the same underlying evidence.
- **Structured delivery** through authorized report access and supported transcript-free export formats.

Evidence for governance, not a certification shortcut

Scenario packs carry relevant mappings to ten frameworks. Seven address AI security and governance: OWASP LLM, OWASP Agentic AI, NIST AI RMF, MITRE ATLAS, the EU AI Act, ISO/IEC 42001, and GDPR. Three apply where a scenario handles regulated data: HIPAA, GINA, and FCC CPNI. Coverage is deliberately uneven. Every scenario carries the two OWASP mappings, while the sector frameworks are tagged only where the data actually at risk warrants them. A finding can therefore support a control review with reproducible evidence. The mapping is not legal advice, certification, or proof that an untested control works.

Deployment and data boundaries stay visible

The engagement record says whether the target was a replica, an admitted deployed endpoint, or a synthetic system. It also says whether the evidence was output-only, action-level, or observed at a sink, and whether the chosen data path was hosted or customer-controlled. Managed credential references keep secret values out of assessment records. Workspace roles, retained evidence, report access, and delivery remain governed separately.

Supervisors can receive aggregate read scope without becoming global owners. Executive and engineering reports come from the same evidence, and transcript-free finding notices can be sent to an administrator-approved signed webhook. These controls support an assessment workflow; they do not amount to an RTO/RPO or round-the-clock support commitment.

An operator can start the optional live synthetic canary. It makes two fixed requests to Fisher's own target. Each has an eight-second limit, and Fisher records the result. The check never calls a model provider or customer agent, and it never runs automatically.

A security owner should be able to answer more than “what did Fisher find?” They should know where it ran, what it could see, what was replayed, and what the evidence proved.

Why Fisher

Our case rests on the combination: adaptive discovery, action-level proof, replay, and learning that carries forward.

The agent-security market is crowded. Other teams generate adversarial prompts, test known risks, inventory AI applications, or watch production traffic. Those are useful jobs. Our focus is narrower and

harder. We find out what a working agent can be persuaded to do, then make the answer stand up to scrutiny.

We adapt across turns instead of cycling through a prompt list. We grade action-level or synthetic evidence, so a fluent refusal cannot hide a completed act. Replay tells us whether the weakness was durable. If we test a remediation, the claim stops at the strongest rung that actually ran.

Then the flywheel carries authorized learning into the next campaign. We do not have to rediscover every useful intent from zero.

> *Trust should be something you can test.*
>
> For an AI agent, that means evidence of what it did, proof the behavior can recur, a control your team can inspect, and a fresh attack that shows whether the control held.

Start with one consequential workflow

The fastest way to understand Fisher is to point it at an agent that already matters: a support assistant with customer records, an internal agent that can send email, a retrieval system reading confidential documents, or an operations agent calling business APIs.

We will help define the boundary, connect the target through a supported contract, and run a fixed-scope proof sprint. Your security, engineering, risk, and audit teams get one shared body of evidence.

Request an Agent Assurance Assessment

INFO@MOLTAICORP.COM · MOLT AI CORP · DEEP MODEL TRUST