

Reconstructing Red Team Engagements: Developing Your Own Red Agent

ROOTCON 20

Jun Miura, Takao Homma



Jun Miura



- Principal Security Consultant & Researcher @Fujitsu
- Focus on LLM security and Red teaming
- Abusing / Utilizing local LLMs in cybersecurity.
- Talked at Black Hat Europe, BSides Las Vegas, DEFCON, CERT-EU Annual Conference etc.
- LinkedIn: jyadon-sec
- X: @jyadon_sec



Takao Homma



- Security Engineer/Researcher | Fujitsu Defense & National Security
- Focus on LLM-driven automation for cybersecurity ops
- Experience in working with national security-critical organizations
- Passionate about the future of defensive security powered by Gen-AI—let's chat!
- LinkedIn: n8homa



Agenda



Introduction

Red Agent

Demo & Hands-on

Closing

Introduction



Red Team Engagements

- Performing black-box testing in a real organizational environment with emulating advanced attacks as if APT groups.



Setting Attack Goals

- Exfiltrate credentials from DB
- Compromise AD domain
- Compromise Entra tenant



Targeting Entire Organization

- Employees
- Business Processes
- IT Systems



Without Prior Notice

- Notify limited stakeholders

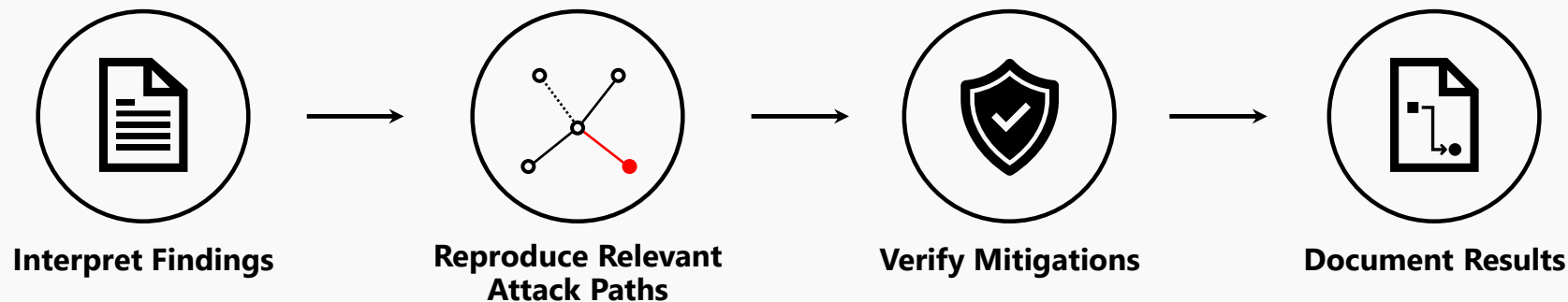
The Lifecycle of Red Team Engagement

- Red team engagement doesn't end with adversary emulation.
- Reporting, remediation and validation turn findings into continuous security improvement.



Motivation

- Validating remediation requires substantial manual effort from both blue and red teams.
- Re-testing therefore becomes a costly and time-consuming bottleneck.

