Wstęp do Reverse engineeringu

O mnie

- Agnieszka Bielec
- Eternal
- pracuję w CERT Polska jako Malware Analyst
- CTFy z p4
- prowadze bloga eternal.red
- lubie ścianki wpinaczkowe i rower
- twitter

Reverse Engineering - co to takiego?

proces badania programu w celu ustalenia jak on dokładnie działa lub jak działają jego konkretne mechanizmy

W języku polskim: analiza wsteczna

A do czego może się przydać?

- Odtworzenie źrodeł programu
- modyfikowanie programu do którego nie mamy źródeł
- tworzenie “hackow”/modów do gier
- sprawdzenie dlaczego program nam się wywala

Zmodyfikujmy gre Icy Tower 1.3.1

Co to za plik?

```
a@x:~/Desktop/Wstep_do_reverse_engineeringu$ file icytower13.exe
icytower13.exe: PE32 executable (GUI) Intel 80386 (stripped to exte
rnal PDB), for MS Windows, UPX compressed
```

```
a@x:~/Desktop/Wstep_do_reverse_engineeringu$ strings icytower13.exe | grep -i upx
UPX0
UPX1
UPX!
```

```
a@x:~/Desktop/Wstep_do_reverse_engineeringu$ objdump -x icytower13.exe | tail -n13

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 UPX0          000b0000  00401000  00401000  00000200  2**2
                  CONTENTS, ALLOC, CODE
  1 UPX1          00045200  004b1000  004b1000  00000200  2**2
                  CONTENTS, ALLOC, LOAD, CODE, DATA
  2 .rsrc         00000600  004f7000  004f7000  00045400  2**2
                  CONTENTS, ALLOC, LOAD, DATA
SYMBOL TABLE:
no symbols
```

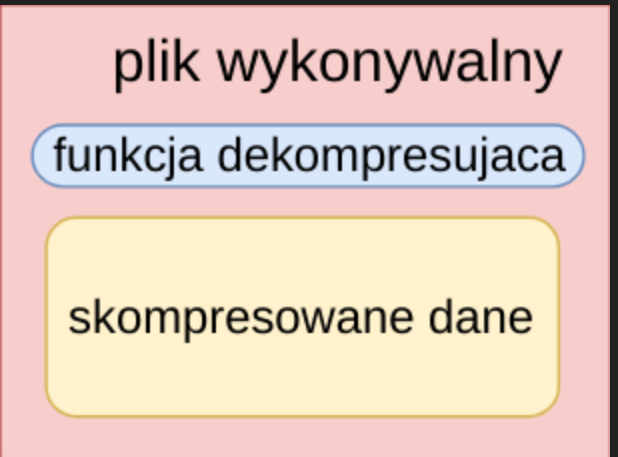
UPX - co to takiego ?

