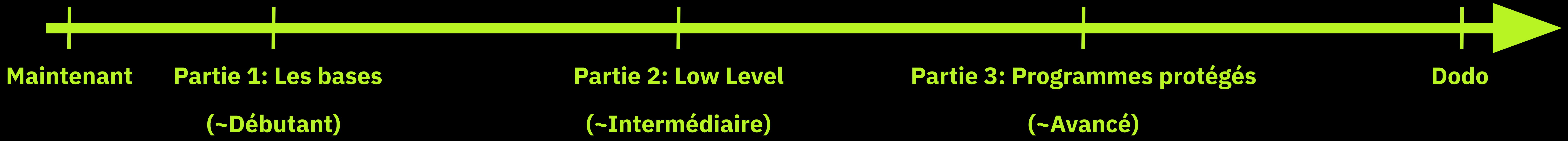
Intro au Reverse Engineering

Timeline


Maintenant
Partie 1: Les bases
(~Débutant)
Partie 2: Low Level
(~Intermédiaire)
Partie 3: Programmes protégés
(~Avancé)
Dodo

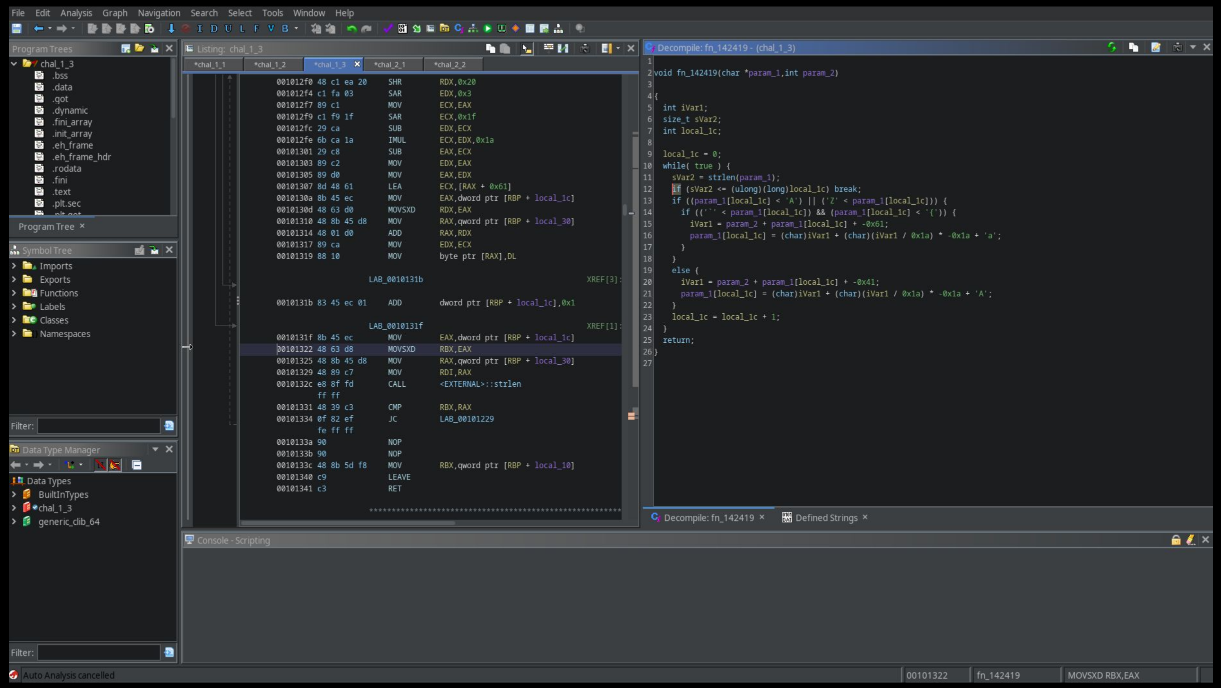
Partie 1: Les bases

Le Reverse Engineering, c’est quoi?

Analyse Statique

Analyse Dynamique

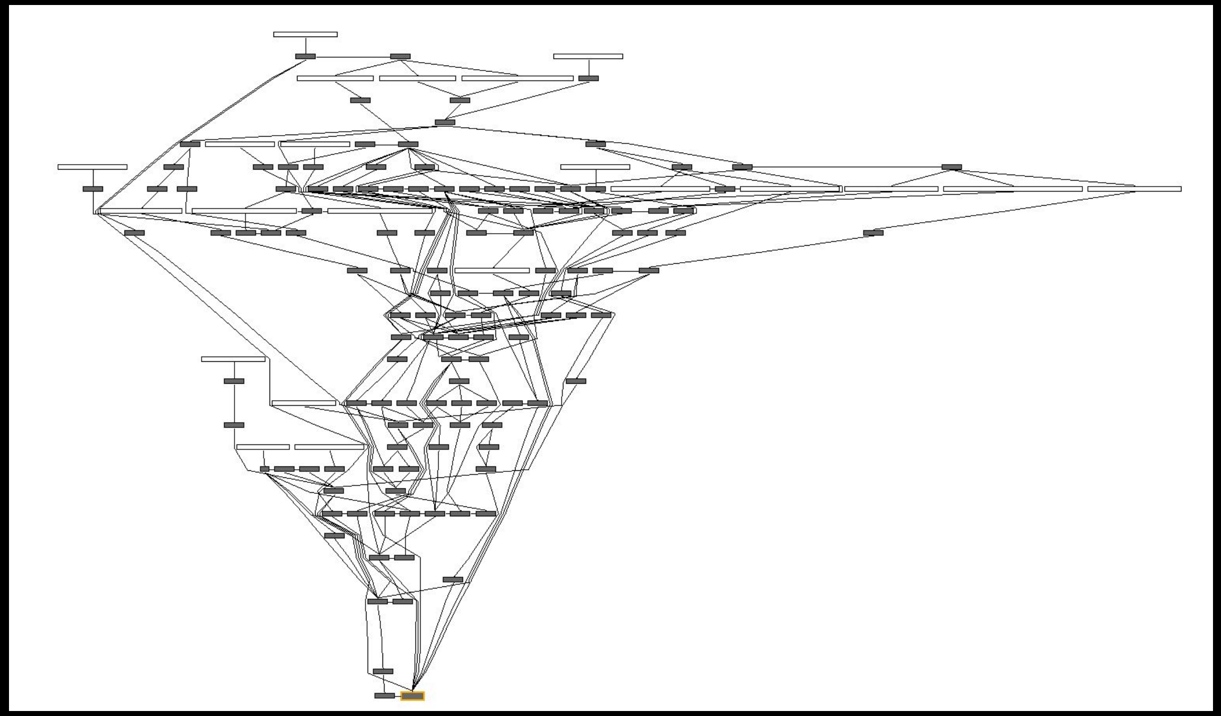
Analyse Statique



```
001012f0 48 c1 ea 20     SHR        RDX,0x20
001012f4 c1 fa 03        SAR        EDX,0x3
001012f7 89 c1           MOV        ECX,EAX
001012f9 c1 f9 1f        SAR        ECX,0x1f
001012fc 29 ca           SUB        EDX,ECX
001012fe 6b ca 1a        IMUL       ECX,EDX,0x1a
00101301 29 c8           SUB        EAX,ECX
00101303 89 c2           MOV        EDX,EAX
00101305 89 d0           MOV        EAX,EDX
00101307 8d 48 61        LEA        ECX,[RAX + 0x61]
0010130a 8b 45 ec        MOV        EAX,dword ptr [RBP + local_1c]
0010130d 48 63 d0        MOVSXD     RDX,EAX
00101310 48 8b 45 d8     MOV        RAX,qword ptr [RBP + local_30]
00101314 48 01 d0        ADD        RAX,RDX
00101317 89 ca           MOV        EDX,ECX
00101319 88 10           MOV        byte ptr [RAX],DL

                     LAB_0010131b                              XREF[3]:

0010131b 83 45 ec 01     ADD        dword ptr [RBP + local_1c],0x1

                     LAB_0010131f                              XREF[1]:
0010131f 8b 45 ec        MOV        EAX,dword ptr [RBP + local_1c]
00101322 48 63 d8        MOVSXD     RBX,EAX
00101325 48 8b 45 d8     MOV        RAX,qword ptr [RBP + local_30]
00101329 48 89 c7        MOV        RDI,RAX
0010132c e8 8f fd        CALL       <EXTERNAL>::strlen
         ff ff
00101331 48 39 c3        CMP        RBX,RAX
00101334 0f 82 ef        JC         LAB_00101229
         fe ff ff
0010133a 90              NOP
0010133b 90              NOP
0010133c 48 8b 5d f8     MOV        RBX,qword ptr [RBP + local_10]
00101340 c9              LEAVE
00101341 c3              RET
```

