Pengenalan PE dan Reverse Engineering PE 32 bit

oleh

Antonius

www.cr0security.com

www.indonesianbacktrack.or.id

www.codewall-security.com

ww.jasaplus.com

www.devilzc0de.org

Portable Executable

- Merupakan file executable di microsoft windows, bisa berekstensi .exe , .dll
- Dikenal mulai windows NT 3.1
- Konon Berasal dari COFF (VMS Executable)

Reverse Engineering ?

Secara Umum

- Proses untuk menganalisis teknologi untuk mengetahui bagaimana teknologi dirancang dan cara kerjanya

Reverse Engineering Software

- Proses menganalisis bagaimana software dirancang dan cara kerjanya

Beberapa Motif Reverse Engineering

- Pembajakan (cracking)
- Tujuan Komersial / Politik
- Modifikasi Software / Mendapat Contoh Source Code / Mendapatkan logika
- Pemeriksaan Keamanan Software

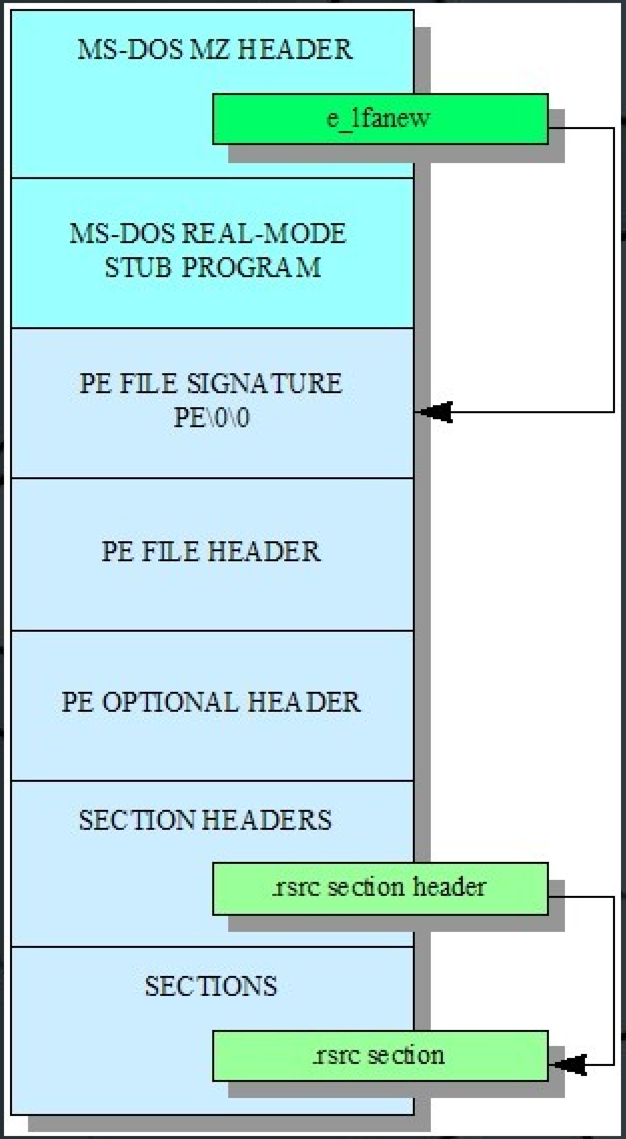
Reverse Engineering PE

- Objek

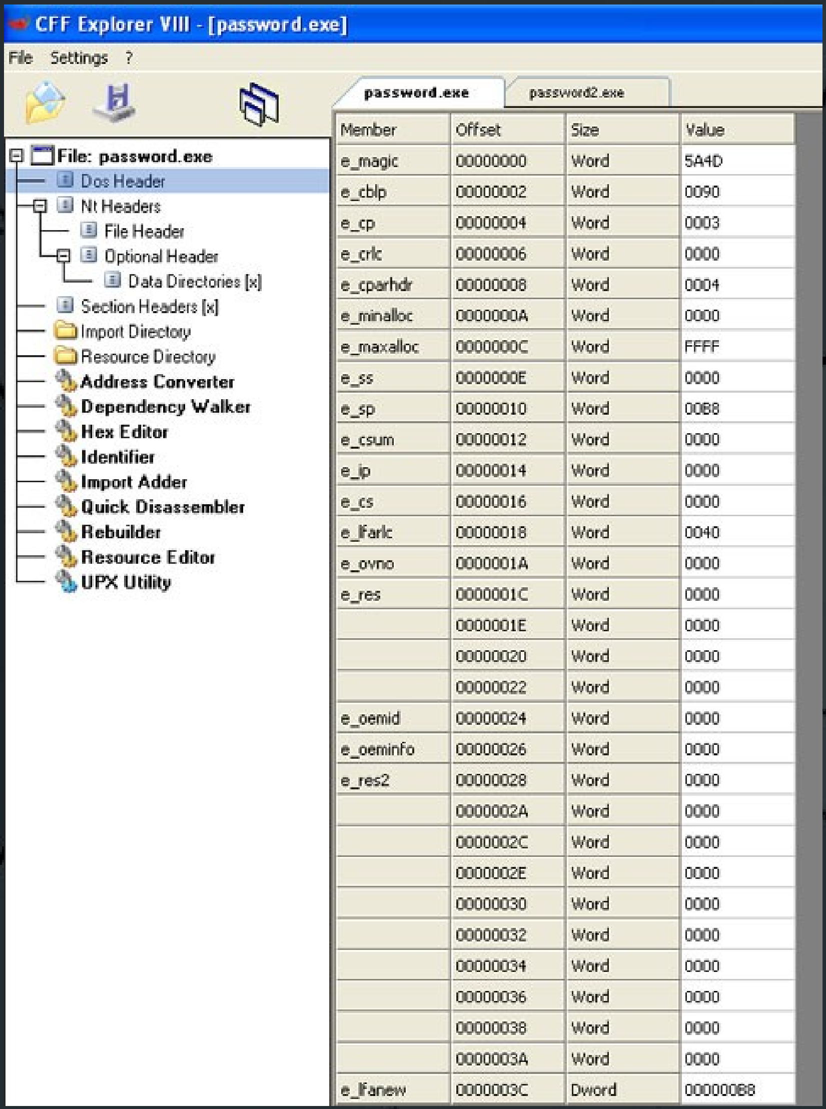
  PE (Portable Executable)
- Peralatan

  CFF Explorer, IDA Pro, hedit

Struktur PE


MS-DOS MZ HEADER
e_lfanew
MS-DOS REAL-MODE STUB PROGRAM
PE FILE SIGNATURE PE\0\0
PE FILE HEADER
PE OPTIONAL HEADER
SECTION HEADERS
.rsrc section header
SECTIONS
.rsrc section

