R2AI:

Applying Language Models for Reverse Engineering

@pancake@infosec.exchange // HackBCN'2025

Who Am I?

Sergi Àlvarez i Capilla
Also known as "**pancake**"

Catalan Hacker from Barcelona
Author and main developer of radare2

Mobile Security Research Engineer

Working at NowSecure:
Saving the world from insecure apps

Free Software Hacker and Developer
UNIX Friendly and Reverser

Reach me out in the Fediverse
@pancake@infosec.exchange

Using Language Models for Reverse Engineering

1 Introduction to AI

2 Setting up R2AI

3 Features and Limitations

4 Decompiling examples

5 Agentic Reversing

6 Demo and Q&A time!

Introduction to AI

Applying Language Models for Reverse Engineering with Radare2

Nowadays we can find AI integrated everywhere

- Growing interest in new use cases.

Radare2 got LLM support in 2023

- Here we focus on large language models
- Autocomplete text using natural language
- Finding the right model
- Using local or remote inference engines

1 Models

2 Providers

3 Prompt

Models

Applying Language Models for Reverse Engineering with Radare2

Selecting the right model for your task is key.

- Very competitive and evolving fast
- Newer is not better, always test!

My recommendations:

- Qwen2.5, Granite, Gemma

Cloud-privative solutions:

- Claude, OpenAI, Gemini, Mistral, Grok

chat | code

uncensor | thinking

Parameter Size

7B | 12B | 32B

API Providers

- Anthropic's Claude
- OpenAI's GPT-4o
- Google's Gemini
- X-AI's Grok
- Mistral's
- ...

Ollama

Easy to use llama cpp server

OpenAPI

Curl https/json endpoints

r2ai-server

Manage models, select inference..