Decompile: fn_142419 - (chal_1_3)

```c
void fn_142419(char *param_1,int param_2)

{
  int iVar1;
  size_t sVar2;
  int local_1c;

  local_1c = 0;
  while( true ) {
    sVar2 = strlen(param_1);
    if (sVar2 <= (ulong)(long)local_1c) break;
    if ((param_1[local_1c] < 'A') || ('Z' < param_1[local_1c])) {
      if (('`' < param_1[local_1c]) && (param_1[local_1c] < '{')) {
        iVar1 = param_2 + param_1[local_1c] + -0x61;
        param_1[local_1c] = (char)iVar1 + (char)(iVar1 / 0x1a) * -0x1a + 'a';
      }
    }
    else {
      iVar1 = param_2 + param_1[local_1c] + -0x41;
      param_1[local_1c] = (char)iVar1 + (char)(iVar1 / 0x1a) * -0x1a + 'A';
    }
    local_1c = local_1c + 1;
  }
  return;
}
```



Les Outils

- IDA
- Ghidra
- Binary Ninja
- Radare 2
- …

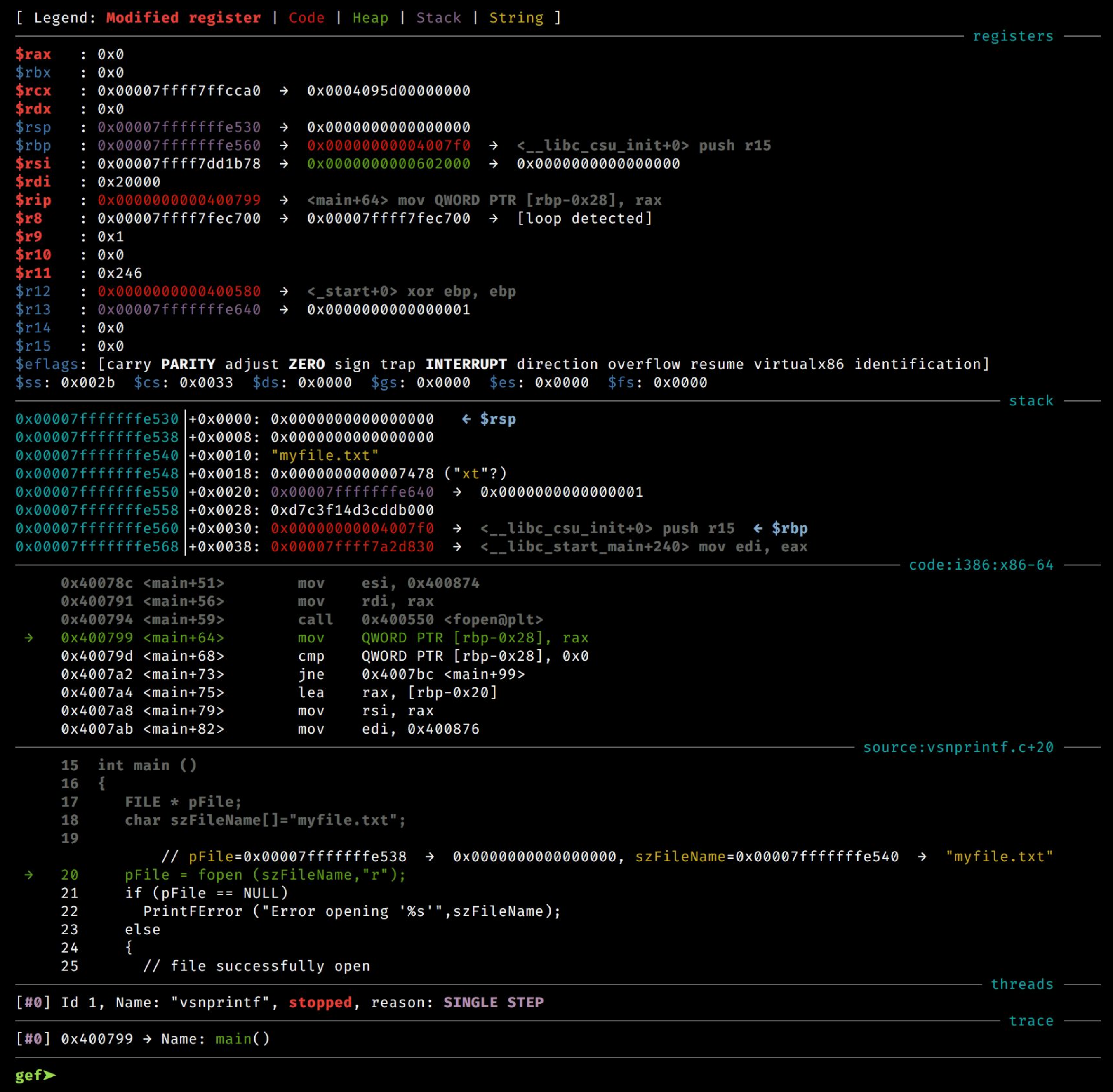
Les Techniques

- Imports/Exports
- Strings
- Xrefs
- Graph to/Graph From

L'incontournable

- RTFM!
- Une bonne partie des réponses se trouvent en ligne
- Man pages

Analyse Dynamique



```
[ Legend: Modified register | Code | Heap | Stack | String ]
────────────────────────────────────────────────────────────────────────────── registers ────
$rax   : 0x0
$rbx   : 0x0
$rcx   : 0x00007ffff7ffcca0  →  0x0004095d00000000
$rdx   : 0x0
$rsp   : 0x00007fffffffe530  →  0x0000000000000000
$rbp   : 0x00007fffffffe560  →  0x00000000004007f0  →  <__libc_csu_init+0> push r15
$rsi   : 0x00007ffff7dd1b78  →  0x0000000000602000  →  0x0000000000000000
$rdi   : 0x20000
$rip   : 0x0000000000400799  →  <main+64> mov QWORD PTR [rbp-0x28], rax
$r8    : 0x00007ffff7fec700  →  0x00007ffff7fec700  →  [loop detected]
$r9    : 0x1
$r10   : 0x0
$r11   : 0x246
$r12   : 0x0000000000400580  →  <_start+0> xor ebp, ebp
$r13   : 0x00007fffffffe640  →  0x0000000000000001
$r14   : 0x0
$r15   : 0x0
$eflags: [carry PARITY adjust ZERO sign trap INTERRUPT direction overflow resume virtualx86 identification]
$ss: 0x002b  $cs: 0x0033  $ds: 0x0000  $gs: 0x0000  $es: 0x0000  $fs: 0x0000
──────────────────────────────────────────────────────────────────────────────── stack ────
0x00007fffffffe530│+0x0000: 0x0000000000000000   ← $rsp
0x00007fffffffe538│+0x0008: 0x0000000000000000
0x00007fffffffe540│+0x0010: "myfile.txt"
0x00007fffffffe548│+0x0018: 0x0000000000007478 ("xt"?)
0x00007fffffffe550│+0x0020: 0x00007fffffffe640  →  0x0000000000000001
0x00007fffffffe558│+0x0028: 0xd7c3f14d3cddb000
0x00007fffffffe560│+0x0030: 0x00000000004007f0  →  <__libc_csu_init+0> push r15  ← $rbp
0x00007fffffffe568│+0x0038: 0x00007ffff7a2d830  →  <__libc_start_main+240> mov edi, eax
────────────────────────────────────────────────────────────────────────── code:i386:x86-64 ────
     0x40078c <main+51>        mov    esi, 0x400874
     0x400791 <main+56>        mov    rdi, rax
     0x400794 <main+59>        call   0x400550 <fopen@plt>
 →   0x400799 <main+64>        mov    QWORD PTR [rbp-0x28], rax
     0x40079d <main+68>        cmp    QWORD PTR [rbp-0x28], 0x0
     0x4007a2 <main+73>        jne    0x4007bc <main+99>
     0x4007a4 <main+75>        lea    rax, [rbp-0x20]
     0x4007a8 <main+79>        mov    rsi, rax
     0x4007ab <main+82>        mov    edi, 0x400876
───────────────────────────────────────────────────────────────────────── source:vsnprintf.c+20 ────
     15  int main ()
     16  {
     17     FILE * pFile;
     18     char szFileName[]="myfile.txt";
     19
             // pFile=0x00007fffffffe538  →  0x0000000000000000, szFileName=0x00007fffffffe540  →  "myfile.txt"
 →   20     pFile = fopen (szFileName,"r");
     21     if (pFile == NULL)
     22       PrintFError ("Error opening '%s'",szFileName);
     23     else
     24     {
     25       // file successfully open
────────────────────────────────────────────────────────────────────────────── threads ────
[#0] Id 1, Name: "vsnprintf", stopped, reason: SINGLE STEP
──────────────────────────────────────────────────────────────────────────────── trace ────
[#0] 0x400799 → Name: main()
─────────────────────────────────────────────────────────────────────────────────────────────
gef➤
```

Les Outils

- GDB (GEF), x64dbg
- ltrace, strace, ptrace
- Wireshark
- Sandboxes (unpac.me, ...)
- Émulateurs
- ...

