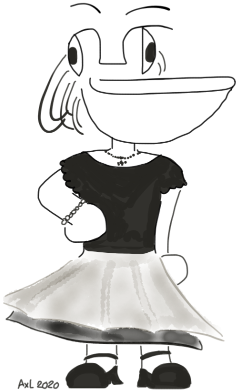
Reverse engineering with r2ai and GhidraMCP

Axelle Apvrille

BruCON, September 2025

Agenda

Who am I?



- Principal security researcher with **Fortinet**
- Reverse mobile malware (Android, iOS) and IoT malware
- Founder of Ph0wn CTF, France
- Talks: `https://cryptax.github.io/talks/papers/`
- @cryptax on Mastodon, BlueSky

Skills pre-requisites

Basic++ knowledge of assembly

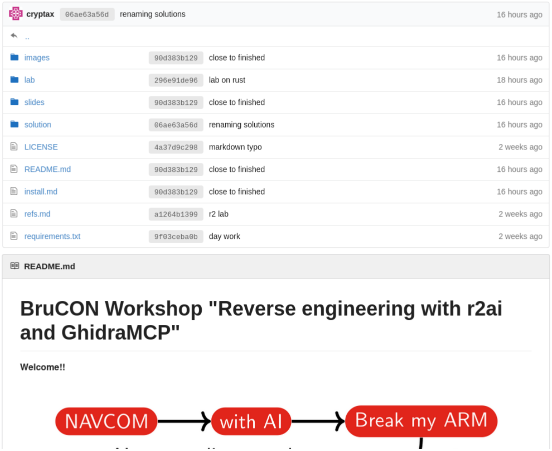
x86, ARM...

or

Have already reversed binaries

IDA Pro? Ghidra? Radare2?

Workshop repository



cryptax	06ae63a56d	renaming solutions	16 hours ago
..			
images	90d383b129	close to finished	16 hours ago
lab	296e91de96	lab on rust	18 hours ago
slides	90d383b129	close to finished	16 hours ago
solution	06ae63a56d	renaming solutions	16 hours ago
LICENSE	4a37d9c298	markdown typo	2 weeks ago
README.md	90d383b129	close to finished	16 hours ago
install.md	90d383b129	close to finished	16 hours ago
refs.md	a1264b1399	r2 lab	2 weeks ago
requirements.txt	9f03ceba0b	day work	2 weeks ago

README.md

BruCON Workshop "Reverse engineering with r2ai and GhidraMCP"

Welcome!!

Clone the repository, or download the ZIP release

`http://34.155.29.84:3000/axelle/r2ai-workshop`

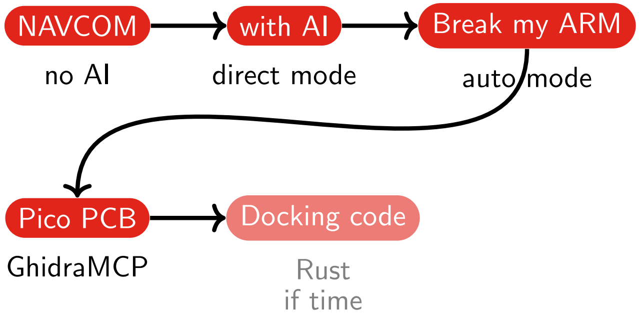
Operating System

This workshop has been tested on
Debian and Linux Mint, on x86-64.

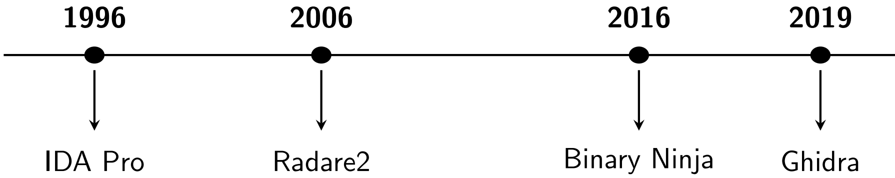
Tool	Windows	macOS x86	macOS ARM
Ghidra	✓	✓	✓
GhidraMCP	✓	✓	Extra effort
Radare2	✓	✓	✓

- With another OS, it's untested, but you probably know what you're doing!
- If you want to isolate your own OS, consider Exegol 🔗, Kali VM 🔗, Linux Docker container 🔗...
- Install Python and Java

Labs


NAVCOM
no AI
with AI
direct mode
Break my ARM
auto mode
Pico PCB
GhidraMCP
Docking code
Rust
if time

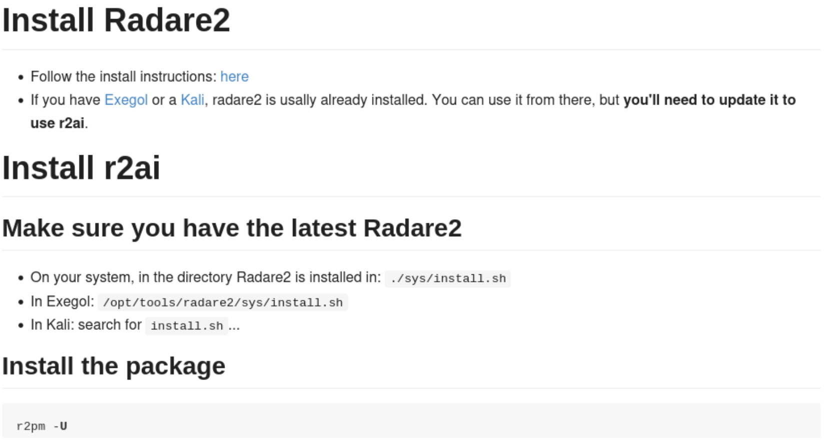
Radare2



https://github.com/radareorg/radare2

open-source, command-line tools, scriptable, many architectures and binary file formats

Install r2



Install Radare2

- Follow the install instructions: here
- If you have Exegol or a Kali, radare2 is usally already installed. You can use it from there, but **you'll need to update it to use r2ai**.

Install r2ai

Make sure you have the latest Radare2

- On your system, in the directory Radare2 is installed in: `./sys/install.sh`
- In Exegol: `/opt/tools/radare2/sys/install.sh`
- In Kali: search for `install.sh` ...

