Retro-ingenierie avec radare2

Parce que l'assembleur, c'est sympathique.

Julien (jvoisin) Voisin

PSES 2014

26 juin 2014

whoami

Julien (jvoisin) Voisin

- Reverseur
- Contributeur à radare2
- Étudiant
- Fatigué

whoami

Julien (jvoisin) Voisin

- Reverseur
- Contributeur à radare2
- Étudiant
- Fatigué

whoami

Julien (jvoisin) Voisin

- Reverseur
- Contributeur à radare2
- Étudiant
- Fatigué

whoami

Julien (jvoisin) Voisin

- Reverseur
- Contributeur à radare2
- Étudiant
- Fatigué

Petit sondage

- Linux
- Compilateur
- C
- Assembleur
- Rétro-ingénierie
- Exploit

Petit sondage

- Linux
- Compilateur
- C
- Assembleur
- Rétro-ingénierie
- Exploit

Petit sondage

- Linux
- Compilateur
- C
- Assembleur
- Rétro-ingénierie
- Exploit

Petit sondage

- Linux
- Compilateur
- C
- Assembleur
- Rétro-ingénierie
- Exploit

Petit sondage

- Linux
- Compilateur
- C
- Assembleur
- Rétro-ingénierie
- Exploit

Petit sondage

- Linux
- Compilateur
- C

- Assembleur
- Rétro-ingénierie
- Exploit

Plan

La rétro-ingénierie, c’est quoi ?

Légalité

Comment on fait en pratique ?

Qu’est-ce que la rétro-ingénierie ?

La rétro-ingénierie consiste à étudier un objet pour en déterminer son fonctionnement interne.

Qu’est-ce que la rétro-ingénierie ?

La rétro-ingénierie consiste à étudier un ~~objet~~ logiciel pour en déterminer son fonctionnement interne.

Pour faire quoi ?

- Espionnage industriel
- Cracking
- Création d'exploits
- Analyse de malwares
- Inter-opérabilité
- Pour le fun

Peu importe la couleur du chapeau.

Pour faire quoi ?

- Espionnage industriel
- Cracking
- Création d'exploits
- Analyse de malwares
- Inter-opérabilité
- Pour le fun

Peu importe la couleur du chapeau.

Pour faire quoi ?

- Espionnage industriel
- Cracking
- Création d'exploits
- Analyse de malwares
- Inter-opérabilité
- Pour le fun

Peu importe la couleur du chapeau.

Pour faire quoi ?

- Espionnage industriel
- Cracking
- Création d'exploits
- Analyse de malwares
- Inter-opérabilité
- Pour le fun

Peu importe la couleur du chapeau.

Pour faire quoi ?

- Espionnage industriel
- Cracking
- Création d'exploits
- Analyse de malwares
- Inter-opérabilité
- Pour le fun

Peu importe la couleur du chapeau.

Pour faire quoi ?

- Espionnage industriel
- Cracking
- Création d'exploits
- Analyse de malwares
- Inter-opérabilité
- Pour le fun

Peu importe la couleur du chapeau.

Pour faire quoi ?

- Espionnage industriel
- Cracking
- Création d'exploits
- Analyse de malwares
- Inter-opérabilité
- Pour le fun

Peu importe la couleur du chapeau.

C’est quoi un programme ?

- code source
- ordinateur
- programmeur

} programme

Code source

```c
#include "stdio.h"

int main(int argc, char* argv[]) {
    if (argc > 2)
        printf("Hello %s!\n", argv[1]);
    return 0;
}

```

Assembleur

```
 1 push ebp
 2 mov ebp, esp
 3 and esp, 0xfffffff0
 4 sub esp, 0x10
 5 cmp dword [ebp+0x8], 0x2
 6 jle 0x8048444
 7 mov eax, [ebp+0xc]
 8 add eax, 0x4
 9 mov eax, [eax]
10 mov [esp+0x4], eax
11 mov dword [esp], 0x80484e0
12 call 0x80482f0
13 mov eax, 0x0
14 leave
15 ret
16
```

C’est quoi un processeur ?