Les Techniques

- Breakpoints
- Memory dump
- Instrumentation du programme

Environnement d'analyse

- Machine virtuelle (VirtualBox, VMWare, etc)
- Contrôle du trafic réseau (INetSim)
- Monitoring (réseau, processus, API calls, registry, ...)

Démo

Hack Time!

Ghidra quick tips:

1. [Decompilateur] > Edit > Tool Options > Listing Fields > Cursor Text Highlighting > Mouse Button To Activate > LEFT

2. [Ghidra Toolbox] > Edit > Theme > Switch Theme > Flat Dark Theme

Ghidra:

GEF:

gdb:

```
sudo apt update &&     \
sudo apt install gdb -y
```

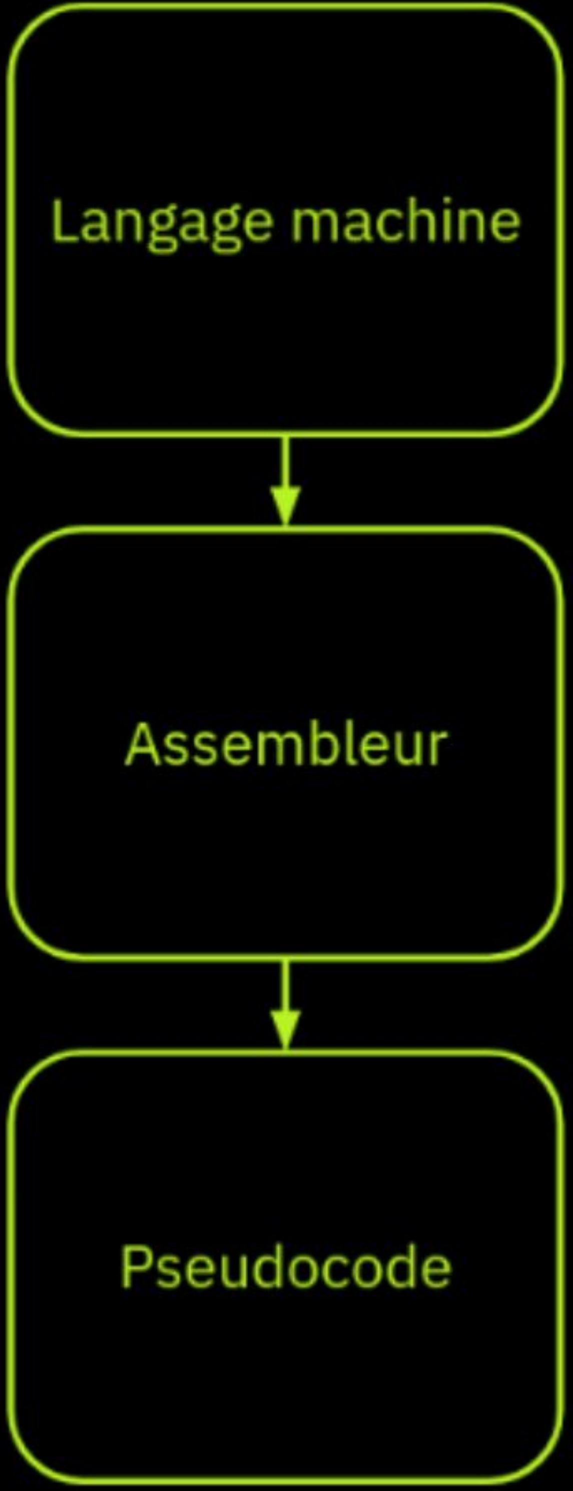
github.com/NationalSecurityAgency/ghidra/releases