Model Types

chat **code**

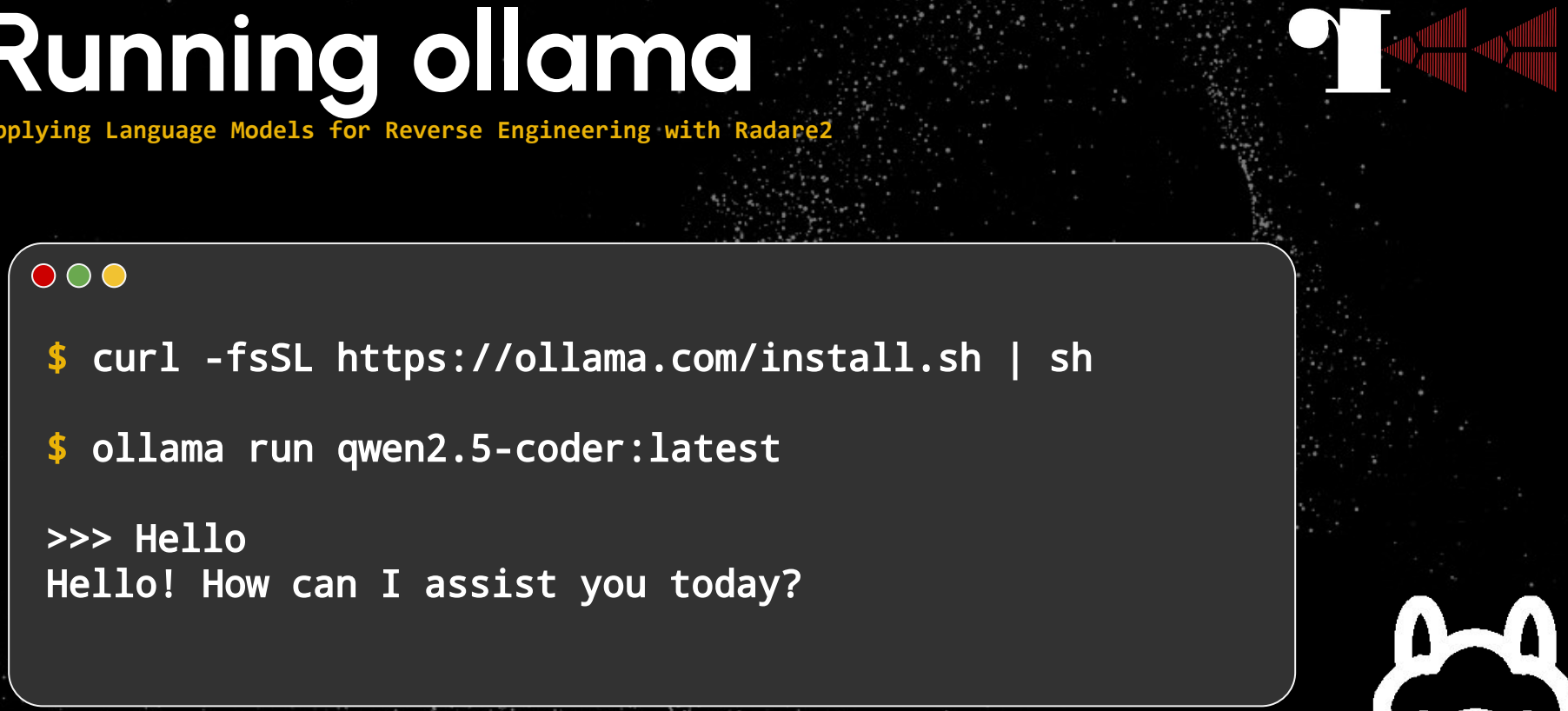
uncensor **mixtral**

"Local models provide privacy in exchange of smaller context and slower response times"

Lower Entrypoints

Running ollama

Applying Language Models for Reverse Engineering with Radare2



```
$ curl -fsSL https://ollama.com/install.sh | sh

$ ollama run qwen2.5-coder:latest

>>> Hello
Hello! How can I assist you today?
```

Prompt Engineering

Applying Language Models for Reverse Engineering with Radare2

Prompt types

- **User prompt**
  - Conversational text
- **System**
  - Formatted with [INST] tokens
  - Privileged LLM role definition
- **Think**
  - Define how the model must reason

Wording your will in the best, try OpenWebUI

Context Limitations

Structure in xml/md

Parameter Size

7B **12B** **32B**

Hallucinations

Applying Language Models for Reverse Engineering with Radare2

Can we trust model responses?

- Like Galicians would say: "it depends"

Always review what we get

- There are ways to reduce errors
- Do not use YOLO mode just in case..

1. Chosen Model
2. Temperature
3. Provide Context
4. Fine tuned models
5. Vector Database
6. More Parameters

Setting up r2ai

Always use r2 from git..

Install plugins with r2pm!

Extending Radare2

Applying Language Models for Reverse Engineering with Radare2

Most versatile reverse engineering framework.

- Started as a forensic tool in 2006
- Added disassemblers, debugging, analysis, exploiting capabilities later
- Command line tool fully opensource since the very first day
- Focus on Fun and follows the UNIX philosophy

Very easy to modify and extend

- Command language or using the embedded Javascript runtime
- Scriptable via r2pipe from literally **any** language
- High level APIs on top of r2pipe (r2papi)
- Native APIs to write your plugins

Understanding how the r2ai subprojects are organized

r2ai

Auto Mode and VDB

decai

Decompiler plugin

r2mcp

Agentic Reversing

Integrations

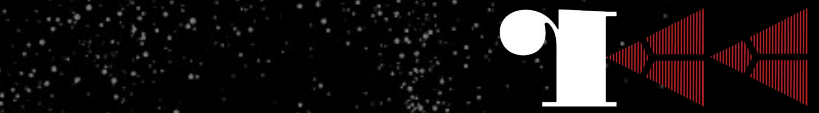
- ollama
- r2pm
- OpenAPI
- Claude

"Conversational Reversing with natural language drastically lowers the learning curve"

Lower Entrypoints

Setting up r2ai

Applying Language Models for Reverse Engineering with Radare2



```
$ git clone https://github.com/radareorg/radare2
$ radare2/sys/install.sh

$ r2pm -U

$ r2pm -ci r2ai-plugin
$ r2pm -ci decai
```

