Generic Data Extraction & Injection with "libr"

The Binary Swiss Army Knife

http://radare.org/

Nibble@develsec.org
http://nibble.develsec.org/

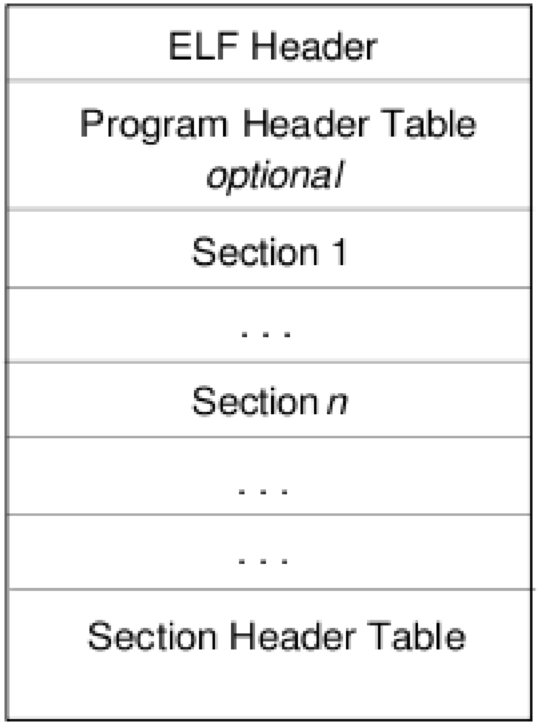
bs3c 2009

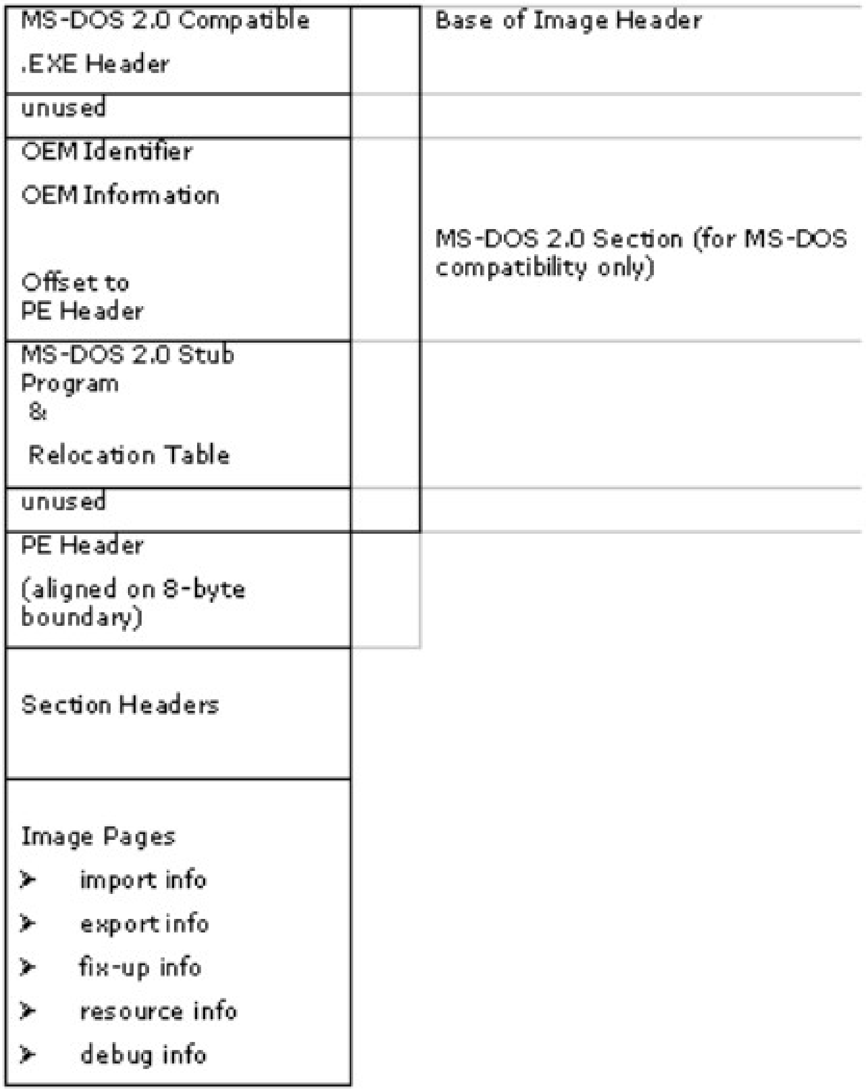
¿Por qué? (I)

Hoy en día existen...

- Demasiadas arquitecturas distintas
  - X86, PPC, MIPS ...
- Demasiados formatos ejecutable distintos
  - PE32, PE32+, ELF32, ELF64, CLASS, MSIL ...

¿Por qué? (II)





bs3c 2009

¿Por qué? (y III)

Por lo tanto...

- Necesitamos distintas librerías para cada formato
- Necesitamos distintos ensambladores
- Necesitamos reescribir programas desde cero para tareas semejantes

En definitiva...

