radare 2

@lacon'10

nibble
nibble@develsec.org

FatBins

Brief description of RBin

Header analysis

Supports:

- ELF32, ELF64, PE32, PE32+, MACH-O, MACH-O64, CLASS...
- FatMach-O, dyld cache, FatBins in general

Completely written from scratch, keeping in mind:

- Reversing (imports, symbols, sections, libs, relocs...)
- Minimalism

API is format-agnostic

FatBins

What?

- Several bins encapsulated into a single file

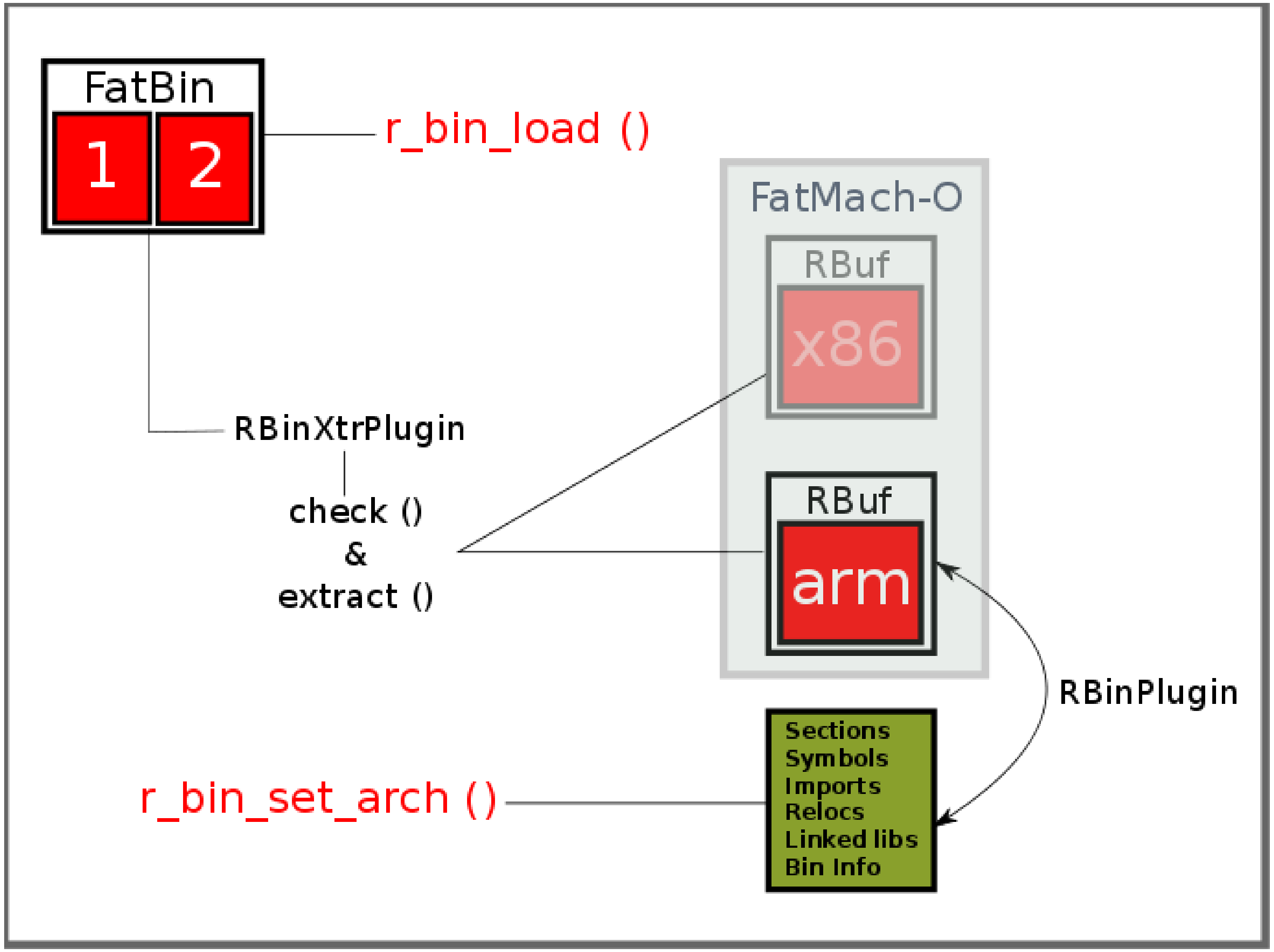
Why?

