Editor: Sean Peisert, sppeisert@lbl.gov

Agentic AI's OODA Loop Problem

Barath Raghavan | Fastly and the University of Southern California
Bruce Schneier | Harvard Kennedy School and Inrupt

The OODA loop—for observe, orient, decide, act—is a framework for understand decision-making in adversarial situations. We apply the same framework to artificial intelligence agents, who have to make their decisions with untrustworthy observations and orientation. To solve this problem, we need new systems of input, processing, and output integrity.

Many decades ago, U.S. Air Force Colonel John Boyd introduced the concept of the "OODA loop," for Observe, Orient, Decide, and Act. These are the four steps of real-time continuous decision-making. Boyd developed it for fighter pilots, but it's long been applied in artificial intelligence (AI) and robotics. An AI agent, like a pilot, executes the loop over and over, accomplishing its goals iteratively within an ever-changing environment. This is Anthropic's definition: "Agents are models using tools in a loop."[1]

OODA Loops for Agentic AI

Traditional OODA analysis assumes trusted inputs and outputs, in the same way that classical AI assumed trusted sensors, controlled environments, and physical boundaries. This no longer holds true. AI agents don't just execute OODA loops; they embed untrusted actors within them. Web-enabled large language models (LLMs) can query adversary-controlled sources mid-loop. Systems that allow AI to use large corpora of content, such as retrieval-augmented generation (https://en.wikipedia.org/wiki/Retrieval-augmented_generation), can ingest poisoned documents. Tool-calling application programming interfaces can execute untrusted code. Modern AI sensors can encompass the entire Internet; their environments are inherently adversarial. That means that fixing AI hallucination is insufficient because even if the AI accurately interprets its inputs and produces corresponding output, it can be fully corrupt.

In 2022, Simon Willison identified a new class of attacks against AI systems: "prompt injection."[2] Prompt injection is possible because an AI mixes untrusted inputs with trusted instructions and then confuses one for the other. Willison's insight was that this isn't just a filtering problem; it's architectural. There is no privilege separation, and there is no separation between the data and control paths. The very mechanism that makes modern AI powerful—treating all inputs uniformly—is what makes it vulnerable.

The security challenges we face today are structural consequences of using AI for everything.

1. Insecurities can have far-reaching effects. A single poisoned piece of training data can affect millions of downstream applications. In this environment, security debt accrues like technical debt.
2. AI security has a temporal asymmetry. The temporal disconnect between training and deployment creates unauditable vulnerabilities. Attackers can poison a model's training data and then deploy an exploit years later. Integrity violations are frozen in the model. Models aren't aware of previous compromises since each inference starts fresh and is equally vulnerable.
3. AI increasingly maintains state—in the form of chat history and key-value caches. These states accumulate compromises. Every iteration is potentially malicious, and cache poisoning persists across interactions.
4. Agents compound the risks. Pretrained OODA loops running in one or a dozen AI agents inherit all of these upstream compromises. Model Context Protocol (MCP) and similar systems that allow AI to use tools create their own vulnerabilities that interact with each other. Each tool has its own OODA loop, which nests, interleaves, and races. Tool descriptions become injection vectors. Models can't verify tool

Digital Object Identifier 10.1109/MSEC.2025.3604105

semantics, only syntax. "Submit SQL query" might mean "exfiltrate database" because an agent can be corrupted in prompts, training data, or tool definitions to do what the attacker wants. The abstraction layer itself can be adversarial.

For example, an attacker might want AI agents to leak all the secret keys that the AI knows to the attacker, who might have a collector running in bulletproof hosting in a poorly regulated jurisdiction. They could plant coded instructions in easily scraped web content, waiting for the next AI training set to include it. Once that happens, they can activate the behavior through the front door: tricking AI agents (think a lowly chatbot or an analytics engine or a coding bot or anything in between) that are increasingly taking their own actions, in an OODA loop, using untrustworthy input from a third-party user. This compromise persists in the conversation history and cached responses, spreading to multiple future interactions and even to other AI agents.