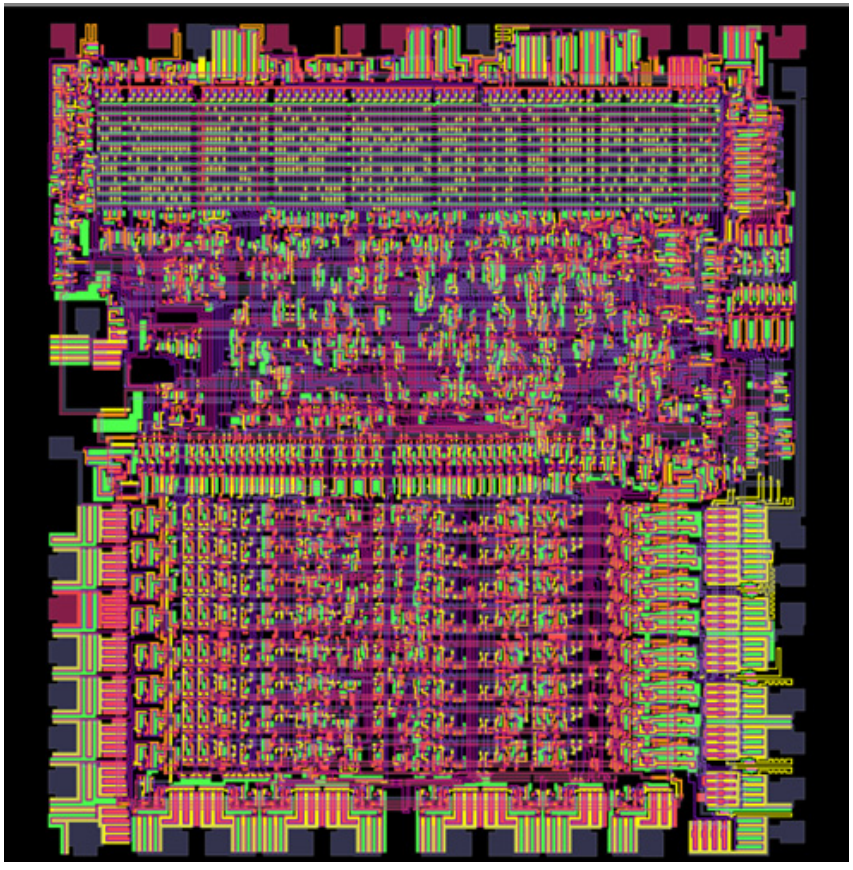


FIGURE : Processeur

Comment ça marche un programme ?

Où sont les zéros et les uns ?

- 11101000 10000000 01001001 00000000
- 74 40 24 80
- je 0x42 ; and al, 0x80

Binaire

Où sont les zéros et les uns ?

- 11101000 10000000 01001001 00000000
- 74 40 24 80
- je 0x42 ; and al, 0x80

Hexadécimal

Où sont les zéros et les uns ?

- 11101000 10000000 01001001 00000000
- 74 40 24 80
- je 0x42 ; and al, 0x80

Mnémonique

Une transformation à sens unique

La compilation n’est pas bijective.

Une transformation à sens unique

La compilation n’est pas inversible.

Plan

La rétro-ingénierie, c’est quoi ?

Légalité

Comment on fait en pratique ?

Est-ce bien légal ?

Article L122-6-1

La reproduction du code du logiciel ou la traduction de la forme de ce code n'est pas soumise à l'autorisation de l'auteur [...] pour obtenir les informations nécessaires à l'interopérabilité d'un logiciel

Mouaif

On peut faire de la rétro-ingénierie pour :

- Corriger des bugs
- Interopérabilité
- Observer/tester/étudier
- Reproduction et traduction du code sous conditions
- On peut tout faire sous couvert de recherche, ou de sécurité.

Plan

La rétro-ingénierie, c’est quoi ?

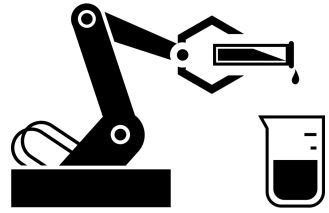
Légalité

Comment on fait en pratique ?