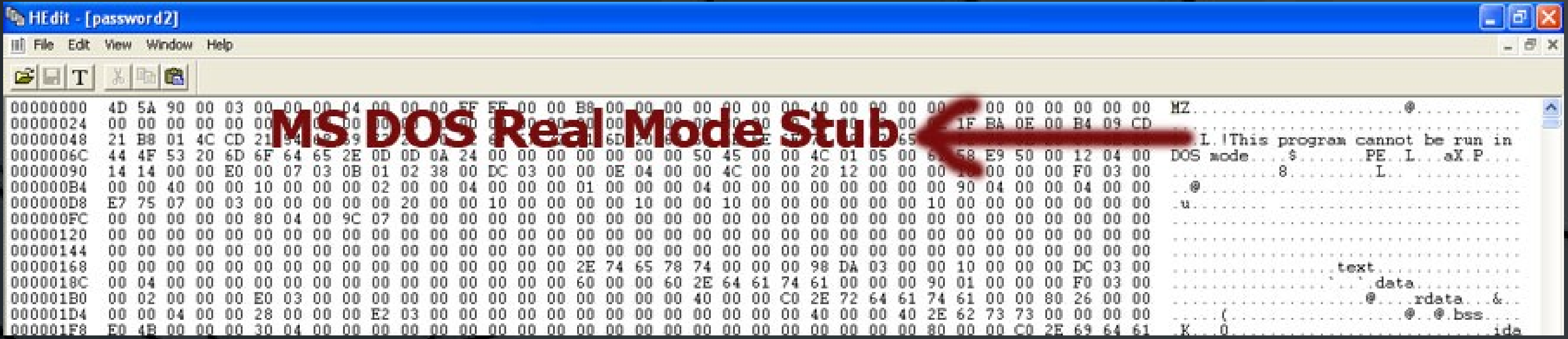
MS Dos Header



Member	Offset	Size	Value
e_magic	00000000	Word	5A4D
e_cblp	00000002	Word	0090
e_cp	00000004	Word	0003
e_crlc	00000006	Word	0000
e_cparhdr	00000008	Word	0004
e_minalloc	0000000A	Word	0000
e_maxalloc	0000000C	Word	FFFF
e_ss	0000000E	Word	0000
e_sp	00000010	Word	00B8
e_csum	00000012	Word	0000
e_ip	00000014	Word	0000
e_cs	00000016	Word	0000
e_lfarlc	00000018	Word	0040
e_ovno	0000001A	Word	0000
e_res	0000001C	Word	0000
	0000001E	Word	0000
	00000020	Word	0000
	00000022	Word	0000
e_oemid	00000024	Word	0000
e_oeminfo	00000026	Word	0000
e_res2	00000028	Word	0000
	0000002A	Word	0000
	0000002C	Word	0000
	0000002E	Word	0000
	00000030	Word	0000
	00000032	Word	0000
	00000034	Word	0000
	00000036	Word	0000
	00000038	Word	0000
	0000003A	Word	0000
e_lfanew	0000003C	Dword	000000B8

```
typedef struct _IMAGE_DOS_HEADER {  // DOS .EXE header
    USHORT e_magic;         // Magic number
    USHORT e_cblp;          // Bytes on last page of file
    USHORT e_cp;            // Pages in file
    USHORT e_crlc;          // Relocations
    USHORT e_cparhdr;       // Size of header in paragraphs
    USHORT e_minalloc;      // Minimum extra paragraphs needed
    USHORT e_maxalloc;      // Maximum extra paragraphs needed
    USHORT e_ss;            // Initial (relative) SS value
    USHORT e_sp;            // Initial SP value
    USHORT e_csum;          // Checksum
    USHORT e_ip;            // Initial IP value
    USHORT e_cs;            // Initial (relative) CS value
    USHORT e_lfarlc;        // File address of relocation table
    USHORT e_ovno;          // Overlay number
    USHORT e_res[4];        // Reserved words
    USHORT e_oemid;         // OEM identifier (for e_oeminfo)
    USHORT e_oeminfo;       // OEM information; e_oemid specific
    USHORT e_res2[10];      // Reserved words
    LONG   e_lfanew;        // File address of new exe header
  } IMAGE_DOS_HEADER, *PIMAGE_DOS_HEADER;
```

