WineRump 🍇 2024

Recherche de patterns en représentation intermédiaire appliquée aux malwares -

Pezier Pierre-Henri

Génération “Classique” de signatures

Repérage de code spécifique

```
mov     rax, 0B1D0B7848570F29h
lea     rdi, [rbp+var_330] ; unsigned __int8 *
mov     [rdi], rax
mov     byte ptr [rdi+0Ah], 4Eh ; 'N'
mov     word ptr [rdi+8], 1F0Eh
mov     esi, 0Bh        ; unsigned int
call    __Z8GetTrickPhj ; GetTrick(uchar *,uint)
mov     rax, 0B052930565C0521h
lea     rdi, [rbp+var_340] ; unsigned __int8 *
mov     [rdi], rax
mov     byte ptr [rdi+0Ah], 32h ; '2'
mov     word ptr [rdi+8], 1417h
mov     esi, 0Bh        ; unsigned int
call    __Z8GetTrickPhj ; GetTrick(uchar *,uint)
mov     rbx, cs:selRef_stringWithCString_encoding_
mov     rdi, cs:classRef_NSString ; id
lea     rdx, [rbp+var_330]
mov     ecx, 1
```

```
starting spotbbl
Found suspicious routine at: 0x100002f25 in function: "sub_100002F16" 4c89c831d248f7f68a04174330040849ffc14c39c975e9
Found suspicious routine at: 0x1000044f6 in function: "sub_10000430F" 89d683e60f8a5c359041301c1648ffc24839d175eb
Found suspicious routine at: 0x100004aef in function: "sub_100004568" be01000000ba000010004c89e74c89e9e81e5f0100488985
Found suspicious routine at: 0x10000518e in function: "sub_100004F84" 89c183e10f8a8c0d20ffffff41300c0648ffc04939c475e8
Found suspicious routine at: 0x100005563 in function: "sub_100004F84" 89ca83e20f8a941520ffffff30140b48ffc14839c875e9
Found suspicious routine at: 0x100009915 in function: "sub_1000097D0" 48c745a0010000004c89e7488d759c4c89f231c94531c045
Found suspicious routine at: 0x100009a32 in function: "sub_1000097D0" 89c1c1e918884c1dbd48ffc3c1e0084883fb0475eb
Found suspicious routine at: 0x100009ac3 in function: "sub_1000097D0" c1e208488b7580c1e8104803b570ffffff884431f489d048
Found suspicious routine at: 0x1000127e9 in function: "sub_1000126BF" 0fb78495c0eefff0fb6f0ff84b5d0f7ffffc1e8060d0004
Found suspicious routine at: 0x10001286d in function: "sub_1000126BF" 0fb7049748d3e80fb6c08bb485d0f3ffff8d5e01899c85d0
Found suspicious routine at: 0x100012fd3 in function: "sub_100012FC4" 4c89c831d248f7f1418a04104230040f49ffc14c39ce75e8
Found suspicious routine at: 0x1000130be in function: "sub_100012FED" 89c183e11f8a0c0b41300c0748ffc04939c675ec
Found suspicious routine at: 0x100017a2d in function: "sub_100017940" 89ca83e21f418a141730140b48ffc14839c875ec
Found suspicious routine at: 0x10001906f in function: "sub_100018210" 31d2a8010f94c2c1e21981caffffff0121ca299485f0fdff
Found suspicious routine at: 0x1000199d5 in function: "sub_1000199BF"
```

```
33          for segea in idautils.Segments():
34              for funcea in idautils.Functions(segea, idc.get_segm_end(segea)):
35                  for bb in idaapi.FlowChart(idaapi.get_func(funcea)):
36
37                      last_inst = idc.prev_head(bb.end_ea)
38                      first_inst = idc.next_head(bb.start_ea)
39
40                      if idc.print_insn_mnem(last_inst).startswith('j') \
41                      and idc.print_insn_mnem(last_inst) != 'jmp'\
42                      and idc.get_operand_type(last_inst, 1) == idc.o_void\
43                      and idc.get_operand_value(last_inst, 0) == bb.start_ea:
44                          while inst != first_inst:
45                              inst = idc.prev_head(inst)
46                              if idc.print_insn_mnem(inst) in {'xor', 'shl', 'shr'}\
47                              and idc.get_operand_type(inst, 0) in {idc.o_phrase, idc.o_displ}:
48                                  bytecode = binascii.hexlify(
49                                      idaapi.get_bytes(bb.start_ea, bb.end_ea - bb.start_ea)
50                                      ).decode('UTF-8')
51                                  print(f'Found suspicious routine at: 0x{bb.start_ea:08x} in function: "{idc.get_func_name(funcea)}" {bytec
```