Selecting The Model

Applying Language Models for Reverse Engineering with Radare2

```
$ r2 /bin/ls
[0x00006d30]>  r2ai -e api=openai
[0x00006d30]>  r2ai -e model=?
..

[0x00006d30]>  decai -e api=claude
[0x00006d30]>  decai -e model=?
claude-3-5-sonnet-20241022
claude-3-7-sonnet-20250219
```

```
$ r2 /bin/ls
[0x00006d30]>  r2ai -e api=?
r2ai
claude
deepseek
gemini
hf
mistral
ollama
openapi
openapi2
openai
vllm
xai
```

r2ai-model

Applying Language Models for Reverse Engineering with Radare2

Custom dataset with focus on finetuning and training models to use radare2

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

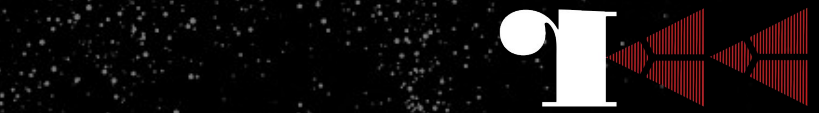
"Everyone is welcome to pull the repo, and submit reviewed statements!"

Sources

- r2book
- blogs
- youtube
- r2cmd

Running r2ai

Applying Language Models for Reverse Engineering with Radare2



```
$ r2 -A server
[0x00006d30]>  r2ai -a find bugs and write the exploits for me
..
```

From Theory To Practice

Let's check some real use cases for r2ai, decai and r2mcp!

Decompilation

Applying Language Models for Reverse Engineering with Radare2

Recovering source code from binary is not an easy task.

There are multiple decompilers available for radare2:

- pdc, r2ghidra, r2dec, jadx, retdec, ...

They are good for some use cases but not all, but with r2ai we can combine them all and improve the output with contextual information and metadata to remove boilerplate from higher level languages like Swift, Dart, ..

Decompiling with Decai

1

Pseudo-disassemble

R2 'pdc' command generates a pseudocode-like full of goto statements and emulation-generated comments to guide the model/reader to provide better context for the reader.

2

Prompting the model

Coding models are good at transforming source code, removing goto statements into high level control flow statements, renaming variables, inlining comments into function arguments, propagating type information..

3

Tweaking / Querying

The output generated by the model can be modified using natural language and provide custom context.

Transpiling into different languages or even adding comments with your mother tongue language.