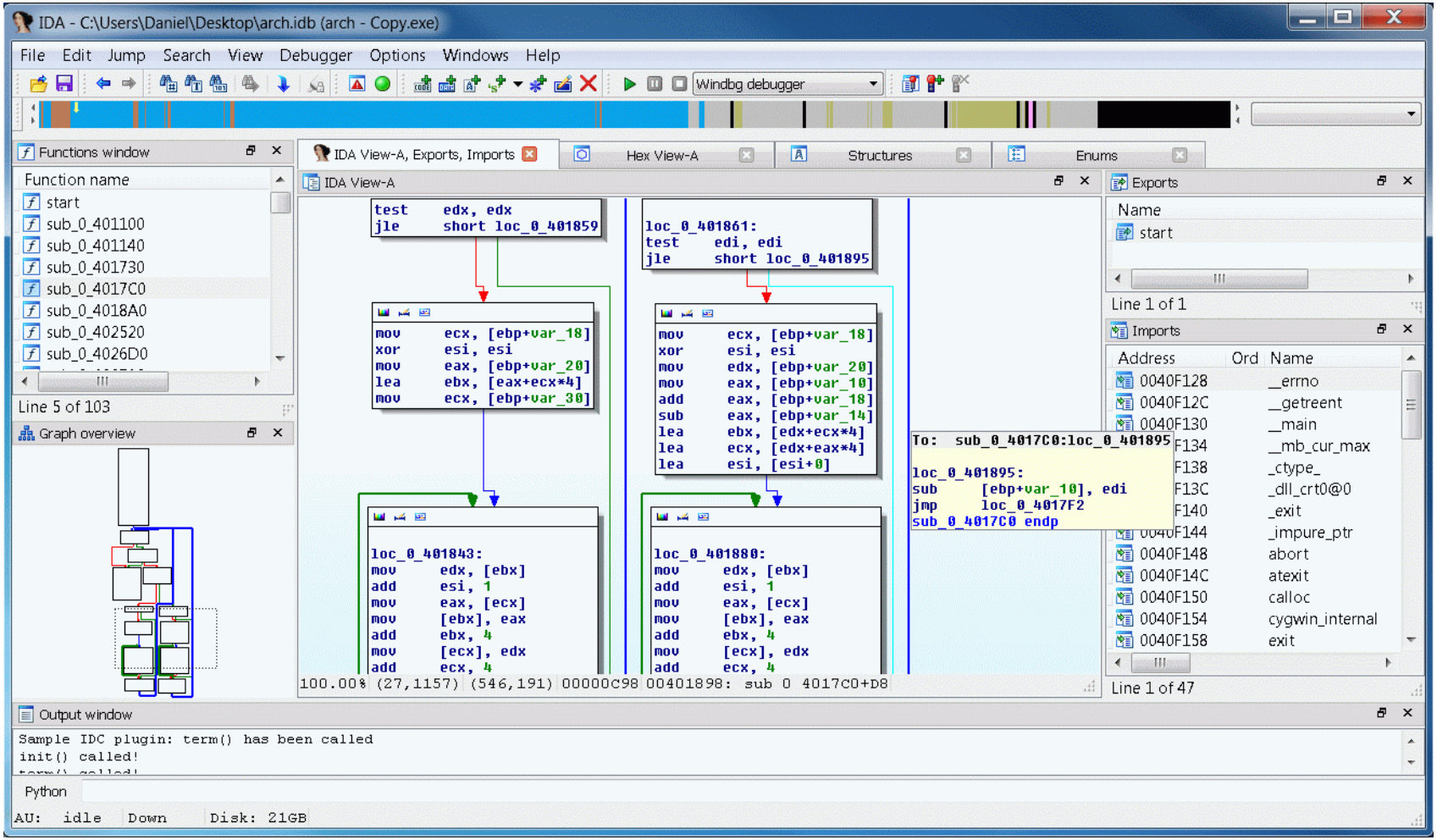
Deux approches

Analyse statique

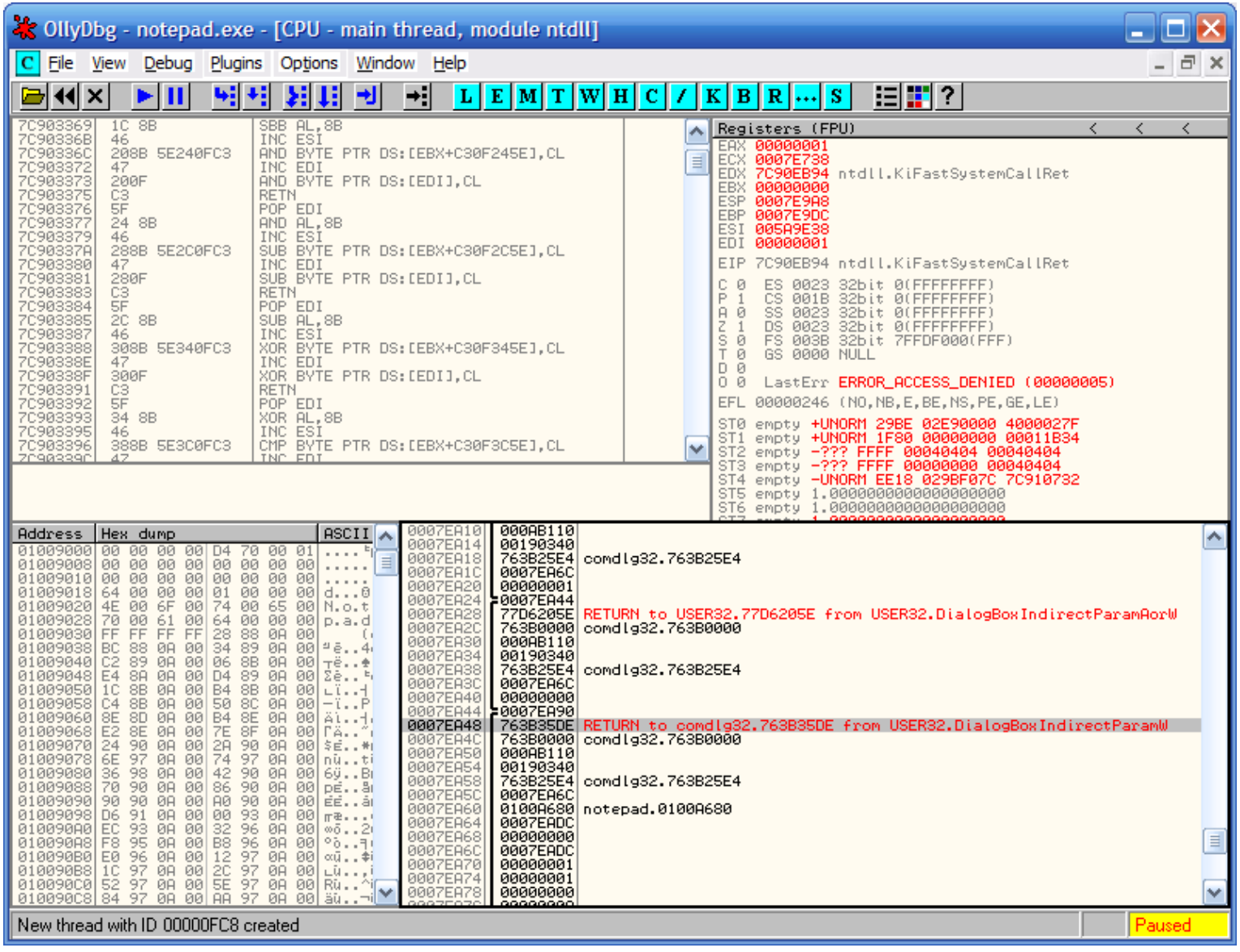
Analyse dynamique



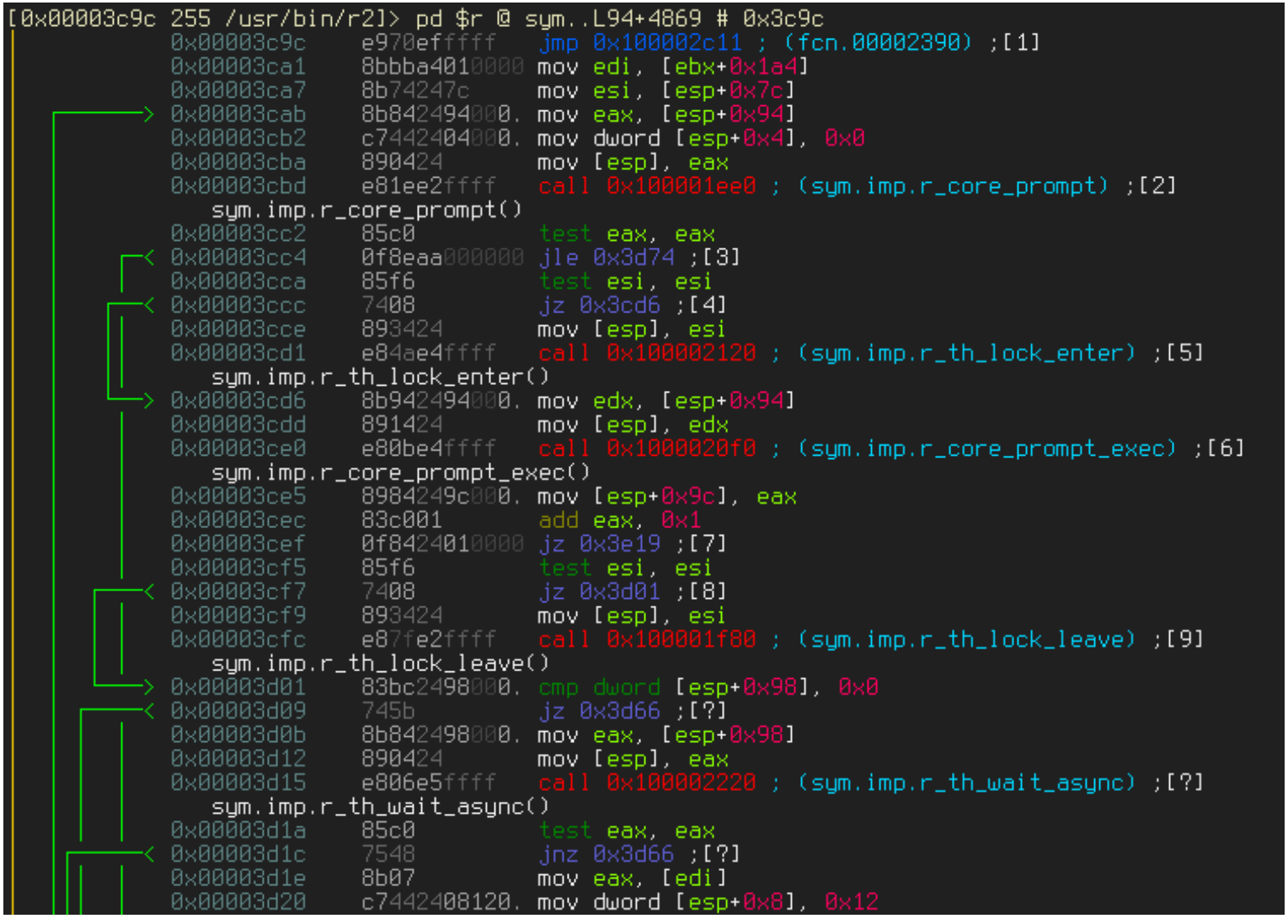
IDA


IDA - C:\Users\Daniel\Desktop\arch.idb (arch - Copy.exe)
File Edit Jump Search View Debugger Options Windows Help
Windbg debugger
Functions window
Function name
start
sub_0_401100
sub_0_401140
sub_0_401730
sub_0_4017C0
sub_0_4018A0
sub_0_402520
sub_0_4026D0
Line 5 of 103
Graph overview
IDA View-A, Exports, Imports
Hex View-A
Structures
Enums
IDA View-A
test edx, edx
jle short loc_0_401859
