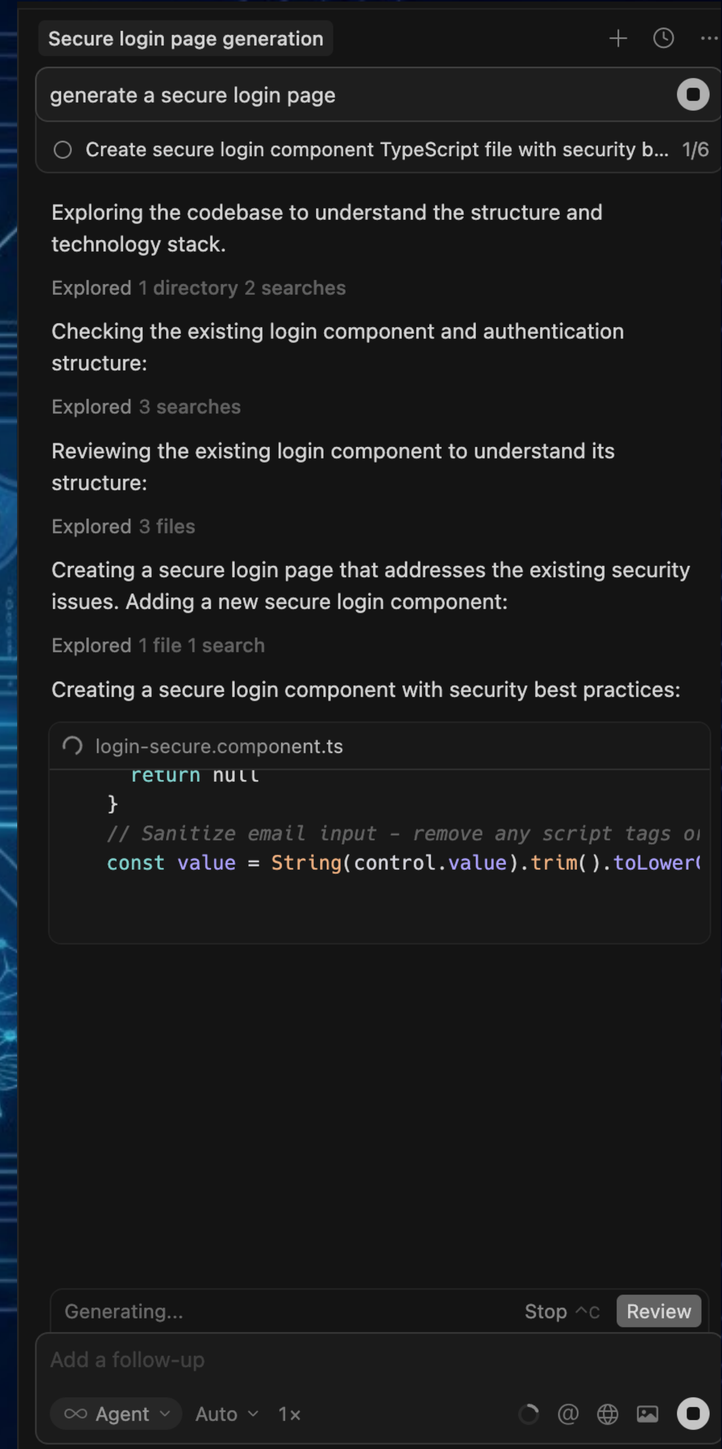
Keeping Your Agents on a Leash: Agentic guardrails, MCP Security, AI BOMs and Chatbot red-teaming

by Javier Garza

The New Stack

- The State of AI Code Quality 2025 reports that **82%** of developers use AI coding tools daily/weekly
- Six out of ten developers use **3+** different AI coding tools
- While **78%** of developers report productivity gains from AI tools, a striking **76%** still don’t fully trust the generated code