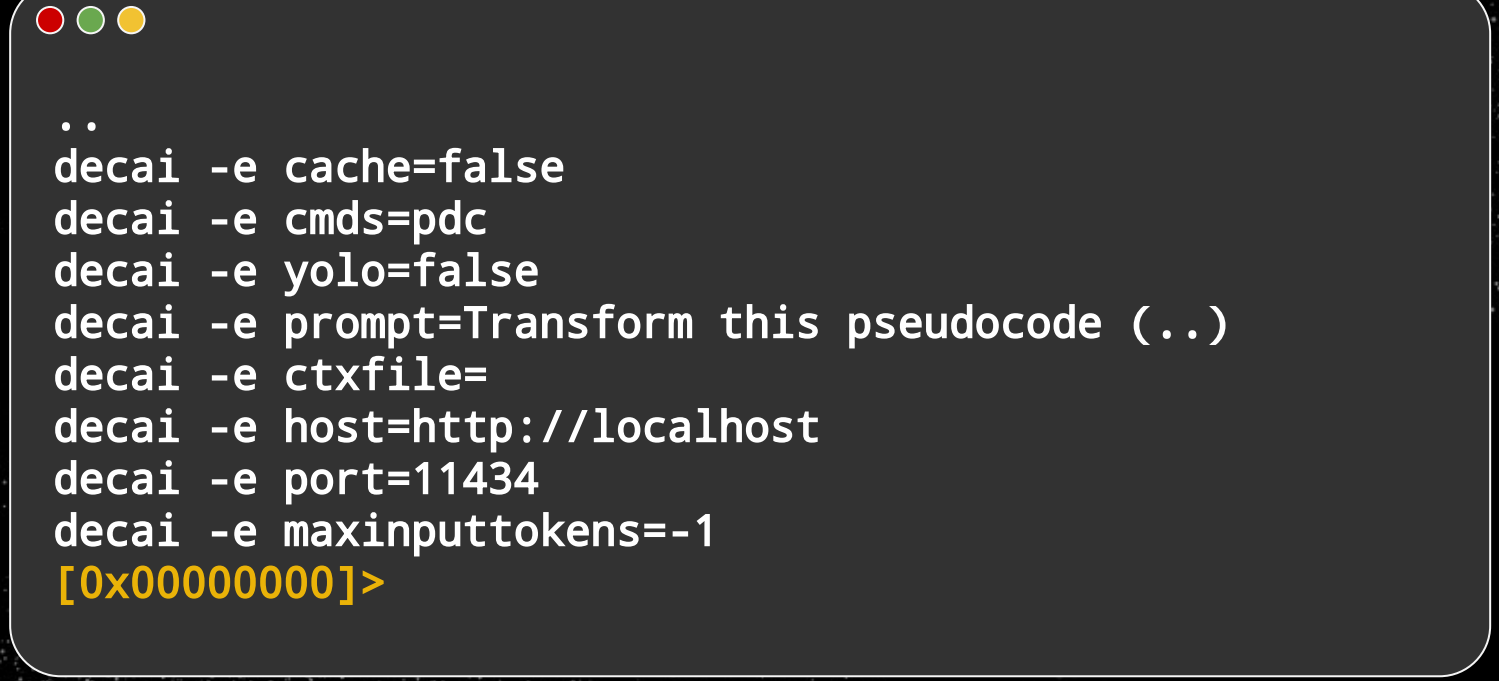
Configuring decai

Applying Language Models for Reverse Engineering with Radare2

```
$ r2 malloc://32
[0x00000000]>  decai -e | more
decai -e model=
decai -e deterministic=true
decai -e think=-1
decai -e debug=false
decai -e api=ollama
decai -e lang=C
decai -e hlang=English
--More--
```

Configuring decai

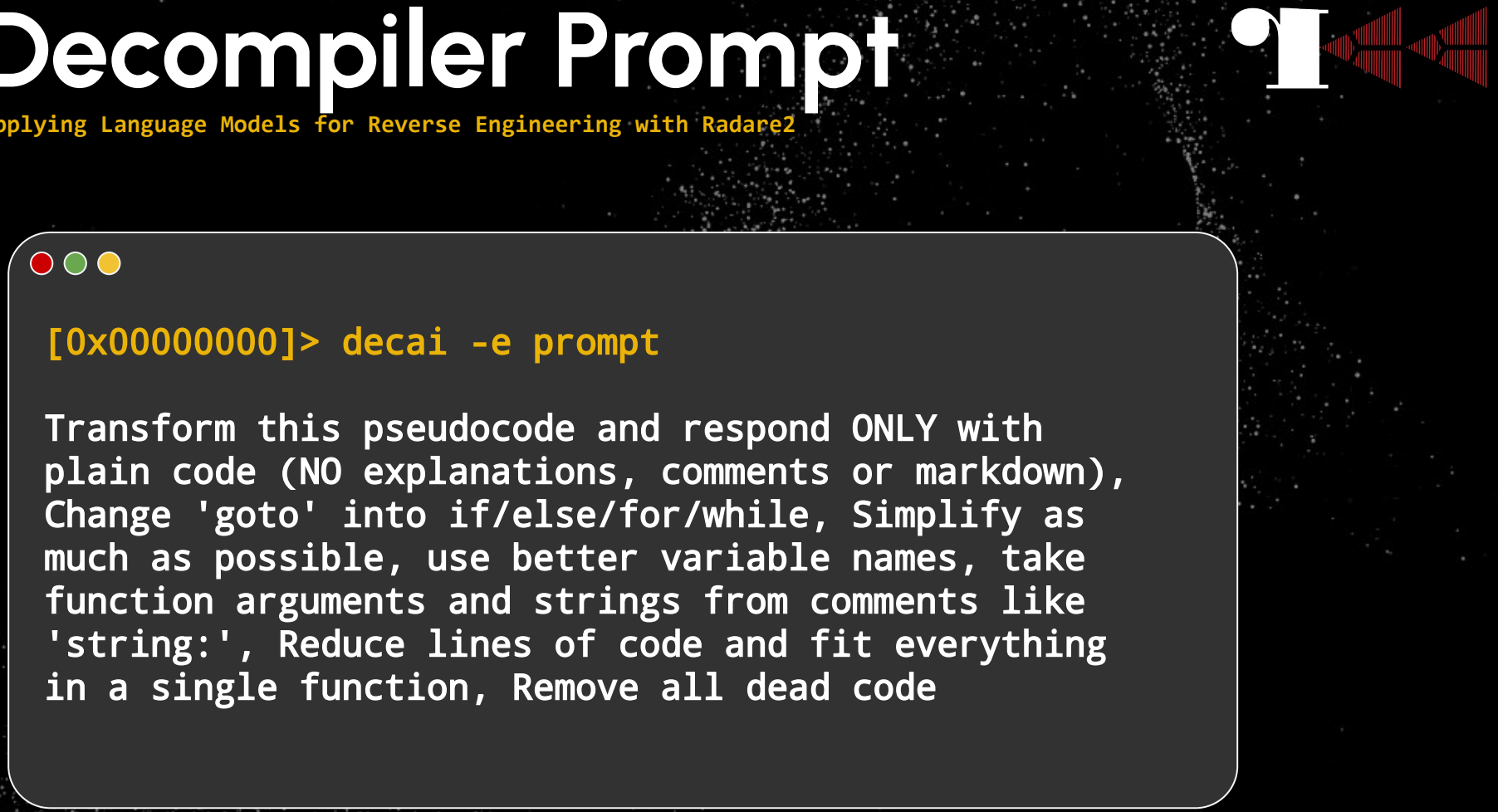
Applying Language Models for Reverse Engineering with Radare2



```
..
decai -e cache=false
decai -e cmds=pdc
decai -e yolo=false
decai -e prompt=Transform this pseudocode (..)
decai -e ctxfile=
decai -e host=http://localhost
decai -e port=11434
decai -e maxinputtokens=-1
[0x00000000]>
```