Install the package

```
r2pm -U
```

In the git repo, go to `install.md`

Are you already a r2 user?

Skip this part

Get the binary: `./lab/navcom/navcom`

Reverse it, get the flag **with r2 but no AI**.

Starting r2

- Start: `r2 ./binary`. There are options, but you can specify them later.
- We don't really care:

```
WARN: Relocs has not been applied. Please use `-e
↪  bin.relocs.apply=true` or `-e bin.cache=true`
↪  next time
```

- This is a **random startup message**, like `motd`. Sometimes funny.

```
-- Execute a command every time a breakpoint is
↪  hit with 'e cmd.bp = !my-program'
```

- This is the **prompt**, waiting for your command. The current address between square brackets.

```
[0x004ccb80]>
```

Radare2 crash course: There are 10 commands to know

1. ?
2. aa and aaa
3. s
4. afl
5. `~word` and `~..`
6. pd, pdc, pdf
7. axt, axf
8. iz, izz
9. /a and /x
10. f

There are many other commands, but not as essential.

Command 1: get help with ?

End any command with `?` to get **help**:

```
[0x004ccb80]> a?
Usage: a  [abdefFghoprxstc] [...]
| a                 alias for aai - analysis
↪  information
| a:[cmd]           run a command implemented by an
↪  analysis plugin (like : for io)
| a*                same as afl*;ah*;ax*
| aa[?]             analyze all (fcns + bbs) (aa0 to
↪  avoid sub renaming)
```

`?` can also be used to **evaluate an expression**:

```
[0x004ccb80]> ? 0x23 + 10
int32   45
uint32  45
hex     0x2d
```

Command 2: analyze with aa

- aa: **a**nalyze **all** (functions)
- aaa: perform a deeper analysis
- aaaa: experimental
- aaaaaaaaaa: a joke among r2 users!

```
[0x0045e5f0]> aa
INFO: Analyze all flags starting with sym. and entry0
↪  (aa)
INFO: Analyze imports (af@@@i)
INFO: Analyze entrypoint (af@ entry0)
INFO: Analyze symbols (af@@@s)
WARN: Function already defined in 0x00401000
INFO: Recovering variables (afva@@@F)
INFO: Analyze all functions arguments/locals
↪  (afva@@@F)
```

Command 3: seek with s

- `s 0x10007310`: **go to** address 0x10007310
- `s sym.main.main`: go to the address of this symbol
- `s- 10`: go 10 bytes backward
- `s+ 10`: go 10 bytes forward
- The address in the prompt adjusts to where you are

```
[0x004ccb76]> s sym.main.ransomware
[0x004cc740]> s- 10
[0x004cc736]>
```

Command 4: list functions with afl

- **a** for “command of type analysis”
- **f** for “function”
- **l** for “list”

```
[0x004cc736]> afl
0x0045e5f0    1       5 entry0
0x00401000    0       0 sym.runtime.text
0x00401780   19    1006 sym.internal_cpu.doinit
```

Command 5: grep and more

- `afl` is often too long!
- `afl~..` to get the output *page by page*
- `afl~main` to list only functions that contain the word *main*

```
[0x004cc736]> afl~main
0x004cc4a0   15    669 sym.main.init.0
0x004cc740   10    340 sym.main.ransomware
0x004cc8b0   11    686 sym.main.start
0x004ccb80    3    138 sym.main.main
```

Command 6: disassemble with pd, pdc, pdf

- `pd 10`: **disassemble 10 instructions** at the current location
- `pd 10 @ 0x004ccc10`: disassemble 10 instructions at that address
- `pd 10 @ sym.crypto.init`: disassemble 10 instructions at that function
- `pd`**f** disassemble the current function
- `pd`**c** provide pseudo decompilation at the current offset. Useful for AI because consumes less tokens than `pdf`.

Command 7: cross references with axt and axf

- axt : references going **to** the current location
- axf : references going **from** the current location
- You can use axt @ addr
- If no references, try aaa or aar first

```
[0x004cc190]> axt
sym.goransom_utils.Decrypt 0x4cc491 [CODE:--x] jmp
↪  sym.goransom_utils.Decrypt
sym.main.ransomware 0x4cc82a [CALL:--x] call
↪  sym.goransom_utils.Decrypt
```

Function sym.goransom_utils.Decrypt is called by itself and by sym.main.ransomware.

Command 8: searching for strings

- iz seaches for strings in the **data** sections
- izz searches in the *whole* binary
- You can use ˜to limit the output

```
[0x004cc190]> iz~http
3028 0x0010ca9c 0x0050ca9c 1247 1248 .rodata    ascii
↪  able\nsync: WaitGroup misuse: Add called
↪  concurrently with Waitbufio.Scanner: SplitFunc
↪  returns advance count beyond inputruntime:
```

Command 9: search for bytes, instructions

- /x e8a8f5: search for the sequence of bytes e8a8f5

```
[0x004cc190]> /x e8a8f5
0x0048d653 hit3_0 e8a8f5
0x004cc883 hit3_1 e8a8f5
[0x0048d653]> pd 1 @ hit3_0
;-- hit3_0:
0x0048d653      e8a8f5fcff     call
↪  sym.runtime.gcWriteBarrier
```

- /a jmp eax: search for this instruction

```
[0x004cbe30]> /a jbe 0x4cc185
0x0041e60d hit1_0 0f86e4020000
[0x004cbe30]> pd 1 @ hit1_0
;-- hit1_0:
0x0041e60d      0f86e4020000   jbe 0x41e8f7
```

Command 10: list “flags” with f

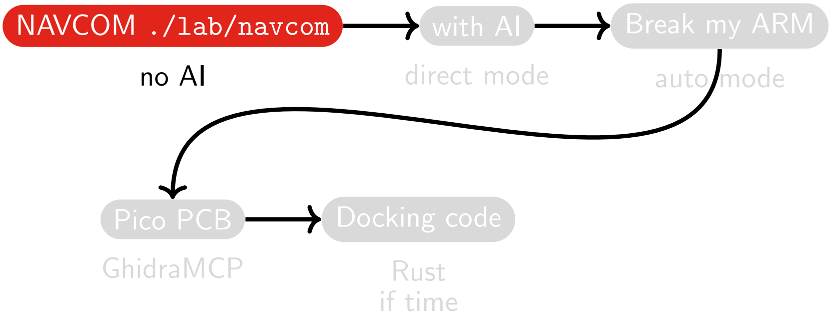
Flag: definition

A *flag* in r2 is a *label for an offset*: functions, objects, symbols, strings...