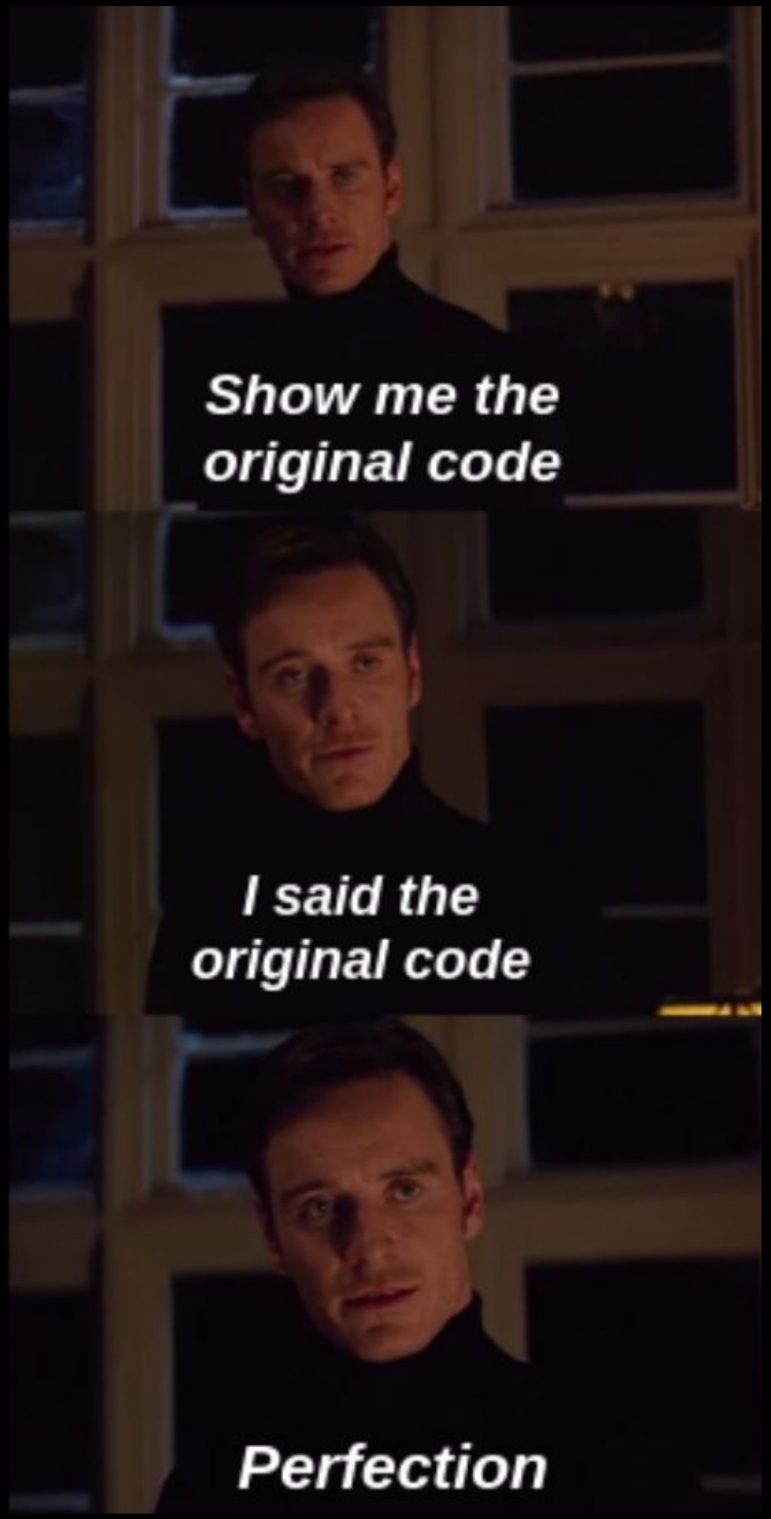
github.com/hugsy/gef

Les défis sont ici: https//github.com/1t1n1/AIRE

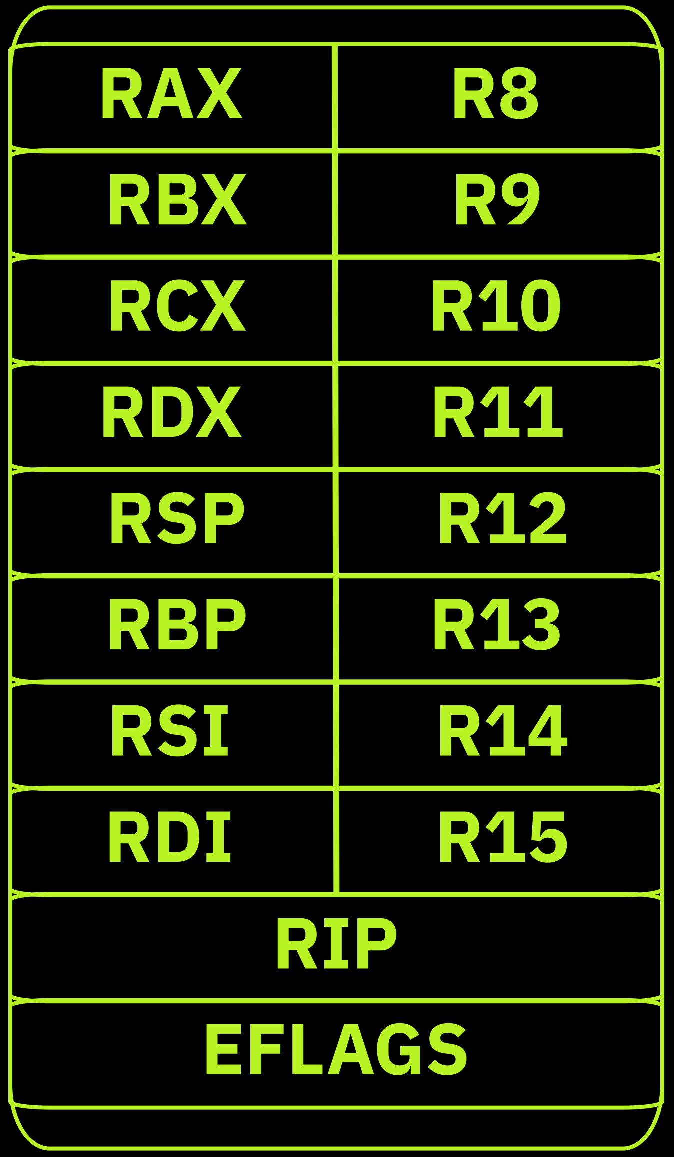
Partie 2: Low level

Décompilation

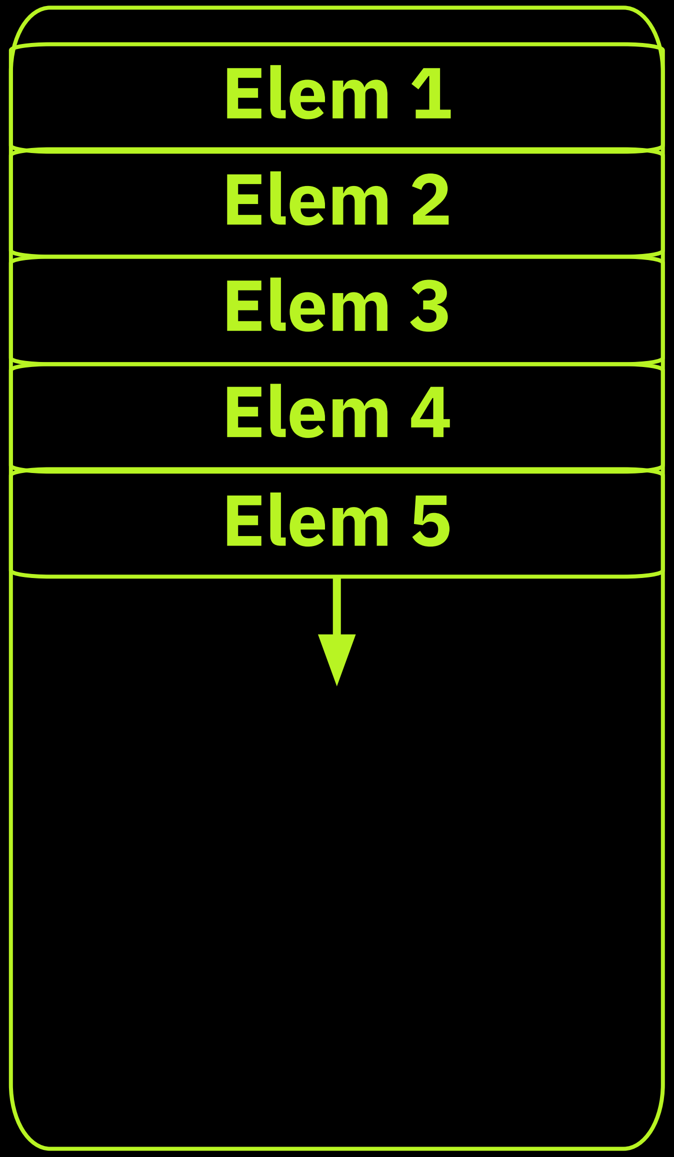




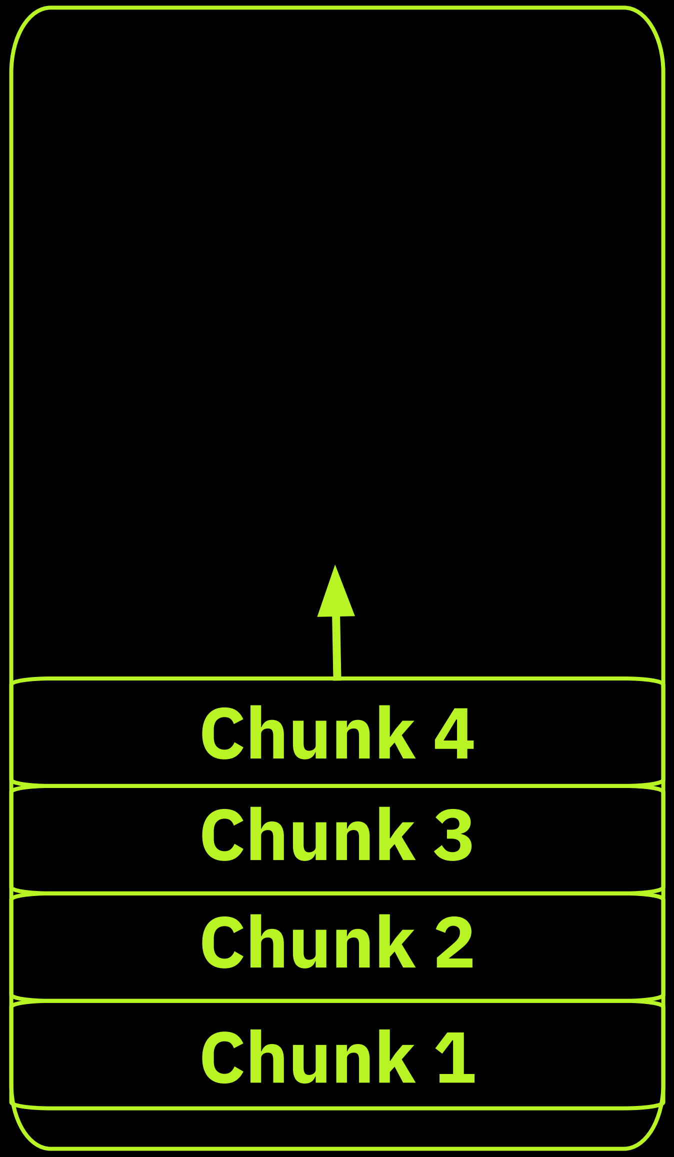
L’assembleur



Registres



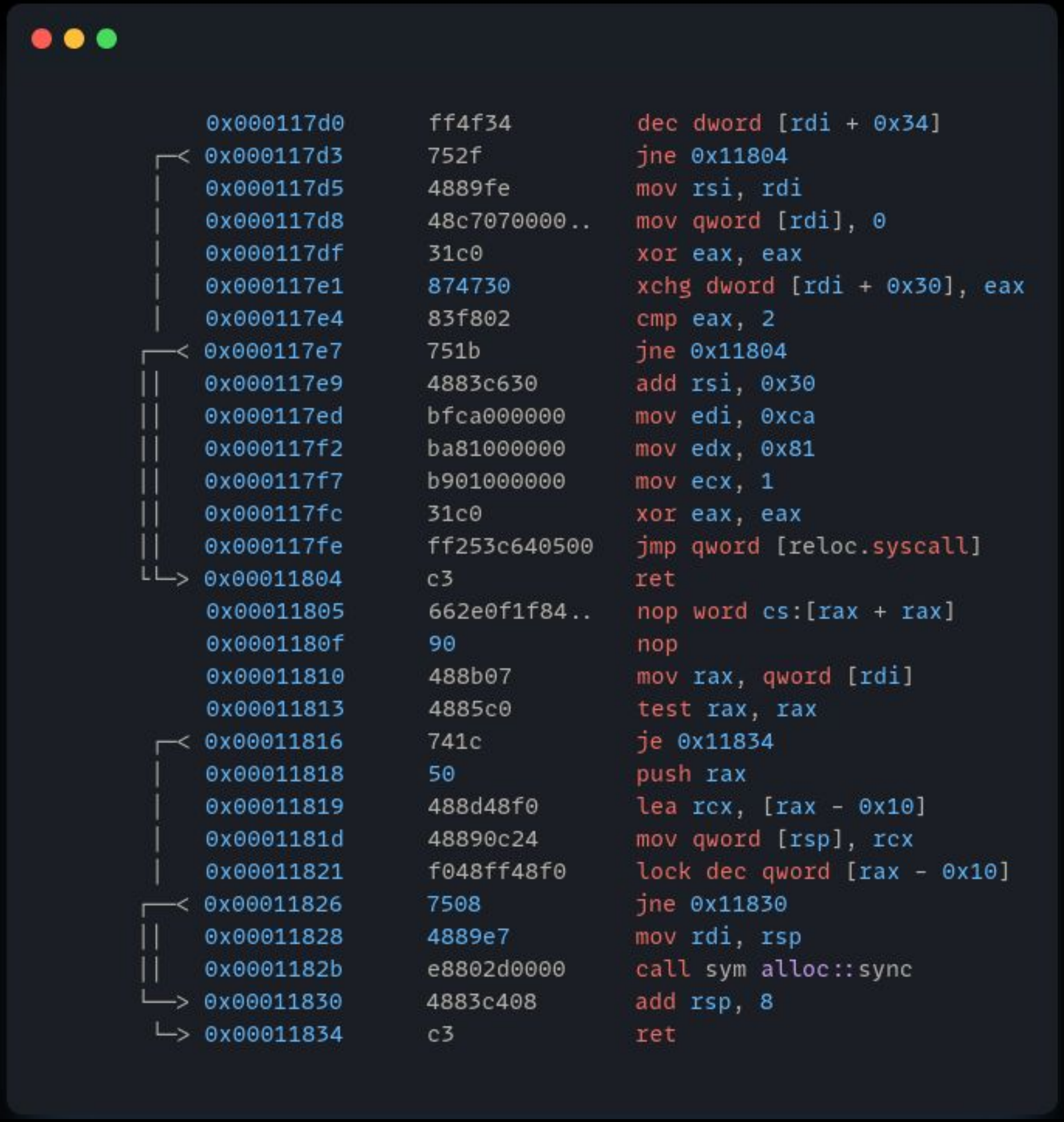
Stack



Heap

L’assembleur

ADD **AND**
CALL **CMP**
DIV **INT**
LEA **LOOP**
MOV **MUL**
NOT **OR**
POP **PUSH**
RET **XOR**
... **...**



```
            0x000117d0      ff4f34          dec dword [rdi + 0x34]
     ┌─< 0x000117d3      752f            jne 0x11804
     │   0x000117d5      4889fe          mov rsi, rdi
     │   0x000117d8      48c7070000..    mov qword [rdi], 0
     │   0x000117df      31c0            xor eax, eax
     │   0x000117e1      874730          xchg dword [rdi + 0x30], eax
     │   0x000117e4      83f802          cmp eax, 2
    ┌──< 0x000117e7      751b            jne 0x11804
    ││   0x000117e9      4883c630        add rsi, 0x30
    ││   0x000117ed      bfca000000      mov edi, 0xca
    ││   0x000117f2      ba81000000      mov edx, 0x81
    ││   0x000117f7      b901000000      mov ecx, 1
    ││   0x000117fc      31c0            xor eax, eax
    ││   0x000117fe      ff253c640500    jmp qword [reloc.syscall]
    └└─> 0x00011804      c3              ret