Exemple de code de désobfuscation

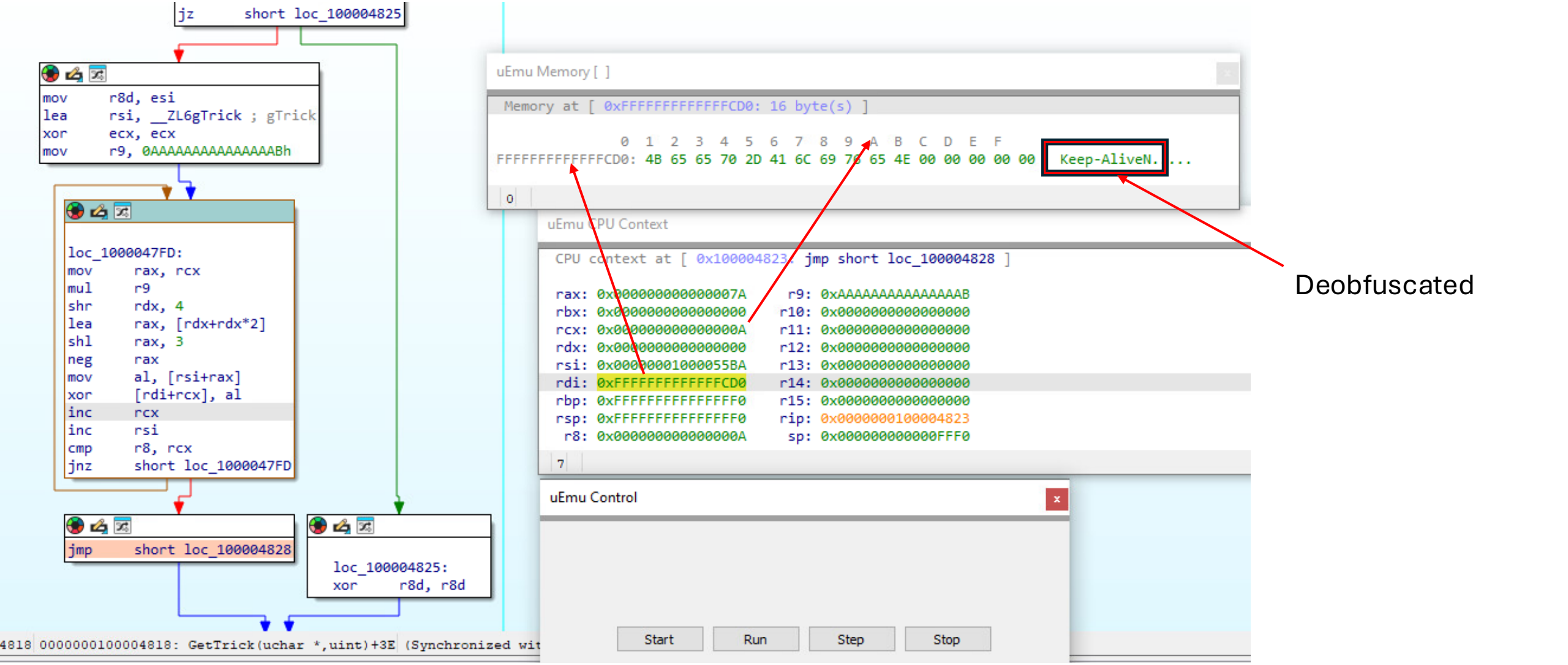
```
mov     rax, cs:_objc_release_ptr
mov     rbx, rax
call    rax ; _objc_release
mov     rdi, r15
call    rbx
mov     rax, 0B1D0B7848570F29h
lea     rdi, [rbp+var_330] ; unsigned __int8 *
mov     [rdi], rax
mov     byte ptr [rdi+0Ah], 4Eh ; 'N'
mov     word ptr [rdi+8], 1F0Eh
mov     esi, 0Bh            ; unsigned int
call    __Z8GetTrickPhj ; GetTrick(uchar *,uint)
```