There are usually lots of flags. Use ˜to limit the length of the output

```
[0x004cbe30]> f~ransom
0x004cbe30 863 sym.goransom_utils.Encrypt
0x004cbe30 1 sym.go.goransom_utils.Encrypt
0x004cc190 774 sym.goransom_utils.Decrypt
0x004cc190 1 sym.go.goransom_utils.Decrypt
0x004cc740 355 sym.main.ransomware
0x004cc740 1 sym.go.main.ransomware
```

Lab 1: hands-on r2 with locked Navcom

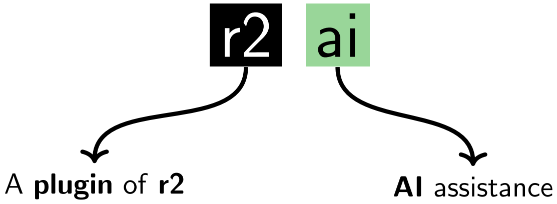


```
== SpaceCom Systems ==
== NAVCOM Module Locked ==
Enter unlock code:
```

Goal: Get the Unlock Code with r2

Agenda

What is r2ai?



Radare2 disassembler (r2) assisted by AI

Radare2 plugins

- r2pm is Radare2's **P**ackage **M**anager
- `r2pm -U`: initialize/**update** the repository
- `r2pm -l`: **list** packages
- `r2pm -s word`: **search** for a package containing *word*
- `r2pm -ci r2ai`: **clean install**
- **r2ai** and **decai** are similar. r2ai is implemented in C, decai in Javascript. We will mostly use **r2ai**.

Connecting r2ai to a model

- **Key**. Put your key in `~/.r2ai.PROVIDER-key`
  e.g `~/.r2ai.mistral-key`, or `~/.r2ai.groq-key`...
- **API**. Inside r2, `r2ai -e api=PROVIDER`
  e.g groq, openai, mistral...
- **Model**. List available models: `r2ai -e model=?`
- Select model: `r2ai -e model=NAME`

The **API** is the way to communication with the LLM server

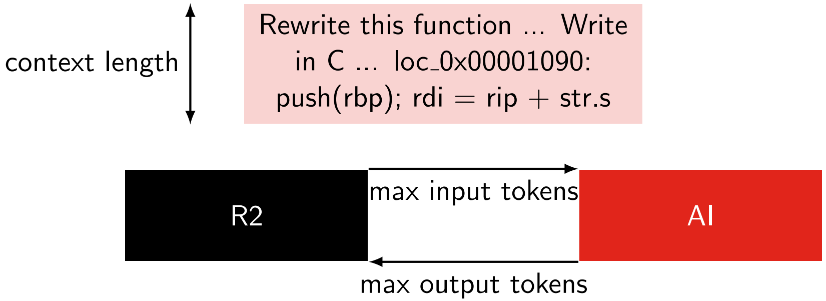
Connecting to your own LLM server

- Install Ollama 🔗, or LM Studio 🔗, and setup your server. GPUs required!
- Start the server: get IP address and port.
- Inside r2,
  - Ollama: `r2ai -e api=ollama`
  - LM Studio: `r2ai -e api=openai` (because it uses the same API as OpenAI)
- `r2ai -e baseurl=http://IPADDRESS:PORT`
- List available models: `r2ai -e model=?`
- Select model: `r2ai -e model=NAME`

Direct mode

1. In r2, go in the function you want to decompile.
2. Select the programming language to use: `r2ai -e lang`.
   By default, **C**.
3. `r2ai -d` decompiles the *current function*.
4. The prompt can be customized with `r2ai -e prompt`, but it works pretty well as is.
5. r2ai sends along the result of `r2ai -e cmds`.
   By default, `cmds=pdc` i.e pseudo code of the function.

The r2ai direct mode: r2ai -d


context length
Rewrite this function ... Write in C ... loc_0x00001090: push(rbp); rdi = rip + str.s
R2
max input tokens
AI
max output tokens

- A single request/response pair.
- Context won't grow. Each new request wipes the previous context.
- Problems may occur with long functions.
  Solution: increase `r2ai -e max_tokens`.

Solving an Output Token Limit

Model	Nb of Requests	Input tokens	Output tokens
Claude Sonnet 3.7	1,000 / min	40,000 / min	16,000 / min

Table: Lower tier limits for August 2025

- R2ai sets it to a *default* value, for all models.
- We need to adjust it **manually**.

```
[0x004130a0]> r2ai -e max_tokens= 16000
[0x004130a0]> r2ai -d
// It works, we get it all :)
```

Asking a quick question with Direct mode

```
r2ai -d your question
```

- This will no longer decompile the function, but answer the question.
- Concerns the current function.
- Alternatively, you can modify the prompt: `r2ai -e prompt=...` and use `r2ai -d` (bare, no question).

New VM features... request too large

While using Groq with gpt-oss-20b:

```
[0x004124f0]> r2ai -d
ERROR: OpenAI API error 413
ERROR: OpenAI API error response: {"error":{"message":"Request too large
↪   for model `openai/gpt-oss-20b` in organization `org_xxx` service
↪   tier `on_demand` on tokens per minute (TPM):
↪   Limit 8000, Requested 9586, please reduce your message size and
↪   try again. Need more tokens? Upgrade to Dev Tier today at
↪   https://console.groq.com/settings/billing","type":"tokens","code":"rate_lim
```

- It's an **input** token issue.
- max_tokens is for *output* token, it won't work.

