#r2con2025

VibeReversingBinaries

with r2ai/decai/r2mcp

by @pancake@infosec.exchange

Who Am I?

Sergi Àlvarez aka **pancake**

- Mobile Security Research Engineer at NowSecure
- Author, leader and main contributor of Radare
- Free Software enthusiast and developer

AI EveryWhere

We are living through an explosion of LLM use cases — every software company is finding new ways to integrate AI to make tools smarter, easier, and more accessible.

- Expanding capabilities across all industries
- Lowering the entry barrier to new users
- Simplifying complex workflows
- Enhancing creativity and productivity

AI NoWhere

All those benefits don’t come for free and raises concerns.

- Training and running models require an insane amount of energy
- Drain natural resources like no other technology did before
- Privacy concerns, your data is no longer yours
- Common tools are not designed to run with small or local models
- Easy to drain your wallet pretty fast
- Learning to use this technology effectively requires practice
- You can’t trust the output

Bit of History

AI support in r2land started in 2022, we were the first to play with all those technologies, approaching them from the bottom.

Nowadays r2 provides several state of the art solutions to work with AI.

We Care About

- Privacy, User control and transparency
- Local models and low consumption
- Owning your data and hardware

R2AI - The origin

The very first implementation of r2ai was based on Python+R2Pipe using the llamacpp apis to run local models.

`https://github.com/radareorg/r2ai`

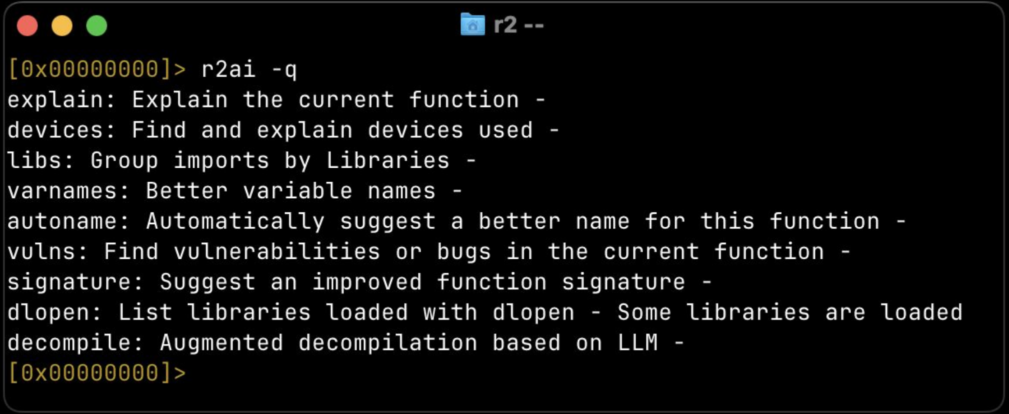
Problems solved since last year

- Models loaded in a separate process, new attention algorithms
- 4KB context sizes are now a thing of the past (256K now)
- Tool calling emulation (tool callings models are not mandatory)
- Python is slow and Vdb, Tools, Prompting was a mess to maintain

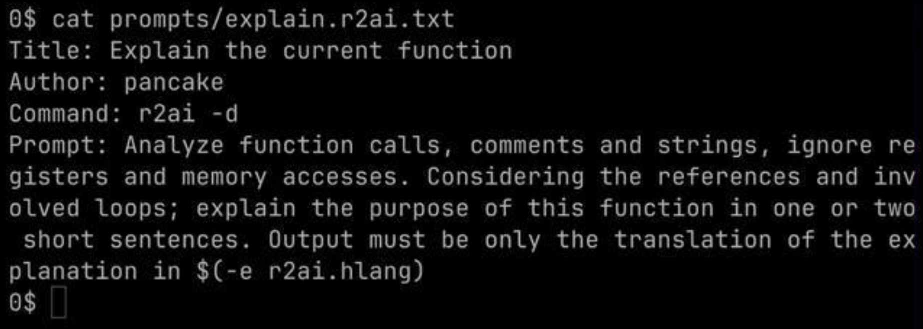
R2AI - Query Prompts

Extensible with custom prompts!