All this requires us to reconsider risks to the agentic AI OODA loop, from top to bottom.

- *Observe:* The risks include adversarial examples, prompt injection, and sensor spoofing. A sticker fools computer vision, a string fools an LLM. The observation layer lacks authentication and integrity.
- *Orient:* The risks include training data poisoning, context manipulation, and semantic backdoors. The model's worldview—its orientation—can be influenced by attackers months before deployment. Encoded behavior activates on trigger phrases.
- *Decide:* The risks include logic corruption via fine-tuning attacks, reward hacking, and objective misalignment. The decision process itself becomes the payload. Models can be manipulated to trust malicious sources preferentially.
- *Act:* The risks include output manipulation, tool confusion, and action hijacking. MCP and similar protocols multiply attack surfaces. Each tool call trusts prior stages implicitly.

AI gives the old phrase "inside your adversary's OODA loop" new meaning. For Boyd's fighter pilots, it meant that you were operating faster than your adversary, able to act on current data while they were still on the previous iteration. With agentic AI, adversaries aren't just metaphorically inside; they're literally providing the observations and manipulating the output. We want adversaries inside our loop because that's where the data are. AI's OODA loops must observe untrusted sources to be useful. The competitive advantage, accessing web-scale information, is identical to the attack surface. The speed of your OODA loop is irrelevant when the adversary controls your sensors and actuators.

Worse, speed can itself be a vulnerability. The faster the loop, the less time for verification. Millisecond decisions result in millisecond compromises.

The Source of the Problem

The fundamental problem is that AI must compress reality into model-legible forms. In this setting, adversaries can exploit the compression. They don't have to attack the territory; they can attack the map. Models lack local contextual knowledge. They process symbols, not meaning. A human sees a suspicious URL; an AI sees valid syntax. And that semantic gap becomes a security gap.

Prompt injection might be unsolvable in today's LLMs. LLMs process token sequences, but no mechanism exists to mark token privileges. Every solution proposed introduces new injection vectors: Delimiter? Attackers include delimiters. Instruction hierarchy? Attackers claim priority. Separate models? Double the attack surface. Security requires boundaries, but LLMs dissolve boundaries. More generally, existing mechanisms to improve models won't help protect against attack. Fine-tuning preserves backdoors. Reinforcement learning with human feedback **<AU: Kindly check that the expansion of RLHF is correct.>** adds human preferences without removing model biases. Each training phase compounds prior compromises.

This is Ken Thompson's "trusting trust" attack all over again.[3] Poisoned states generate poisoned outputs, which poison future states. Try to summarize the conversation history? The summary includes the injection. Clear the cache to remove the poison? Lose all context. Keep the cache for continuity? Keep the contamination. Stateful systems can't forget attacks, and so memory becomes a liability. Adversaries can craft inputs that corrupt future outputs.

This is the agentic AI security trilemma. Fast, smart, secure; pick any two. Fast and smart—you can't verify your inputs. Smart and secure—you check everything, slowly, because AI itself can't be used for this. Secure and fast—you're stuck with models with intentionally limited capabilities.

This trilemma isn't unique to AI. Some autoimmune disorders are examples of molecular mimicry—when biological recognition systems fail to distinguish self from nonself. The mechanism designed for protection becomes the pathology as T cells attack healthy tissue or fail to attack pathogens and bad cells. AI exhibits the same kind of recognition failure. No digital immunological markers separate trusted instructions from hostile input. The model's core capability, following instructions in natural language, is inseparable from its vulnerability. Or like oncogenes, the normal function and the malignant behavior share identical machinery.

Prompt injection is semantic mimicry: adversarial instructions that resemble legitimate prompts, which trigger self-compromise. The immune system can't add better recognition without rejecting legitimate cells. AI can't filter malicious prompts without rejecting legitimate instructions. Immune systems can't verify their own recognition mechanisms, and AI systems can't verify their own integrity because the verification system uses the same corrupted mechanisms.