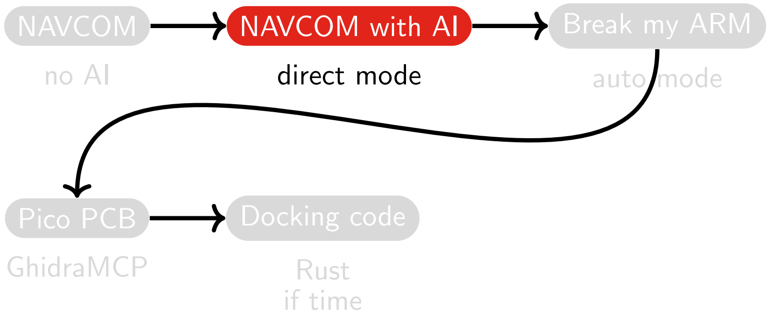
Solution: shorten the context, or raise the limit!

- `pd 20 @ main` or `pdc` instead of `pdf`
- `afl~main` instead of `afl`
- If you have access to the model: increase **context length**.
- or select another model, a higher tier etc

Other settings

- List all config: `r2ai -e`
- Modify the r2 command result to send with Direct mode: `r2ai -e cmds`
- Speak in another language: `r2ai -e hlang=fr`
- Be more creative: raise `r2ai -e temperature`

Lab 2: r2ai direct mode on Navcom



```
== SpaceCom Systems ==
== NAVCOM Module Locked ==
Enter unlock code:
```

Goal: Get the Unlock Code using r2ai in direct mode

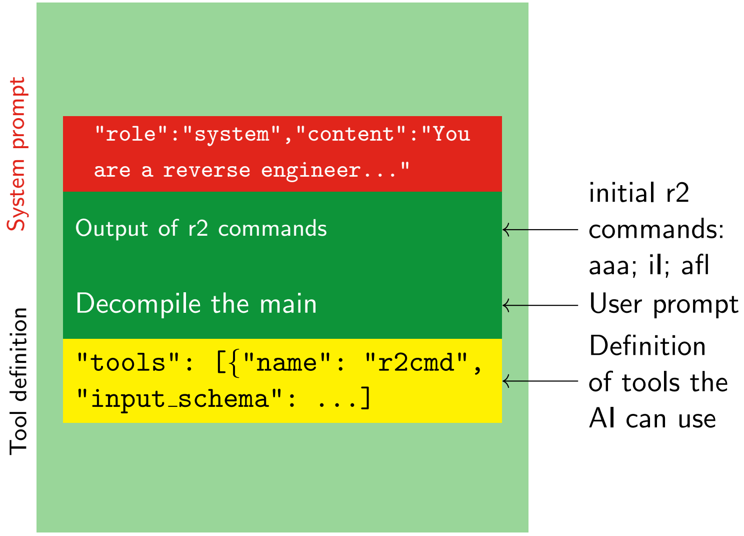
The r2ai Auto Mode: r2ai -a

The AI can **run tools** on **your** host.

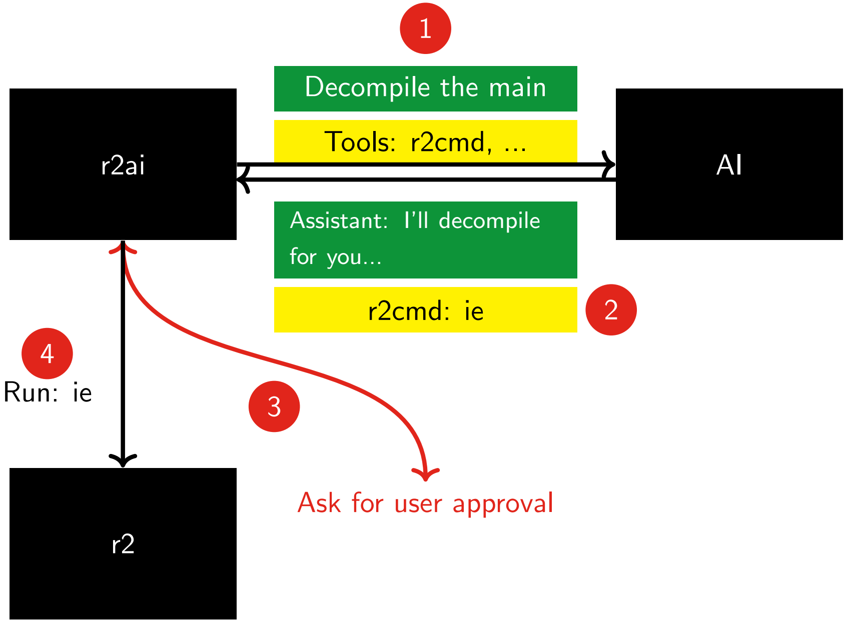
- **r2cmd**: run a r2 command and return the output.
- **execute_js**: runs a Javascript program, using `QuickJS` engine (built in Radare2).

User approval is required.

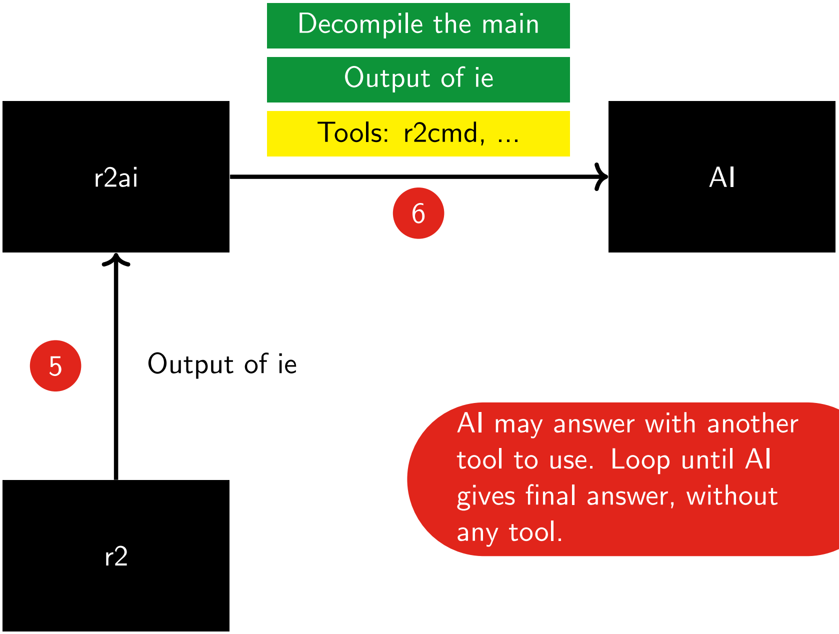
The context in r2ai auto mode


System prompt
Tool definition
"role":"system","content":"You are a reverse engineer..."
Output of r2 commands
Decompile the main
"tools": [{"name": "r2cmd", "input_schema": ...]
initial r2 commands: aaa; il; afl
User prompt
Definition of tools the AI can use

Auto mode flow



Auto mode flow



AI may answer with another tool to use. Loop until AI gives final answer, without any tool.