Decompiler Prompt

Applying Language Models for Reverse Engineering with Radare2



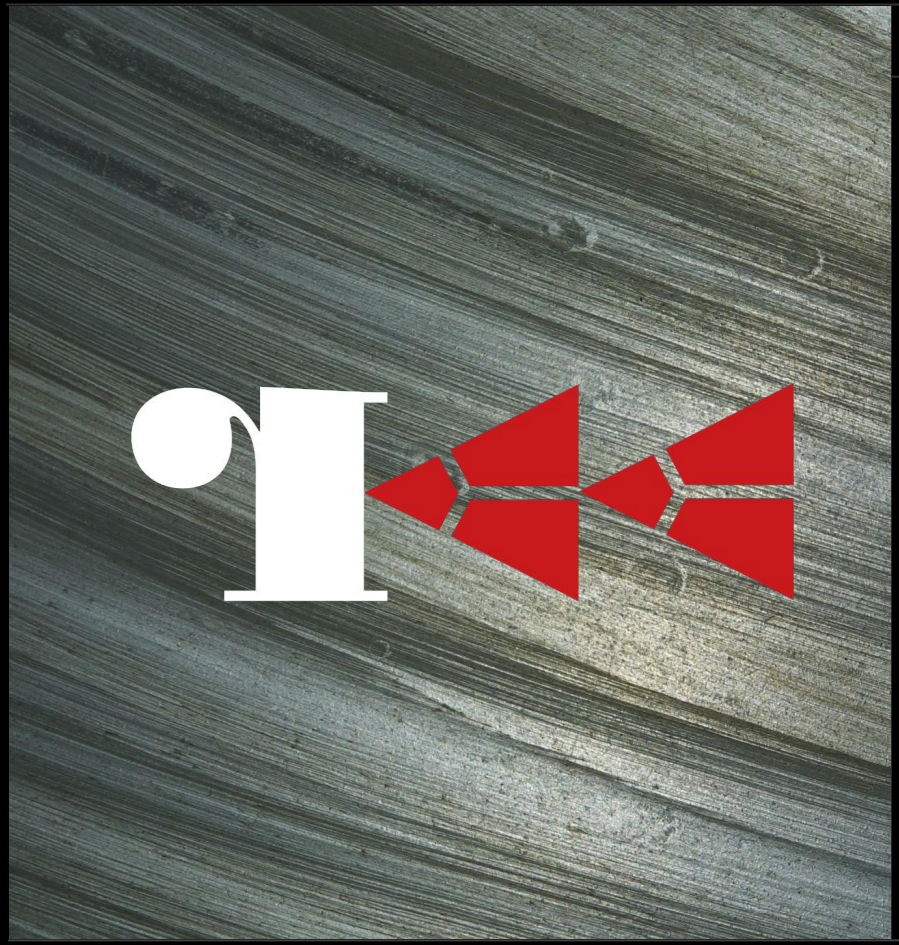
Demo Time!