Obfuscated string

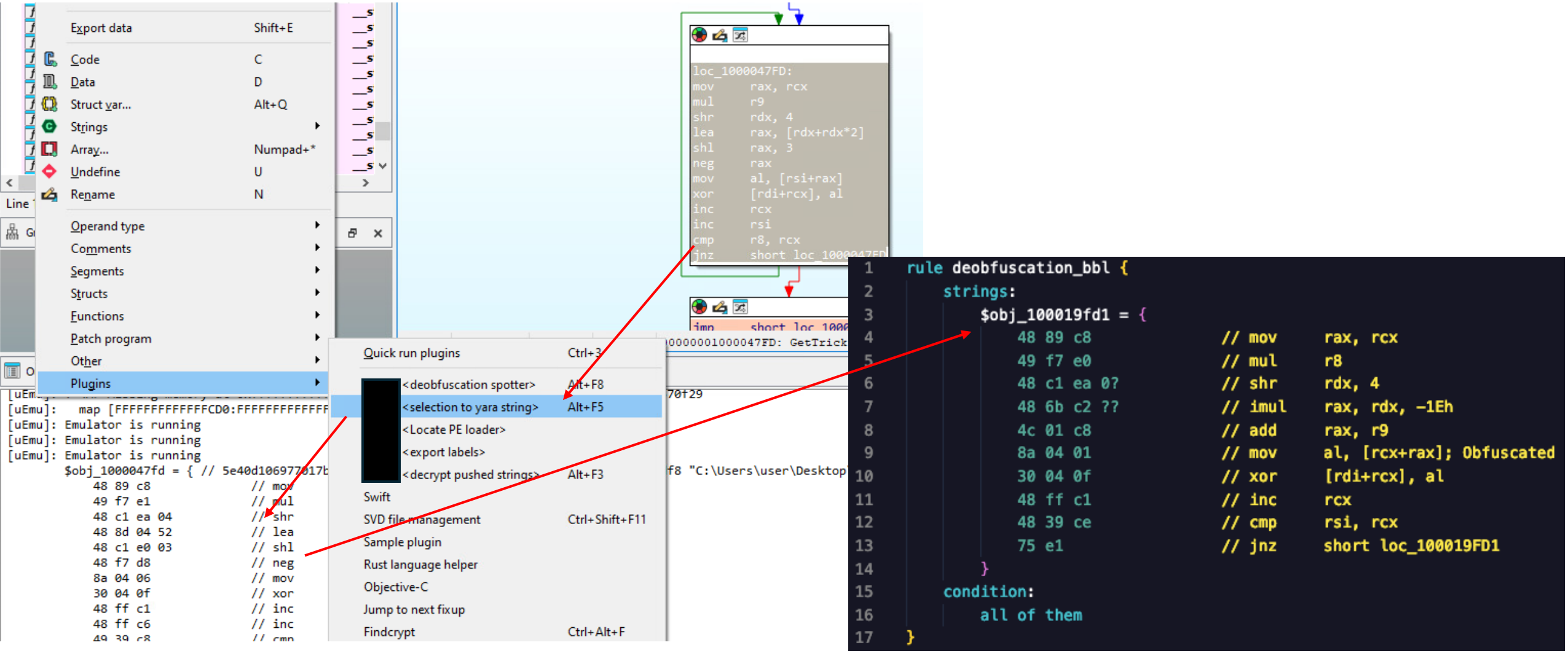
Key

Decryption routine

Exemple de code de désobfuscation

