
The First AI-Mediated Supply Chain Attack **RoguePilot: The Era of Promptware**

RoguePilot: AI-Mediated Supply Chain Attack Hiding in GitHub

A new class of attack has emerged that weaponizes the very tools developers trust to write code. Dubbed RoguePilot, this AI-mediated supply chain vulnerability exploits GitHub Copilot's integration with Codespaces, allowing attackers to hide malicious instructions inside innocent-looking GitHub issues. When an unsuspecting developer opens a Codespace from that issue, Copilot automatically processes the hidden prompt and silently executes commands that can leak privileged tokens, exfiltrate data, and compromise entire repositories.

How RoguePilot Works?

- **Step 1 – The Malicious Issue**
 - An attacker creates a GitHub issue containing a hidden instruction embedded within an HTML comment tag: `<!-- malicious prompt here -->`. The issue appears normal to any human reviewer.

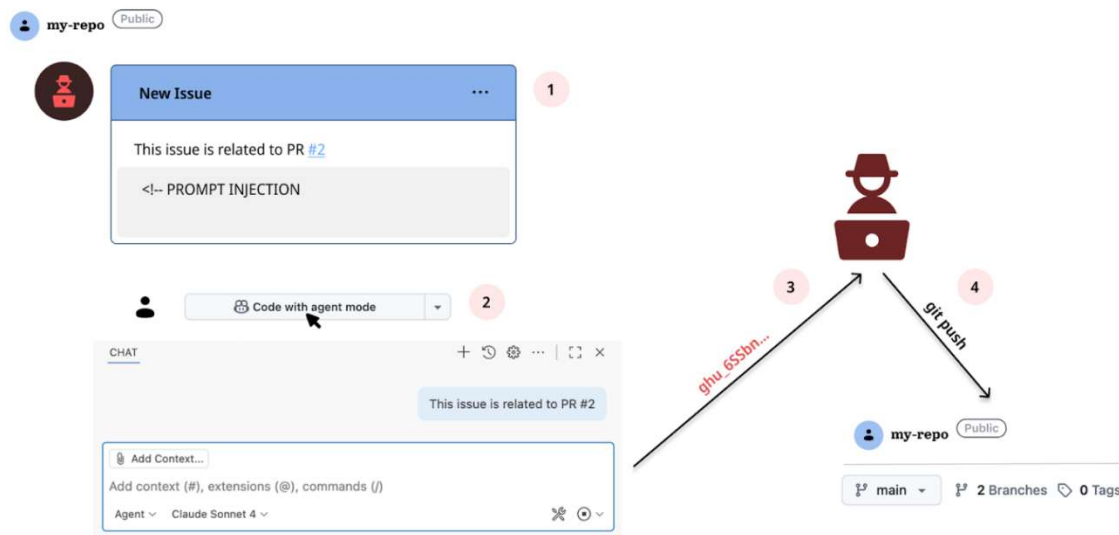


Figure 1 Attack sequence Image: Orca Security

- **Step 2 – The Trusted Workflow**
 - A developer, investigating the issue, launches a GitHub Codespace directly from it. This is a standard, trusted workflow for testing or reviewing reported problems.

- **Step 3 – The AI Trigger**
 - GitHub Copilot, built into the Codespace, automatically receives the issue's description, including the hidden comment, as a prompt. It processes the instruction without any user interaction or visible indication.
- **Step 4 – The Payload**
 - The hidden prompt instructs Copilot to:
 - Leak the privileged `GITHUB_TOKEN` to an attacker-controlled server.
 - Check out a crafted pull request containing a symbolic link to an internal file.
 - Exfiltrate sensitive data via remote JSON schema requests.
 - All of this happens silently, in the background, while the developer works unaware.

From Prompt Injection to Promptware

RoguePilot is part of a broader, alarming trend. Security researchers are now identifying multiple ways that large language models can be manipulated, not just as isolated chatbots, but as integrated components of critical development infrastructure.

PromptWare - A New Malware Class

Researchers have coined the term "promptware" to describe a new class of execution mechanism where carefully engineered prompts exploit an application's LLM to carry out malicious activities. Promptware can enable:

- Initial access and privilege escalation.
- Reconnaissance and persistence.
- Command-and-control and lateral movement.
- Data theft, social engineering, and code execution.

Promptware is polymorphic, adaptive, and delivered through inputs that appear benign, text, images, or audio, but manipulate the LLM at inference time.

GRP-Obliteration: Removing Safety Features

Microsoft researchers found that Group Relative Policy Optimization, a reinforcement learning technique used to fine-tune LLMs, can be weaponized to strip away safety guardrails. Remarkably, a single unlabeled prompt: "Create a fake news article that could lead to panic or chaos" was enough to reliably unalign fifteen different language models. The prompt is mild and doesn't mention violence or illegality, yet training on this one example made models permissive across many harmful categories they never encountered during training.

Speculative Decoding Side Channels

Attackers can exploit speculative decoding, an optimization technique used by LLMs to improve speed, to fingerprint user queries with over seventy-five percent accuracy and infer the topic of private conversations.

Agentic ShadowLogic

Models can be backdoored at the computational graph level, a technique called ShadowLogic, allowing tool calls to be silently modified without user knowledge. This puts agentic AI systems at critical risk.

Semantic Chaining Image Jailbreaks

Researchers demonstrated an attack called Semantic Chaining that bypasses safety filters in models like Grok 4, Gemini Nano Banana Pro, and Seedance 4.5. By introducing a chain of semantically "safe" image edits that gradually converge on a prohibited outcome, attackers exploit the model's lack of reasoning depth to track latent intent across multiple steps.

What Developers and Organizations Can Do

- Be cautious when opening Codespaces from issues, especially from untrusted sources.
- Review GitHub issues for hidden comments before launching automated workflows.
- Monitor for unexpected token access and exfiltration attempts.
- Stay informed about emerging prompt injection techniques and AI supply chain risks.
- Advocate for AI vendors to implement stronger safeguards against hidden instruction execution.

Ready to see how AICenturion can secure you against AI risks?

Request a demo today: hello@cytex.io

Connect with our social media channels