Decompiling an iOS Swift app

```
$ r2 PasswordCheck
[0x00000000]> decai -d
```

Finding Vulnerabilities

r2ai and decai have all the tools to feed language models to analyze the decompiled code to spot vulnerable functions and guide you fix the bugs or exploit them!



Demo Time!

Basic binary overflow analysis

```
$ r2 hello-overflow
[0x00000000]> decai -v
```

Auto-Pilot mode for Reversing!

r2ai is more than an assistant, it can also take control over your r2 session and run all the commands, analyse the output and take decisions until the proposed task is resolved.

Agentic Reversing with the Auto mode

1

Reasoning with almost any model

Use decai -a or r2ai -a with the prompt to be resolved.

2

Plan the task and iterate

The model will enter a loop trying to request all the necessary information in order to determine how to solve the problem.

Provide feedback and corrections to train the model for your needs

3

Yolo mode

Be careful not to enable this mode unless you fully trust the model, running commands unassisted can be dangerous!

Demo Time!

Brainless crackme solving and find vulnerabilities in binaries

```
[0x00000000]> decai -a
solve this crackme

..
```

Agentic Reversing

Extending Claude or OpenWebUI with MCP agent servers to automate complex tasks with more clear target

Agentic Reversing with the r2mcp

1

Installing r2mcp

$ r2pm -ci r2mcp

2

Configure Claude

Cmd+comma and paste the fancy json to get the r2mcp agent loaded just restart the app

3

Prompt and wait

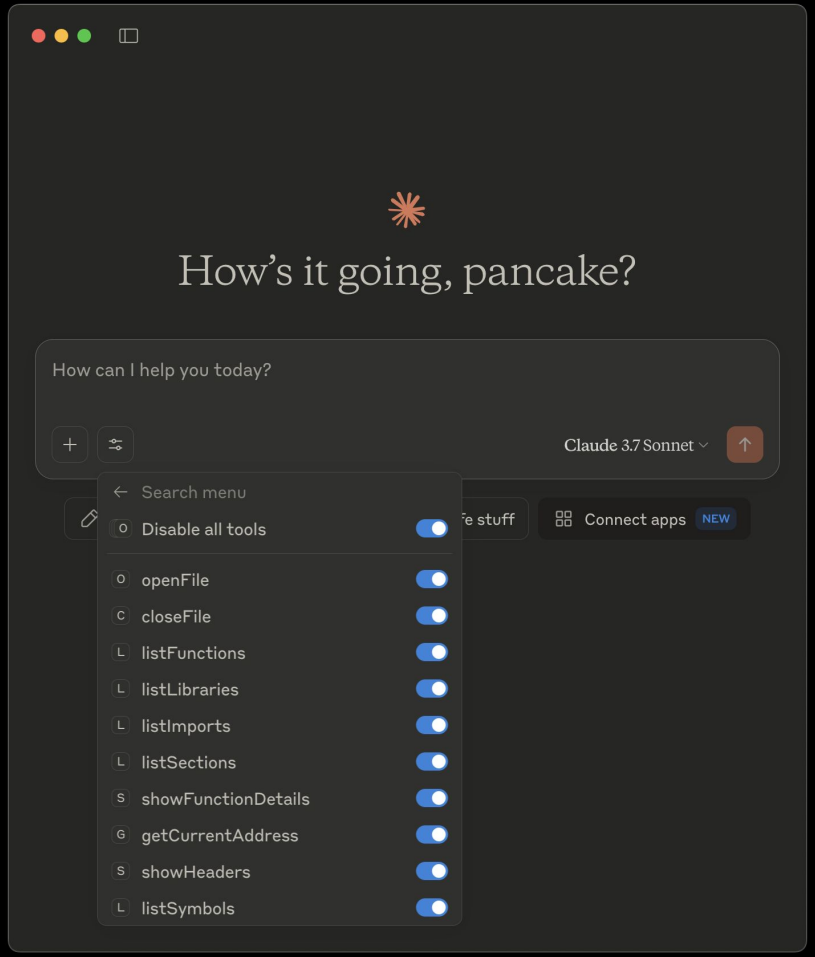
Ask Claude to open /tmp/deepseek with radare2 and follow the owasp guidelines to find vulnerabilities and bad privacy practices in order to write an structured report showing all the spotted findings.