- Portability
- Makes linkage easier in the case of libs
- Optimization

How?

- RBinXtrPlugin

RBinXtrPlugin



Demos (RBin)

RBinXtrPlugin

Agnostic API

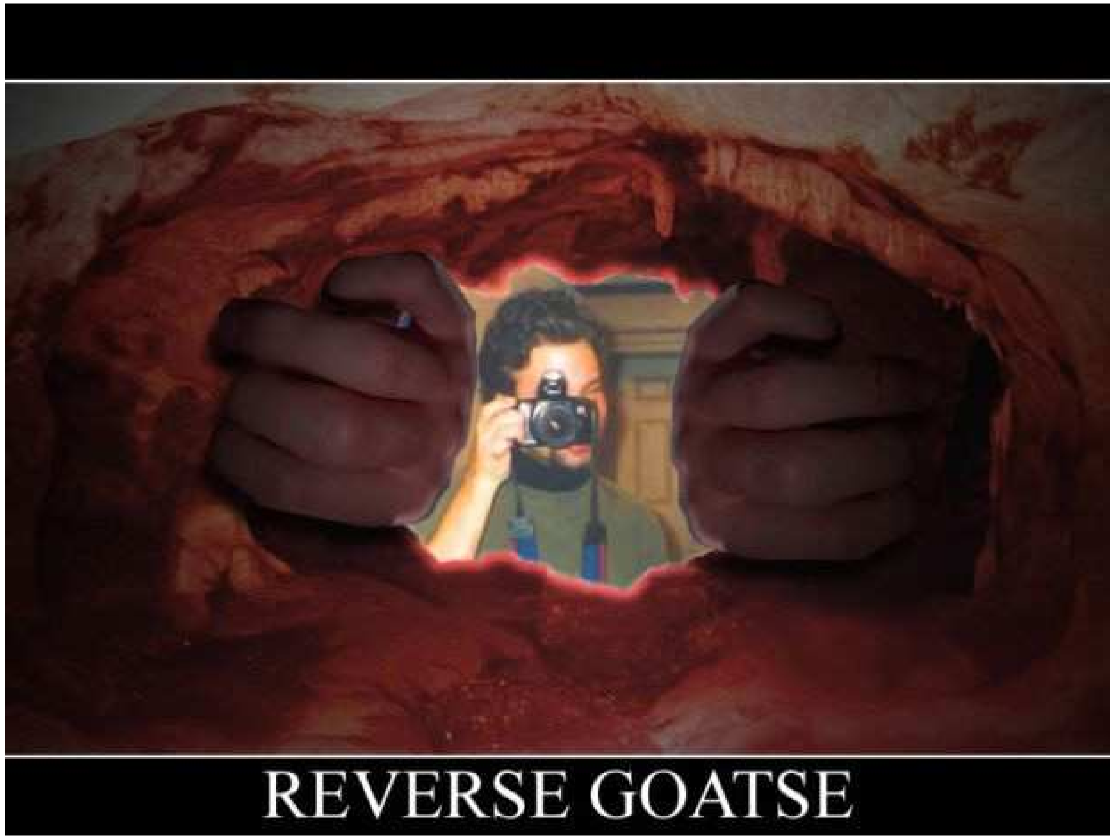
```
var bin = new RBin ();
if (bin.load (args[1], false) != 1)
            error ("Cannot open binary file0);

var baddr = bin.get_baddr();
foreach (var scn in bin.get_sections ())
            print ("0x%08"+uint64.FORMAT_MODIFIER+
                     "x - %05i %s0, baddr+scn.rva,
                     scn.size, scn.name);
```