         0x00011805      662e0f1f84..    nop word cs:[rax + rax]
         0x0001180f      90              nop
         0x00011810      488b07          mov rax, qword [rdi]
         0x00011813      4885c0          test rax, rax
     ┌─< 0x00011816      741c            je 0x11834
     │   0x00011818      50              push rax
     │   0x00011819      488d48f0        lea rcx, [rax - 0x10]
     │   0x0001181d      48890c24        mov qword [rsp], rcx
     │   0x00011821      f048ff48f0      lock dec qword [rax - 0x10]
    ┌──< 0x00011826      7508            jne 0x11830
    ││   0x00011828      4889e7          mov rdi, rsp
    ││   0x0001182b      e8802d0000      call sym alloc::sync
    └──> 0x00011830      4883c408        add rsp, 8
     └─> 0x00011834      c3              ret
```

Le langage machine

xor eax, eax	=	0x31C0
cmp rax, 0x15	=	0x4883F815
mov eax, 1	=	0xB801000000

Boutisme (endianness)

Little-endian

0x78	0x56	0x34	0x12
0x0100	0x0101	0x0102	0x0103

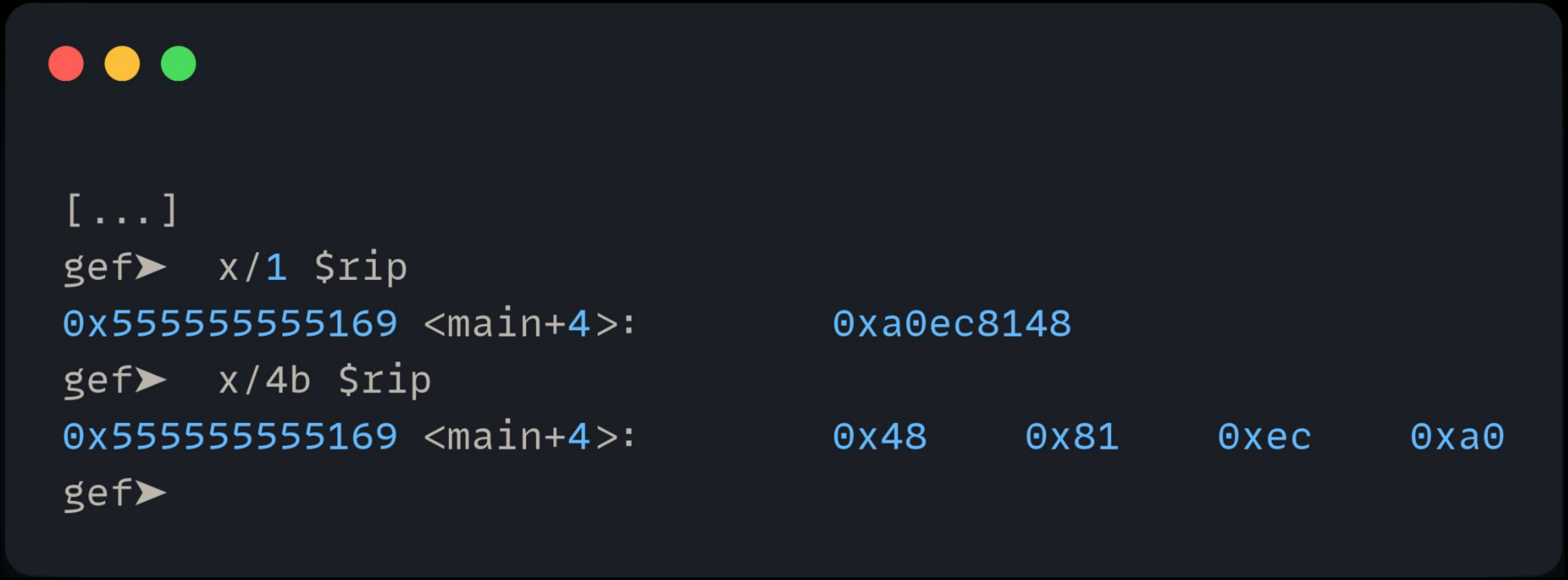
0x12345678

= 305419896

Big-endian

0x12	0x34	0x56	0x78
0x0100	0x0101	0x0102	0x0103

Boutisme (endianness)



```
[...]
gef➤  x/1 $rip
0x555555555169 <main+4>:        0xa0ec8148
gef➤  x/4b $rip
0x555555555169 <main+4>:        0x48    0x81    0xec    0xa0
gef➤
```

Hack Time!