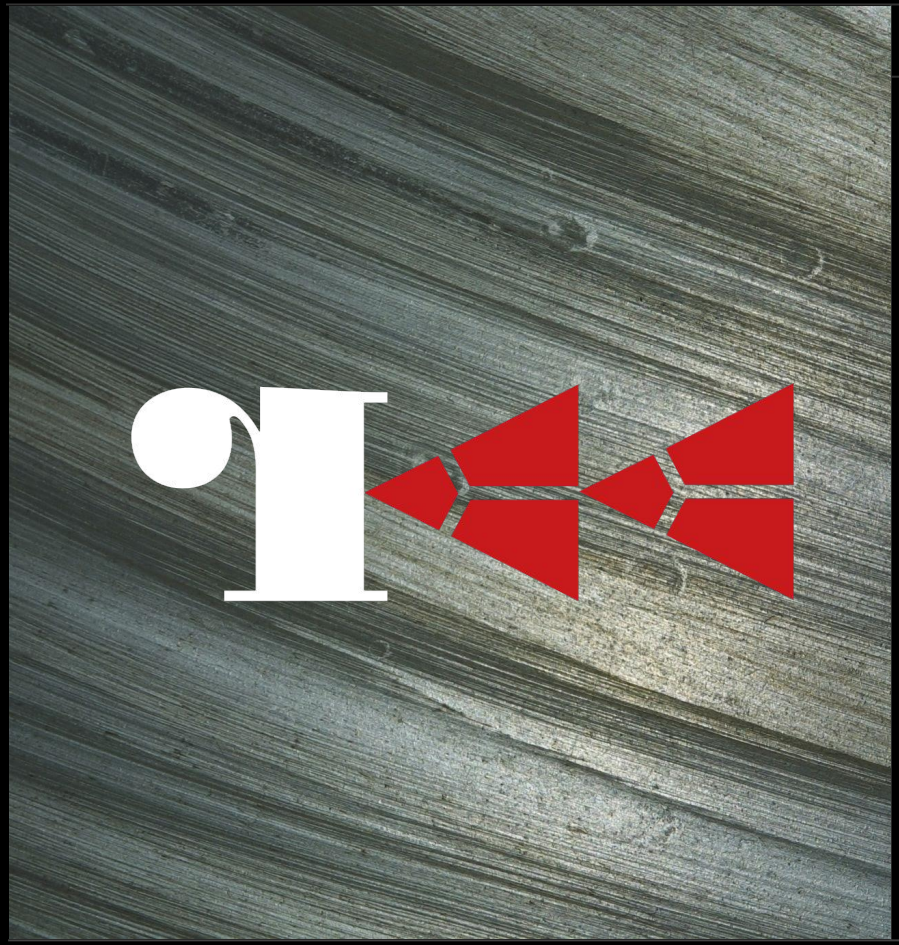
Claude with r2mcp

Provide an stdin/stdout daemon like inetd but using json.

- Exposes an abstracted interface to use radare2
  - Tool Name
  - Tool Description
  - Tool Arguments

The model knows what needs to be executed and when.





Demo Time!