loc_0_401861:
test edi, edi
jle short loc_0_401895
mov ecx, [ebp+var_18]
xor esi, esi
mov eax, [ebp+var_20]
lea ebx, [eax+ecx*4]
mov ecx, [ebp+var_30]
mov ecx, [ebp+var_18]
xor esi, esi
mov edx, [ebp+var_20]
mov eax, [ebp+var_10]
add eax, [ebp+var_18]
sub eax, [ebp+var_14]
lea ebx, [edx+ecx*4]
lea ecx, [edx+eax*4]
lea esi, [esi+0]
loc_0_401843:
mov edx, [ebx]
add esi, 1
mov eax, [ecx]
mov [ebx], eax
add ebx, 4
mov [ecx], edx
add ecx, 4
loc_0_401880:
mov edx, [ebx]
add esi, 1
mov eax, [ecx]
mov [ebx], eax
add ebx, 4
mov [ecx], edx
add ecx, 4
To: sub_0_4017C0:loc_0_401895
loc_0_401895:
sub [ebp+var_10], edi
jmp loc_0_4017F2
sub_0_4017C0 endp
100.00% (27,1157) (546,191) 00000C98 00401898: sub_0_4017C0+D8
Exports
Name
start
Line 1 of 1
Imports
Address Ord Name
0040F128 __errno
0040F12C __getreent
0040F130 __main
0040F134 __mb_cur_max
0040F138 _ctype_
0040F13C _dll_crt0@0
0040F140 _exit
0040F144 _impure_ptr
0040F148 abort
0040F14C atexit
0040F150 calloc
0040F154 cygwin_internal
0040F158 exit
Line 1 of 47
Output window
Sample IDC plugin: term() has been called
init() called!
Python
AU: idle Down Disk: 21GB