- Jest jednym z packerów

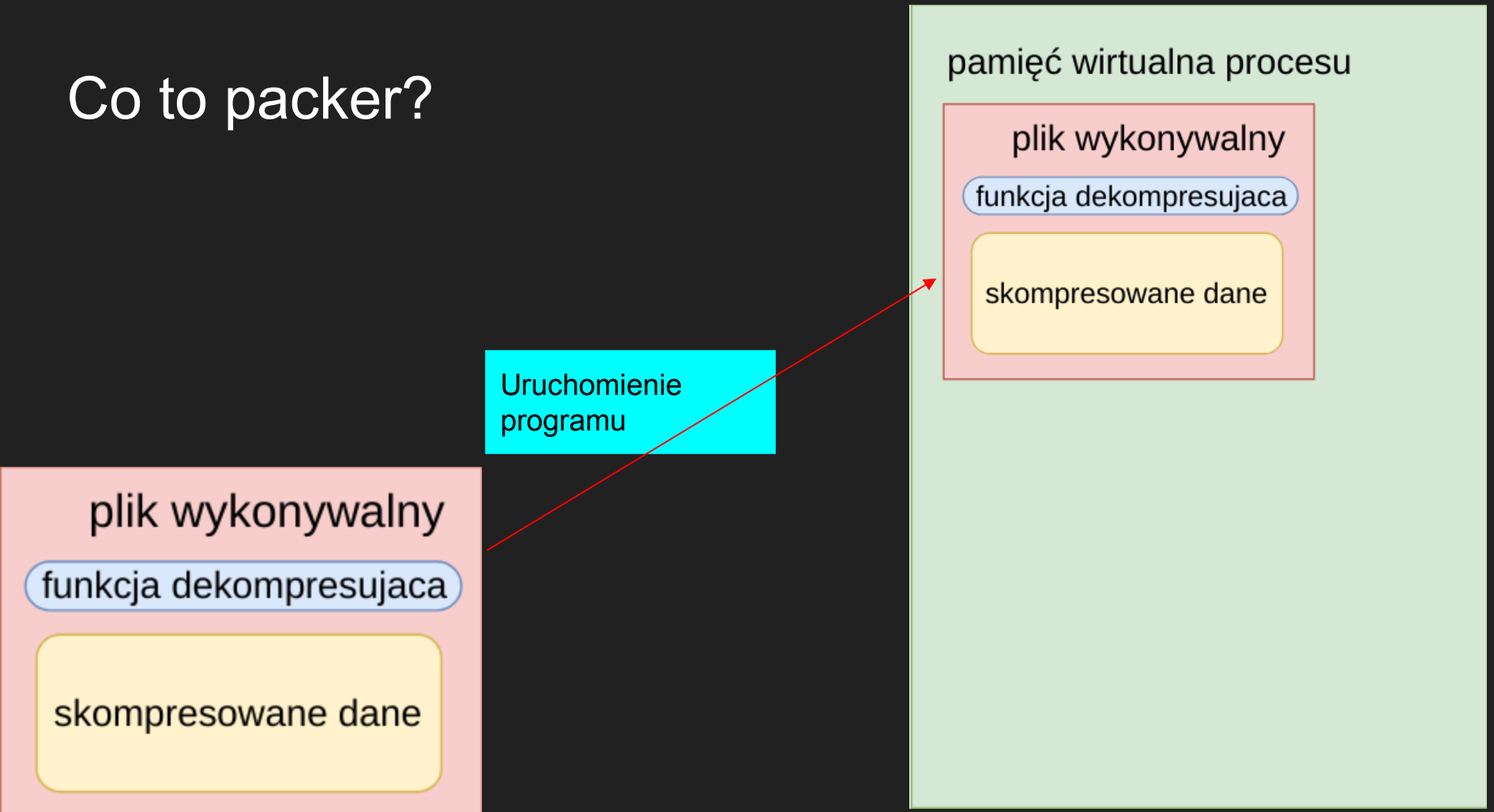
Co to packer?