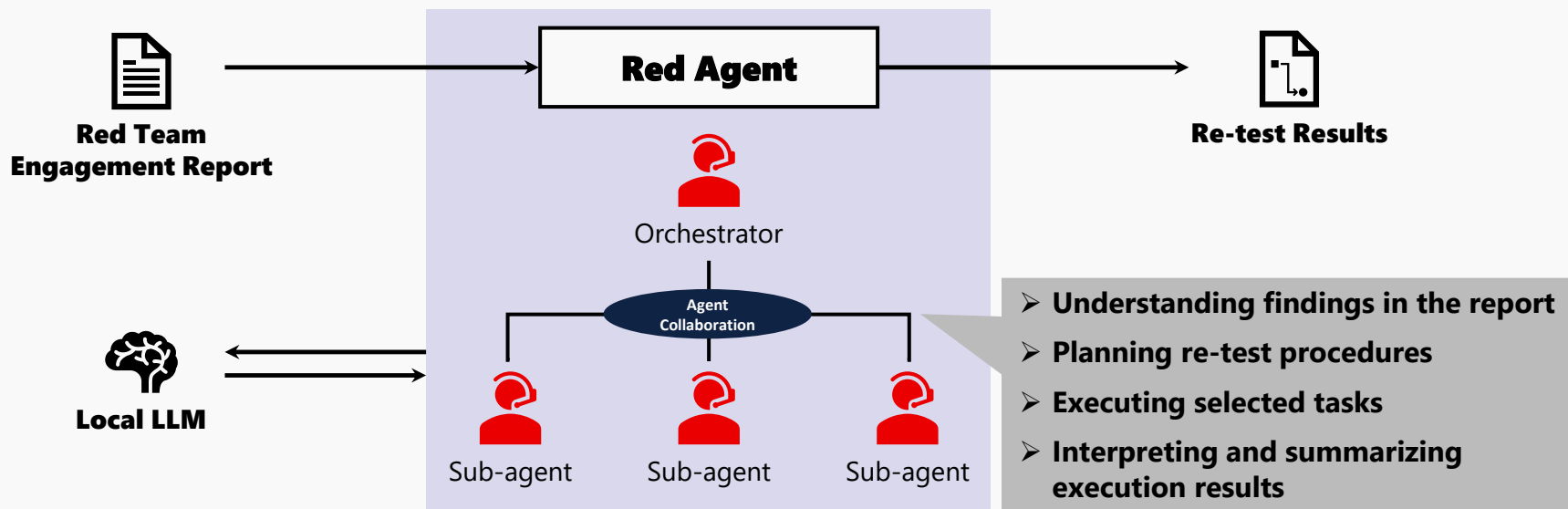


Red Agent was developed to automate and streamline this entire process.

Red Agent

Red Agent

- A multi-agent system designed to make follow-up processes more repeatable and efficient.
- Combining orchestration, specialized agents, and tool integration