- Perdemos demasiado tiempo
- Trabajamos de manera improductiva

libr y radare2

```
            +---------+
          .-| config  |
         /  +---------+
 +-------+  +-------+ +-------+ +-------+
 | core  |--| cons  | | asm   | | diff  |
 +-------+  | line  | | bin   | | sign  |
   |     \  +-------+ | anal  | | hash  |    ,______.
 +-----+  \           +---.---+ +--.----+   +.______.+
 | io  |   +--------------'-------/         |        |
 +-----+   | cmd, search, print   |<------->| flags  |
   |       +------.---------------\         | meta   |
[ lib ]      +----'------------+ +-`------+ +.______.+
   |    .----| debug, bp, vm   | | lang   |
   |    |    | reg, syscall    | | macro  |
+-------'--+ |  var, trace    | +--------+
| plugins  | +-----------------+     |
+------.---+                         |
       `-----------------------------'
```

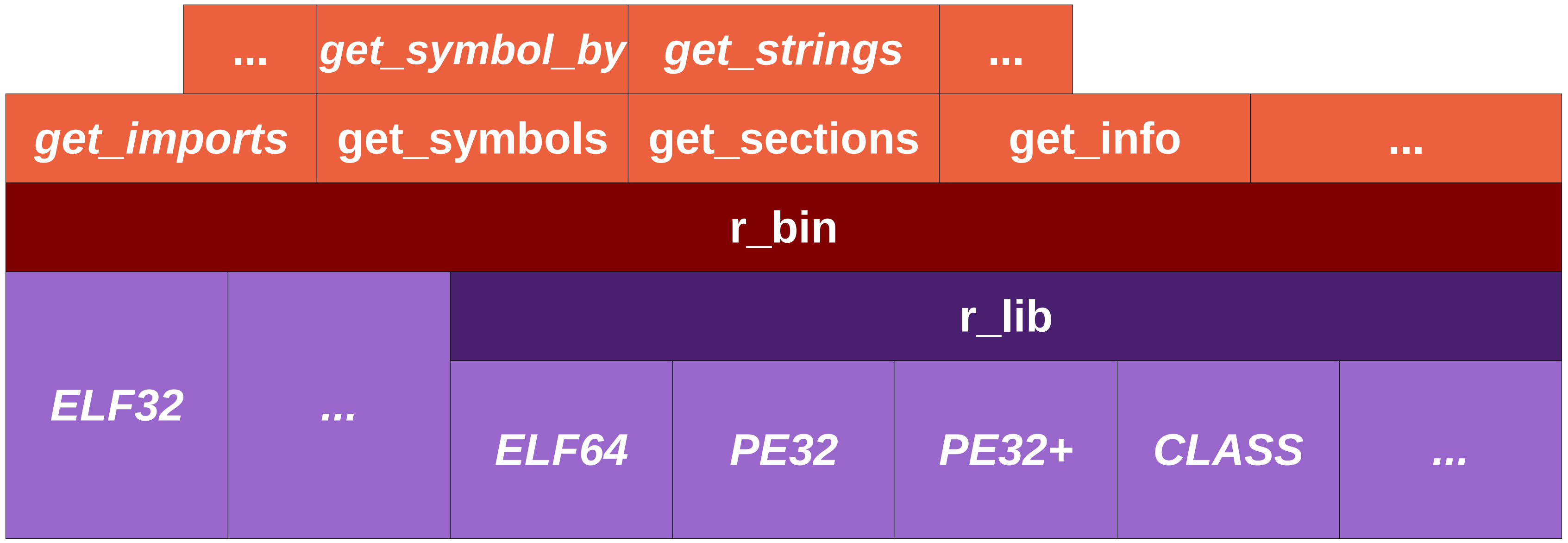
r_bin (I)

- Análisis de cabeceras
- Múltiples fomatos
  - ELF32, ELF64, PE32, PE32+, CLASS...
- Modular (r_lib, static)
- Portable

r_bin (II)

- Soporte
  - Binary info
  - Imports
  - Symbols
  - Sections
  - Libs
  - Strings
  - Section resize

r_bin (III)


...
get_symbol_by
get_strings
...
get_imports
get_symbols
get_sections
get_info
...
r_bin
ELF32
...
r_lib
ELF64
PE32
PE32+
CLASS
...

r_bin (IV)