Packery służą do kompresji plików wykonywalnych

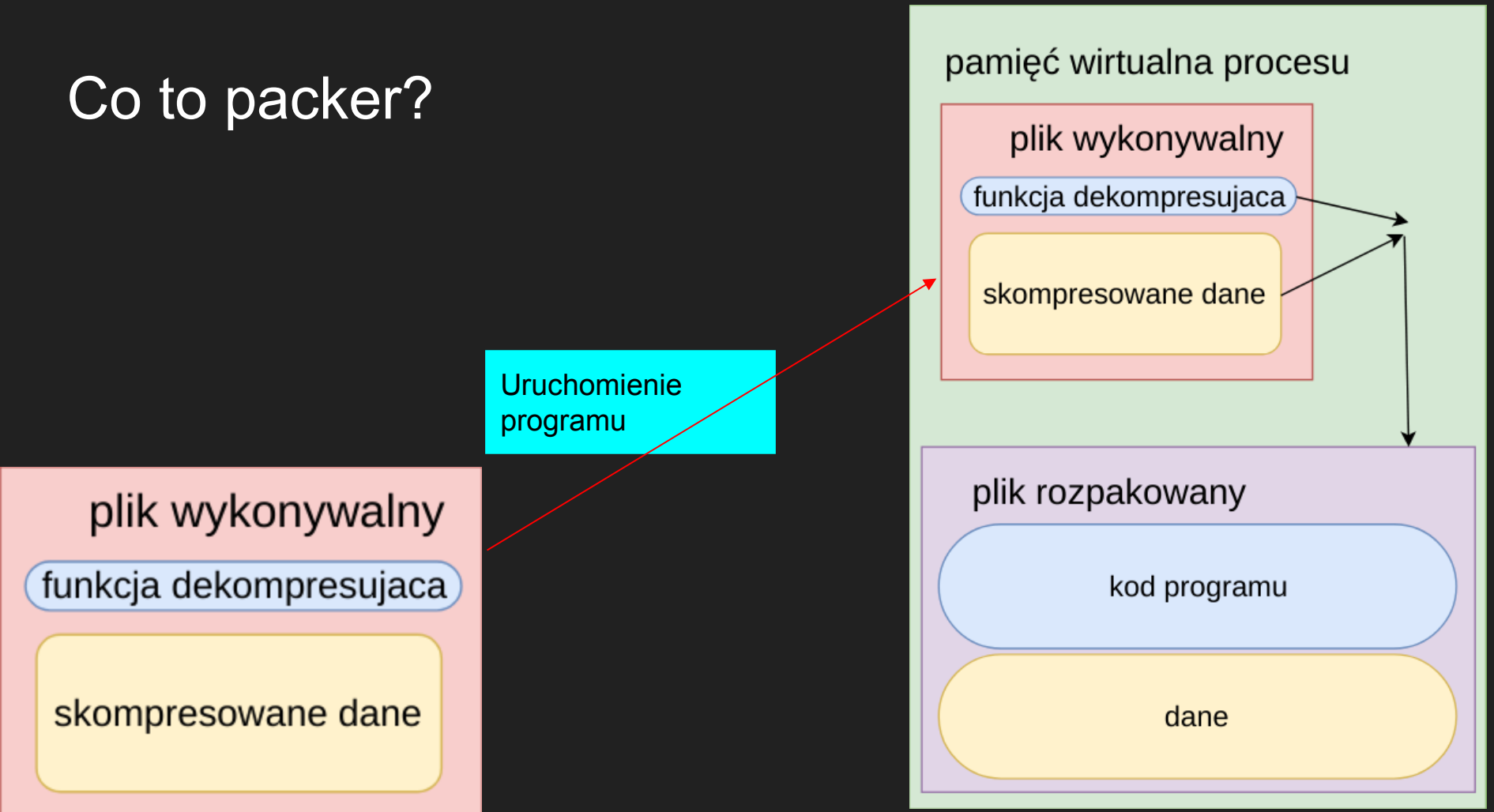
Co to packer?


plik wykonywalny
funkcja dekompresujaca
skompresowane dane

Co to packer?


pamięć wirtualna procesu
plik wykonywalny
funkcja dekompresujaca
skompresowane dane
Uruchomienie programu
plik wykonywalny
funkcja dekompresujaca
skompresowane dane

Co to packer?


pamięć wirtualna procesu
plik wykonywalny
funkcja dekompresujaca
skompresowane dane
Uruchomienie programu
plik wykonywalny
funkcja dekompresujaca
skompresowane dane
plik rozpakowany
kod programu
dane

rozpakowanie UPX’a

```
a@x:~/Desktop/Wstep_do_reverse_engineeringu$ upx -d icytower13.exe
                       Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2017
UPX 3.94        Markus Oberhumer, Laszlo Molnar & John Reiser   May 12th 2017

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
    715264 <-    285184   39.87%    win32/pe     icytower13.exe

Unpacked 1 file.
```

W jakim języku został napisany Icy Tower?

W jakim języku został napisany plik?

- metoda 1: strings

```
a@x:~/Desktop/Wstep_do_reverse_engineeringu$ strings icytower13.exe > stringi
```

W jakim języku został napisany plik?

