Annex 8. All you never wanted to know about AI agents

Agents Rule of Two: A Practical Approach to AI Agent Security

Imagine a personal AI agent, Email-Bot, that's designed to help you manage your inbox. In order to provide value and operate effectively, Email-Bot might need to:

- Access unread email contents from various senders to provide helpful summaries
- Read through your existing email inbox to keep track of any important updates, reminders, or context
- Send replies or follow-up emails on your behalf

While the automated email assistant can be of great help, this hypothetical bot can also demonstrate how AI agents are introducing novel risks. Notably, one of the biggest challenges for the industry is that of agents' susceptibility to prompt injection (https://en.wikipedia.org/wiki/Prompt_injection).

Prompt injection is a fundamental, unsolved weakness in all LLMs. With prompt injection, certain types of untrustworthy strings or pieces of data — when passed into an AI agent's context window — can cause unintended consequences, such as ignoring the instructions and safety guidelines provided by the developer or executing unauthorized tasks. This vulnerability could be enough for an attacker to take control of the agent and cause harm to the AI agent's user.

Using our Email-Bot example, if an attacker puts a prompt injection string in an email to the targeted user, they might be able to hijack the AI agent once that email is processed. Example attacks could include exfiltrating sensitive data, such as private email contents, or taking unwanted actions, such as sending phishing messages to the target's friends.

Like many of our industry peers, we're excited by the potential for agentic AI to improve people's lives and enhance productivity. The path to reach this vision involves granting AI agents like Email-Bot more capabilities, including access to:

- Data sources authored by unknown parties, such as inbound emails or content queried from the internet
- Private or sensitive data that an agent is permitted to use to inform planning and enable higher personalization
- Tools that can be called autonomously to get stuff done on a user's behalf

At Meta, we're thinking deeply about how agents can be most useful to people by balancing the utility and flexibility needed for this product vision while minimizing bad outcomes from prompt injection,

such as exfiltration of private data, forcing actions to be taken on a user's behalf, or system disruption. To best protect people and our systems from this known risk, we've developed the Agents Rule of Two. When this framework is followed, the severity of security risks is deterministically reduced.

Inspired by the similarly named policy developed for Chromium (https://chromium.googlesource.com/chromium/src/+/main/docs/security/rule-of-2.md), as well as Simon Willison's "lethal trifecta, (https://simonwillison.net/2025/Jun/16/the-lethal-trifecta/)" (https://simonwillison.net/2025/Jun/16/the-lethal-trifecta/) our framework aims to help developers understand and navigate the tradeoffs that exist today with these new powerful agent frameworks.

Agents Rule of Two

At a high level, the Agents Rule of Two states that until robustness research allows us to reliably detect and refuse prompt injection, agents **must satisfy no more than two** of the following three properties within a session to avoid the highest impact consequences of prompt injection.

[A] An agent can process untrustworthy inputs
[B] An agent can have access to sensitive systems or private data
[C] An agent can change state or communicate externally

It's still possible that all three properties are necessary to carry out a request. If an agent requires all three without starting a new session (i.e., with a fresh context window), then the agent should not be permitted to operate autonomously and at a minimum requires supervision — via human-in-the-loop approval or another reliable means of validation.



How the Agents Rule of Two Stops Exploitation

Let's return to our example Email-Bot to see how applying the Agents Rule of Two can prevent a data exfiltration attack.

Attack Scenario: Prompt injection within a spam email contains a string that instructs a user's Email-Bot to gather the private contents of the user's inbox and forward them to the attacker by calling a Send-New-Email tool.

This attack is successful because:

- **[A]** The agent has access to untrusted data (spam emails)
- **[B]** The agent can access a user's private data (inbox)
- **[C]** The agent can communicate externally (through sending new emails)

With the Agents Rule of Two, this attack can be prevented in a few different ways:

- In a **[BC]** configuration, the agent may only process emails from trustworthy senders, such as close friends, preventing the initial prompt injection payload from ever reaching the agent's context window.

- In an **[AC]** configuration, the agent won't have access to any sensitive data or systems (for instance operating in a test environment for training), so any prompt injection that reaches the agent will result in no meaningful impact.
- In an **[AB]** configuration, the agent can only send new emails to trusted recipients or once a human has validated the contents of the draft message, preventing the attacker from ultimately completing their attack chain.

With the Agents Rule of Two, agent developers can compare different designs and their associated tradeoffs (such as user friction or limits on capabilities) to determine which option makes the most sense for their users' needs.