MS Dos Stub

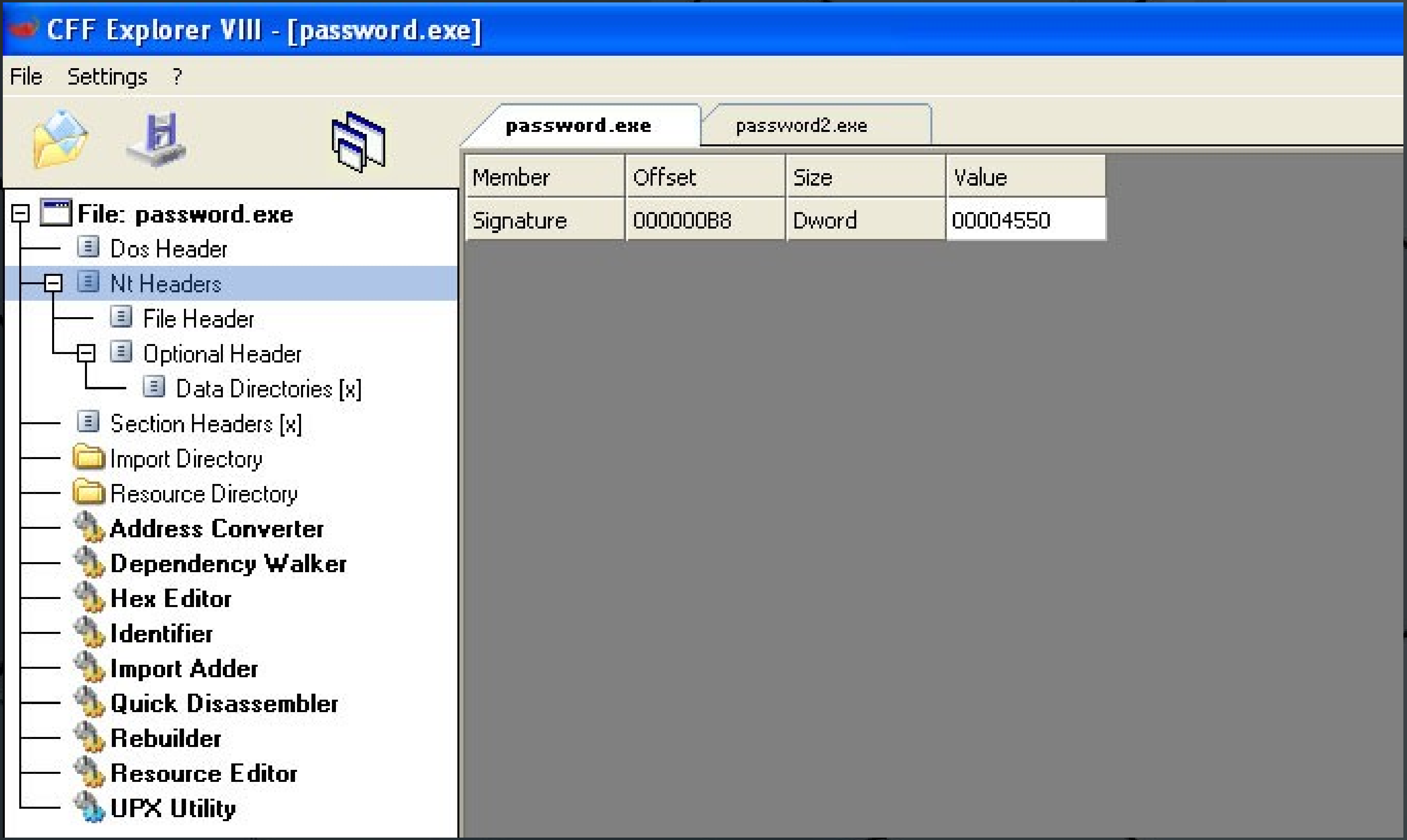


MS DOS Real Mode Stub

```
00000000  4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 B8 00 00 00 00 00 00 00 40 00 00 00 00 [illegible] 00 00 00 00 00 00  MZ......................@...........
00000024  00 00 00 00 00 00 [illegible] 00 [illegible] 00 [illegible] 1F BA 0E 00 B4 09 CD  ....................................
00000048  21 B8 01 4C CD 21 [illegible] 6D 20 [illegible] 65 [illegible]  [illegible]L.!This program cannot be run in
0000006C  44 4F 53 20 6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00 50 45 00 00 4C 01 05 00 61 58 E9 50 00 12 04 00  DOS mode....$.......PE..L...aX.P....
00000090  14 14 00 00 E0 00 07 03 0B 01 02 38 00 DC 03 00 00 0E 04 00 00 4C 00 00 20 12 00 00 00 [illegible] 00 00 00 F0 03 00  ............8.........L.............
000000B4  00 00 40 00 00 10 00 00 00 02 00 00 04 00 00 00 01 00 00 00 04 00 00 00 00 00 00 00 00 90 04 00 00 04 00 00  ..@.................................
000000D8  E7 75 07 00 03 00 00 00 00 00 20 00 00 10 00 00 00 00 10 00 00 10 00 00 00 00 00 00 10 00 00 00 00 00 00 00  .u..................................
000000FC  00 00 00 00 00 80 04 00 9C 07 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ....................................
00000120  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ....................................