- metoda 1: strings

```
-LIBGCCW32-EH-2-SJLJ-GTHR-MINGW32
w32_sharedptr->size == sizeof(W32_EH_SHARED)
%s:%u: failed assertion `%s'
../../gcc/gcc/config/i386/w32-shared-ptr.c
```

W jakim języku został napisany plik?

- metoda 1: strings

```
GetPaletteEntries
GetStockObject
GetSystemPaletteEntries
RealizePalette
SelectObject
SelectPalette
SetPaletteEntries
StretchBlt
StretchDIBits
AddAtomA
CloseHandle
CreateEventA
DeleteCriticalSection
DuplicateHandle
EnterCriticalSection
```

```
-LIBGCCW32-EH-2-SJLJ-GTHR-MINGW32
w32_sharedptr->size == sizeof(W32_EH_SHARED)
%s:%u: failed assertion `%s'
../../gcc/gcc/config/i386/w32-shared-ptr.c
```

W jakim języku został napisany plik?

- metoda 1: strings

```
GetPaletteEntries
GetStockObject
GetSystemPaletteEntries
RealizePalette
SelectObject
SelectPalette
SetPaletteEntries
StretchBlt
StretchDIBits
AddAtomA
CloseHandle
CreateEventA
DeleteCriticalSection
DuplicateHandle
EnterCriticalSection
```

```
-LIBGCCW32-EH-2-SJLJ-GTHR-MINGW32
w32_sharedptr->size == sizeof(W32_EH_SHARED)
%s:%u: failed assertion `%s'
../../gcc/gcc/config/i386/w32-shared-ptr.c
```

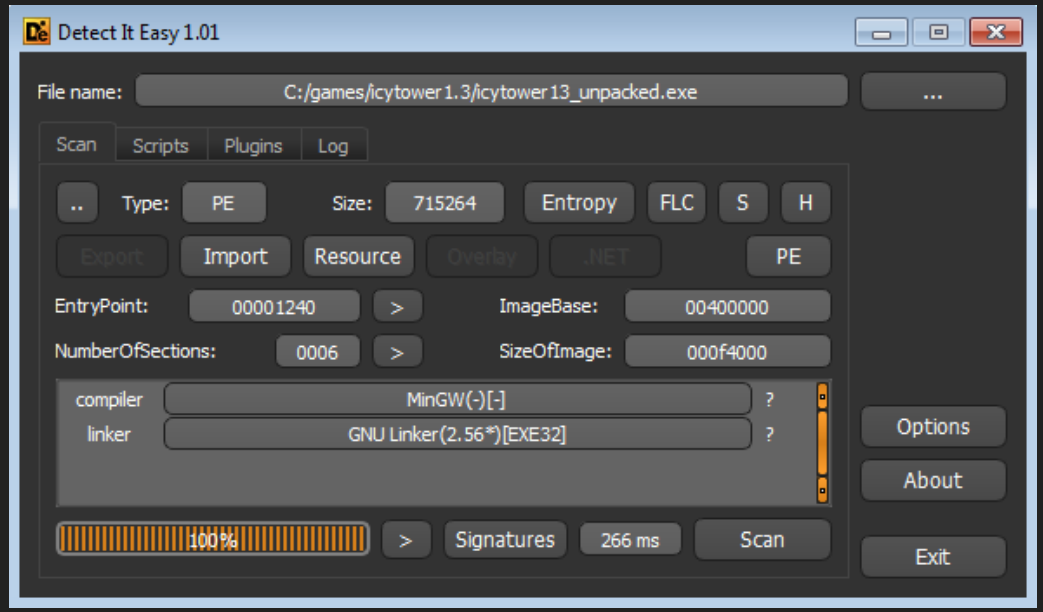
```
Play Again
Watch Replay
Save Replay
Main Menu
Start Game
Load Replay
Instructions
Options
Exit
```

W jakim języku został napisany plik?