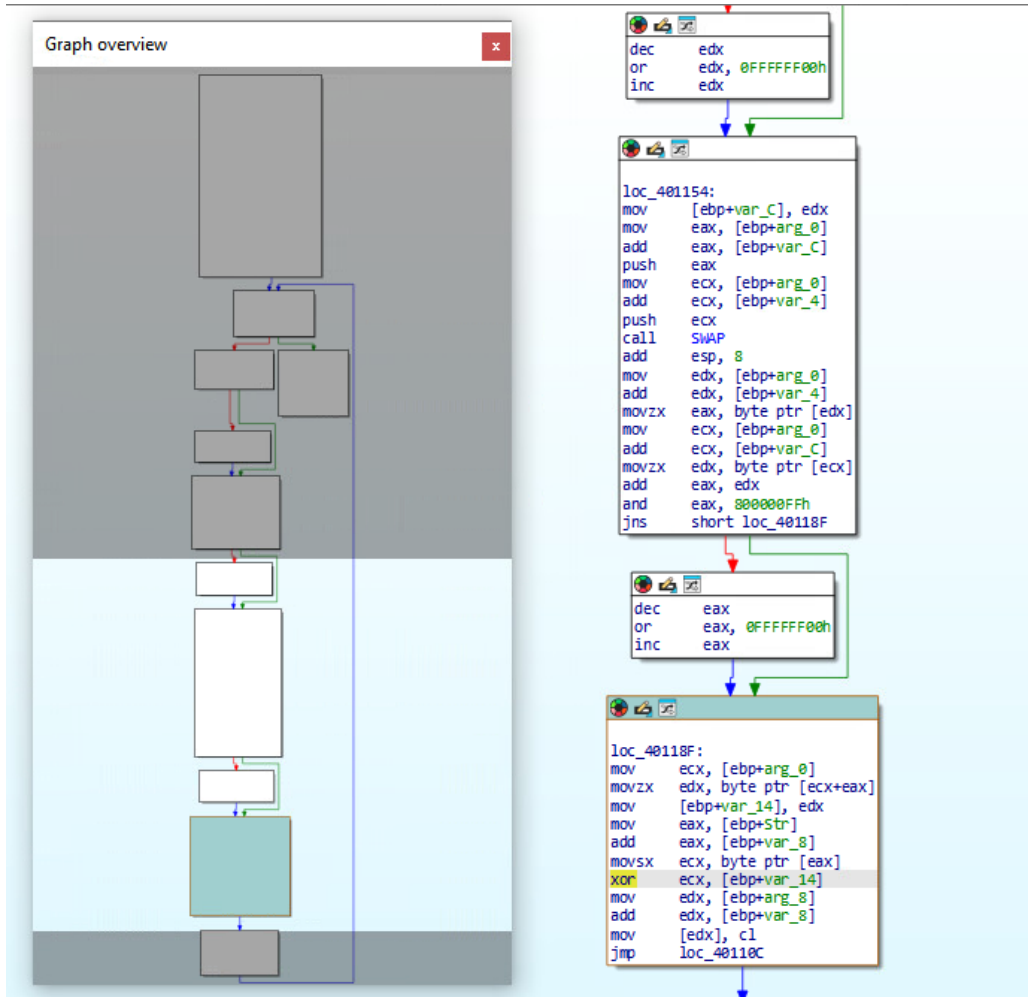


Génération des règles

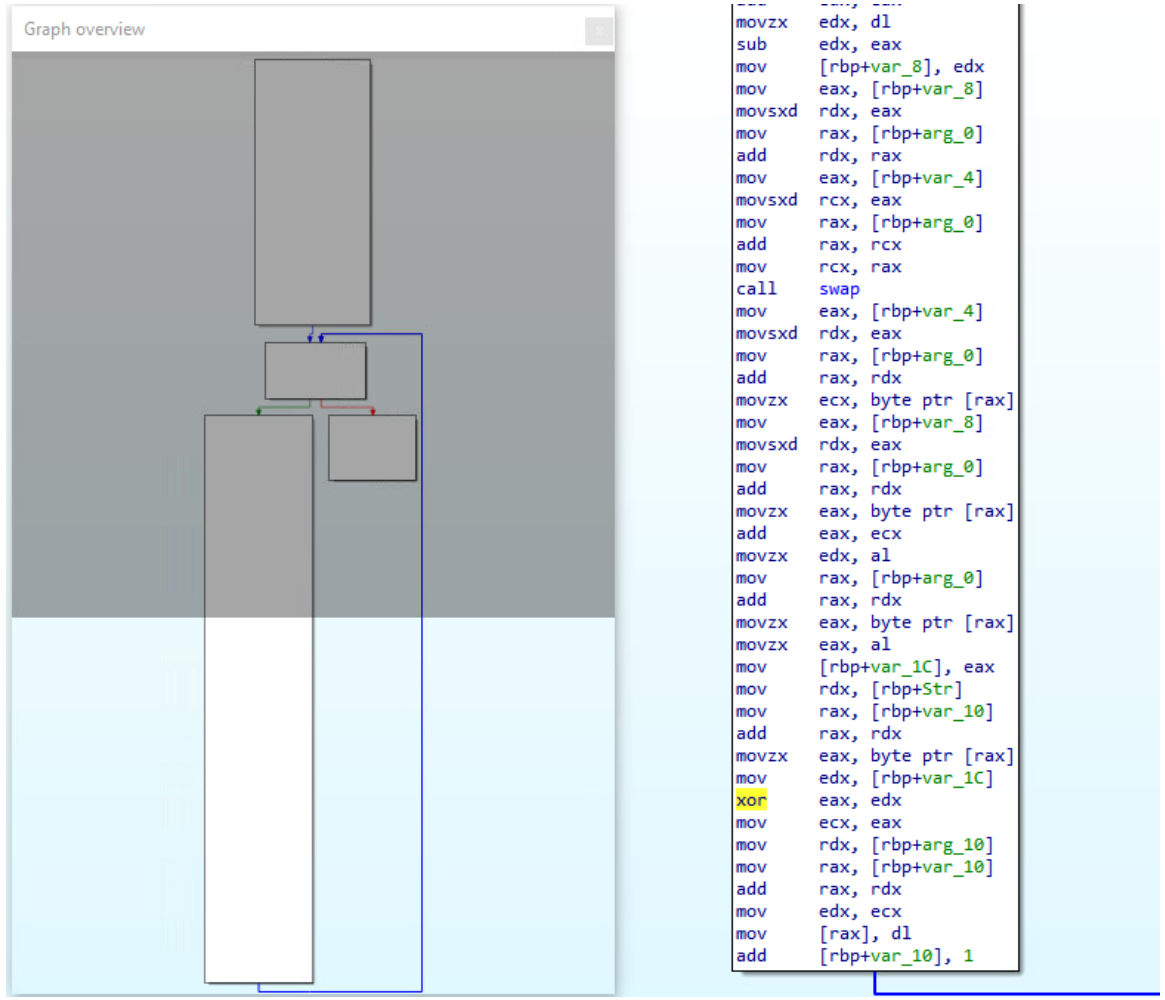


Limites (rverton/RC4.c)