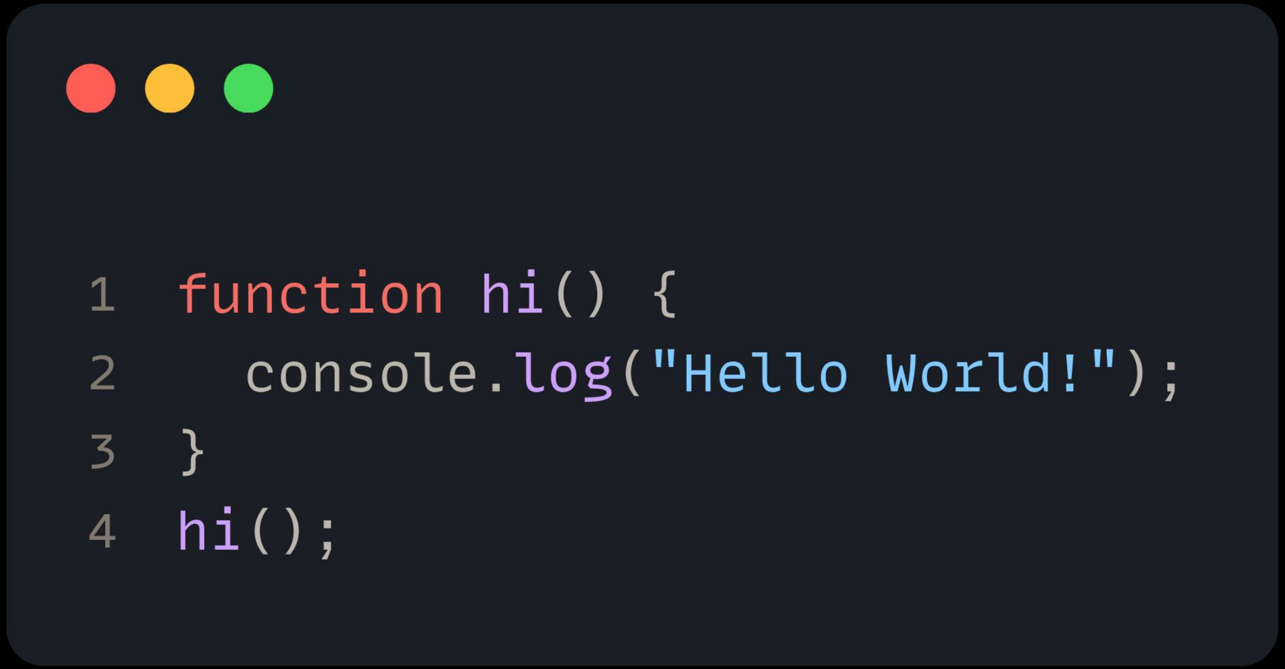
https://felixcloutier.com/x86

Les défis sont ici: https//github.com/1t1n1/AIRE

Partie 3: Programmes protégés

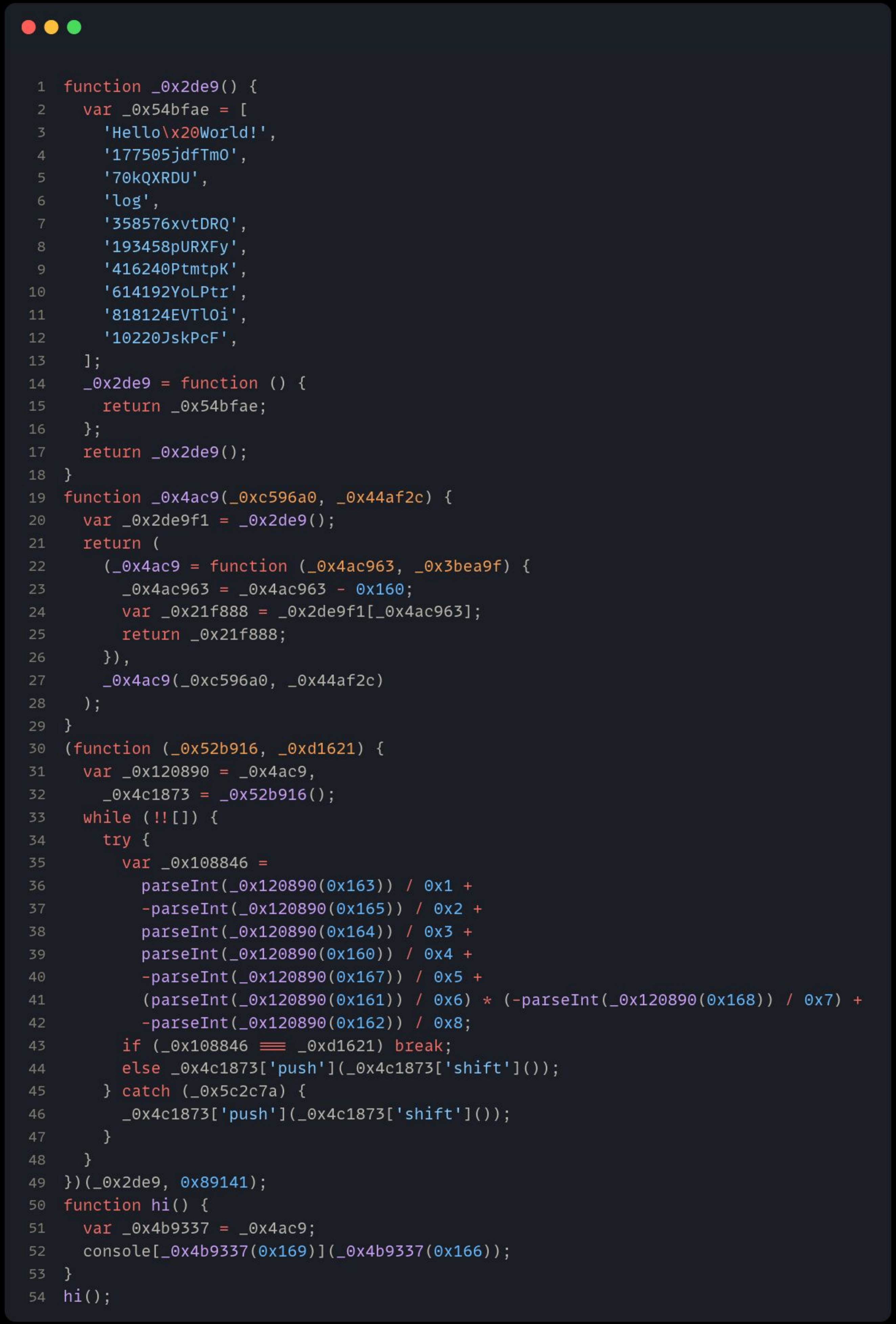
1t1n1

2024



```
function hi() {
  console.log("Hello World!");
}
hi();
```

==



```
function _0x2de9() {
  var _0x54bfae = [
    'Hello\x20World!',
    '177505jdfTmO',
    '70kQXRDU',
    'log',
    '358576xvtDRQ',
    '193458pURXFy',
    '416240PtmtpK',
    '614192YoLPtr',
    '818124EVTlOi',
    '10220JskPcF',
  ];
  _0x2de9 = function () {
    return _0x54bfae;
  };
  return _0x2de9();
}
function _0x4ac9(_0xc596a0, _0x44af2c) {
  var _0x2de9f1 = _0x2de9();
  return (
    (_0x4ac9 = function (_0x4ac963, _0x3bea9f) {
      _0x4ac963 = _0x4ac963 - 0x160;
      var _0x21f888 = _0x2de9f1[_0x4ac963];
      return _0x21f888;
    }),
    _0x4ac9(_0xc596a0, _0x44af2c)
  );
}
(function (_0x52b916, _0xd1621) {
  var _0x120890 = _0x4ac9,
    _0x4c1873 = _0x52b916();
  while (!![]) {
    try {
      var _0x108846 =
        parseInt(_0x120890(0x163)) / 0x1 +
        -parseInt(_0x120890(0x165)) / 0x2 +
        parseInt(_0x120890(0x164)) / 0x3 +
        parseInt(_0x120890(0x160)) / 0x4 +
        -parseInt(_0x120890(0x167)) / 0x5 +
        (parseInt(_0x120890(0x161)) / 0x6) * (-parseInt(_0x120890(0x168)) / 0x7) +
        -parseInt(_0x120890(0x162)) / 0x8;
      if (_0x108846 === _0xd1621) break;
      else _0x4c1873['push'](_0x4c1873['shift']());
    } catch (_0x5c2c7a) {
      _0x4c1873['push'](_0x4c1873['shift']());
    }
  }
})(_0x2de9, 0x89141);
function hi() {
  var _0x4b9337 = _0x4ac9;
  console[_0x4b9337(0x169)](_0x4b9337(0x166));
}
hi();
```

Anti-Analyse

Obfuscation

- Opaque Predicate
- Stack-Strings
- Control Flow Flattening (CFF)
- Virtual Machines (custom instruction set)
- ...