OllyDBG


OllyDbg - notepad.exe - [CPU - main thread, module ntdll]
File View Debug Plugins Options Window Help
7C903369 1C 8B SBB AL,8B
7C90336B 46 INC ESI
7C90336C 208B 5E240FC3 AND BYTE PTR DS:[EBX+C30F245E],CL
7C903372 47 INC EDI
7C903373 200F AND BYTE PTR DS:[EDI],CL
7C903375 C3 RETN
7C903376 5F POP EDI
7C903377 24 8B AND AL,8B
7C903379 46 INC ESI
7C90337A 288B 5E2C0FC3 SUB BYTE PTR DS:[EBX+C30F2C5E],CL
7C903380 47 INC EDI
7C903381 280F SUB BYTE PTR DS:[EDI],CL
7C903383 C3 RETN
7C903384 5F POP EDI
7C903385 2C 8B SUB AL,8B
7C903387 46 INC ESI
7C903388 308B 5E340FC3 XOR BYTE PTR DS:[EBX+C30F345E],CL
7C90338E 47 INC EDI
7C90338F 300F XOR BYTE PTR DS:[EDI],CL
7C903391 C3 RETN
7C903392 5F POP EDI
7C903393 34 8B XOR AL,8B
7C903395 46 INC ESI
7C903396 388B 5E3C0FC3 CMP BYTE PTR DS:[EBX+C30F3C5E],CL
Registers (FPU)
EAX 00000001
ECX 0007E738
EDX 7C90EB94 ntdll.KiFastSystemCallRet
EBX 00000000
ESP 0007E9A8
EBP 0007E9DC
ESI 005A9E38
EDI 00000001
EIP 7C90EB94 ntdll.KiFastSystemCallRet
C 0 ES 0023 32bit 0(FFFFFFFF)
P 1 CS 001B 32bit 0(FFFFFFFF)
A 0 SS 0023 32bit 0(FFFFFFFF)
Z 1 DS 0023 32bit 0(FFFFFFFF)
S 0 FS 003B 32bit 7FFDF000(FFF)
T 0 GS 0000 NULL
D 0
O 0 LastErr ERROR_ACCESS_DENIED (00000005)
EFL 00000246 (NO,NB,E,BE,NS,PE,GE,LE)
ST0 empty +UNORM 29BE 02E90000 4000027F
ST1 empty +UNORM 1F80 00000000 00011B34
ST2 empty -??? FFFF 00040404 00040404
ST3 empty -??? FFFF 00000000 00040404
ST4 empty -UNORM EE18 029BF07C 7C910732
ST5 empty 1.0000000000000000000
ST6 empty 1.0000000000000000000
Address Hex dump ASCII
01009000 00 00 00 00 D4 70 00 01
01009008 00 00 00 00 00 00 00 00
01009010 00 00 00 00 00 00 00 00
01009018 64 00 00 00 01 00 00 00
01009020 4E 00 6F 00 74 00 65 00
01009028 70 00 61 00 64 00 00 00
01009030 FF FF FF FF 28 88 0A 00
01009038 BC 88 0A 00 34 89 0A 00
01009040 C2 89 0A 00 06 8B 0A 00
01009048 E4 8A 0A 00 D4 89 0A 00
01009050 1C 8B 0A 00 B4 8B 0A 00
01009058 C4 8B 0A 00 50 8C 0A 00
01009060 8E 8D 0A 00 B4 8E 0A 00
01009068 E2 8E 0A 00 7E 8F 0A 00
01009070 24 90 0A 00 2A 90 0A 00
01009078 6E 97 0A 00 74 97 0A 00
01009080 36 98 0A 00 42 90 0A 00
01009088 70 90 0A 00 86 90 0A 00
01009090 90 90 0A 00 A0 90 0A 00
01009098 D6 91 0A 00 00 93 0A 00
010090A0 EC 93 0A 00 32 96 0A 00
010090A8 F8 95 0A 00 B8 96 0A 00
010090B0 E0 96 0A 00 12 97 0A 00
010090B8 1C 97 0A 00 2C 97 0A 00
010090C0 52 97 0A 00 5E 97 0A 00
010090C8 84 97 0A 00 AA 97 0A 00
0007EA10 000AB110
0007EA14 00190340
0007EA18 763B25E4 comdlg32.763B25E4
0007EA1C 0007EA6C
0007EA20 00000001
0007EA24 0007EA44
0007EA28 77D6205E RETURN to USER32.77D6205E from USER32.DialogBoxIndirectParamAorW
0007EA2C 763B0000 comdlg32.763B0000
0007EA30 000AB110
0007EA34 00190340
0007EA38 763B25E4 comdlg32.763B25E4
0007EA3C 0007EA6C
0007EA40 00000000
0007EA44 0007EA90
0007EA48 763B35DE RETURN to comdlg32.763B35DE from USER32.DialogBoxIndirectParamW
0007EA4C 763B0000 comdlg32.763B0000
0007EA50 000AB110
0007EA54 00190340
0007EA58 763B25E4 comdlg32.763B25E4
0007EA5C 0007EA6C
0007EA60 0100A680 notepad.0100A680
0007EA64 0007EADC
0007EA68 00000000
0007EA6C 0007EADC
0007EA70 00000001
0007EA74 00000001
0007EA78 00000000
New thread with ID 00000FC8 created
Paused