VS code x86_32

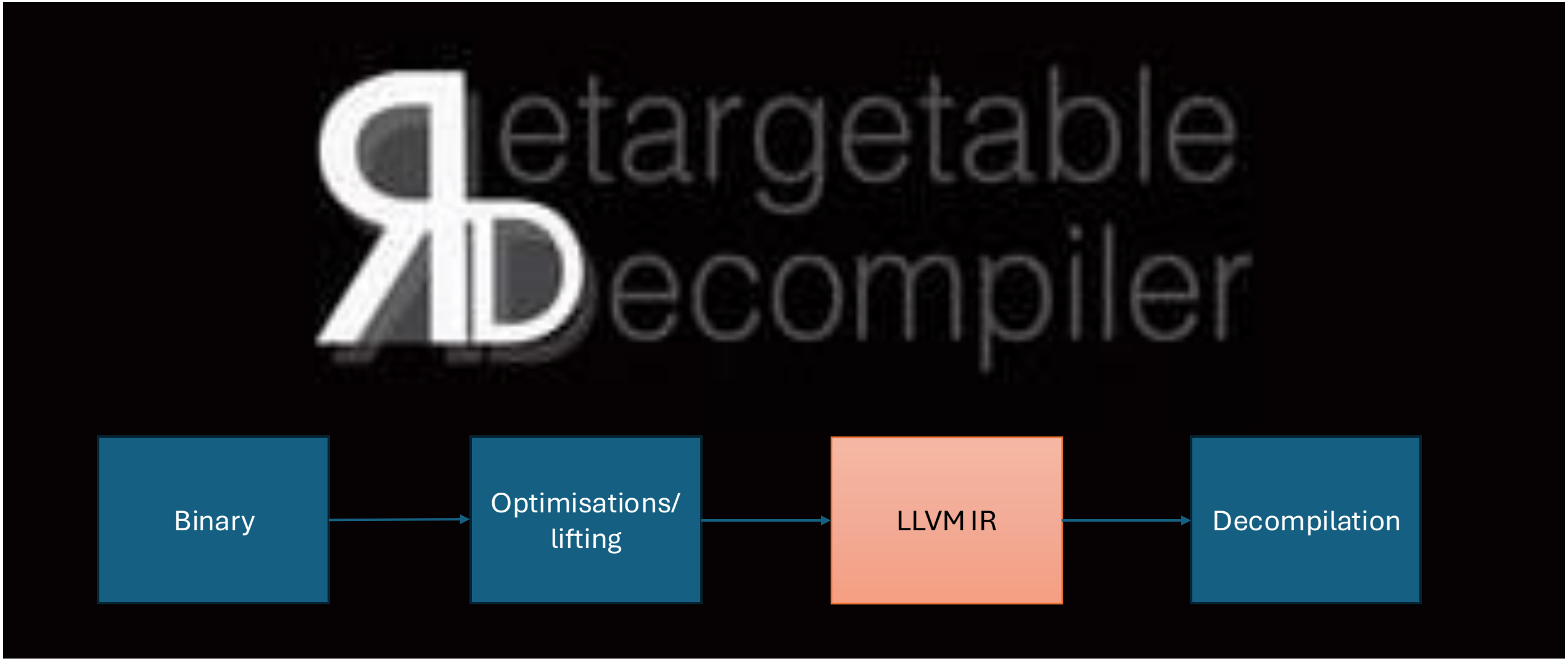


GCC code x86_64



Passage en représentation intermédiaire

Retargetable decompiler



Représentation intermédiaire

```
retdec-decompiler.exe C:\Users\user\Desktop\winerump\test_cross_platform\RC4_gcc_win64.exe --select-ranges 0x01400014E0-0x01400015B7 > NUL 2>&1

retdec-decompiler.exe C:\Users\user\Desktop\winerump\test_cross_platform\RC4_VS_win32.exe --select-ranges 0x04010E0-0x04011B7 > NUL 2>&1
```

```python
async def bin_to_ir(file_path: pathlib.Path) -> None:
    dst_file = BDD / (str(file_path.parent).replace("/", "_") + "_"  + file_path.stem + ".ll.zlib")
    if dst_file.exists():
        return
    print("processing:", file_path)
    try:
        with tempfile.TemporaryDirectory(dir="/dev/shm") as _tmp:
            tmp = pathlib.Path(_tmp)
            proc = await asyncio.create_subprocess_exec(RETDEC_DECOMPILER_PATH,
                                                        "-o",
                                                        tmp / file_path.stem,
                                                        file_path,
                                                        stdout=asyncio.subprocess.DEVNULL,
                                                        stderr=asyncio.subprocess.DEVNULL,
                                                        )
            await proc.communicate()
            ll_file = tmp / f"{file_path.stem}.ll"
            if ll_file.exists():
                dst_file.write_bytes(zlib.compress(ll_file.read_bytes()))
    except OSError:
        return

async def crawl() -> None:
    async with asyncio.Semaphore(10):
        await asyncio.gather(*map(bin_to_ir, MALWARE_SAMPLE_PATH.glob("**/*.exe")))
```

Similarités (PRGA RC4)

VS code x86_32

```
dec_label_pc_401154:                              ; preds = %dec_label_pc_40114c, %
  %edx.0.reload = load i32, i32* %edx.0.reg2mem
  %26 = add i32 %edx.0.reload, %3, !insn.addr !18
  store i32 %26, i32* %6, align 4, !insn.addr !5
  store i32 %15, i32* %8, align 4, !insn.addr !6
  %27 = call i32 @function_401000(i32 %edx.0.reload, i32 %eax.0), !insn.addr !19
  %28 = load i8, i8* %16, align 1, !insn.addr !20
  %29 = zext i8 %28 to i32, !insn.addr !20
  %30 = inttoptr i32 %26 to i8*, !insn.addr !21
  %31 = load i8, i8* %30, align 1, !insn.addr !21
  %32 = zext i8 %31 to i32, !insn.addr !21
  %33 = add nuw nsw i32 %32, %29, !insn.addr !22
  %34 = and i32 %33, 255, !insn.addr !23
  %35 = add i32 %34, %3, !insn.addr !24
  %36 = inttoptr i32 %35 to i8*, !insn.addr !24
  %37 = load i8, i8* %36, align 1, !insn.addr !24
  %38 = zext i8 %37 to i32, !insn.addr !24
  store i32 %38, i32* %stack_var_-24, align 4, !insn.addr !25
  %39 = add i32 %stack_var_-12.01.reload, %arg2, !insn.addr !26
  %40 = inttoptr i32 %39 to i8*, !insn.addr !27
  %41 = load i8, i8* %40, align 1, !insn.addr !27
  %42 = add i32 %stack_var_-12.01.reload, %arg3, !insn.addr !28
  %43 = xor i8 %41, %37, !insn.addr !29
  %44 = inttoptr i32 %42 to i8*, !insn.addr !29
  store i8 %43, i8* %44, align 1, !insn.addr !29
  %45 = add i32 %stack_var_-12.01.reload, 1, !insn.addr !30
  %46 = icmp ult i32 %45, %0, !insn.addr !2
  %47 = icmp eq i1 %46, false, !insn.addr !3
  store i32 %eax.0, i32* %stack_var_-8.03.reg2mem, !insn.addr !3
  store i32 %edx.0.reload, i32* %stack_var_-16.02.reg2mem, !insn.addr !3
  store i32 %45, i32* %stack_var_-12.01.reg2mem, !insn.addr !3
  br i1 %47, label %dec_label_pc_4011b2, label %dec_label_pc_401121, !insn.addr !3
```