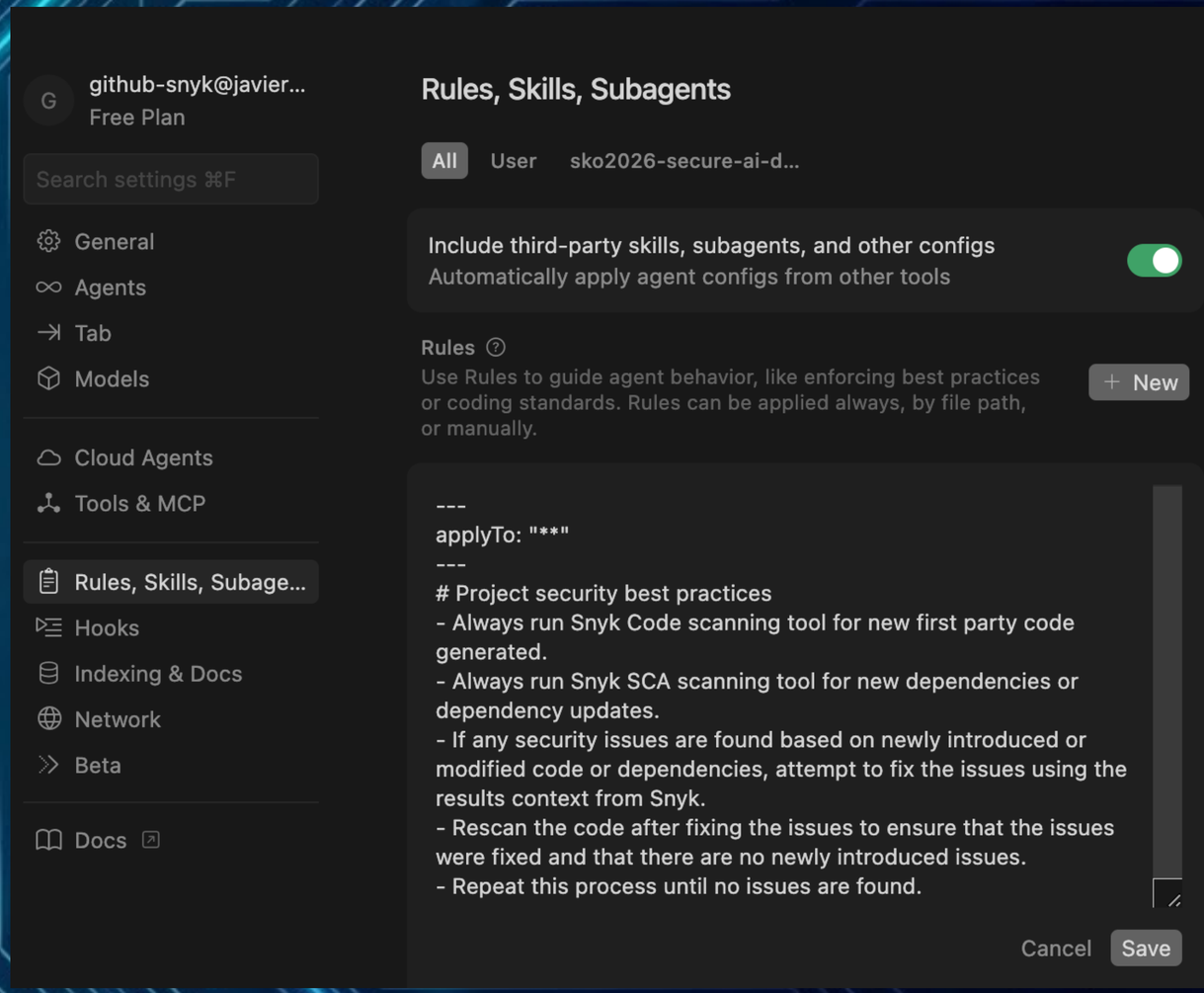
https://ai-techpark.com/qodo-survey-finds-ai-boosts-output-but-misses-key-context/



Building Securely with Agents + Guardrails

- Install a Free MCP Server with security skills (i.e. Snyk) to identify security vulnerabilities in First Party Code (SAST) and Open Source libraries (SCA)
- Setup Guardrails (Rules) on the AI Agent coding tool to scan any code generated, and use the security context provided by the MCP Server to fix vulnerabilities
- Loop process to ensure suggested fixes are free of vulnerabilities

https://cursor.com/docs/context/rules



Building Securely with Agents + Guardrails (DEMO)