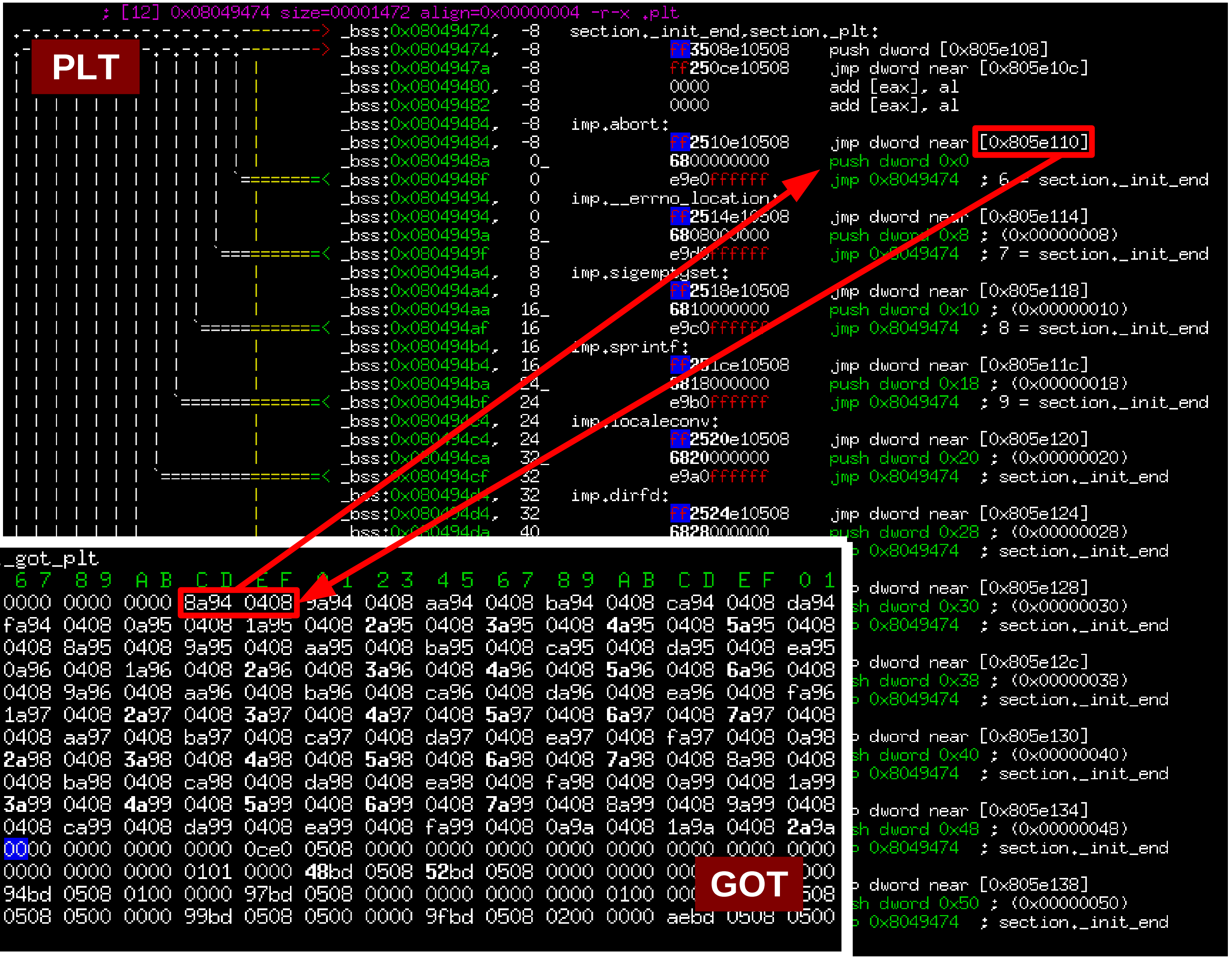
Demo rabin2

r_bin (V)

r_bin_get_imports

r_bin (VI)