Tweaking the auto mode

- `r2ai -e auto.init_commands`: the result of r2 commands **to send with the first context**.
- `r2ai -e auto.max_runs`: the **max number of interactions**. 30 is enough.
- `r2ai -e auto.hide_tool_output`: hide (or not) the output of commands which were run.
- `r2ai -e chat.show_cost`: hide (or not) the cost of your requests (approximate).

Keep low cost

- `aaa;iI;afl` is often too many tokens. Do `aaa` before. Do `iI` if you must. As first command, just do `afl~main`.
- Prefer `pdc` over `pdf`. Even better `pd 20` etc.
- `iz` is likely to be long. Prefer `iz~blah` (use part of an interesting string).
- As soon as you see the conversation is going nowhere, **stop it**. Erase the context `r2ai -R` and ask a better question.
- Recent models are often more expensive. If you don't need the most recent, use an older one: it will be cheaper. Inspect rate limits.
- Example: with Anthropic, a full malware analysis costs 2-3 US dollars.

Too Many Requests

```
WARN: Server returned 429 response code
INFO: Retrying request (4/10) after error...
```

You sent too many requests to the LLM in a short period of time.
The server blocks them.

Solution: wait (or purchase a higher tier)
Terminate conversations that look like a dead end.

r2/r2ai issue workaround

```
[0x0041380c]> pdf @
↪  sym.SCANFOLDER_ENCRYPTORERASEFOLDER_crc0EEABFEB
ERROR: curl failed: execl: Argument list too long
```

Solution:

- `r2ai -e http.use_files = true`
- and/or `r2ai -e http.backend = system`

More issues...

- Tool support. Mistral behaves badly in Auto mode.
- Insufficient funds.

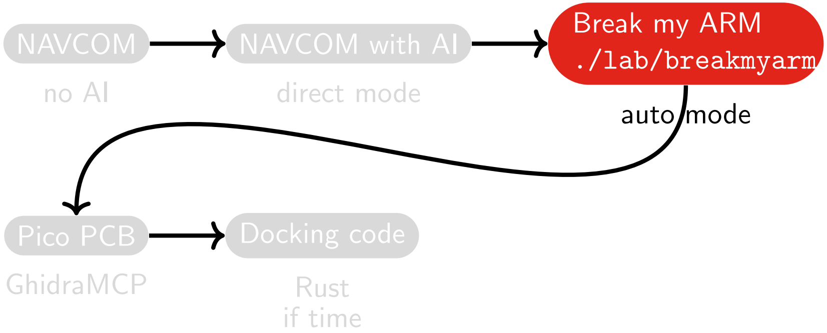
```
"error":{"type":"invalid_request_error","message":"Your credit
↪  balance is too low"}
```

- Input rate limit.

```
"error":{"type":"rate_limit_error","message":"This request would
↪  exceed your organization's rate limit of 40,000 input tokens
↪  per minute"}
```

- `Server returned 429 response code`: too many requests, slow down!
- `Server returned 529 response code`: timeout.
- Context length. Use "economic" commands: `pd 20 @ main` instead of `pdf` etc.

Lab 3: r2ai auto mode on Break my ARM

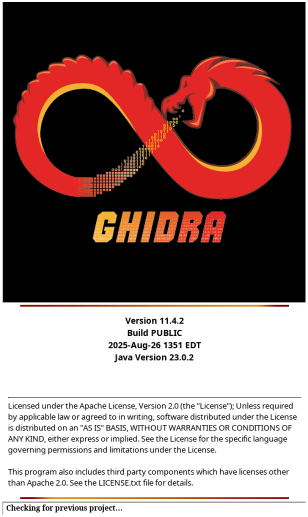


Easy challenge from Ph0wn CTF 2019

Goal: solve it using the **auto** mode

Agenda

Ghidra



Version 11.4.2
Build PUBLIC
2025-Aug-26 1351 EDT
Java Version 23.0.2

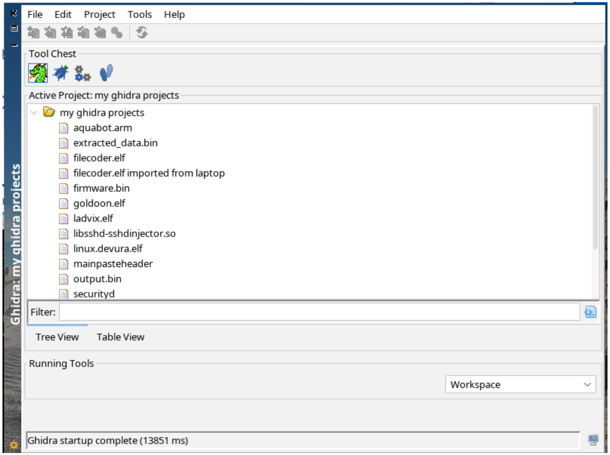
Licensed under the Apache License, Version 2.0 (the "License"); Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

This program also includes third party components which have licenses other than Apache 2.0. See the LICENSE.txt file for details.

Checking for previous project...