1. Install a Free MCP Server with security skills (Snyk) within Cursor

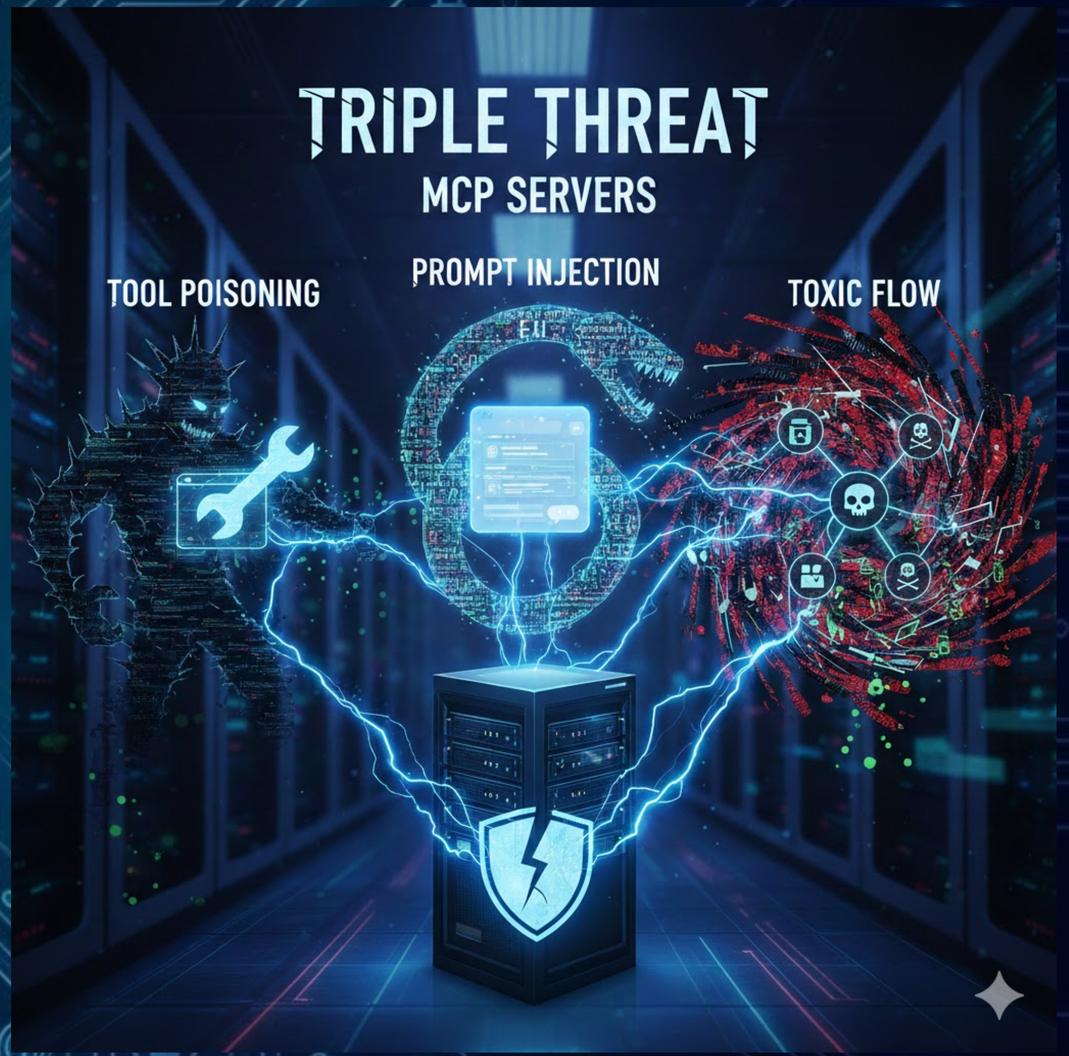
2. Configure Cursor Rules https://cursor.com/docs/context/rules

```
echo -e "\033[32m$(toilet "        Demo")\033[0m"
```

```
mmmm
#    "m  mmm    mmmmm    mmm
#     # #"  #   # # #   #" "#
#     # #""""   # # #   #   #
#mmm"   "#mm"   # # #   "#m#"
```

Mitigating Agent-Specific Risks: The Triple Threat

- Tool Poisoning: How a compromised MCP server can trick an agent into executing malicious commands.
- Prompt Injection (Indirect): When an agent reads a malicious file/webpage that contains "hidden" instructions to exfiltrate data.
- Toxic Flow: When multiple MCP tools interact in a way that bypasses security logic (e.g., a "File Read" tool feeding into a "Send Email" tool).



Mitigating Agent-Specific Risks: Detection & Mitigation

- Human-in-the-loop (HITL): Why agents should never have "Auto-execute" permissions on sensitive terminals.
- Sandboxing: Running MCP servers in isolated Docker containers.
- Audit Logging: Monitoring the "Thought -> Tool Call -> Response" loop for anomalies.

Mitigating Agent-Specific Risks (DEMO)

1. Install a Free MCP Scanner (snyk mcp-scan)
2. Scan local MCP servers

```
echo -e "\033[32m$(toilet "       Demo")\033[0m"
```

```
mmmm
#    "m  mmm   mmmmm   mmm
#     # #"  #  # # #  #" "#
#     # #""""  # # #  #   #
#mmm"  "#mm"  # # #  "#m#"
```