ELF



```
; [12] 0x08049474 size=00001472 align=0x00000004 -r-x .plt
_bss:0x08049474,  -8   section._init_end,section._plt:
_bss:0x08049474,  -8             ff3508e10508     push dword [0x805e108]
_bss:0x0804947a   -8             ff250ce10508     jmp dword near [0x805e10c]
_bss:0x08049480,  -8             0000             add [eax], al
_bss:0x08049482   -8             0000             add [eax], al
_bss:0x08049484,  -8   imp.abort:
_bss:0x08049484,  -8             ff2510e10508     jmp dword near [0x805e110]
_bss:0x0804948a    0_            6800000000       push dword 0x0
_bss:0x0804948f    0             e9e0ffffff       jmp 0x8049474   ; 6 = section._init_end
_bss:0x08049494,   0   imp.__errno_location:
_bss:0x08049494,   0             ff2514e10508     jmp dword near [0x805e114]
_bss:0x0804949a    8_            6808000000       push dword 0x8 ; (0x00000008)
_bss:0x0804949f    8             e9d0ffffff       jmp 0x8049474   ; 7 = section._init_end
_bss:0x080494a4,   8   imp.sigemptyset:
_bss:0x080494a4,   8             ff2518e10508     jmp dword near [0x805e118]
_bss:0x080494aa   16_            6810000000       push dword 0x10 ; (0x00000010)
_bss:0x080494af   16             e9c0ffffff       jmp 0x8049474   ; 8 = section._init_end
_bss:0x080494b4,  16   imp.sprintf:
_bss:0x080494b4,  16             ff251ce10508     jmp dword near [0x805e11c]
_bss:0x080494ba   24_            6818000000       push dword 0x18 ; (0x00000018)
_bss:0x080494bf   24             e9b0ffffff       jmp 0x8049474   ; 9 = section._init_end
_bss:0x080494c4,  24   imp.localeconv:
_bss:0x080494c4,  24             ff2520e10508     jmp dword near [0x805e120]
_bss:0x080494ca   32_            6820000000       push dword 0x20 ; (0x00000020)
_bss:0x080494cf   32             e9a0ffffff       jmp 0x8049474   ; section._init_end
_bss:0x080494d4,  32   imp.dirfd:
_bss:0x080494d4,  32             ff2524e10508     jmp dword near [0x805e124]
_bss:0x080494da   40             6828000000       push dword 0x28 ; (0x00000028)
                                                  p 0x8049474   ; section._init_end
                                                  p dword near [0x805e128]
                                                  sh dword 0x30 ; (0x00000030)
                                                  p 0x8049474   ; section._init_end
                                                  p dword near [0x805e12c]
                                                  sh dword 0x38 ; (0x00000038)
                                                  p 0x8049474   ; section._init_end
                                                  p dword near [0x805e130]
                                                  sh dword 0x40 ; (0x00000040)
                                                  p 0x8049474   ; section._init_end
                                                  p dword near [0x805e134]
                                                  sh dword 0x48 ; (0x00000048)
                                                  p 0x8049474   ; section._init_end
                                                  p dword near [0x805e138]
                                                  sh dword 0x50 ; (0x00000050)
                                                  p 0x8049474   ; section._init_end
```

```
[0x0804944B]> px @ section._got_plt
   offset    0 1  2 3  4 5  6 7  8 9  A B  C D  E F  0 1  2 3  4 5  6 7  8 9  A B  C D  E F  0 1
0x0805e104, 14e0 0508 0000 0000 0000 0000 8a94 0408 9a94 0408 aa94 0408 ba94 0408 ca94 0408 da94
0x0805e126  0408 ea94 0408 fa94 0408 0a95 0408 1a95 0408 2a95 0408 3a95 0408 4a95 0408 5a95 0408
0x0805e148, 6a95 0408 7a95 0408 8a95 0408 9a95 0408 aa95 0408 ba95 0408 ca95 0408 da95 0408 ea95
0x0805e16a  0408 fa95 0408 0a96 0408 1a96 0408 2a96 0408 3a96 0408 4a96 0408 5a96 0408 6a96 0408
0x0805e18c, 7a96 0408 8a96 0408 9a96 0408 aa96 0408 ba96 0408 ca96 0408 da96 0408 ea96 0408 fa96
0x0805e1ae  0408 0a97 0408 1a97 0408 2a97 0408 3a97 0408 4a97 0408 5a97 0408 6a97 0408 7a97 0408
0x0805e1d0, 8a97 0408 9a97 0408 aa97 0408 ba97 0408 ca97 0408 da97 0408 ea97 0408 fa97 0408 0a98
0x0805e1f2  0408 1a98 0408 2a98 0408 3a98 0408 4a98 0408 5a98 0408 6a98 0408 7a98 0408 8a98 0408
0x0805e214, 9a98 0408 aa98 0408 ba98 0408 ca98 0408 da98 0408 ea98 0408 fa98 0408 0a99 0408 1a99
0x0805e236  0408 2a99 0408 3a99 0408 4a99 0408 5a99 0408 6a99 0408 7a99 0408 8a99 0408 9a99 0408
0x0805e258, aa99 0408 ba99 0408 ca99 0408 da99 0408 ea99 0408 fa99 0408 0a9a 0408 1a9a 0408 2a9a
0x0805e27a  0408 0000 0000 0000 0000 0000 0000 0ce0 0508 0000 0000 0000 0000 0000 0000 0000 0000
0x0805e29c, 0000 0000 0100 0000 0000 0000 0101 0000 48bd 0508 52bd 0508 0000 0000 000      000
0x0805e2be  0000 0200 0000 94bd 0508 0100 0000 97bd 0508 0000 0000 0000 0000 0100 000      508
0x0805e2e0, 0100 0000 06b9 0508 0500 0000 99bd 0508 0500 0000 9fbd 0508 0200 0000 aebd 0508 0500
0x0805e302  0000
```

r_bin (VII)

Recorremos las secciones hasta que:

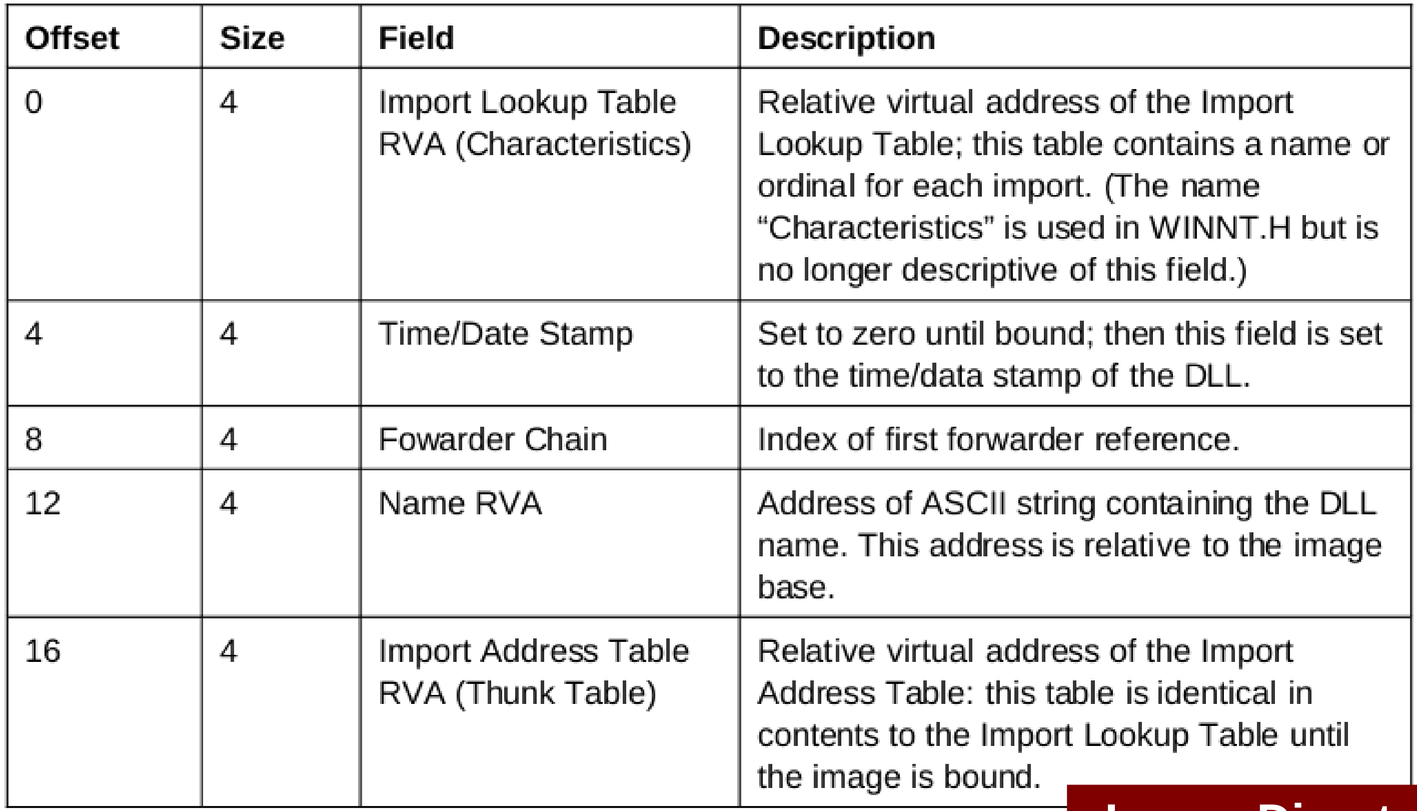
shdr->sh_type == (bin->ehdr.e_type == ET_REL? SHT_SYMTAB:SHT_DYNSYM)

Recorremos los syms de esa sección:

```
If (sym->st_value)
    offset = sym->st_value
Else {
    K = índice del import dentro de symtab o dynsym
    Recorremos .rel.plt (o rela.plt en elf64)
    if (ELF_R_SYM(rel->r_info) == k))
        Offset = read(rel->r_offset-bin->base_addr) - 6
}
```

r_bin (VIII)