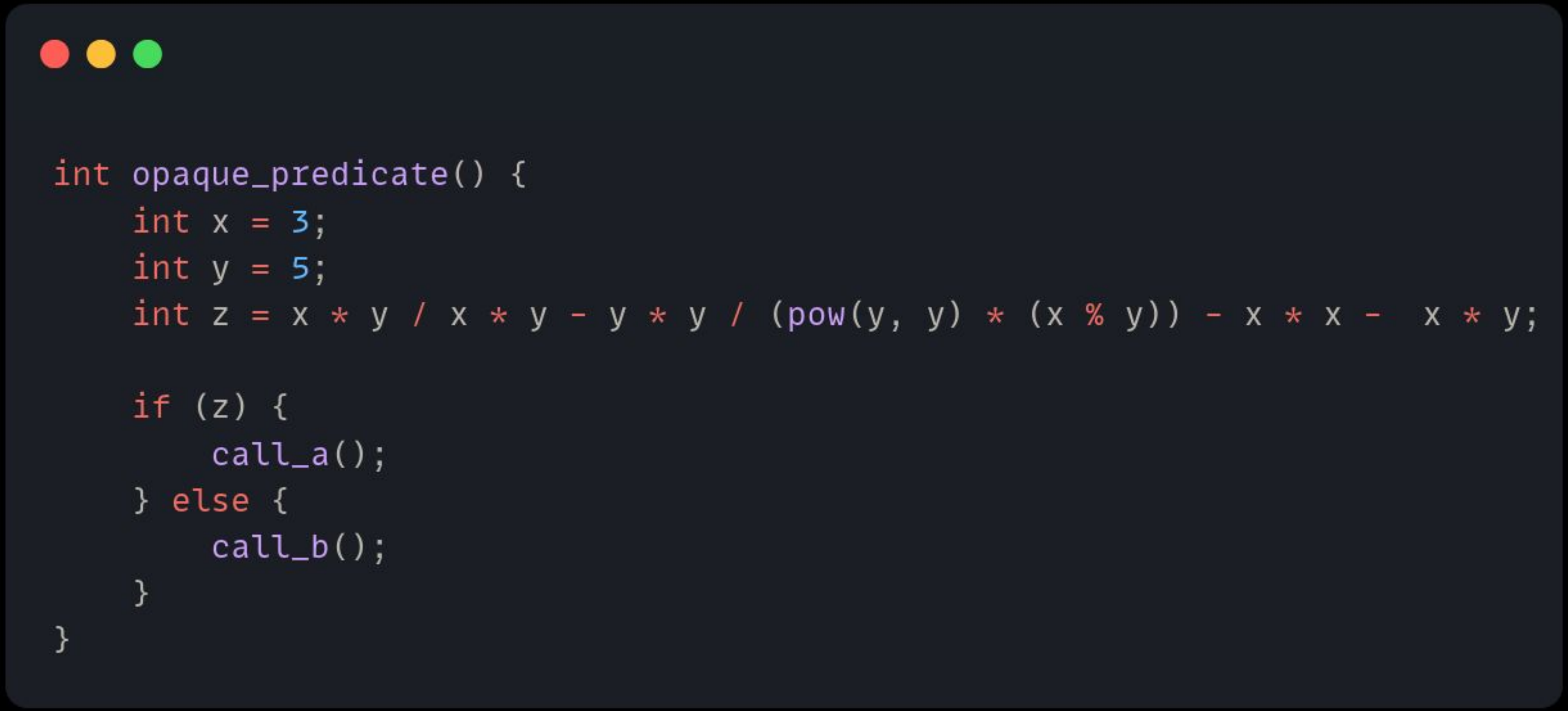
Anti-Debugging

- IsDebuggerPresent()
- GetTickCount()
- NtQueryInformationProcess()
- ScyllaHide / TitanHide
- ...

Anti-VM

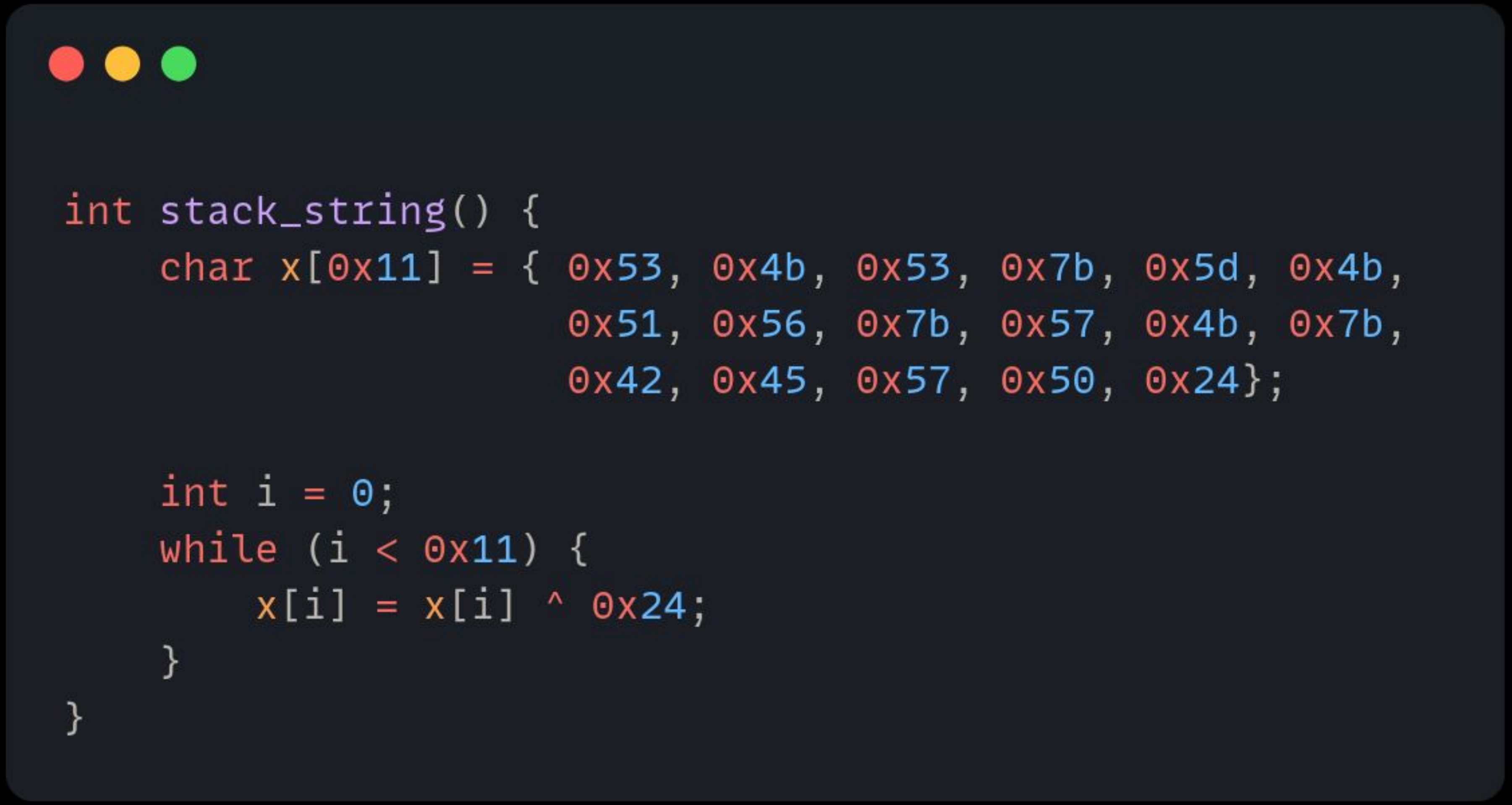
- Hostname / Username
- Registry Keys
- CPUID
- IN
- ...