Melting together the analysis libs

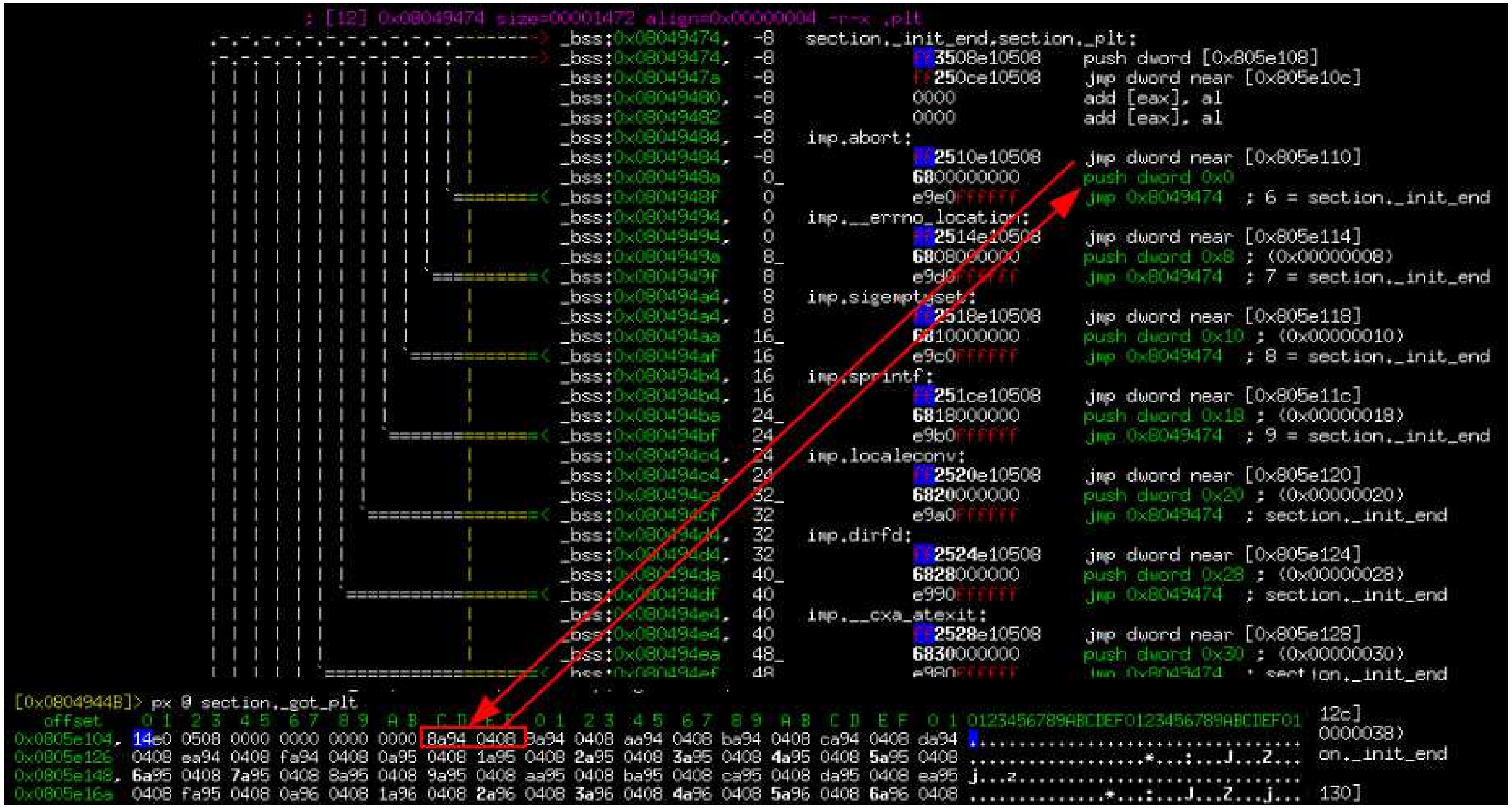
In r2, analysis is handled by...