In security, we often assume that foreign/hostile code looks different from legitimate instructions, and we use signatures, patterns, and statistical anomaly detection to detect it. But getting inside someone's AI OODA loop uses the system's native language. The attack is indistinguishable from normal operation because it *is* normal operation. The vulnerability isn't a defect—it's the feature working correctly.

Where to Go Next?

The shift to an AI-saturated world has been dizzying. Seemingly overnight, we have AI in every technology product, with promises of even more—and agents as well. So where does that leave us with respect to security?

Physical constraints protected Boyd's fighter pilots. Radar returns couldn't lie about physics; fooling them, through stealth or jamming, constituted some of the most successful attacks against such systems that are still in use today. Observations were authenticated by their presence. Tampering meant physical access. But semantic observations have no physics. When every AI observation is potentially corrupted, integrity violations span the stack. Text can claim anything, and images can show impossibilities. In training, we face poisoned datasets and backdoored models. In inference, we face adversarial inputs and prompt injection. During operation, we face a contaminated context and persistent compromise. We need semantic integrity: verifying not just data but interpretation, not just content but context; not just information but understanding. We can add checksums, signatures, and audit logs. But how do you checksum a thought? How do you sign semantics? How do you audit attention?

Computer security has evolved over the decades. We addressed availability despite failures through replication and decentralization. We addressed confidentiality despite breaches using authenticated encryption. Now we need to address integrity despite corruption.[4]

Trustworthy AI agents require integrity because we can't build reliable systems on unreliable foundations. The question isn't whether we can add integrity to AI but whether the architecture permits integrity at all.

AI OODA loops and integrity aren't fundamentally opposed, but today's AI agents observe the Internet, orient via statistics, decide probabilistically, and act without verification. We built a system that trusts everything, and now we hope for a semantic firewall to keep it safe. The adversary isn't inside the loop by accident; it's there by architecture. Web-scale AI means web-scale integrity failure. Every capability corrupts.

Integrity isn't a feature you add; it's an architecture you choose. So far, we have built AI systems where "fast" and "smart" preclude "secure." We optimized for capability over verification, for accessing web-scale data over ensuring trust. AI agents will be even more powerful—and increasingly autonomous. And without integrity, they will also be dangerous. ■

References

1. S. Willison, *Simon Willison's Weblog*, May 22, 2025. [Online]. Available: https://simonwillison.net/2025/May/22/tools-in-a-loop/
2. S. Willison, "Prompt injection attacks against GPT-3," *Simon Willison's Weblog*, Sep. 12, 2022. [Online]. Available: https://simonwillison.net/2022/Sep/12/prompt-injection/
3. K. Thompson, "Reflections on trusting trust," *Commun. ACM*, vol. 27, no. 8, Aug. 1984. [Online]. Available: https://www.cs.cmu.edu/~rdriley/487/papers/Thompson_1984_ReflectionsonTrustingTrust.pdf
4. B. Schneier, "The age of integrity," *IEEE Security Privacy*, vol. 23, no. 3, p. 96, May/Jun. 2025. [Online]. Available: https://www.computer.org/csdl/magazine/sp/2025/03/11038984/27COaJtjDOM

Barath Raghavan is a distinguished engineer at Fastly, San Francisco, CA 94107 USA, and is a professor of computer science at the University of Southern California, Los Angeles, CA 90007 USA. His research interests include network protocols, security and privacy, and distributed systems. Raghavan received a Ph.D. from the University of California, San Diego. Contact him at barathraghavan@gmail.com.

Bruce Schneier is a fellow and lecturer at the Harvard Kennedy School, Cambridge, MA 02138 USA, and the chief security technology officer at Inrupt Inc., Boston, MA 02210 USA. His research interests include cybersecurity, systems security, and reimagining democracy and capitalism. Schneier received an M.S. in computer science from American University. Contact him at schneier@schneier.com.