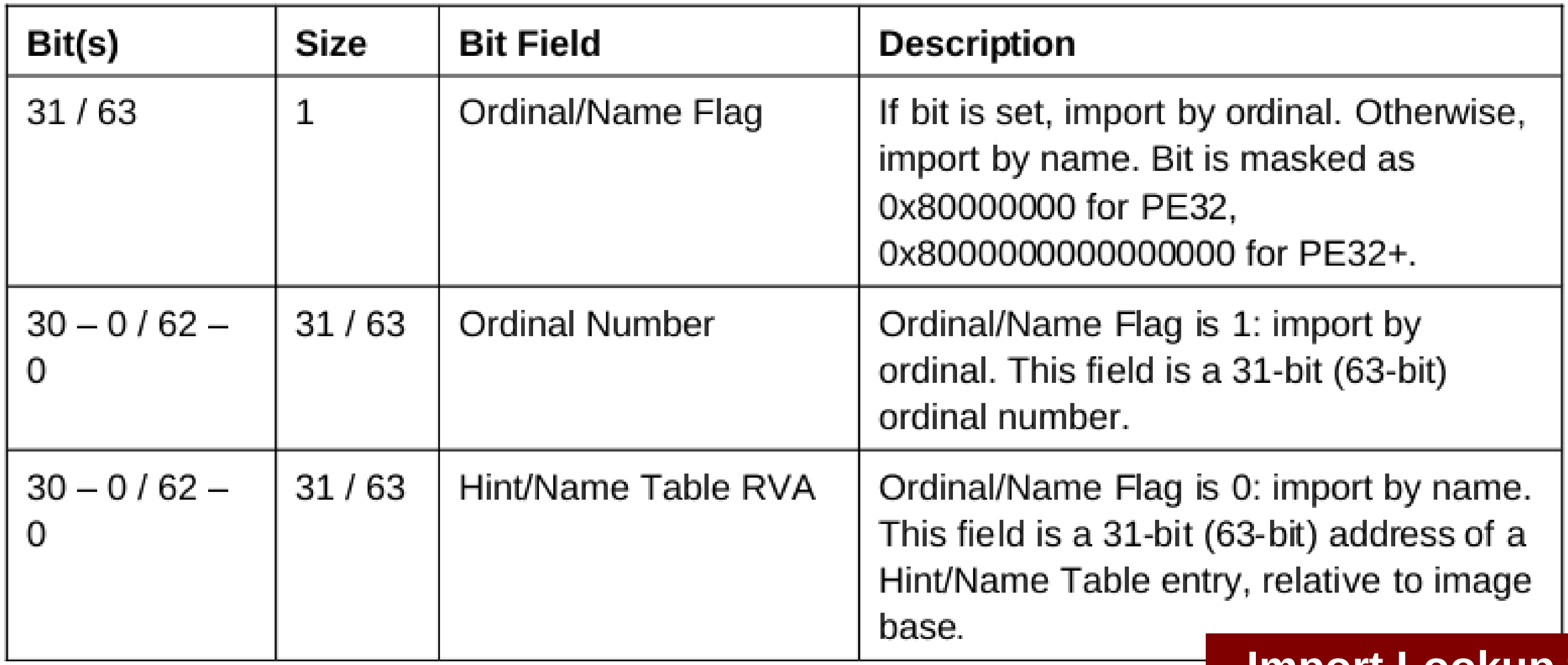
PE



Offset	Size	Field	Description
0	4	Import Lookup Table RVA (Characteristics)	Relative virtual address of the Import Lookup Table; this table contains a name or ordinal for each import. (The name “Characteristics” is used in WINNT.H but is no longer descriptive of this field.)
4	4	Time/Date Stamp	Set to zero until bound; then this field is set to the time/data stamp of the DLL.
8	4	Fowarder Chain	Index of first forwarder reference.
12	4	Name RVA	Address of ASCII string containing the DLL name. This address is relative to the image base.
16	4	Import Address Table RVA (Thunk Table)	Relative virtual address of the Import Address Table: this table is identical in contents to the Import Lookup Table until the image is bound.

Image Directory Table

r_bin (IX)



Bit(s)	Size	Bit Field	Description
31 / 63	1	Ordinal/Name Flag	If bit is set, import by ordinal. Otherwise, import by name. Bit is masked as 0x80000000 for PE32, 0x8000000000000000 for PE32+.
30 – 0 / 62 – 0	31 / 63	Ordinal Number	Ordinal/Name Flag is 1: import by ordinal. This field is a 31-bit (63-bit) ordinal number.
30 – 0 / 62 – 0	31 / 63	Hint/Name Table RVA	Ordinal/Name Flag is 0: import by name. This field is a 31-bit (63-bit) address of a Hint/Name Table entry, relative to image base.

Import Lookup table

r_bin (X)

- Recorremos las “Import Directory Table”:
  - Obtenemos el nombre de la DLL
  - Offset “Import Lookup Table” (ILT)
  - Recorremos las ILT:
    - Obtenemos el nombre del import o su ordinal
    - Obtenemos rva (la de la propia ILT)

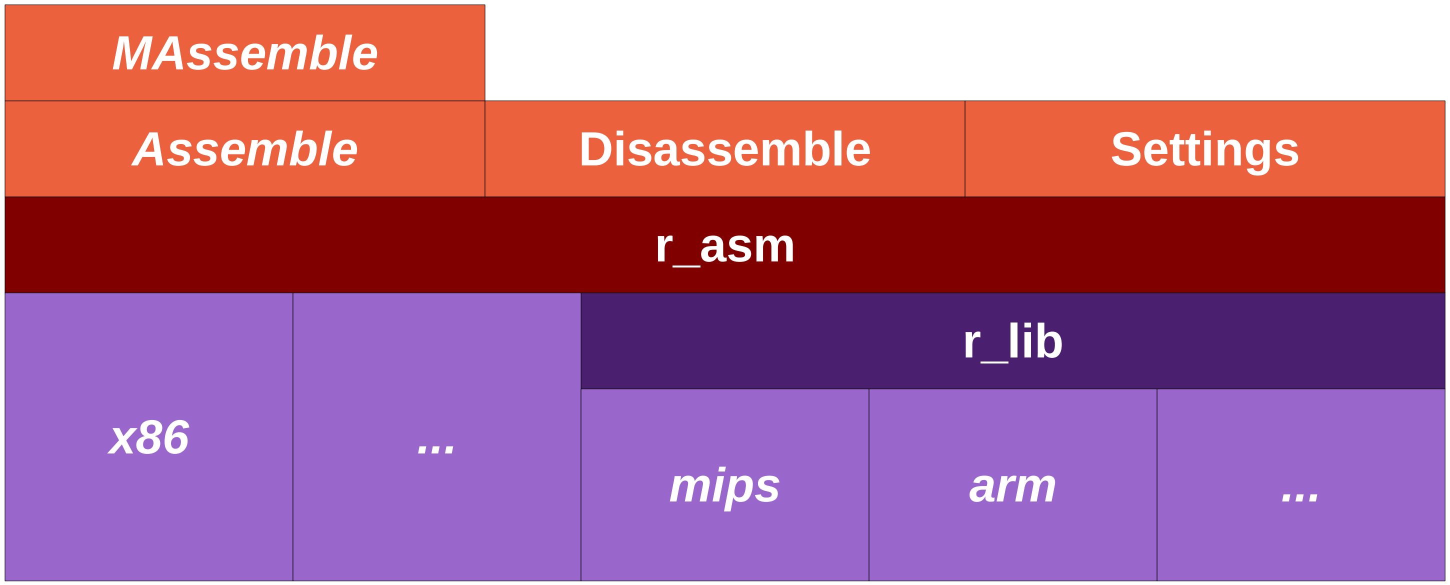
r_bin (y XI)

Demos data1 y data2

r_asm (I)

- Ensamblado/Desensamblado
- Múltiples arquitecturas
  - Arch (x86, ppc, mips, arm, sparc, brainfuck...)
  - Wordsize
  - Endianness
- Soporte
  - Pseudo-instrucciones (.org, .arch, .bits, .byte, .string...)
  - Labels
- Modular (r_lib, static)

r_asm (II)


MAssemble
Assemble
Disassemble
Settings
r_asm
r_lib
x86
...
mips
arm
...