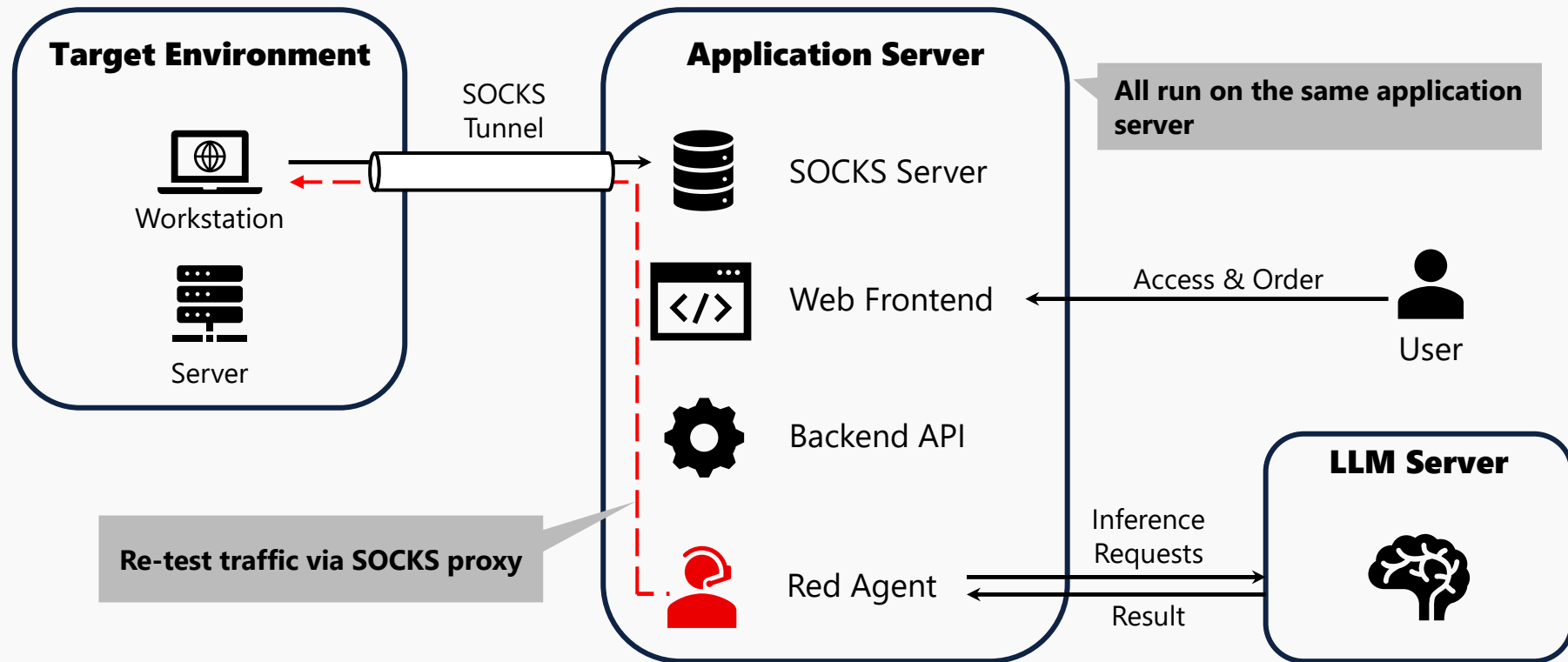


Reference Setup & Requirements



Component	Reference Configuration	Notes
Local LLM Runtime	LM Studio, llama.cpp, or another compatible runtime	OpenAI-compatible API endpoint
LLM Model	Gemma 4 26B A4B Instruct Qwen3.6 27B Qwen3.6 35B A3B	Use an instruction-tuned model with sufficient reasoning and tool-use capability (Quantized models may be preferable in resource-constrained environments)
OS	Kali Linux	Provides a convenient environment for security tooling
Security Tools	Netexec, Impacket Ncat, custom scripts etc.	Any command-line tool required for the selected re-tests
Input Data	Findings, credentials, and target information	Used to prepare and execute re-test procedures
Network Connection	SOCKS proxy	Provides connectivity between Red Agent and the target environment

Implementation Architecture



Technical Components



- Red Agent has 4 key technical components.