Radare2


[0x00003c9c 255 /usr/bin/r2]> pd $r @ sym..L94+4869 # 0x3c9c
0x00003c9c e970efffff jmp 0x100002c11 ; (fcn.00002390) ;[1]
0x00003ca1 8bbba4010000 mov edi, [ebx+0x1a4]
0x00003ca7 8b74247c mov esi, [esp+0x7c]
0x00003cab 8b842494000. mov eax, [esp+0x94]
0x00003cb2 c7442404000. mov dword [esp+0x4], 0x0
0x00003cba 890424 mov [esp], eax
0x00003cbd e81ee2ffff call 0x100001ee0 ; (sym.imp.r_core_prompt) ;[2]
sym.imp.r_core_prompt()
0x00003cc2 85c0 test eax, eax
0x00003cc4 0f8eaa000000 jle 0x3d74 ;[3]
0x00003cca 85f6 test esi, esi
0x00003ccc 7408 jz 0x3cd6 ;[4]
0x00003cce 893424 mov [esp], esi
0x00003cd1 e84ae4ffff call 0x100002120 ; (sym.imp.r_th_lock_enter) ;[5]
sym.imp.r_th_lock_enter()
0x00003cd6 8b942494000. mov edx, [esp+0x94]
0x00003cdd 891424 mov [esp], edx
0x00003ce0 e80be4ffff call 0x1000020f0 ; (sym.imp.r_core_prompt_exec) ;[6]
sym.imp.r_core_prompt_exec()
0x00003ce5 8984249c000. mov [esp+0x9c], eax
0x00003cec 83c001 add eax, 0x1
0x00003cef 0f8424010000 jz 0x3e19 ;[7]
0x00003cf5 85f6 test esi, esi
0x00003cf7 7408 jz 0x3d01 ;[8]
0x00003cf9 893424 mov [esp], esi
0x00003cfc e87fe2ffff call 0x100001f80 ; (sym.imp.r_th_lock_leave) ;[9]
sym.imp.r_th_lock_leave()
0x00003d01 83bc2498000. cmp dword [esp+0x98], 0x0
0x00003d09 745b jz 0x3d66 ;[?]
0x00003d0b 8b842498000. mov eax, [esp+0x98]
0x00003d12 890424 mov [esp], eax
0x00003d15 e806e5ffff call 0x100002220 ; (sym.imp.r_th_wait_async) ;[?]
sym.imp.r_th_wait_async()
0x00003d1a 85c0 test eax, eax
0x00003d1c 7548 jnz 0x3d66 ;[?]
0x00003d1e 8b07 mov eax, [edi]
0x00003d20 c7442408120. mov dword [esp+0x8], 0x12

Historique

- Radare vient de fêter son 8e anniversaire (2006)
- Re-écrit en 2010
- Originellement un éditeur hexadécimal
- 450.000 lignes de code
- 5000^{eme} commit hier !
- $\sim$ 10 contributeurs actifs

Historique

- Radare vient de fêter son 8e anniversaire (2006)
- Re-écrit en 2010
- Originellement un éditeur hexadécimal
- 450.000 lignes de code
- 5000^{eme} commit hier !
- $\sim$ 10 contributeurs actifs

Historique

- Radare vient de fêter son 8e anniversaire (2006)
- Re-écrit en 2010
- Originellement un éditeur hexadécimal
- 450.000 lignes de code
- 5000^{eme} commit hier !
- $\sim$ 10 contributeurs actifs

Historique

- Radare vient de fêter son 8e anniversaire (2006)
- Re-écrit en 2010
- Originellement un éditeur hexadécimal
- 450.000 lignes de code
- 5000^{eme} commit hier !
- $\sim$ 10 contributeurs actifs