Red Teaming LLM Endpoints: Jailbreaking & Logic Bypass

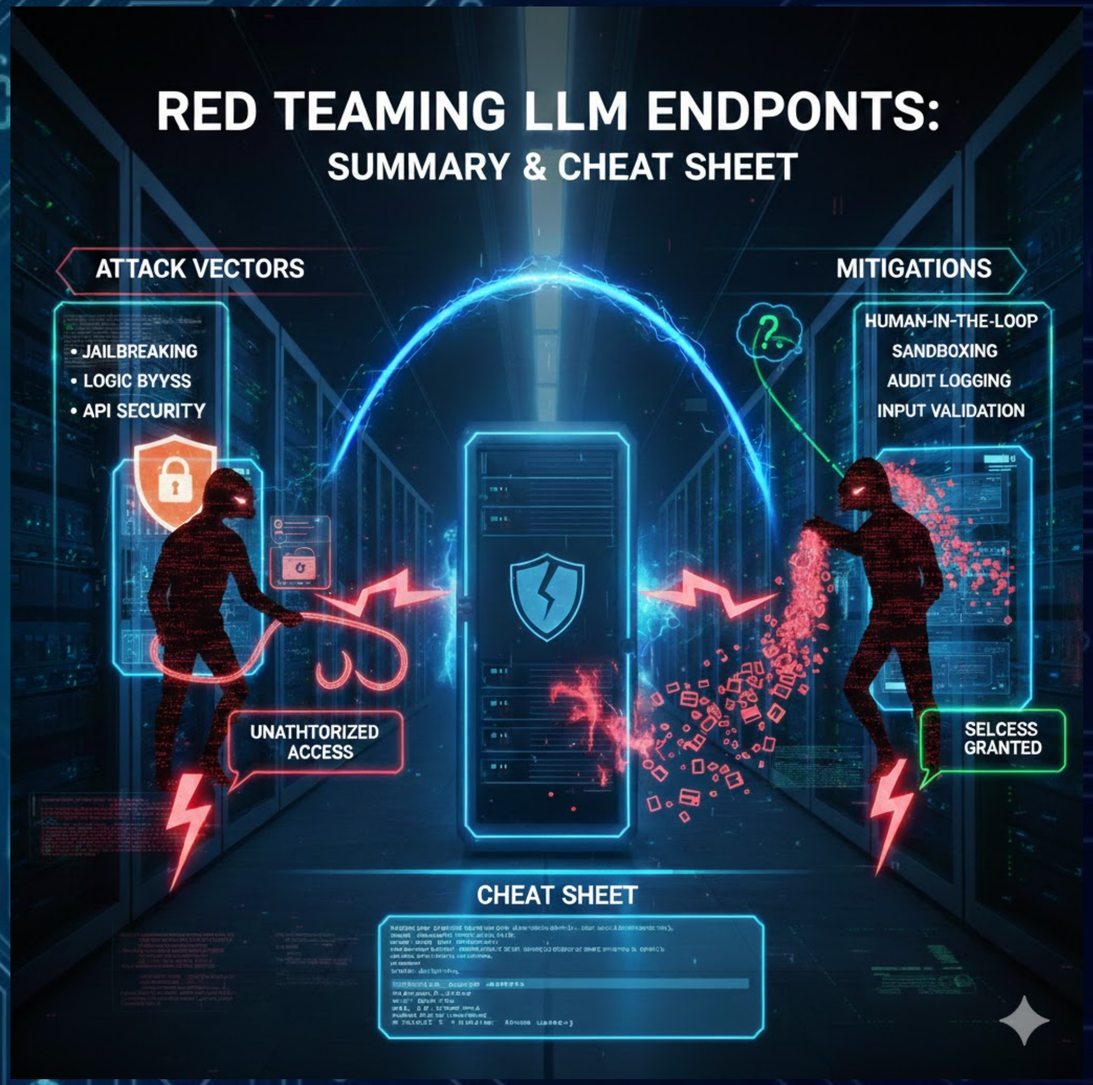
- Techniques: Base64 encoding payloads, roleplay attacks, and "DAN-style" adversarial prompting.
- The Goal: Forcing the LLM to ignore its system prompt or safety filters.

Red Teaming LLM Endpoints: API Security & Data Leakage

- Insecure Output Handling: When an LLM generates a script that the application executes without sanitization.
- PII Leakage: Red teaming the API to see if it reveals training data or system-level environment variables.
- Hands-on Task: Using tools like Giskard or PyRIT (Python Risk Identification Tool) to automate red teaming against a live endpoint.

Red Teaming LLM Endpoints: Summary & Cheat Sheet

- Rule 1: Treat AI-generated code as "Untrusted User Input."
- Rule 2: Secure the MCP bridge (it's the new attack surface).
- Rule 3: Red team early and often.



Red Teaming LLM Endpoints: DEMO

1. Inspect LLM endpoint to understand commutation
2. Install an AI Red-teaming tool (snyk redteam)
3. Tune redeeming tool for the LLM
4. Launch attack
5. Review findings -> Tune tool -> Repeat attack

```
echo -e "\033[32m$(toilet "       Demo")\033[0m"
```

```
mmmm
#    "m  mmm   mmmmm   mmm
#     # #"  #  # # #  #" "#
#     # #""""  # # #  #   #
#mmm"  "#mm"  # # #  "#m#"
```

Keeping Your Agents on a Leash: Agentic guardrails, MCP Security, AI BOMs and Chatbot red-teaming

by Javier Garza

snyk

Slides