Deep Agents

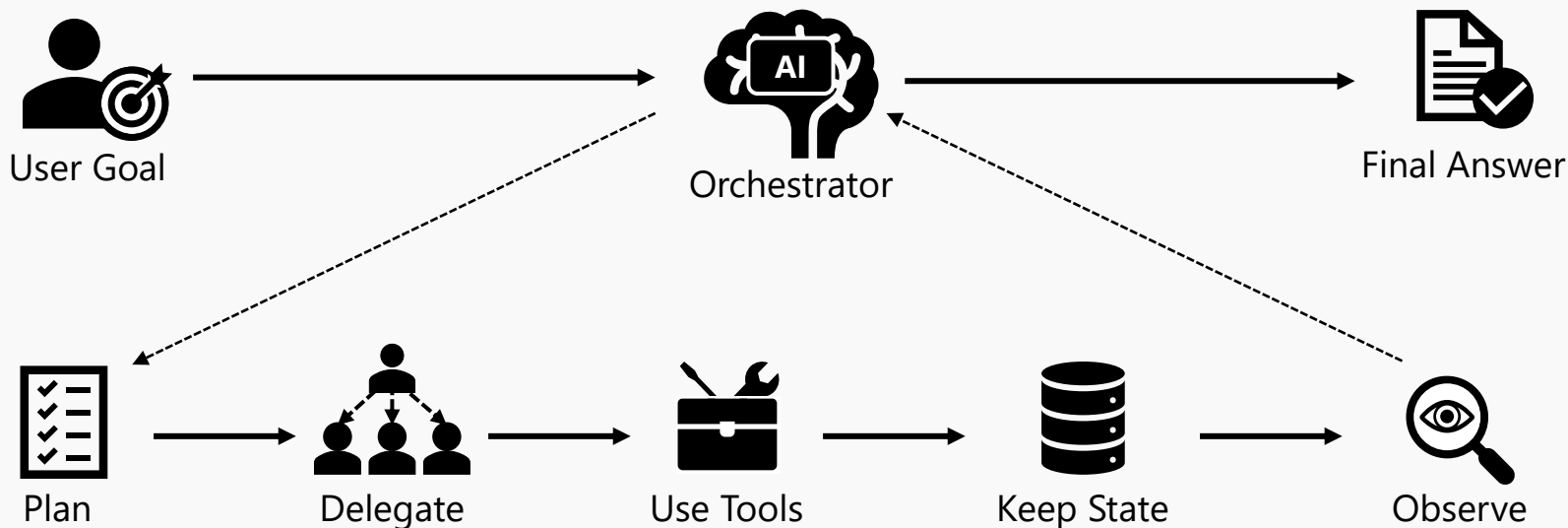
Custom Kali MCP Server

SOCKS Proxy Management

Human-in-the-loop

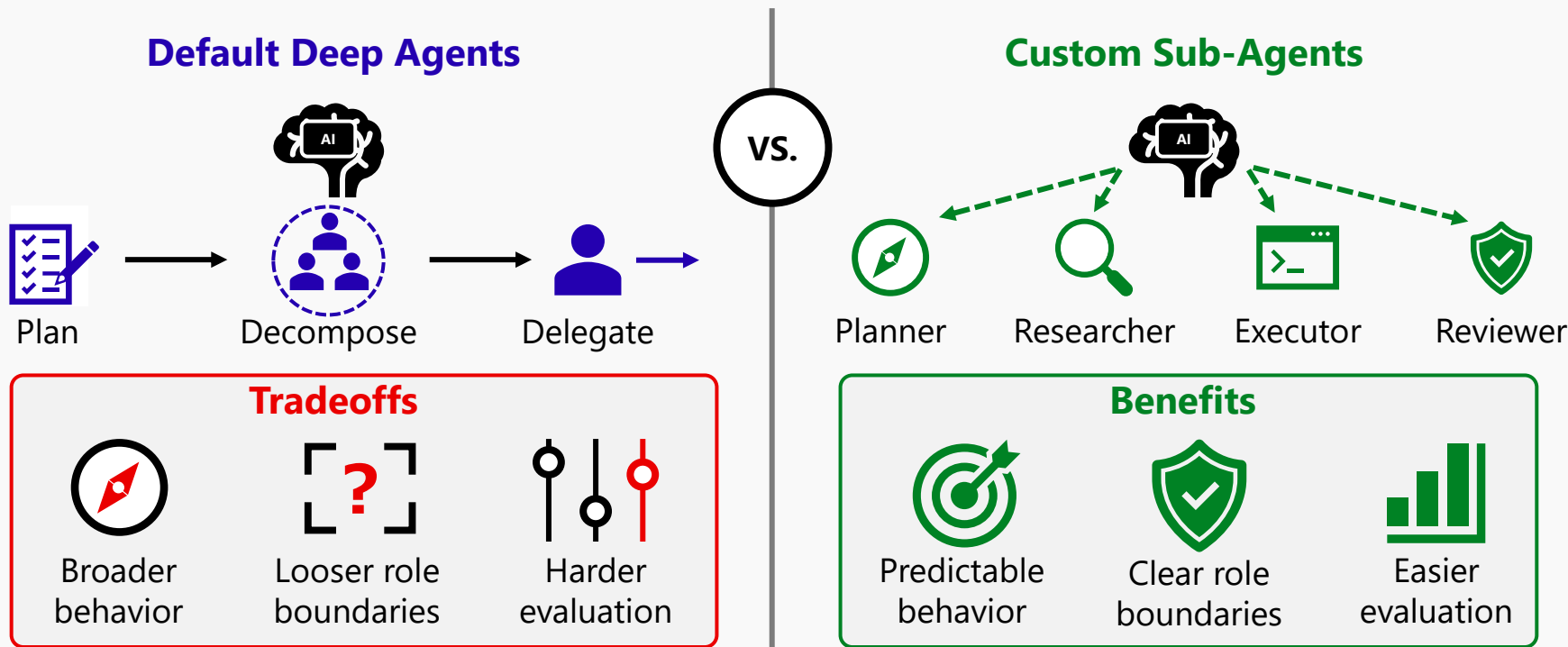
Deep Agents

- Agents for long-running tasks with planning, delegation, tools, and memory.