RBin: Header level

RAnal: Code and Data levels (functions, bbs, refs...)



Demo (OneTimeHooks)



```
; [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)
_bss:0x080494df   40              e990ffffff       jmp 0x8049474  ; section._init_end
_bss:0x080494e4,  40   imp.__cxa_atexit:
_bss:0x080494e4,  40              ff2528e10508     jmp dword near [0x805e128]
_bss:0x080494ea   48_             6830000000       push dword 0x30 ; (0x00000030)
_bss:0x080494ef   48              e980ffffff       jmp 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 0123456789ABCDEF0123456789ABCDEF01   12c]
0x0805e104, 14e0 0508 0000 0000 0000 0000 8a94 0408 9a94 0408 aa94 0408 ba94 0408 ca94 0408 da94 ..................................   0000038)
0x0805e126  0408 ea94 0408 fa94 0408 0a95 0408 1a95 0408 2a95 0408 3a95 0408 4a95 0408 5a95 0408 .................*...:...J...Z...   on._init_end
0x0805e148, 6a95 0408 7a95 0408 8a95 0408 9a95 0408 aa95 0408 ba95 0408 ca95 0408 da95 0408 ea95 j...z.............................
0x0805e16a  0408 fa95 0408 0a96 0408 1a96 0408 2a96 0408 3a96 0408 4a96 0408 5a96 0408 6a96 0408 ..............*...:...J...Z...j...   130]
```

Demo (OneTimeHooks)

Analyze entrypoint

- Get init address

Write the payload into init

- Push MAGIC
- jump to the first PLT entry

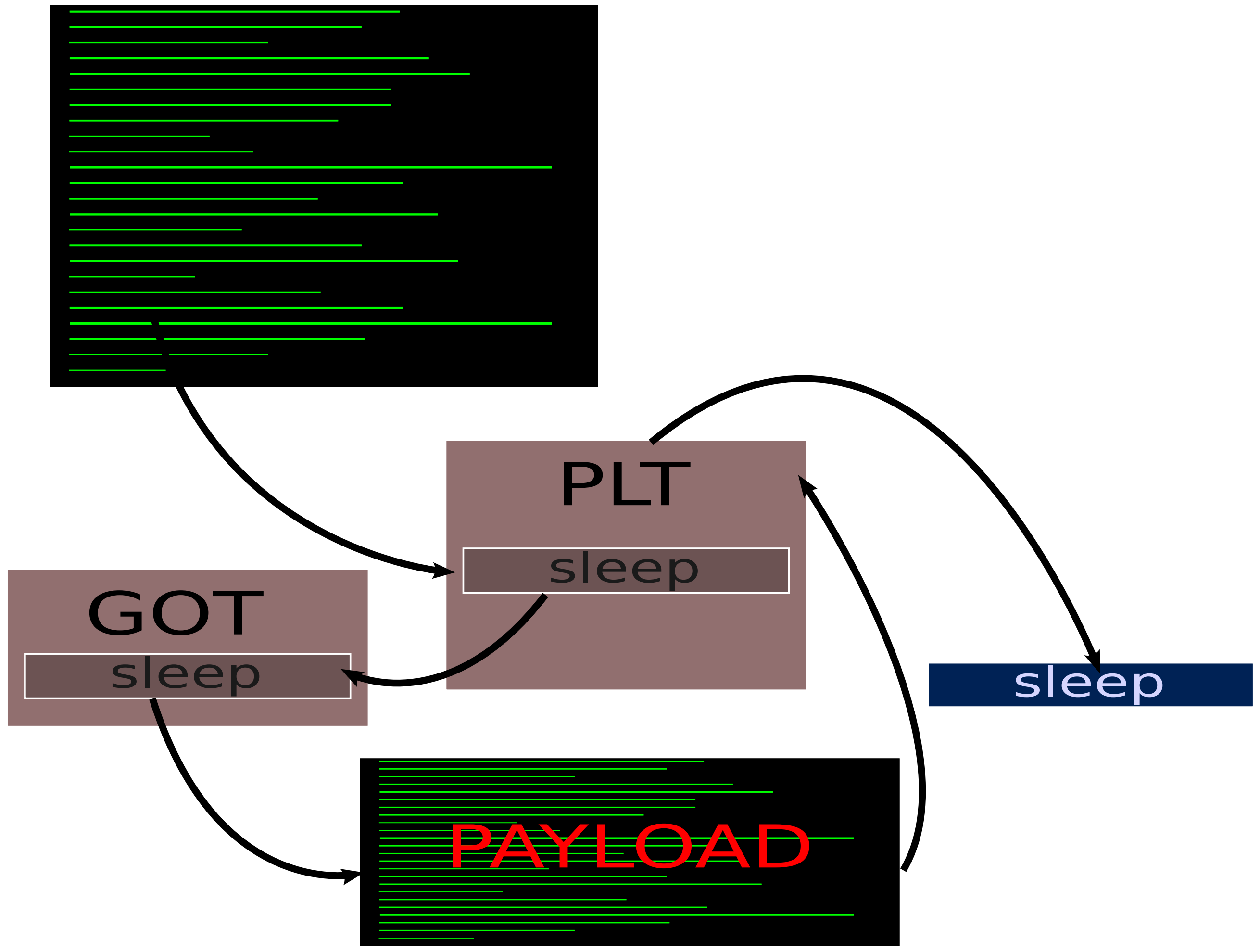
Get the reloc offset of the target import