- metoda 1: strings

```
a@x:~/Desktop/Wstep_do_reverse_engineeringu$ strings icytower13.exe > stringi
```

- metoda 2: Detect It Easy (DIE)



Prosta modyfikacja programu - hexedytorem

Start Game -> Rozpocznij Gre

hexedytor HxD

Nazwy funkcji

```
a@x:~/Desktop/Wstep_do_reverse_engineeringu$ objdump --syms icytower13.exe

icytower13.exe:     file format pei-i386

SYMBOL TABLE:
no symbols
```

Nazwy funkcji w innej wersji Icy Towera 1.5

```
a@x:~/Desktop/Wstep_do_reverse_engineeringu$ objdump --syms icytower15.exe | grep jump
[254](sec  1)(fl 0x00)(ty  20)(scl   2) (nx 1) 0x000030f4 _add_jump_sequence
[424](sec  1)(fl 0x00)(ty  20)(scl   2) (nx 0) 0x00005ecc _play_jump_sound
[660](sec  1)(fl 0x00)(ty  20)(scl   2) (nx 0) 0x00017678 _jump_player
[10499](sec  4)(fl 0x00)(ty   0)(scl   2) (nx 0) 0x0001d728 _jumpSequence
[10687](sec  4)(fl 0x00)(ty   0)(scl   2) (nx 0) 0x00020cd8 _rejump
[10788](sec  4)(fl 0x00)(ty   0)(scl   2) (nx 0) 0x000002b0 _jump_sound
```

strip

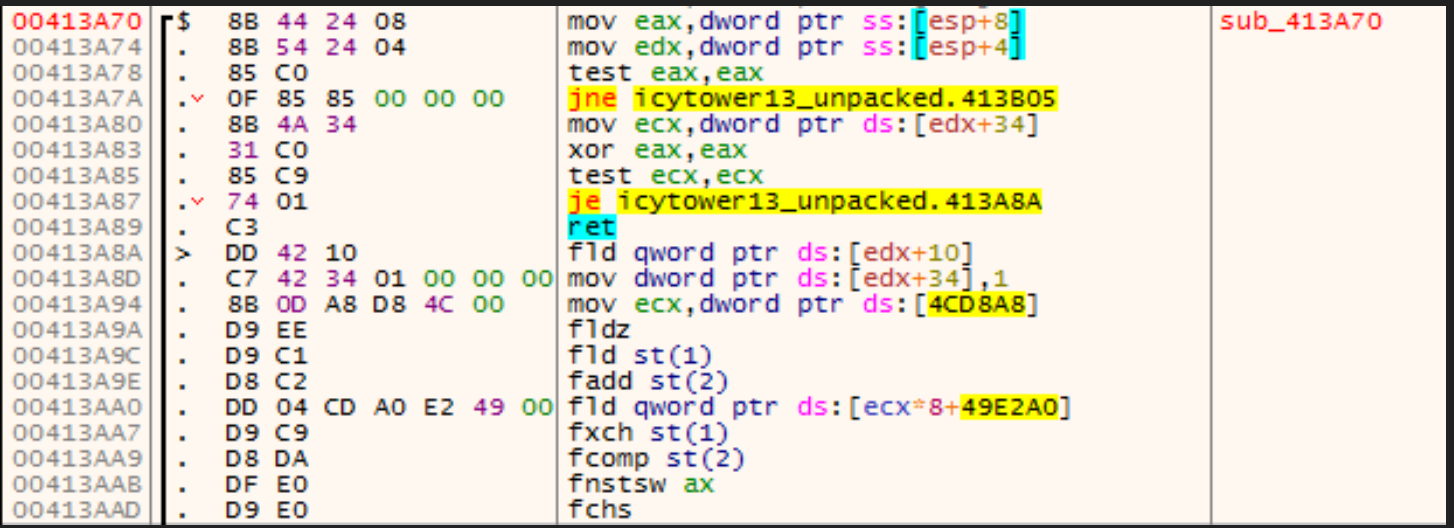
- program służący do usuwania symboli z plików wykonywalnych

To jak znajdziemy funkcję odpowiedzialną za skok?