`https://github.com/NationalSecurityAgency/ghidra`

Ghidra: Project Window



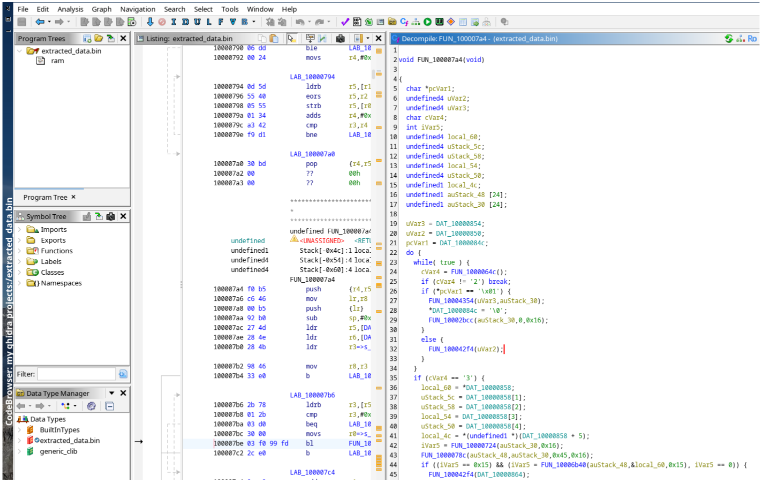
Active Project: my ghidra projects

- my ghidra projects
  - aquabot.arm
  - extracted_data.bin
  - filecoder.elf
  - filecoder.elf imported from laptop
  - firmware.bin
  - goldoon.elf
  - ladvix.elf
  - libsshd-sshdinjector.so
  - linux.devura.elf
  - mainpasteheader
  - output.bin
  - securityd

Ghidra startup complete (13851 ms)

File, Import File

Ghidra: Code Browser



File Edit Analysis Graph Navigation Search Select Tools Window Help
Program Trees
extracted_data.bin
ram
Program Tree
Symbol Tree
Imports
Exports
Functions
Labels
Classes
Namespaces
Filter:
Data Type Manager
Data Types
BuiltInTypes
extracted_data.bin
generic_clib
CodeBrowser: my ghidra projects:/extracted_data.bin
Listing: extracted_data.bin
Decompile: FUN_100007a4 - (extracted_data.bin)

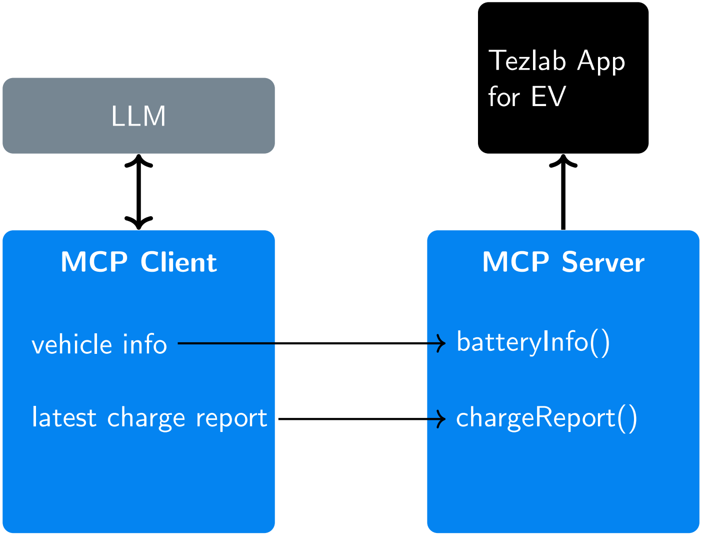
```
10000790 06 dd        ble       LAB_10
10000792 00 24        movs      r4,#0x

                  LAB_10000794
10000794 0d 5d        ldrb      r5,[r1
10000796 55 40        eors      r5,r2
10000798 05 55        strb      r5,[r0
1000079a 01 34        adds      r4,#0x
1000079c a3 42        cmp       r3,r4
1000079e f9 d1        bne       LAB_10

                  LAB_100007a0
100007a0 30 bd        pop       {r4,r5
100007a2 00           ??        00h
100007a3 00           ??        00h

                  **********************
                  *
                  **********************
                  undefined FUN_100007a4
    undefined     <UNASSIGNED>    <RETU
    undefined1      Stack[-0x4c]:1 local
    undefined4      Stack[-0x54]:4 local
    undefined4      Stack[-0x60]:4 local
                  FUN_100007a4
100007a4 f0 b5        push      {r4,r5
100007a6 c6 46        mov       lr,r8
100007a8 00 b5        push      {lr}
100007aa 92 b0        sub       sp,#0x
100007ac 27 4d        ldr       r5,[DA
100007ae 28 4e        ldr       r6,[DA
100007b0 28 4b        ldr       r3=>s_

100007b2 98 46        mov       r8,r3
100007b4 33 e0        b         LAB_10

                  LAB_100007b6
100007b6 2b 78        ldrb      r3,[r5
100007b8 01 2b        cmp       r3,#0x
100007ba 03 d0        beq       LAB_10
100007bc 30 00        movs      r0=>s_
100007be 03 f0 99 fd  bl        FUN_10
100007c2 2c e0        b         LAB_10

                  LAB_100007c4
```

```c
void FUN_100007a4(void)

{
  char *pcVar1;
  undefined4 uVar2;
  undefined4 uVar3;
  char cVar4;
  int iVar5;
  undefined4 local_60;
  undefined4 uStack_5c;
  undefined4 uStack_58;
  undefined4 local_54;
  undefined4 uStack_50;
  undefined1 local_4c;
  undefined1 auStack_48 [24];
  undefined1 auStack_30 [24];

  uVar3 = DAT_10000854;
  uVar2 = DAT_10000850;
  pcVar1 = DAT_1000084c;
  do {
    while( true ) {
      cVar4 = FUN_1000064c();
      if (cVar4 != '2') break;
      if (*pcVar1 == '\x01') {
        FUN_10004354(uVar3,auStack_30);
        *DAT_1000084c = '\0';
        FUN_10002bcc(auStack_30,0,0x16);
      }
      else {
        FUN_100042f4(uVar2);
      }
    }
    if (cVar4 == '3') {
      local_60 = *DAT_10000858;
      uStack_5c = DAT_10000858[1];
      uStack_58 = DAT_10000858[2];
      local_54 = DAT_10000858[3];
      uStack_50 = DAT_10000858[4];
      local_4c = *(undefined1 *)(DAT_10000858 + 5);
      iVar5 = FUN_10000724(auStack_30,0x16);
      FUN_1000078c(auStack_48,auStack_30,0x45,0x16);
      if ((iVar5 == 0x15) && (iVar5 = FUN_10006b40(auStack_48,&local_60,0x15), iVar5 == 0)) {
        FUN_100042f4(DAT_10000864);
```