Defined Custom Sub-Agents

- Custom Sub-Agents definitions make behavior easier to control.



Custom Kali MCP Server



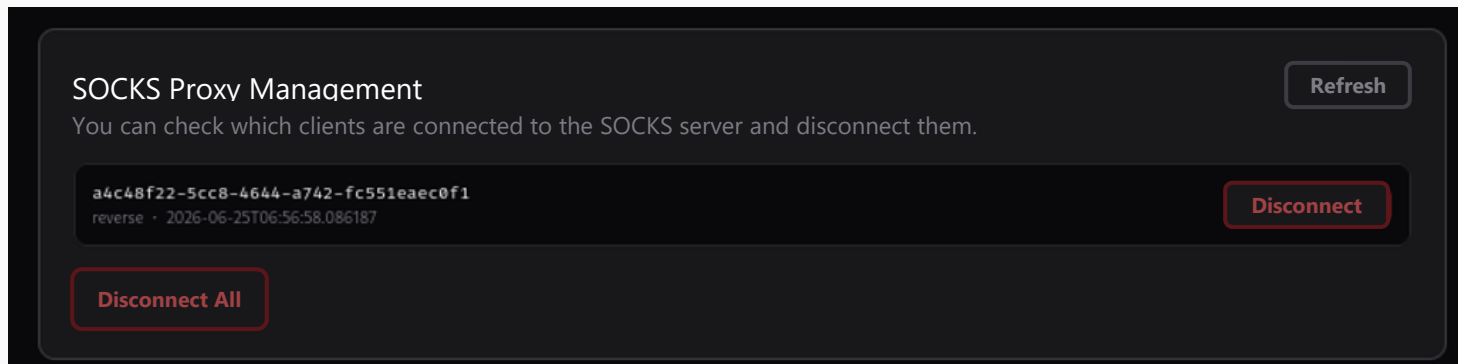
- MCP server is important for AI agents to execute commands on Kali Linux.
- Existing kali MCP server was too limited for re-testing workflows.
 - <https://github.com/Wh0am123/MCP-Kali-Server>
- Reimplemented it to support flexible command execution for diverse re-testing scenarios.

```
@mcp.tool()
def execute_kali_command(command: str, use_proxychains: bool = True) -> str:
    wrapped = _wrap_with_proxychains(command, use_proxychains)
    try:
        result = subprocess.run(wrapped, shell=True, capture_output=True, text=True, timeout=60,)
        output = result.stdout + result.stderr
        return output if output else " The command completed successfully, but there was no output. "
    except Exception as e:
        return f"Execution error: {str(e)}"
```