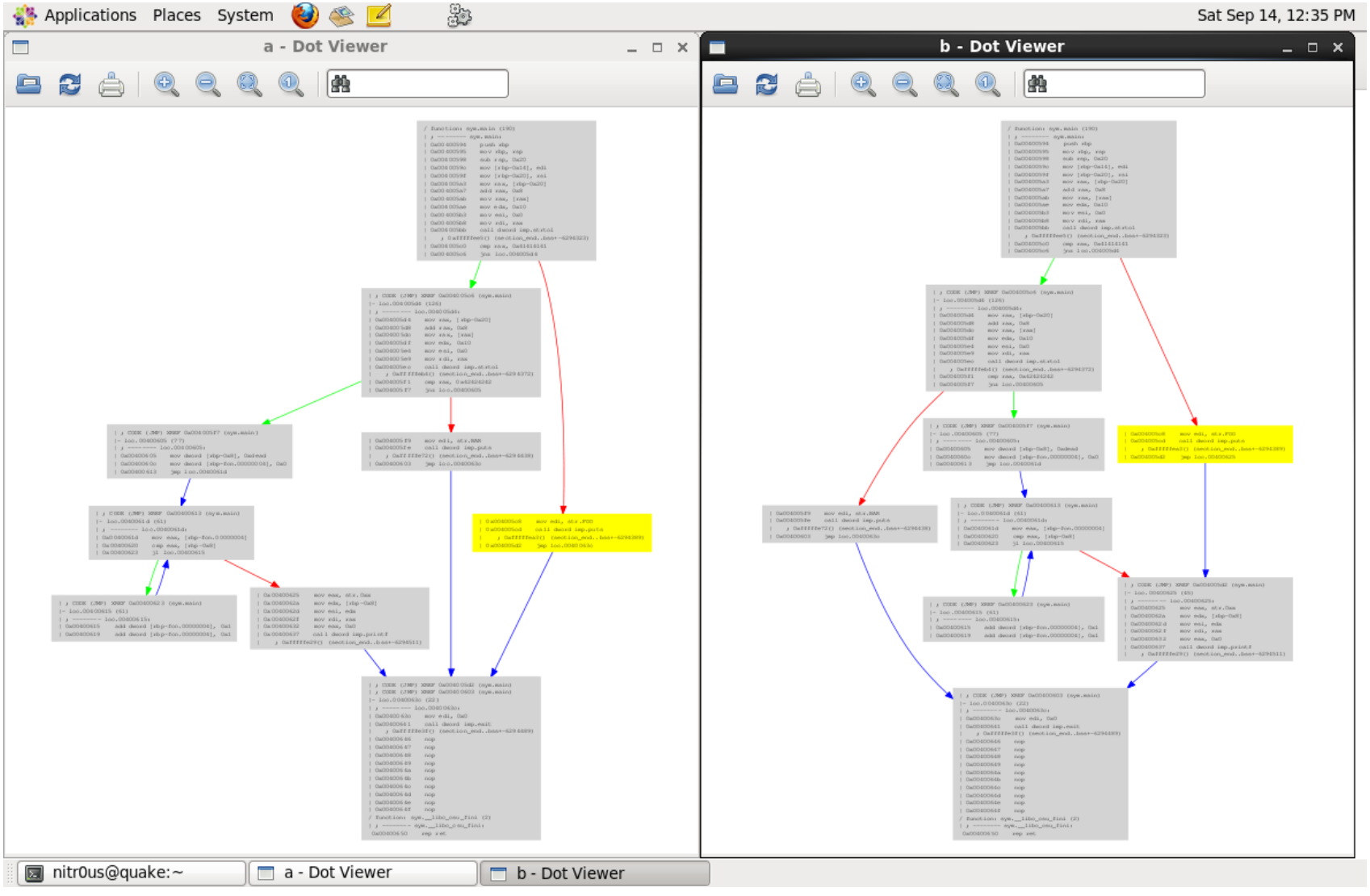
Historique

- Radare vient de fêter son 8e anniversaire (2006)
- Re-écrit en 2010
- Originellement un éditeur hexadécimal
- 450.000 lignes de code
- 5000^{eme} commit hier !
- $\sim$ 10 contributeurs actifs

Historique

- Radare vient de fêter son 8e anniversaire (2006)
- Re-écrit en 2010
- Originellement un éditeur hexadécimal
- 450.000 lignes de code
- 5000^{eme} commit hier !
- $\sim$ 10 contributeurs actifs

Bindiff


Applications
Places
System
Sat Sep 14, 12:35 PM
a - Dot Viewer
b - Dot Viewer
nitr0us@quake:~
a - Dot Viewer
b - Dot Viewer

Analyse

```
[0x08048320]> pdf@sym.main
|           ; DATA XREF from 0x08048337 (fcn.08048316)
          0x0804841d    55              push ebp
          0x0804841e    89e5            mov ebp, esp
          0x08048420    83e4f0          and esp, 0xfffffff0
          0x08048423    83ec10          sub esp, 0x10
          0x08048426    837d0802        cmp dword [ebp+0x8], 0x2
      ,=< 0x0804842a    7e18            jle loc.08048444
      |   0x0804842c    8b450c          mov eax, [ebp+0xc]
      |   0x0804842f    83c004          add eax, 0x4
      |   0x08048432    8b00            mov eax, [eax]
      |   0x08048434    89442404        mov [esp+0x4], eax
      |   0x08048438    c70424e0840. mov dword [esp], str.Hello_s_ ;  0x080484e0
      |   0x0804843f    e8acfeffff      call sym.imp.printf
|     `-  ; JMP XREF from 0x0804842a (unk)
      `-> 0x08048444    b800000000      mov eax, 0x0
          0x08048449    c9              leave
          0x0804844a    c3              ret
```

Graphes


[0x080483f0]> VV @ sym.frame_dummy (nodes 4)
0x080483f0
mov eax, [0x8049f10]
test eax, eax
je 0x8048418
mov eax, 0x0
0x08048418
jmp 0x8048390
push ebp
mov ebp, esp
and esp, 0xfffffff0
sub esp, 0x10
cmp dword [ebp+0x8], 0x2
jle 0x8048444
0x0804842c
mov eax, [ebp+0xc]
add eax, 0x4
mov eax, [eax]
mov [esp+0x4], eax
mov dword [esp], 0x80484e0
call 0x80482f0
0x08048444
mov eax, 0x0
leave
ret

Mais aussi

- Assembleur
- Hexedit
- Debugger
- Shellcodes

Ça tourne sur quoi ?

- Windows
- GNU/Linux
- MacOS
- Android
- iPhone
- BSD
- Solaris
- Haiku

Architectures

- i8080
- 8051
- x86
- avr
- arc
- arm
- c55x+
- dalvik
- ebc
- nios2
- powerpc
- z80
- psosvm
- m68k
- whitespace
- brainfuck
- malbolge
- msil
- gb
- java
- sparc
- mips
- snes
- dcpu16
- csr

Démo

Ok, mais comment on s’en sert ?