- Point it to the payload

LD_PRELOAD library containing hijacked import

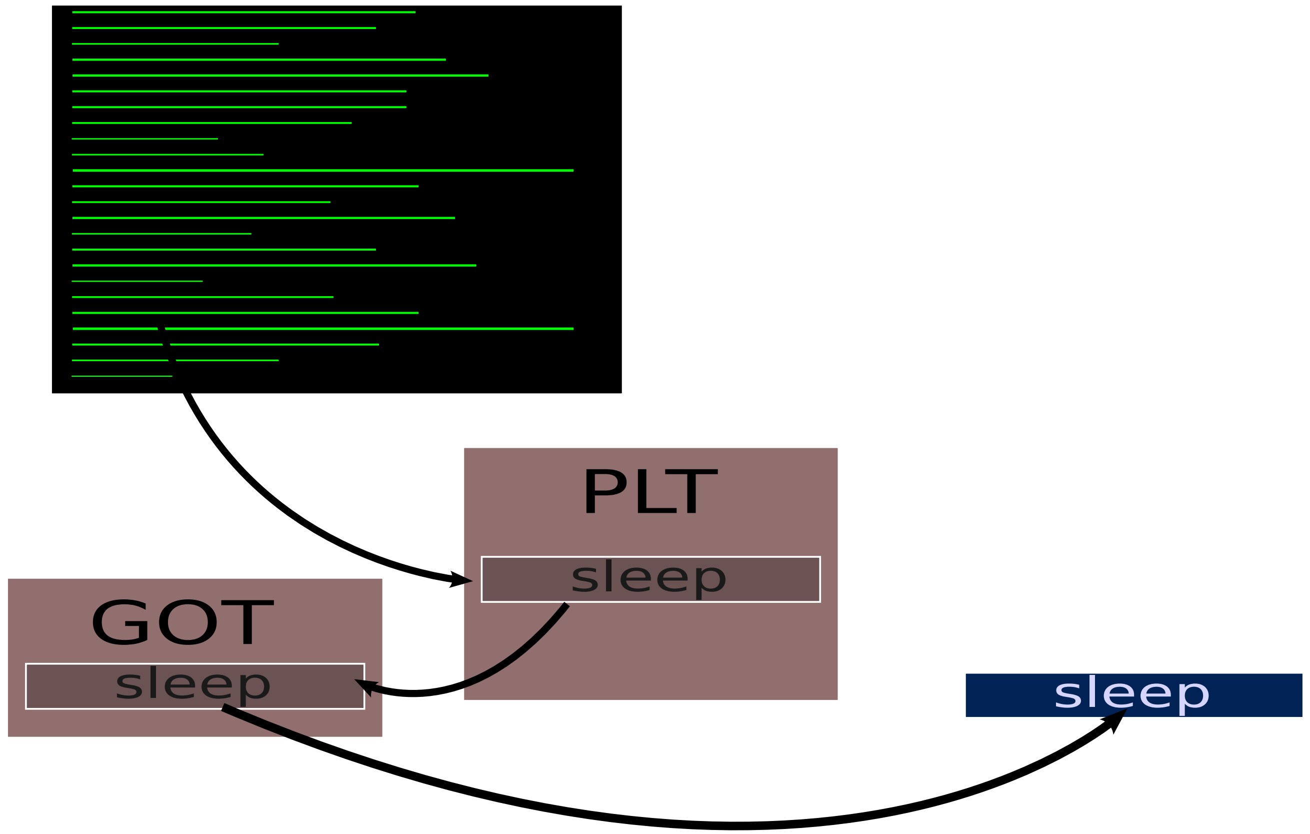
Demo (OneTimeHooks)