GCC code x86_64

```
dec_label_pc_1400015f7:                           ; preds = %dec_label_pc_1400015b8, %dec
  %stack_var_-12.01.reload = load i32, i32* %stack_var_-12.01.reg2mem
  %stack_var_-16.02.reload = load i32, i32* %stack_var_-16.02.reg2mem
  %stack_var_-24.03.reload = load i64, i64* %stack_var_-24.03.reg2mem
  %5 = add nsw i32 %stack_var_-12.01.reload, 1, !insn.addr !4
  %6 = srem i32 %5, 256, !insn.addr !5
  %7 = sext i32 %6 to i64, !insn.addr !6
  %8 = add i64 %7, %0, !insn.addr !7
  %9 = inttoptr i64 %8 to i8*, !insn.addr !8
  %10 = load i8, i8* %9, align 1, !insn.addr !8
  %11 = zext i8 %10 to i32, !insn.addr !9
  %12 = add nsw i32 %stack_var_-16.02.reload, %11, !insn.addr !9
  %13 = srem i32 %12, 256, !insn.addr !10
  %14 = sext i32 %13 to i64, !insn.addr !11
  %15 = add i64 %14, %0, !insn.addr !12
  %16 = call i64 @swap(i64 %8, i64 %15), !insn.addr !13
  %17 = load i8, i8* %9, align 1, !insn.addr !14
  %18 = inttoptr i64 %15 to i8*, !insn.addr !15
  %19 = load i8, i8* %18, align 1, !insn.addr !15
  %20 = add i8 %19, %17, !insn.addr !16
  %21 = zext i8 %20 to i64, !insn.addr !17
  %22 = add i64 %21, %0, !insn.addr !18
  %23 = inttoptr i64 %22 to i8*, !insn.addr !19
  %24 = load i8, i8* %23, align 1, !insn.addr !19
  %25 = add i64 %stack_var_-24.03.reload, %arg2, !insn.addr !20
  %26 = inttoptr i64 %25 to i8*, !insn.addr !21
  %27 = load i8, i8* %26, align 1, !insn.addr !21
  %28 = xor i8 %27, %24
  %29 = add i64 %stack_var_-24.03.reload, %arg3, !insn.addr !22
  %30 = inttoptr i64 %29 to i8*, !insn.addr !23
  store i8 %28, i8* %30, align 1, !insn.addr !23
  %31 = add nuw i64 %stack_var_-24.03.reload, 1, !insn.addr !24
  %32 = icmp ult i64 %31, %3, !insn.addr !2
  store i64 %31, i64* %stack_var_-24.03.reg2mem, !insn.addr !3
  store i32 %13, i32* %stack_var_-16.02.reg2mem, !insn.addr !3
  store i32 %6, i32* %stack_var_-12.01.reg2mem, !insn.addr !3
  br i1 %32, label %dec_label_pc_1400015f7, label %dec_label_pc_1400016c7, !insn.addr !3
```

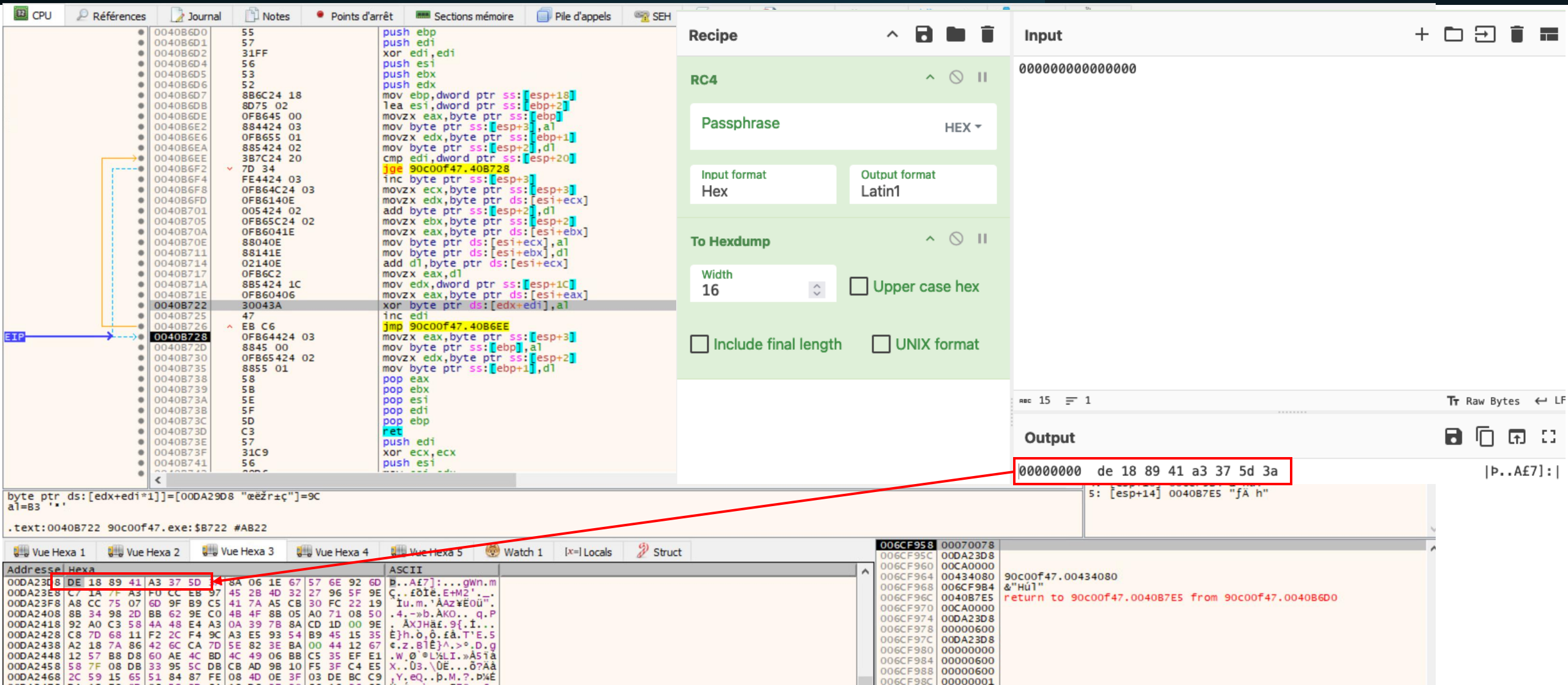
Signature

- POC de détection rc4

```
bbl.count("call") < 2
and bbl.count("load") < 20
and not any(x in bbl for x in ("sub", "shl", "shr", "mul", "and", "div", "trunc", "select"))
and all(len(re.findall(x, bbl)) == 1 for x in [r"br i1 \%[a-z\d]+, label",
                                               r"xor\si8\s\%\d+,\s\%\d+",
                                               ])
```

- Amélioration avec llvmlite (beaucoup de refactor)

Win!