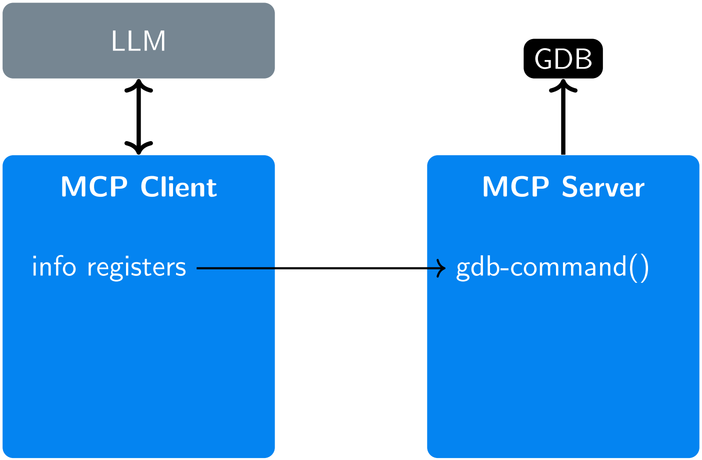
Model Context Protocol

- Announced by **Anthropic** in **November 2024**
- **Open Protocol** (not limited to Anthropic)
- Standardizes how *applications* provide *context* to LLMs
- MCP is about **finding, connecting and use external tools** (and resources)
- r2ai **auto** mode interacted with tools *before MCP existed*. Its own way.
- https://modelcontextprotocol.io/

MCP Example: Accessing Vehicle Information

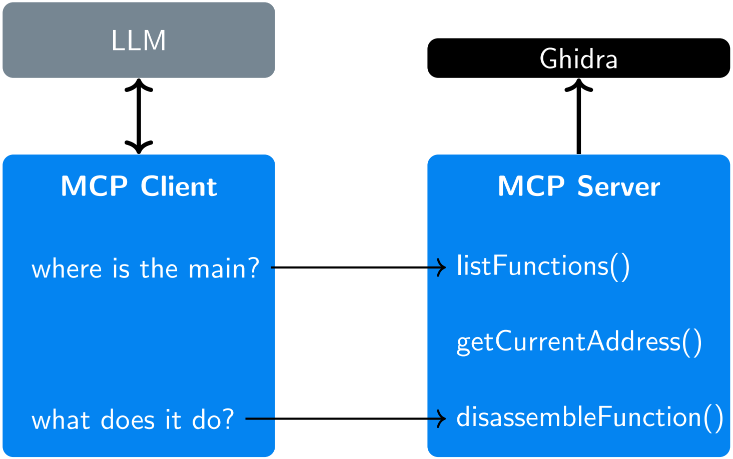


MCP Example: GDB



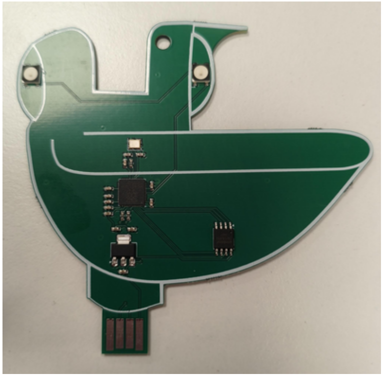
https://github.com/jtang613/gdb-mcp

Ghidra MCP



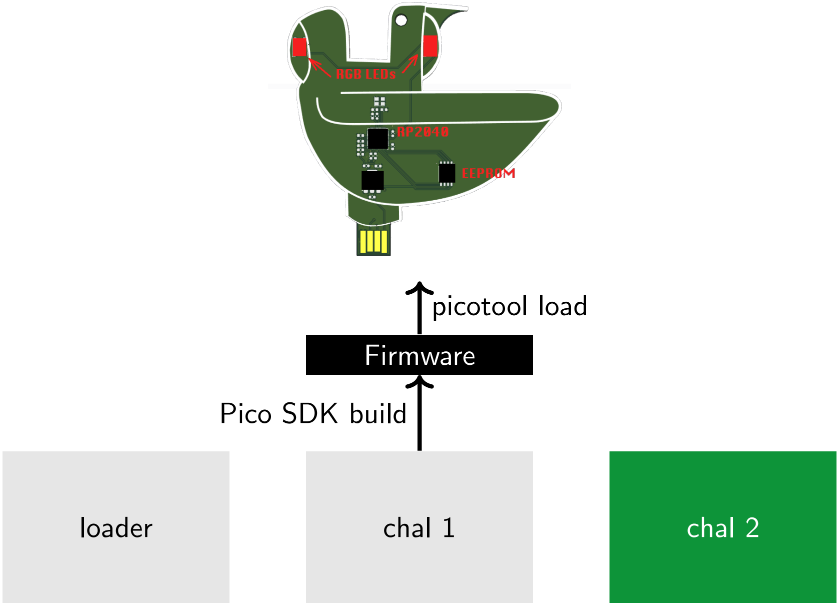
https://github.com/LaurieWired/GhidraMCP

Pico PCB challenge



- Ph0wn CTF 2024
- Contains 2 challenges: 1 hardware, **1 reverse**
- RP2040 chip (Raspberry Pico) https://www.raspberrypi.com/products/rp2040/specifications/
- Flashes white lights when you plug it in

Pico PCB: firmware



Pico PCB: serial connection

```
$ picocom -b 115200 /dev/ttyACM0
Pico PCB Loader v0.15...
------------------------------
Welcome to the Pico PCB Board
Stage 1: Hardware
Stage 2: Car
Select challenge:
```

- When you connect, eyes light up yellow.
- Hardware challenge is out of scope (and it requires the physical badge).
- We **focus on stage 2**.