1st time:



Demo (OneTimeHooks)

Next time:



Exploiting features (ROP)

Real case:

- Finding rop gadgets

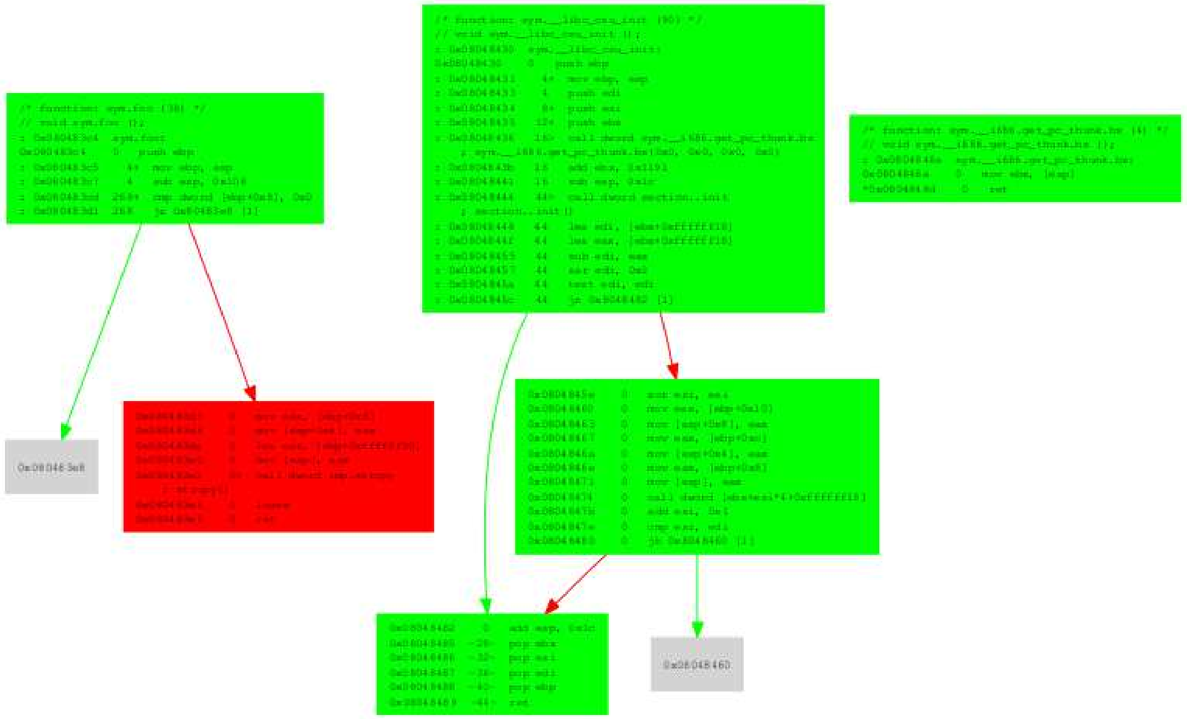
Interesting features:

- Code search
  - Using string patterns
  - Assembling instructions
- Backward disassembly
- Arch-Agnostic API

Exploiting features (GraphDiff)

Real case:

- Diff an old bin with the patched one



```
/* function: sym.foo (38) */
// void sym.foo ();
: 0x080483c4  sym.foo:
0x080483c4      0    push ebp
: 0x080483c5     4+  mov ebp, esp
: 0x080483c7     4   sub esp, 0x108
: 0x080483cd  268+  cmp dword [ebp+0x8], 0x0
: 0x080483d1  268    je 0x80483e8 [1]
```

```
0x080483e8
```

```
0x080483d3      0    mov eax, [ebp+0x8]
0x080483d6      0    mov [esp+0x4], eax
0x080483da      0    lea eax, [ebp+0xfffffef8]
0x080483e0      0    mov [esp], eax
0x080483e3     0?    call dword imp.strcpy
       ; strcpy()
0x080483e8      0    leave
0x080483e9      0    ret
```

```
/* function: sym.__libc_csu_init (90) */
// void sym.__libc_csu_init ();
: 0x08048430  sym.__libc_csu_init:
0x08048430     0    push ebp
: 0x08048431     4+  mov ebp, esp
: 0x08048433     4   push edi
: 0x08048434     8+  push esi
: 0x08048435    12+  push ebx
: 0x08048436    16>  call dword sym.__i686.get_pc_thunk.bx
    ; sym.__i686.get_pc_thunk.bx(0x0, 0x0, 0x0, 0x0)
: 0x0804843b    16   add ebx, 0x1191
: 0x08048441    16   sub esp, 0x1c
: 0x08048444    44>  call dword section..init
    ; section..init()
: 0x08048449    44   lea edi, [ebx+0xffffff18]
: 0x0804844f    44   lea eax, [ebx+0xffffff18]
: 0x08048455    44   sub edi, eax
: 0x08048457    44   sar edi, 0x2
: 0x0804845a    44   test edi, edi
: 0x0804845c    44   je 0x8048482 [1]
```

```
/* function: sym.__i686.get_pc_thunk.bx (4) */
// void sym.__i686.get_pc_thunk.bx ();
: 0x0804848a  sym.__i686.get_pc_thunk.bx:
0x0804848a     0    mov ebx, [esp]
*0x0804848d    0    ret
```

```
0x0804845e     0    xor esi, esi
0x08048460     0    mov eax, [ebp+0x10]
0x08048463     0    mov [esp+0x8], eax
0x08048467     0    mov eax, [ebp+0xc]
0x0804846a     0    mov [esp+0x4], eax
0x0804846e     0    mov eax, [ebp+0x8]
0x08048471     0    mov [esp], eax
0x08048474     0    call dword [ebx+esi*4+0xffffff18]
0x0804847b     0    add esi, 0x1
0x0804847e     0    cmp esi, edi
0x08048480     0    jb 0x8048460 [1]
```

```
0x08048482     0    add esp, 0x1c
0x08048485   -28-   pop ebx
0x08048486   -32-   pop esi
0x08048487   -36-   pop edi
0x08048488   -40-   pop ebp
0x08048489   -44-   ret
```

```
0x08048460
```

Work in progress

Demos (Exploiting)

Simple code search

Gadget search (with context)

Return Oriented Programming Assistant

GraphDiff

Questions?

Ideas, questions?

Yes, of course, here is the ascii penis 8========D

Thanks for listening!