CPU
Références
Journal
Notes
Points d'arrêt
Sections mémoire
Pile d'appels
SEH
0040B6D0 55 push ebp
0040B6D1 57 push edi
0040B6D2 31FF xor edi,edi
0040B6D4 56 push esi
0040B6D5 53 push ebx
0040B6D6 52 push edx
0040B6D7 8B6C24 18 mov ebp,dword ptr ss:[esp+18]
0040B6DB 8D75 02 lea esi,dword ptr ss:[ebp+2]
0040B6DE 0FB645 00 movzx eax,byte ptr ss:[ebp]
0040B6E2 884424 03 mov byte ptr ss:[esp+3],al
0040B6E6 0FB655 01 movzx edx,byte ptr ss:[ebp+1]
0040B6EA 885424 02 mov byte ptr ss:[esp+2],dl
0040B6EE 3B7C24 20 cmp edi,dword ptr ss:[esp+20]
0040B6F2 7D 34 jge 90c00f47.40B728
0040B6F4 FE4424 03 inc byte ptr ss:[esp+3]
0040B6F8 0FB64C24 03 movzx ecx,byte ptr ss:[esp+3]
0040B6FD 0FB6140E movzx edx,byte ptr ds:[esi+ecx]
0040B701 005424 02 add byte ptr ss:[esp+2],dl
0040B705 0FB65C24 02 movzx ebx,byte ptr ss:[esp+2]
0040B70A 0FB6041E movzx eax,byte ptr ds:[esi+ebx]
0040B70E 88040E mov byte ptr ds:[esi+ecx],al
0040B711 88141E mov byte ptr ds:[esi+ebx],dl
0040B714 02140E add dl,byte ptr ds:[esi+ecx]
0040B717 0FB6C2 movzx eax,dl
0040B71A 8B5424 1C mov edx,dword ptr ss:[esp+1C]
0040B71E 0FB60406 movzx eax,byte ptr ds:[esi+eax]
0040B722 30043A xor byte ptr ds:[edx+edi],al
0040B725 47 inc edi
0040B726 EB C6 jmp 90c00f47.40B6EE
EIP
0040B728 0FB64424 03 movzx eax,byte ptr ss:[esp+3]
0040B72D 8845 00 mov byte ptr ss:[ebp],al
0040B730 0FB65424 02 movzx edx,byte ptr ss:[esp+2]
0040B735 8855 01 mov byte ptr ss:[ebp+1],dl
0040B738 58 pop eax
0040B739 5B pop ebx
0040B73A 5E pop esi
0040B73B 5F pop edi
0040B73C 5D pop ebp
0040B73D C3 ret
0040B73E 57 push edi
0040B73F 31C9 xor ecx,ecx
0040B741 56 push esi
Recipe
RC4
Passphrase
HEX
Input format
Hex
Output format
Latin1
To Hexdump
Width
16
Upper case hex
Include final length
UNIX format
Input
000000000000000
15
1
Raw Bytes
LF
Output
00000000 de 18 89 41 a3 37 5d 3a
|Þ..A£7]:|
byte ptr ds:[edx+edi*1]]=[00DA29D8 "œëžr±ç"]=9C
al=B3 '³'
.text:0040B722 90c00f47.exe:$B722 #AB22
5: [esp+14] 0040B7E5 "ƒÁ h"
Vue Hexa 1
Vue Hexa 2
Vue Hexa 3
Vue Hexa 4
Vue Hexa 5
Watch 1
[x=] Locals
Struct
Addresse Hexa ASCII
00DA23D8 DE 18 89 41 A3 37 5D 8A 06 1E 67 57 6E 92 6D Þ..A£7]:...gWn.m
00DA23E8 C7 1A 7F A3 F0 CC EB 97 45 2B 4D 32 27 96 5F 9E Ç..£ðÌë.E+M2'._.
00DA23F8 A8 CC 75 07 6D 9F B9 C5 41 7A A5 CB 30 FC 22 19 ¨Ìu.m.'ÅAz¥Ë0ü".
00DA2408 8B 34 98 2D BB 62 9E C0 4B 4F 8B 05 A0 71 08 50 .4.-»b.ÀKO.. q.P
00DA2418 92 A0 C3 58 4A 48 E4 A3 0A 39 7B 8A CD 1D 00 9E . ÃXJHä£.9{.Í...
00DA2428 C8 7D 68 11 F2 2C F4 9C A3 E5 93 54 B9 45 15 35 È}h.ò,ô.£å.T'E.5
00DA2438 A2 18 7A 86 42 6C CA 7D 5E 82 3E BA 00 44 12 67 ¢.z.BlÊ}^.>°.D.g
00DA2448 12 57 B8 D8 60 AE 4C BD 4C 49 06 BB C5 35 EF E1 .W¸Ø`®L½LI.»Å5ïá
00DA2458 58 7F 08 DB 33 95 5C DB CB AD 9B 10 F5 3F C4 E5 X..Û3.\ÛË...õ?Äå
00DA2468 2C 59 15 65 51 84 87 FE 08 4D 0E 3F 03 DE BC C9 ,Y.eQ..þ.M.?.Þ¼É
006CF958 00070078
006CF95C 00DA23D8
006CF960 00CA0000
006CF964 00434080 90c00f47.00434080
006CF968 006CF984 &"Hú1"
006CF96C 0040B7E5 return to 90c00f47.0040B7E5 from 90c00f47.0040B6D0
006CF970 00CA0000
006CF974 00DA23D8
006CF978 00000600
006CF97C 00DA23D8
006CF980 00000000
006CF984 00000600
006CF988 00000600
006CF98C 00000001