użyjemy do tego debuggera x64dbg

1. PPM on code -> Analysis -> Analyse Module
2. Menu -> View -> Functions
3. PPM -> Set breakpoint on all commands
4. Rób wszystko, wyłącz breakpointy, a później skocz

Początek funkcji odpowiedzialnej za skok

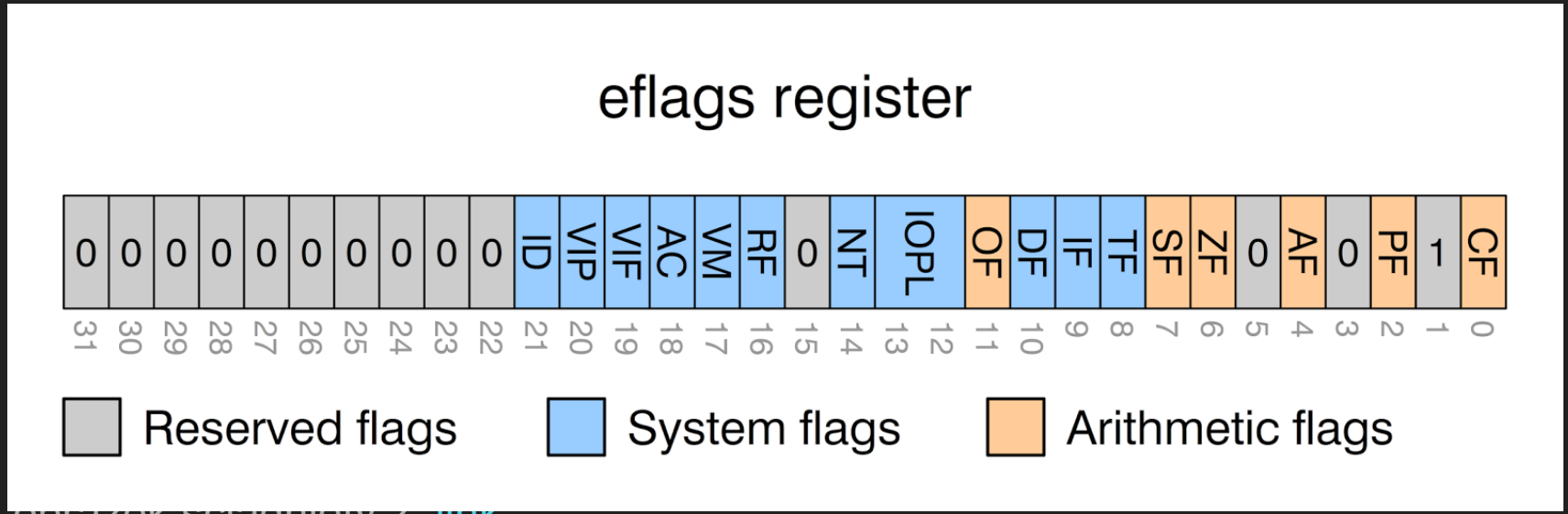


Asembler x86-32 - Jak działają skoki warunkowe?

- Asembler posiada zbiór instrukcji które wykonują operacje na liczbach i ustawiają flagi w rejestrze EFLAGS. Przykładowe instrukcje to:
  - test
  - cmp
  - sub
  - add
- Przykładowe flagi rejestru EFLAGS to:
  - ZF (zero flag) - wynik operacji wyniósł zero
  - SF (sign flag) - wynik operacji jest liczbą ujemną - najstarszy bit jest ustawiony na 1
  - CF (carry flag) - wystąpił integer overflow (unsigned) - gdy wynik operacji nie zmieści się w pamięci docelowej
  - OF (overflow flag) - wystąpił integer overflow (signed)

Rejestr EFLAGS

Jest to 32-bitowy rejestr w którym poszczególne bity symbolizują flagi



obrazek sciągnięty 2: link

1 - flaga ustawiona, 0 - flaga wygaszona

Instrukcja test

- test a, b - wykonuje operacje and na 2 liczbach i ustawia flagę ZF jeśli wynikiem było 0, wynik operacji and nie jest nigdzie zapisany
- test a, a - ustawia ZF jeśli a==0

Instrukcja test

- test a, b - wykonuje operacje and na 2 liczbach i ustawia flagę ZF jeśli wynikiem było 0, wynik operacji and nie jest nigdzie zapisany
- test a, a - ustawia ZF jeśli a==0

Przykład:

- test 2, 2 ?
- test 0, 0 ?
- test 3, 4 ?
- test 1, 3 ?