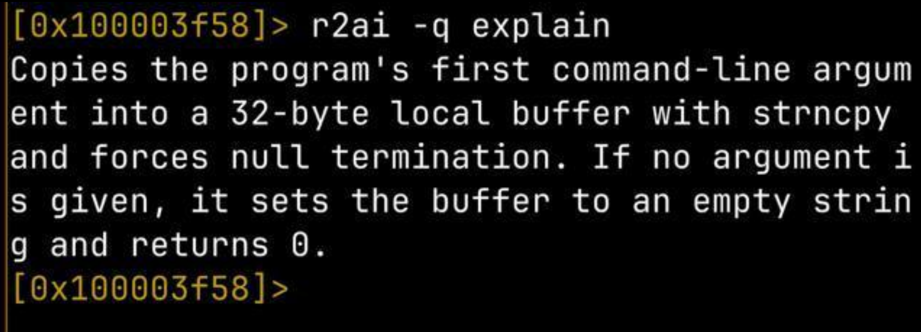
- Plaintext scripts with commands and prompts!
- Easy to script with oneliners



```
[0x00000000]> r2ai -q
explain: Explain the current function -
devices: Find and explain devices used -
libs: Group imports by Libraries -
varnames: Better variable names -
autoname: Automatically suggest a better name for this function -
vulns: Find vulnerabilities or bugs in the current function -
signature: Suggest an improved function signature -
dlopen: List libraries loaded with dlopen - Some libraries are loaded
decompile: Augmented decompilation based on LLM -
[0x00000000]>
```



```
0$ cat prompts/explain.r2ai.txt
Title: Explain the current function
Author: pancake
Command: r2ai -d
Prompt: Analyze function calls, comments and strings, ignore re
gisters and memory accesses. Considering the references and inv
olved loops; explain the purpose of this function in one or two
 short sentences. Output must be only the translation of the ex
planation in $(-e r2ai.hlang)
0$ 
```



```
[0x100003f58]> r2ai -q explain
Copies the program's first command-line argum
ent into a 32-byte local buffer with strncpy
and forces null termination. If no argument i
s given, it sets the buffer to an empty strin
g and returns 0.
[0x100003f58]>
```

R2AI - Tool Calling without Tool Models

Tool calling is not available in all models. But we can emulate it!

```
r2ai -e r2ai.auto.raw = true
```

Lessons learned from **MAI**, all models can use grammars, but not all of them follow the instructions described when the query is long.

- Submit a prompt with the TASK, TOOLS and expect ACTION
- Iterate until problem solved.

Known as “auto” mode (r2ai –a) but it’s also named “reson+act loops”

R2AI - Demo

Basic setup and introductory session:

- Installing r2ai
- Launching r2ai from the CLI
- Launch some queries with r2ai -q
- Calling tools without Tools models!

Decai - Decompiling with AI

An **r2js** plugin with focus on decompilation features.

- Playground that helped redesign the old r2ai
    - Javascript is easier to **experiment** and modify
    - Easy iterations and fun to modify and hack
- **Combine** the output of N decompilers
- Supports **‘auto’** mode, without requiring tool models
- Support reasoning models

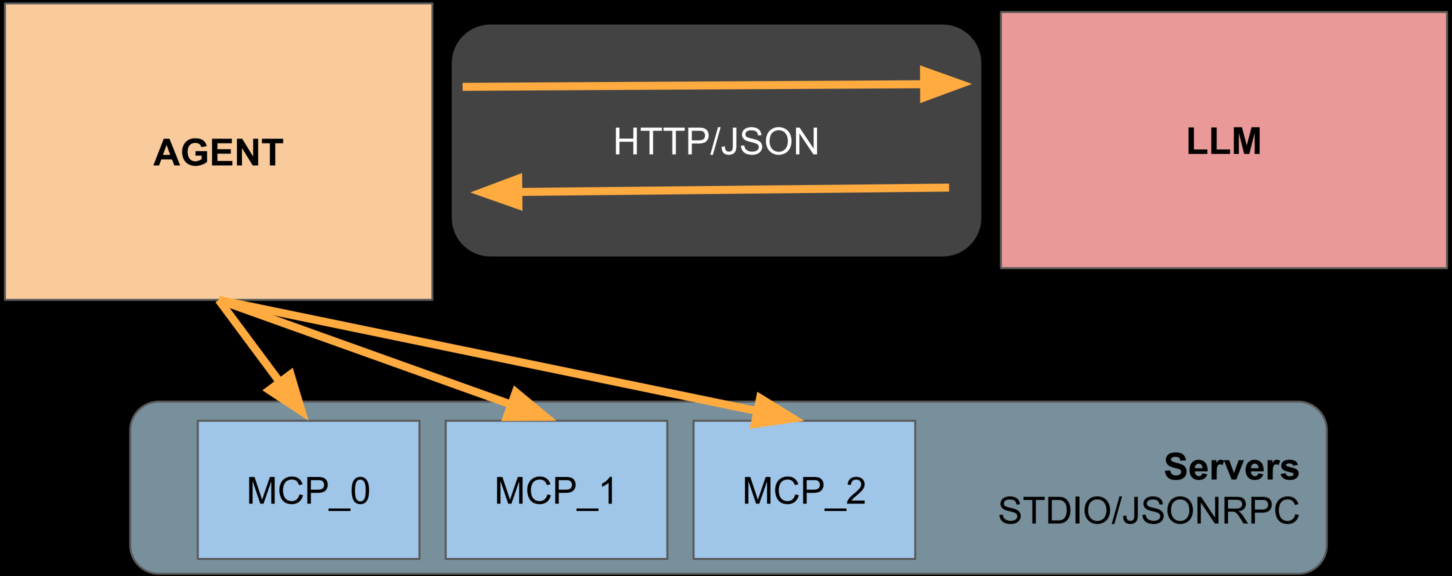
In one line: `r2pm -ci decai; r2 -qAc ‘decai -d’ /bin/ls`

Decai logo?