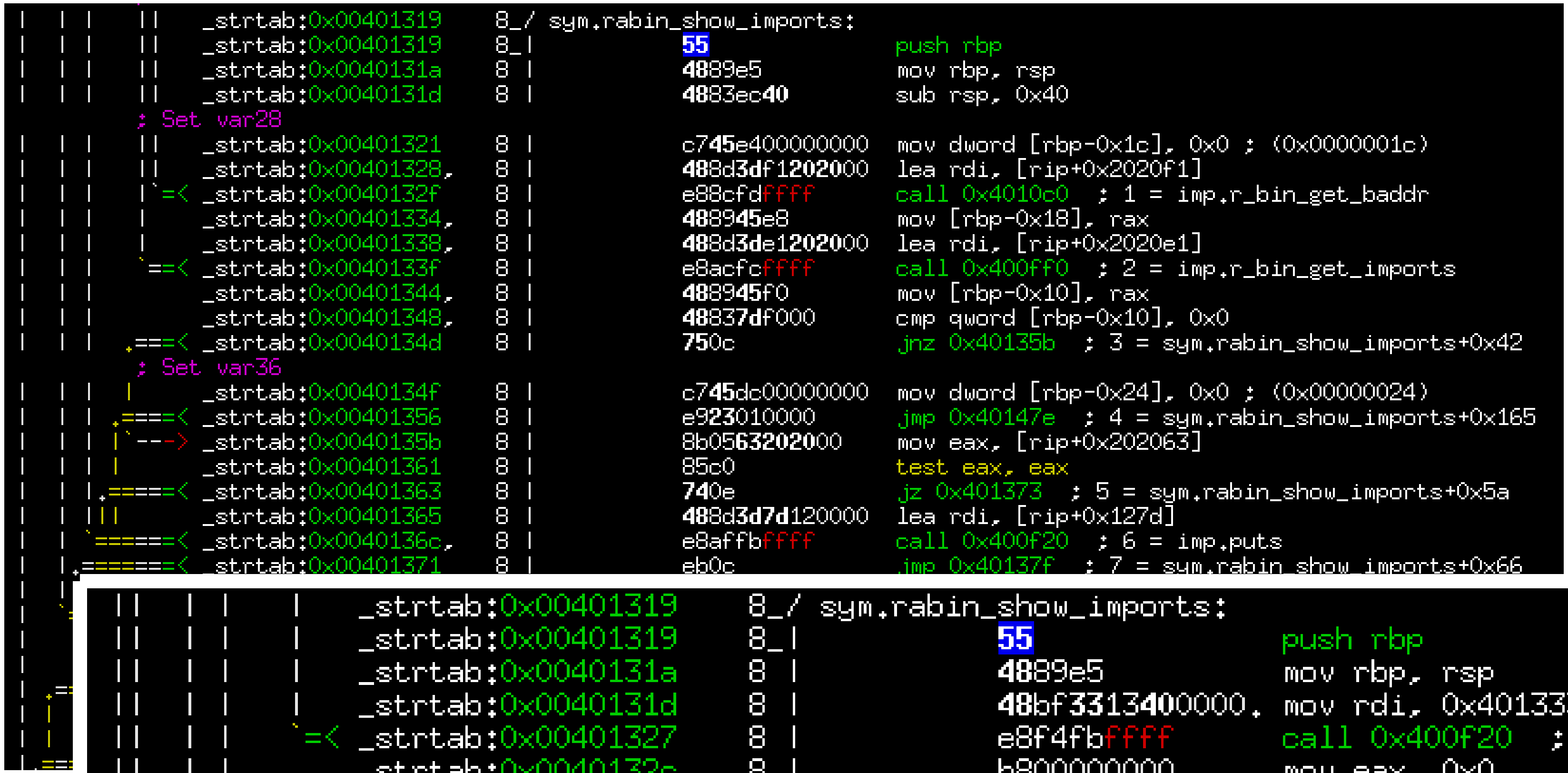
r_asm (y III)