Pico PCB: Car

```
Starting challenge: Car

      ______     +-------------+
   __//_||_\__   |  Pico       |
  |     ||    |__| Car Status  |
  '--(_)--(_)-'  +-------------+

Lights: OFF Motor: OFF
-----------------------
 1. Turn lights ON
 2. Start engine

Enter your choice:
```

- Eyes light up red.
- We can turn lights ON and OFF.

Pico PCB: the car stalled

```
      ______     +------------+
   __//_||_\__   |  Pico      |
  |     ||    |__| Car Status |
  '--(_)--(_)-'  +------------+

Lights: OFF Motor: OFF
-----------------------
 1. Turn lights ON
 2. Start engine

Enter your choice: 2
 Ouch! The engine stalled!!!
```

We can't start the engine. No way. That's the challenge!

Pico PCB: what we know

- In stage 1, we retrieved the firmware (UF2) and extracted a *binary*.
- RP2040 has an ARM Cortex-M0+ processor. **Important: ARM, Little Endian, 32 bits**.
- **Base address: 0x10000000**.
- It's important to tell the disassembler (Ghidra).
- The source code is quite simple, but it's RP2040. That's the difficult part.

Lab 4: Pico PCB

`./lab/picopcb`

- Goal: **Solve it with Ghidra and Ghidra MCP**
- This lab is less guided.
- Don't ask your LLM to solve it in one go. It (probably) won't work.
- Think as you normally would, use Ghidra to understand what's happening.
- Ask the LLM to speed up some tasks: rename functions? explain an algorithm? what does this do? etc.
- Hint: Something is hidden, and you need to find it.

Agenda

Rust: why is it complicated?

- A simple *Hello World* in Rust weighs several MB.
- The compiler inserts safety checks, string formatting + uses static linking (no dependancy on libc).
- Memory safety is handled at compile-time by the borrow checker.

Consequences for reverse engineering

Over 500 functions... (`aflc`)

Strings are fat pointers

- Inline strings are **pointer + length**.
- Strings with type `String` have **pointer + length + capacity**.
- Consequently, the assembly doesn't directly show the string. There's an indirection.

```
lea rsi, reloc.fixup.Hello ; 0x555f8
[0x00008250]> px 16 @ 0x555f8
- offset -   F8F9 FAFB FCFD FEFF  0 1  2 3  4 5  6 7  89ABCDEF01234567
0x000555f8  cb72 0400 0000 0000 0700 0000 0000 0000  .r..............
[0x00008250]> px 7 @ 0x0472cb
- offset -   CBCC CDCE CFD0 D1D2 D3D4 D5D6 D7D8 D9DA  BCDEF0123456789A
0x000472cb  4865 6c6c 6f2c 20                        Hello,
```

Monomorphism

Polymorphism: 1 function, many types

- The compiler emits *one* piece of code, that works for many types.
- At runtime, use a **vtable** or *pointer indirection* to pick the right function.

Monomorphism: 1 function for each type

- **Rust compiler** takes *generic* code and produces *specialized* machine code for each type.
- No longer need vtable.
- e.g. `square()` for i32, another `square()` for f64...

Rust closures

```rust
fn main() {
    let x = 5;
    let add_x = |y| x + y;
    println!("{}", add_x(3));
}
```

- A **closure** captures the environment.
- `add_x(3)` returns 8.
- Compiler creates a hidden **struct** for `add_x`.
- The struct has a method named `call()`. Yes, Rust supports methods in structures...
- The compiler monomorphizes `call()` so that we have a **specialized** function that does the addition.

Calling convention

r2 command: `afci` to display the calling convention

On x86-64

- First argument in RDI
- Second argument in RSI
- Third argument in RDX
- ...
- Return in RAX. If it doesn't fit in a single register, sometimes, RDX is used too.

Rust macros

Rust's `println!` is a **macro**.

1. Format string: `core::fmt::Arguments`
2. Generic print: `std::io::_print`
3. Print it: `sym.std::io::stdio::stdout`

Read *Bite Sized Rust RE: 1 Deconstructing Hello World* 🔗

Rust Desugaring

Functions that work over an object,
e.g `myobject.blah()`

are often *desugared* by the **compiler** as
`class::blah(&myobject)`.
Turned into more fundamental core language.

Indirect structure return

When a function needs to return a big structure such as a vector, the compiler uses **Indirect return**:

- The value is *not* returned in RAX.
- Instead, space is allocated **on the stack**.
- and that space is provided as *first argument* to the function.

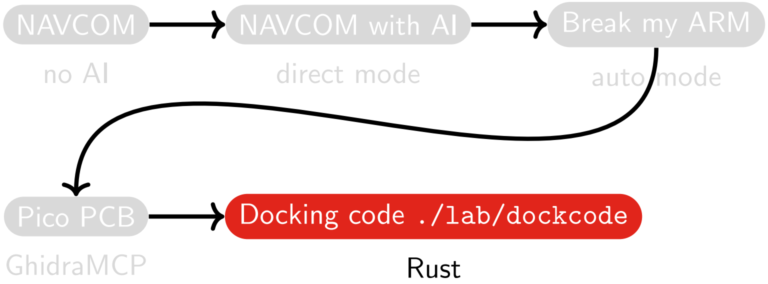
Example of indirect return

```
struct Pair { a: i32, b: i32 }
fn make_pair(x: i32, y: i32) -> Pair {
    Pair { a: x, b: y }
}
```

The pair to return is allocated on the stack

```
sub     rsp, 16                 ; allocate space for return struct Pair on
↪   stack
mov     esi, 5                  ; x = 5
mov     edx, 7                  ; y = 7
lea     rdi, [rsp]              ; pointer to return space
call    make_pair               ; call function
```

Lab 5: reversing Rust



```
$ ./dockcode
Enter docking authorization code:
```

Thank You

- https://github.com/radareorg/r2ai
- @cryptax (Blue Sky, Mastodon, Discord)