Instrukcja test

- test a, b - wykonuje operacje and na 2 liczbach i ustawia flagę ZF jeśli wynikiem było 0, wynik operacji and nie jest nigdzie zapisany
- test a, a - ustawia ZF jeśli a==0

Przykład:

- test 2, 2 ZF<-0
- test 0, 0 ?
- test 3, 4 ?
- test 1, 3 ?

Instrukcja test

- test a, b - wykonuje operacje and na 2 liczbach i ustawia flagę ZF jeśli wynikiem było 0, wynik operacji and nie jest nigdzie zapisany
- test a, a - ustawia ZF jeśli a==0

Przykład:

- test 2, 2 ZF<-0
- test 0, 0 ZF<-1
- test 3, 4 ?
- test 1, 3 ?

Instrukcja test

- test a, b - wykonuje operacje and na 2 liczbach i ustawia flagę ZF jeśli wynikiem było 0, wynik operacji and nie jest nigdzie zapisany
- test a, a - ustawia ZF jeśli a==0

Przykład:

- test 2, 2 ZF<-0
- test 0, 0 ZF<-1
- test 3, 4 ZF<-1
- test 1, 3 ?

Instrukcja test

- test a, b - wykonuje operacje and na 2 liczbach i ustawia flagę ZF jeśli wynikiem było 0, wynik operacji and nie jest nigdzie zapisany
- test a, a - ustawia ZF jeśli a==0

Przykład:

- test 2, 2 ZF<-0
- test 0, 0 ZF<-1
- test 3, 4 ZF<-1
- test 1, 3 ZF<-0

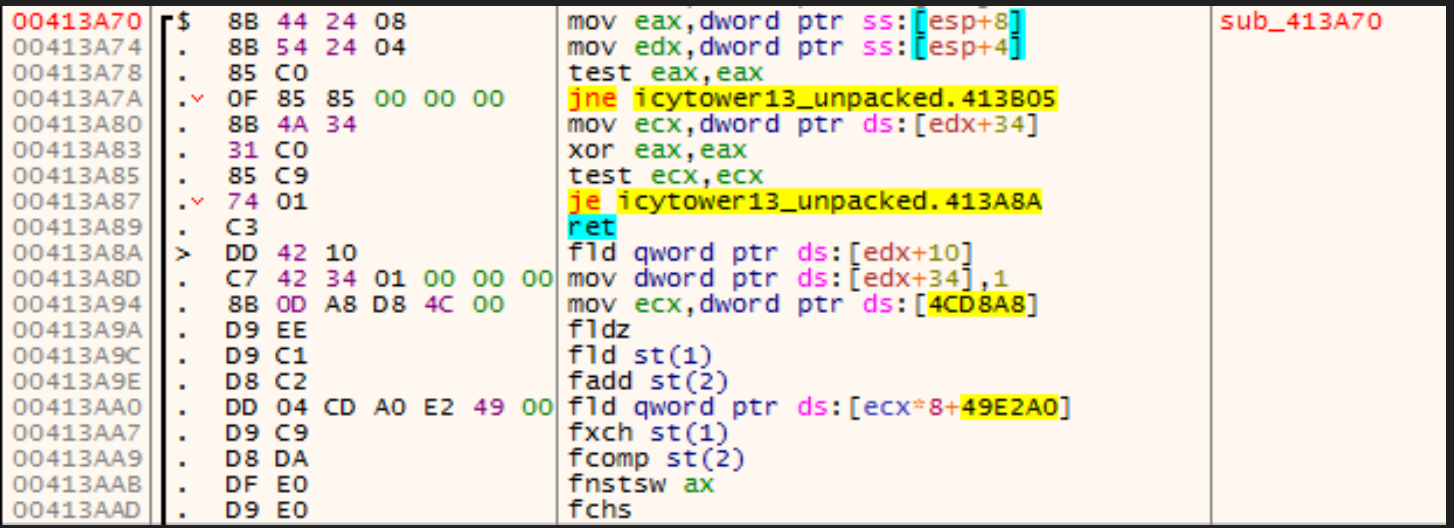
Skoki warunkowe w asemblerze

- je/jz [adres] - jump if zero flag is set
- jne [adres]- jump if not zero flag is set
- inne

Inne instrukcje asemblera x86

- mov a, b - a<-b
- [...] w mov oznacza dereferencje
- ret - wychodzimy z funkcji
- xor a, b - a<- a xor b
- fld, fldz, fadd, fcomp itp - to funkcje operujące na koprocesorze (ułamkach) - nie zamierzamy się w to bawić na konferencji

Początek funkcji odpowiedzialnej za skok (demo)



Reverse engineering to też odtwarzanie źródeł programu

Użyjmy dekompilatora!

x64dbg (snowman) - open source

```
struct s0 {
    int8_t[24] pad24;
    int32_t f24;
    int32_t f28;
    int8_t[20] pad52;
    int32_t f52;
    int8_t[24] pad80;
    int32_t f80;
    int32_t f84;
};

int32_t fun_413a70(struct s0* a1, int32_t a2) {
    uint1_t zf3;

    if (a2) {
        a1->f52 = 1;
        a1->f24 = 0;
        a1->f28 = 0xc0280000;
        return -1;
    } else {
        zf3 = (uint1_t)(a1->f52 == 0);
        if (zf3) {
            asm("fld qword [edx+0x10]");
            a1->f52 = 1;
            asm("fldz");
            asm("fld st1");
            asm("fadd st0, st2");
            asm("fld qword [ecx*8+0x49e2a0]");
            asm("fxch st0, st1");
            asm("fcomp st0, st2");
            asm("fnstsw ax");
            asm("fchs");
            asm("sahf");
            if (0) {
                asm("fld st2");
                asm("fadd st0, st3");
            } else {
                asm("fld dword [0x4ad820]");
                asm("fmul st0, st3");
            }
```

Użyjmy dekompilatora!

IDA Pro (Hex-Rays) - nie open source

potrafi dekompilować x86, x86-64, ARM, MIPS

```
signed int __cdecl sub_413A70(int a1, int a2)
{
  signed int result; // eax@2
  long double v3; // fst7@3
  long double v4; // fst4@4
  long double v5; // fst6@6

  if ( a2 )
  {
    *(_DWORD *)(a1 + 52) = 1;
    result = -1;
    *(_DWORD *)(a1 + 24) = 0;
    *(_DWORD *)(a1 + 28) = -1071120384;
  }
  else
  {
    result = 0;
    if ( !*(_DWORD *)(a1 + 52) )
    {
      v3 = *(double *)(a1 + 16);
      *(_DWORD *)(a1 + 52) = 1;
      if ( v3 + v3 < 0.0 )
        v4 = v3 + v3;
      else
        v4 = -2.0 * v3;
      if ( v4 >= -dbl_49E2A0[dword_4CD8A8] )
      {
        v5 = -dbl_49E2A0[dword_4CD8A8];
      }
      else
      {
        v5 = v3 + v3;
        if ( v5 >= 0.0 )
          v5 = -2.0 * v3;
      }
      *(double *)(a1 + 24) = v5;
      *(double *)(a1 + 32) = v3;
      if ( v5 < -22.0 )
        *(_DWORD *)(a1 + 80) = 1;
      *(_DWORD *)(a1 + 84) = 0;
      result = -1;
    }
  }
  return result;
}
```

Inne dekompilatory

- RetDec - open source - x86, ARM, MIPS, PowerPC, PIC32 (tylko 32 bitowe architektury)

Gry multiplayer

Dziękuję za uwagę

Referencje