Resultats pour RC4

- Fonctions vues par le script et pas par:
  - polymorf/findcrypt-yara
  - shellcromancer/alg_crypto_rc4.yara

- Détection surNetwire, Emotet, Macoute, MysticStealer, Samples de ref (GCC, VS)…
- Aucun FP sur 15 000 samples

Success?

- Performances médiocres
  - Retdec sur thanos (916500065fb0037de6e95bdbeafaa69a8d3932af10e81acb02f88c6a65cb577e)

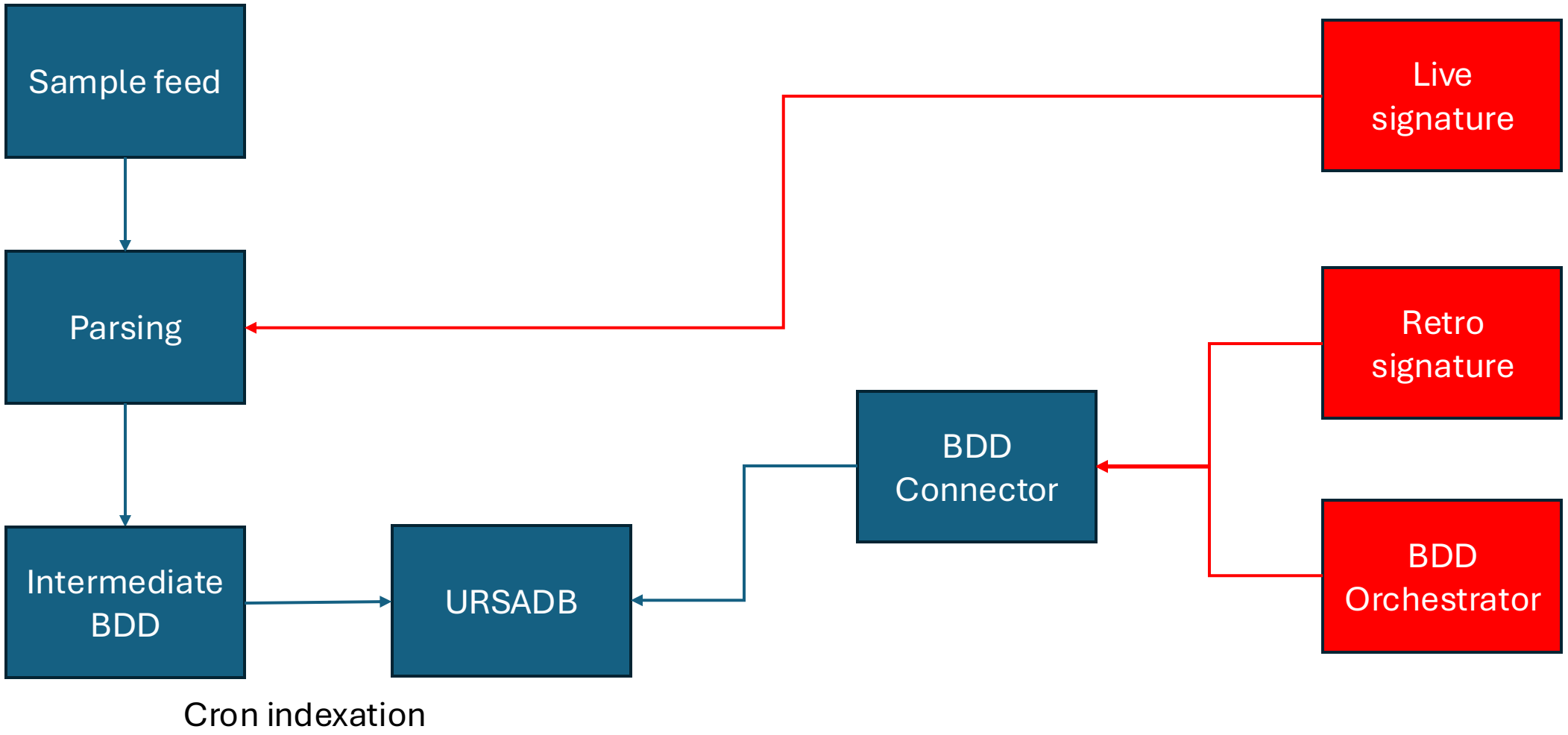
    real 0m12.826s
    user 0m12.667s
    sys 0m0.071s

  - 1 signature sur 15 000 samples goodware/malware mix (python script)

    real 9m43.781s
    user 9m19.179s
    sys 0m7.150s

BDD n-gram

VT like architecture



URSADB

- Python script:
  - real 9m43.781s
  - user 9m19.179s
  - sys 0m7.150s
- CERT-Polska/ursadb
  - real 0m0.222s
  - user 0m0.044s
  - sys 0m0.041s
  - Limites dans les types de query

Limites

Limites

- Résultats encourageants mais:
  - Resultats ARM décevants
  - Multiples bugs en cas d’obfuscations
  - Besoin de tester davantage de lifters

Fin