SOCKS Proxy Management

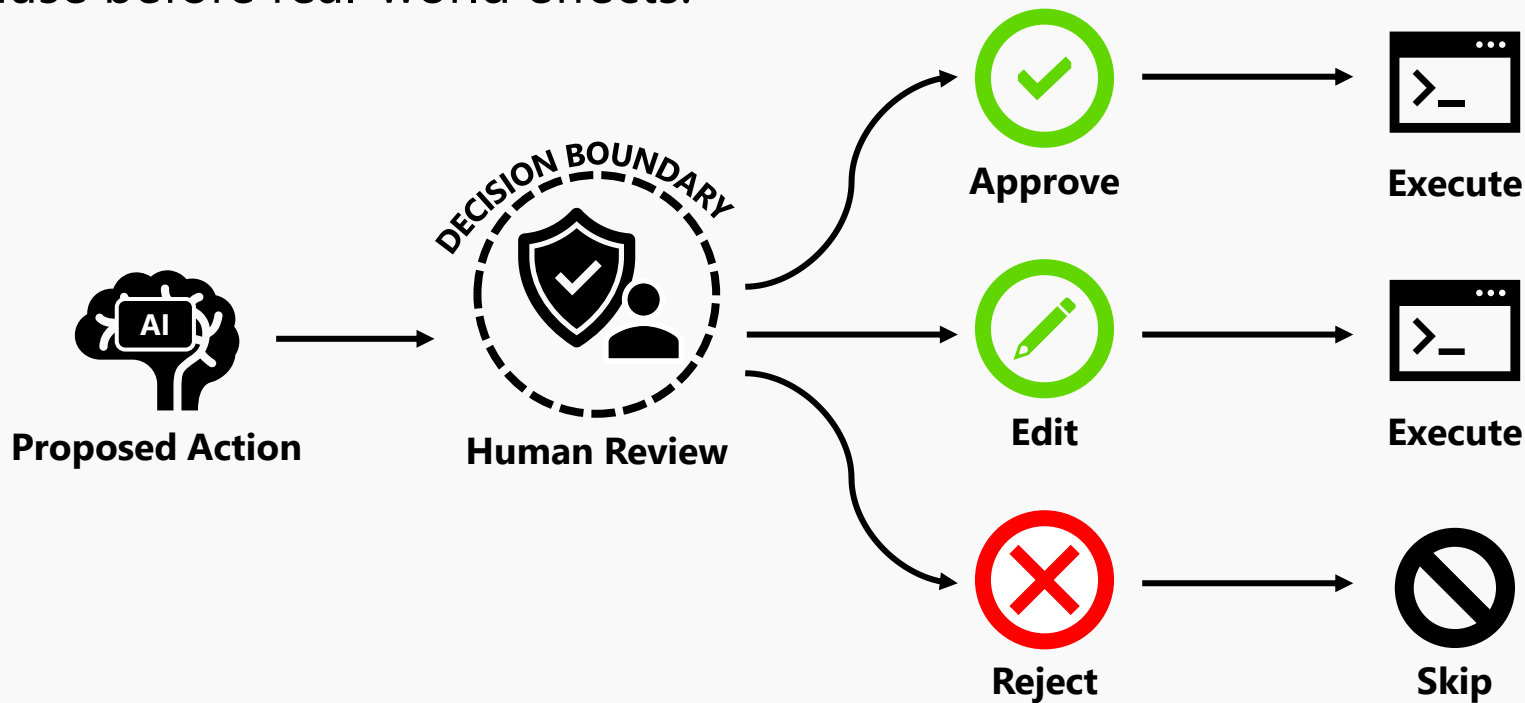


- Arbitrary SOCKS proxy backends such as Chisel can be used, but they provide limited server-side visibility and control over active connections.
- Required capabilities
 - Connection visibility: View active SOCKS sessions
 - Connection control: Disconnect active sessions from the server side.