Obfuscation



```c
int opaque_predicate() {
    int x = 3;
    int y = 5;
    int z = x * y / x * y - y * y / (pow(y, y) * (x % y)) - x * x -  x * y;

    if (z) {
        call_a();
    } else {
        call_b();
    }
}
```

Obfuscation

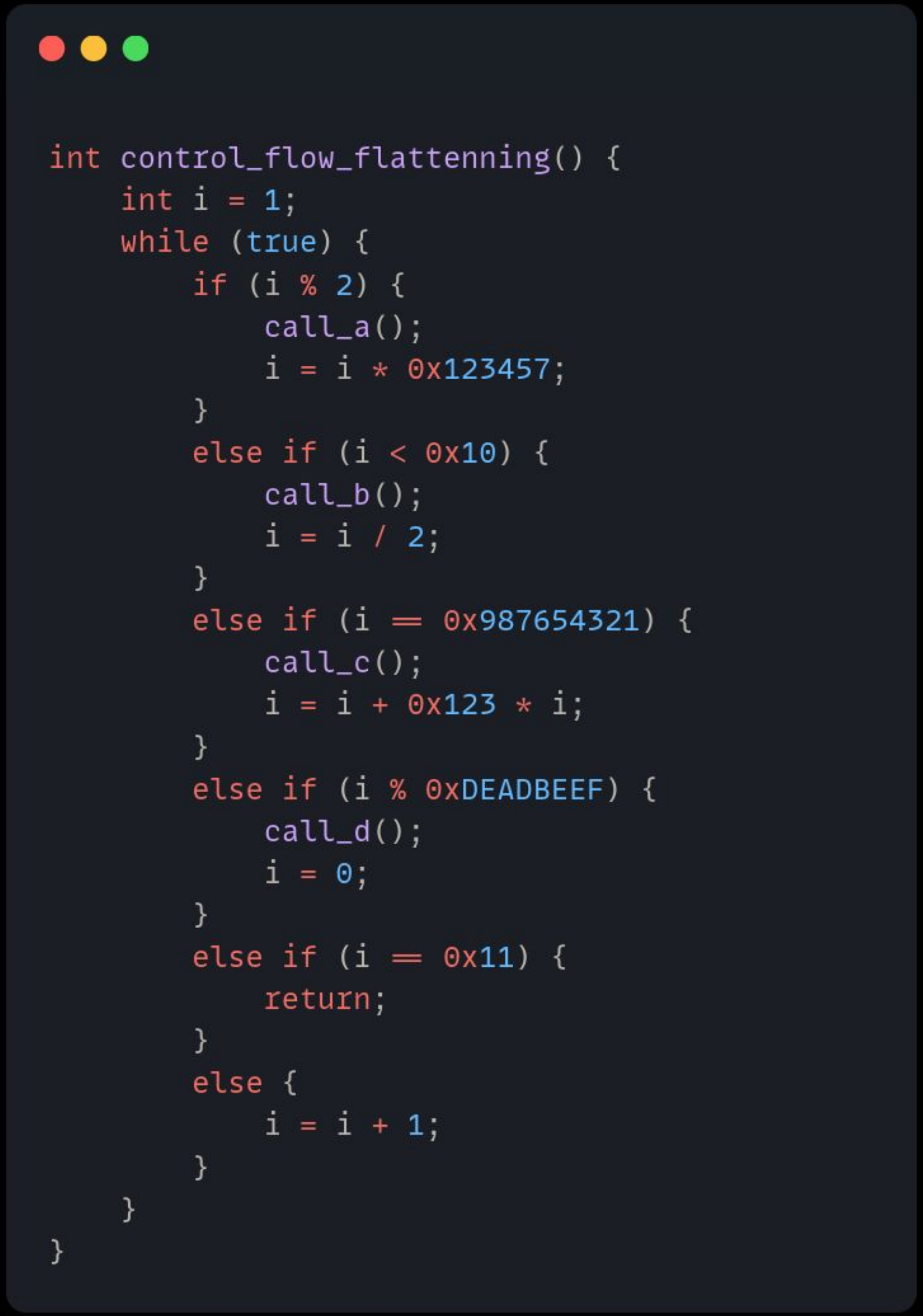


```c
int stack_string() {
    char x[0x11] = { 0x53, 0x4b, 0x53, 0x7b, 0x5d, 0x4b,
                     0x51, 0x56, 0x7b, 0x57, 0x4b, 0x7b,
                     0x42, 0x45, 0x57, 0x50, 0x24};

    int i = 0;
    while (i < 0x11) {
        x[i] = x[i] ^ 0x24;
    }
}
```

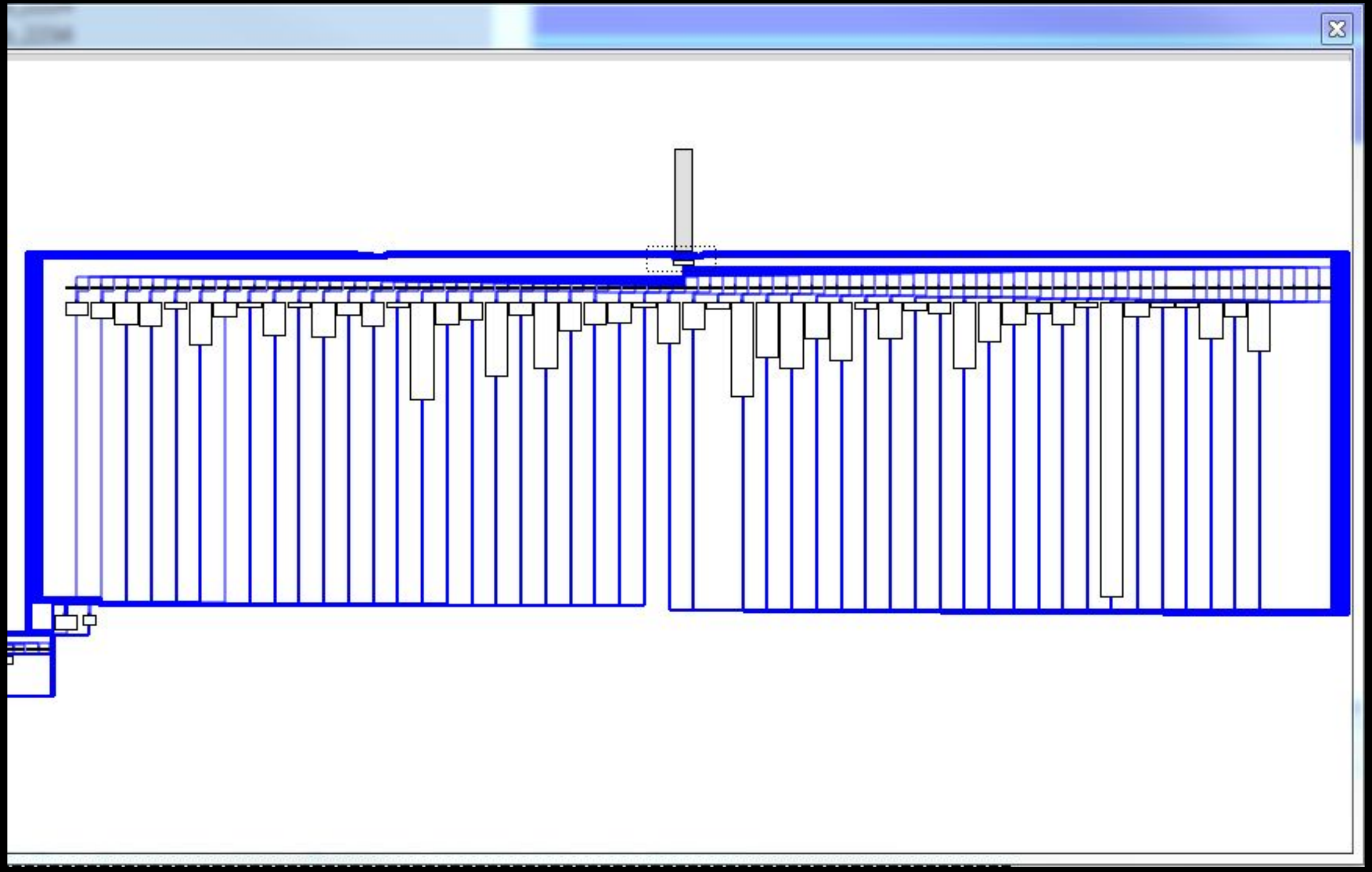
1t1n1

2024

Obfuscation



```c
int control_flow_flattenning() {
    int i = 1;
    while (true) {
        if (i % 2) {
            call_a();
            i = i * 0x123457;
        }
        else if (i < 0x10) {
            call_b();
            i = i / 2;
        }
        else if (i == 0x987654321) {
            call_c();
            i = i + 0x123 * i;
        }
        else if (i % 0xDEADBEEF) {
            call_d();
            i = 0;
        }
        else if (i == 0x11) {
            return;
        }
        else {
            i = i + 1;
        }
    }
}
```



Hack Time!

Les défis sont ici: https//github.com/1t1n1/AIRE

That's all Folks!