Demo rasm2

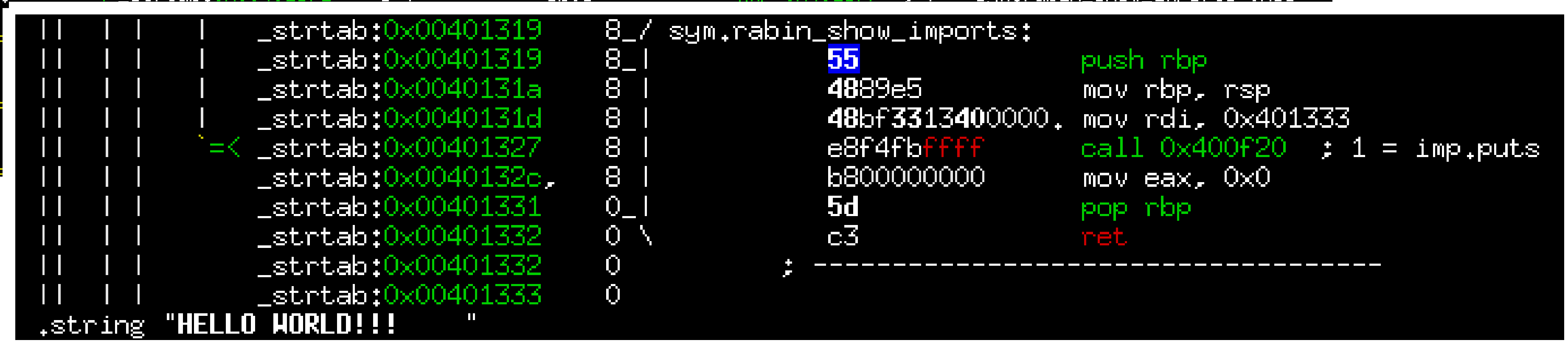
Demo vala asm widget

r_bin + r_asm

Demo “simple binary injection”



```
|   | |   ||   _strtab:0x00401319    8_/ sym.rabin_show_imports:
|   | |   ||   _strtab:0x00401319    8_|              55                push rbp
|   | |   ||   _strtab:0x0040131a    8 |              4889e5            mov rbp, rsp
|   | |   ||   _strtab:0x0040131d    8 |              4883ec40          sub rsp, 0x40
          ; Set var28
|   | |   ||   _strtab:0x00401321    8 |              c745e400000000    mov dword [rbp-0x1c], 0x0 ; (0x0000001c)
|   | |   ||   _strtab:0x00401328,   8 |              488d3df1202000    lea rdi, [rip+0x2020f1]
|   | |   |`=< _strtab:0x0040132f    8 |              e88cfdffff        call 0x4010c0   ; 1 = imp.r_bin_get_baddr
|   | |   |    _strtab:0x00401334,   8 |              488945e8          mov [rbp-0x18], rax
|   | |   |    _strtab:0x00401338,   8 |              488d3de1202000    lea rdi, [rip+0x2020e1]
|   | |   `==< _strtab:0x0040133f    8 |              e8acfcffff        call 0x400ff0   ; 2 = imp.r_bin_get_imports
|   | |        _strtab:0x00401344,   8 |              488945f0          mov [rbp-0x10], rax
|   | |        _strtab:0x00401348,   8 |              48837df000        cmp qword [rbp-0x10], 0x0
|   | |  .===< _strtab:0x0040134d    8 |              750c              jnz 0x40135b    ; 3 = sym.rabin_show_imports+0x42
          ; Set var36
|   | |  |     _strtab:0x0040134f    8 |              c745dc00000000    mov dword [rbp-0x24], 0x0 ; (0x00000024)
|   | | .====< _strtab:0x00401356    8 |              e923010000        jmp 0x40147e    ; 4 = sym.rabin_show_imports+0x165
|   | | |`---> _strtab:0x0040135b    8 |              8b0563202000      mov eax, [rip+0x202063]
|   | | |      _strtab:0x00401361    8 |              85c0              test eax, eax
|   | |.=====< _strtab:0x00401363    8 |              740e              jz 0x401373     ; 5 = sym.rabin_show_imports+0x5a
|   | |||      _strtab:0x00401365    8 |              488d3d7d120000    lea rdi, [rip+0x127d]
|   | `======< _strtab:0x0040136c,   8 |              e8affbffff        call 0x400f20   ; 6 = imp.puts
|   |.=======< _strtab:0x00401371    8 |              eb0c              jmp 0x40137f    ; 7 = sym.rabin_show_imports+0x66
```



```
||  | |   |   _strtab:0x00401319   8_/ sym.rabin_show_imports:
||  | |   |   _strtab:0x00401319   8_|         55                push rbp
||  | |   |   _strtab:0x0040131a   8 |         4889e5            mov rbp, rsp
||  | |   |   _strtab:0x0040131d   8 |         48bf3313400000,   mov rdi, 0x401333
||  | |   `=< _strtab:0x00401327   8 |         e8f4fbffff        call 0x400f20   ; 1 = imp.puts
||  | |       _strtab:0x0040132c,  8 |         b800000000        mov eax, 0x0
||  | |       _strtab:0x00401331   0_|         5d                pop rbp
||  | |       _strtab:0x00401332   0 \         c3                ret
||  | |       _strtab:0x00401332   0        ; ------------------------------------
||  | |       _strtab:0x00401333   0
 .string "HELLO WORLD!!!    "
```

bs3c 2009

Generic Data Extraction & Injection with "libr"

The Binary Swiss Army Knife

http://radare.org/

Nibble@develsec.org
http://nibble.develsec.org/

bs3c 2009