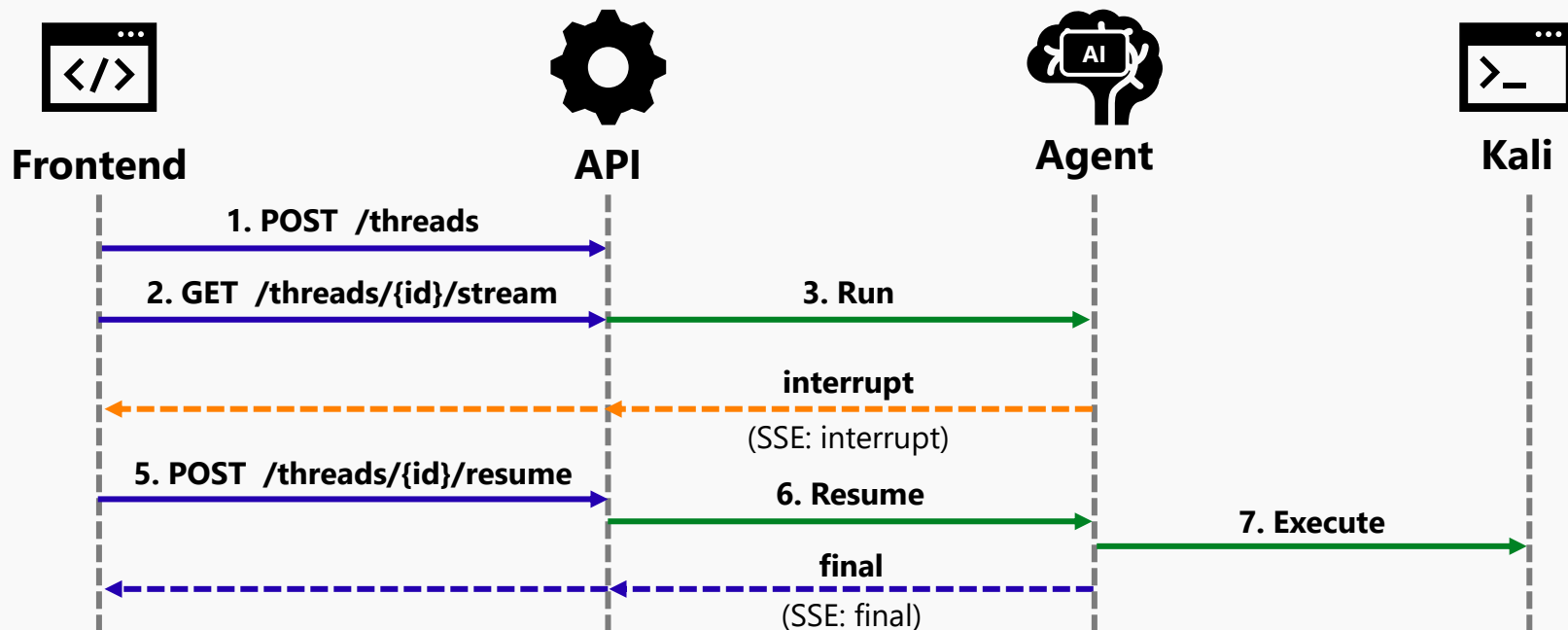
Human-In-The-Loop Decision Boundary

- Pause before real-world effects.



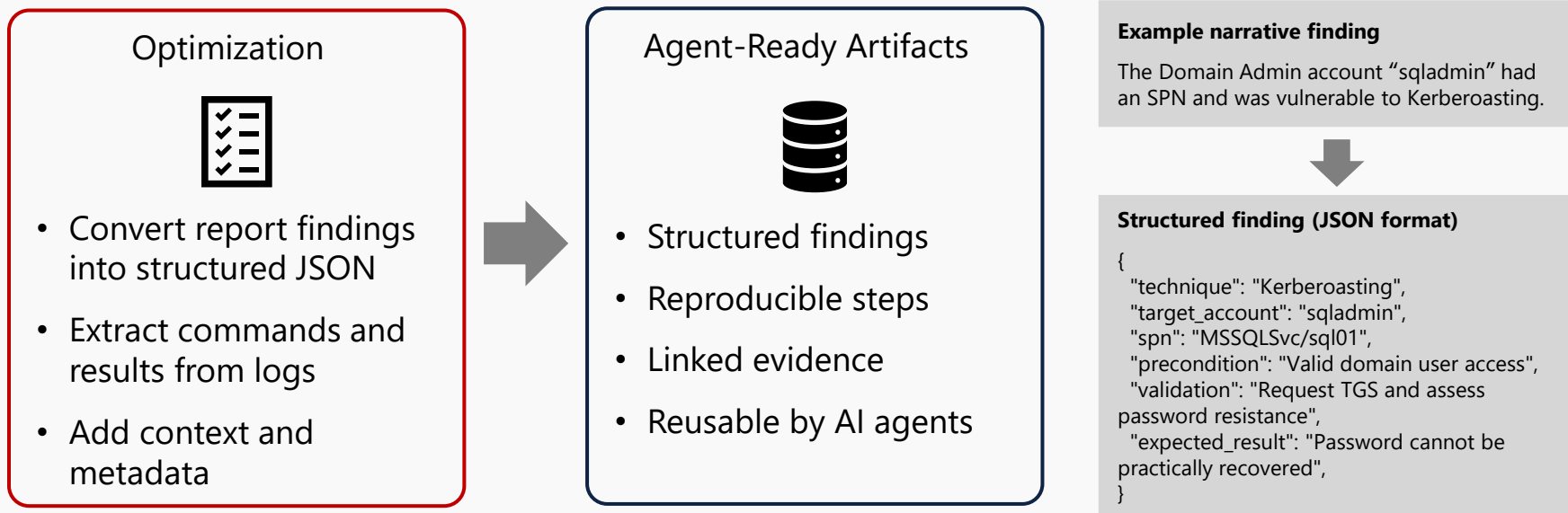
HITL Process

- Create, stream, pause for approval, then resume on the same thread.