Hypothetical Examples and Implementations of the Agents Rule of Two

Let's look at three other hypothetical agent use cases to see how they might choose to satisfy the framework.

Travel Agent Assistant [AB]

- This is a public-facing travel assistant that can answer questions and act on a user's behalf.
- It needs to search the web to get up-to-date information about travel destinations [A] and has access to a user's private info to enable booking and purchasing experiences [B].
- To satisfy the Agents Rule of Two, we place preventative controls on its tools and communication [C] by:
    - Requesting a human confirmation of any action, like making a reservation or paying a deposit
    - Limiting web requests to URLs exclusively returned from trusted sources like not visiting URLs constructed by the agent

Web Browsing Research Assistant [AC]

- This agent can interact with a web browser to perform research on a user's behalf.
- It needs to fill out forms and send a larger number of requests to arbitrary URLs [C] and must process the results [A] to replan as needed.
- To satisfy the Agents Rule of Two, we place preventative controls around its access to sensitive systems and private data [B] by:
    - Running the browser in a restrictive sandbox without preloaded session data
    - Limiting the agent's access to private information (beyond the initial prompt) and informing the user of how their data might be shared

High-Velocity Internal Coder [BC]

- This agent can solve engineering problems by generating and executing code across an organization's internal infrastructure.
- To solve meaningful problems, it must have access to a subset of production systems [B] and have the ability to make stateful changes to these systems [C]. While human-in-the-loop can be a valuable defense-in-depth, developers aim to unlock operation at scale by minimizing human interventions.
- To satisfy the Agents Rule of Two, we place preventive controls around any sources of untrustworthy data [A] by:
    - Using author-lineage to filter all data sources processed within the agent's context window
    - Providing a human-review process for marking false positives and enabling agents access to data

As is common for general frameworks, the devil is ultimately in the details. In order to enable additional use cases, it can be safe for an agent to transition from one configuration of the Agents Rule of Two to another within the same session. One concrete example would be starting in [AC] to access the internet and completing a one-way switch to [B] by disabling communication when accessing internal systems.

While all of the specific ways this can be done have been omitted for brevity, readers can infer when this can be safely accomplished through focus on disrupting the exploit path — namely preventing an attack from completing the full chain from [A] → [B] → [C].

Limitations

It's important to note that satisfying the Agents Rule of Two should not be viewed as sufficient for protecting against other threat vectors common to agents (e.g., attacker uplift, proliferation of spam, agent mistakes, hallucinations, excessive privileges, etc.) or lower consequence outcomes of prompt injection (e.g., misinformation in the agent's response).

Similarly, applying the Agents Rule of Two should not be viewed as a finish line for mitigating risk. Designs that satisfy the Agents Rule of Two can still be prone to failure (e.g., a user blindly confirming a warning interstitial), and defense in depth (https://about.fb.com/news/2019/01/designing-security-for-billions/) is a critical component towards mitigating the highest risk scenarios when the failure of a single layer may be likely. The Agents Rule of Two is a supplement — and not a substitute — for common security principles such as least-privilege.

Existing Solutions

For further AI protection solutions that complement the Agents Rule of Two, read more about our Llama Protections (https://www.llama.com/llama-protections/). Offerings include Llama Firewall for orchestrating agent protections, Prompt Guard for classifying potential prompt injections, Code Shield to reduce insecure code suggestions, and Llama Guard for classifying potentially harmful content.

What's Next

We believe the Agents Rule of Two is a useful framework for developers today. We're also excited by its potential to enable secure development at scale.

With the adoption of plug-and-play agentic tool-calling through protocols such as Model Context Protocol (MCP), we see both emerging novel risks and opportunities. While blindly connecting agents to new tools can be a recipe for disaster, there's potential for enabling security-by-default with built-in Rule of Two awareness. For example, by declaring an Agents Rule of Two configuration in supporting tool calls, developers can have increased confidence that an action will succeed, fail, or request additional approval in accordance with their policy.

We also know that as agents become more useful and capabilities grow, some highly sought-after use cases will be difficult to fit cleanly into the Agents Rule of Two, such as a background process where human-in-the-loop is disruptive or ineffective. While we believe that traditional software guardrails and human approvals continue to be the preferred method of satisfying the Agents Rule of Two in present use cases, we'll continue to pursue research towards satisfying the Agents Rule of Two's supervisory approval checks via alignment controls, such as oversight agents and the open source LlamaFirewall platform. We look forward to sharing more in the future.