Model Context Protocol

Open JSONRPC Standards defined by Anthropic

Extend agents with **resources**, **prompts** and **tools**



Setting up MCPs in your favourite editor

Claude Code, OpenCode, Visual Studio Code, .. even MAI

Use the very same mcpServers json configuration:

```json
{
  "$schema": "https://opencode.ai/config.json",
  "mcp": {
    "r2mcp": {
      "type": "local",
      "command": [
        "r2pm", "-r", "r2mcp", "-m", "-u", "http://localhost:9090/cmd"
      ],
      "enabled": true
    }
  }
}
```

r2mcp - The Radare2 MCP Server

- Comunicates like inetd using **JSONRPC** via stdio or tcp
  - Uses radare2 C API or connects to a remote r2pipe
- Works with any Agent!
  - Cursor, VSCode, OpenCode, Mai, …

Security features

- Mini mode (expose less tools)
- Sandbox and **readonly** modes
- SuperVisor console!

- MiniMode
- DSL/Tests
- ExecuteJS
- Logging
- Scripts

r2copilot

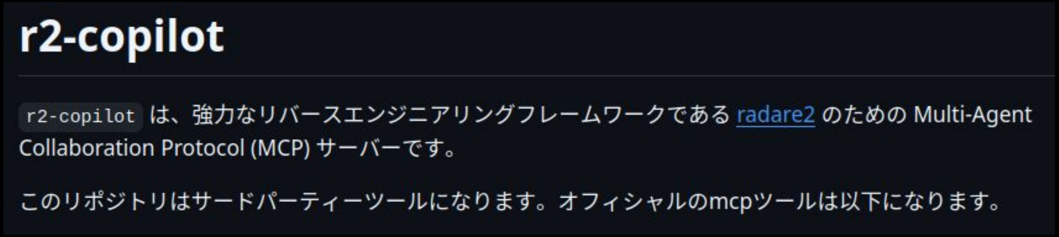
Another MCP written in Python using r2pipe

```
r2pm -Uci r2-copilot
```

- Written by darallium with focus on solving CTF challenges
- Support multiple files / sessions
- Security-related features like ROP search, binary patching, debugging

```
$ mai-wmcp -t ./r2-copilot/start.sh | grep ^ToolName |wc -l

43
```



r2-copilot

`r2-copilot` は、強力なリバースエンジニアリングフレームワークである radare2 のための Multi-Agent Collaboration Protocol (MCP) サーバーです。

このリポジトリはサードパーティーツールになります。オフィシャルのmcpツールは以下になります。

R2MCP - Demo Time

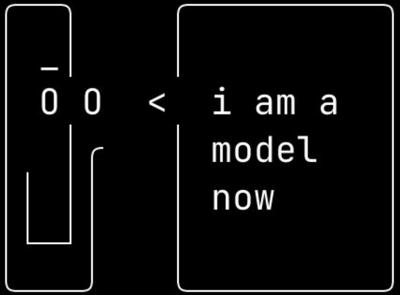
Setting up r2mcp with Mai

- Using local models
- The mai-wmcp proxy
- SuperVisor Mode

Extending Coding agents like OpenCode with r2mcp

- Configuration ~/.config/opencode/config.json

r2ai-model - The Foundation Dataset



Dataset to finetune models to instruct them to use r2.

https://github.com/radareorg/r2ai-model

1. Q/A format by AI and reviewed by humans
2. AI reviewing statements and commands
3. FineTuned models and repeat 1+2
4. Create reasoning plans
5. Extract book statements, improve documentation
6. Run r2 oneliners to verify everything
7. Compare source code to learn and suggest improvements
8. **Total World Domination**

Current Stage

HONORABLE MENTION: MAI

Stands for “My Artificial Intelligence”

LLM Framework providing cli tools, mcp library, GUI and more

https://github.com/trufae/mai

- Run fancy oneliners
- Interactive and powerful REPL
- Integrated with VIM
- Tool calling without grammars
- Jailbreak models with raw messages

Questions?

Reach me out at @pancake@infosec.exchange