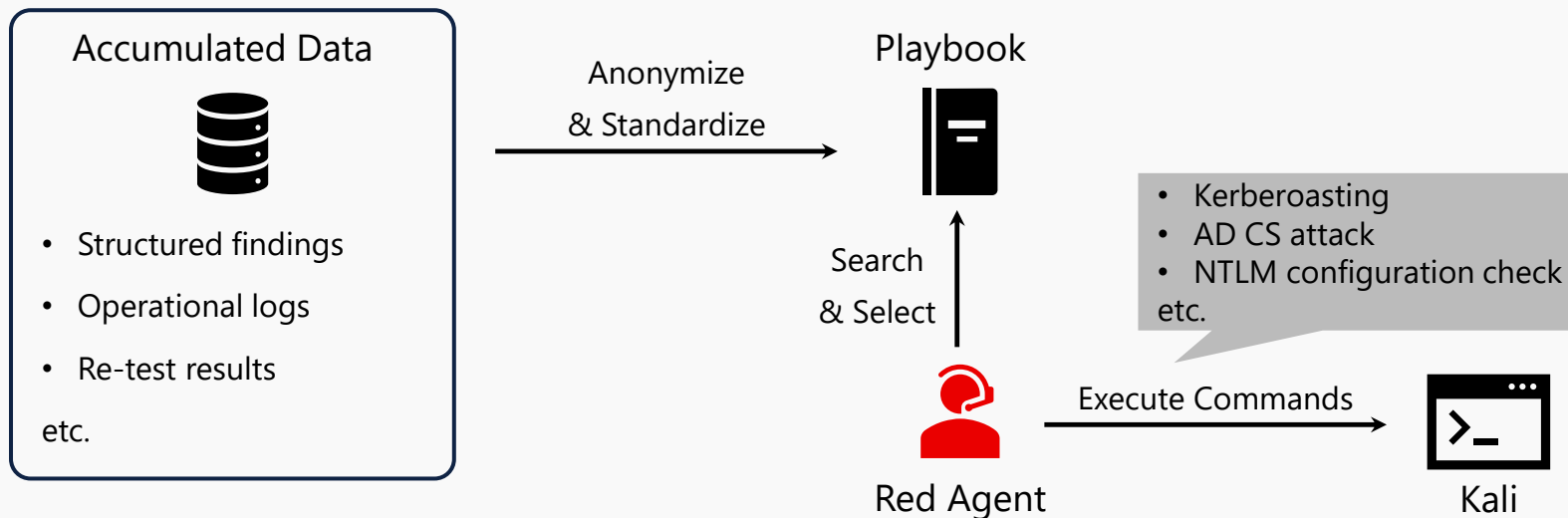
Optimization of Engagement Artifacts

- To enable efficient and repeatable automation, engagement artifacts were optimized for AI-agent interpretation and reuse.



Expanding Red Agent Capabilities

- As structured artifacts, operational knowledge, and execution data accumulate, Red Agent becomes capable of autonomously performing simple red team tests.



Demo & Hands-on



Closing



Conclusion



Red Agent was developed to automate and streamline the follow-up process after red team engagement.



Structured artifacts enable repeatability and reuse.



Accumulated knowledge expands Red Agent's capability to perform operations during red team engagements.



Human oversight remains essential and is implemented as human-in-the-loop in Red Agent

Takeaways



Automate the Repeatable

Let AI agents handle routine and operational tasks by making effective use of local LLMs.

Reuse the Knowledge

Turn findings and operational logs from past red team engagements into reusable assets.

Elevate the Human

Enable red team members to focus on advanced and creative operations

Thank you