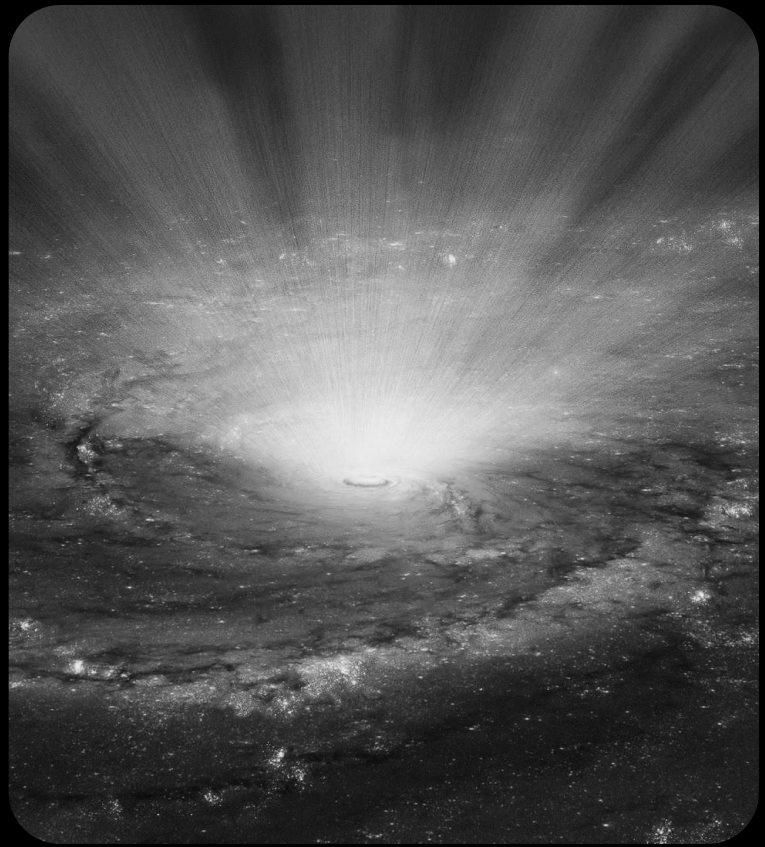
Running Claude with r2mcp

- Generate a report analyzing the Deepseek iOS app

Future Plans

- r2ai is fun, as user or developer
  - **Give it a try!**

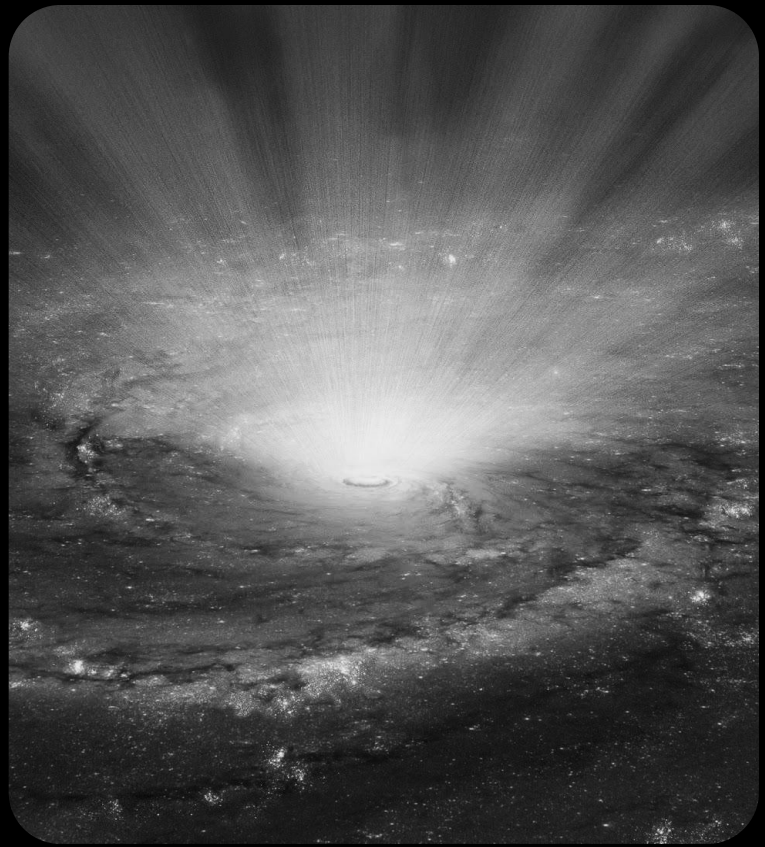
- Finetune models with our dataset
- Improved local support, reducing hardware requirements
- Event driven and bg reasoning
- Realtime voice interactions



Learn more!

Check out last r2con 2024 recordings!

- **Dnakov** autosolving crackmes
- **Cryptax** analysing malware
- **Pancake** decompiling bins
- **Brainstorm** reversing firmwares
- **Murphy's** cracking Unity games
- ...



NSConnect Spam

I'll be presenting at **#NSConnect** soon!

Online conference organized by **NowSecure** with focus on mobile security

https://academy.nowsecure.com/page/nowsecure-connect

Thanks For Watching!

Q&A Time!

@pancake@infosec.exchange // HackBCN'2025