00000144  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ....................................
00000168  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 2E 74 65 78 74 00 00 00 98 DA 03 00 00 10 00 00 00 DC 03 00  .................text...............
0000018C  00 04 00 00 00 00 00 00 00 00 00 00 00 00 00 00 60 00 00 60 2E 64 61 74 61 00 00 00 90 01 00 00 00 F0 03 00  ................`..`.data...........
000001B0  00 02 00 00 00 E0 03 00 00 00 00 00 00 00 00 00 00 00 00 00 40 00 00 C0 2E 72 64 61 74 61 00 00 80 26 00 00  ....................@....rdata...&..
000001D4  00 00 04 00 00 28 00 00 00 E2 03 00 00 00 00 00 00 00 00 00 00 00 00 00 40 00 00 40 2E 62 73 73 00 00 00 00  .....(..................@..@.bss....
000001F8  E0 4B 00 00 00 30 04 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 80 00 00 C0 2E 69 64 61  .K...0...........................ida
```

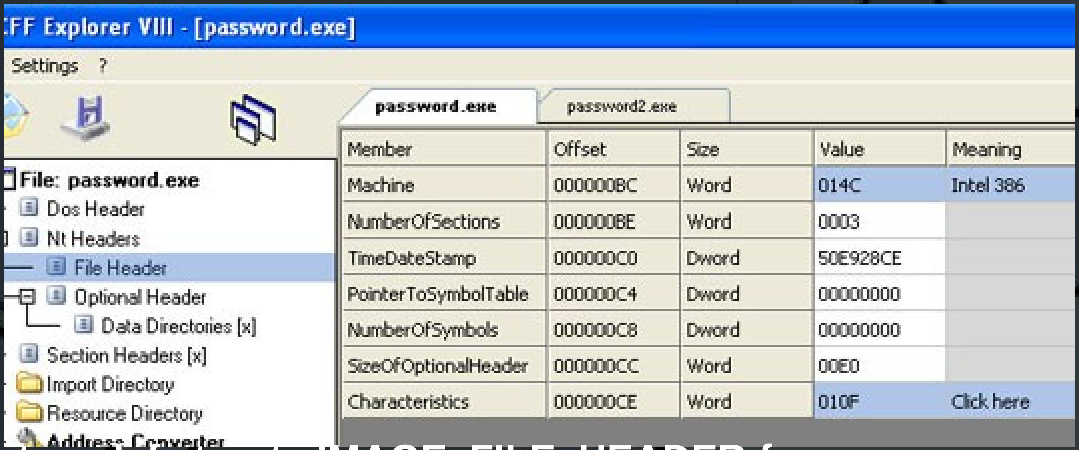
Dijalankan MS Dos saat program pertama kali diload !

PE File Signature



Member	Offset	Size	Value
Signature	000000B8	Dword	00004550

PE File Header



Member	Offset	Size	Value	Meaning
Machine	000000BC	Word	014C	Intel 386
NumberOfSections	000000BE	Word	0003	
TimeDateStamp	000000C0	Dword	50E928CE	
PointerToSymbolTable	000000C4	Dword	00000000	
NumberOfSymbols	000000C8	Dword	00000000	
SizeOfOptionalHeader	000000CC	Word	00E0	
Characteristics	000000CE	Word	010F	Click here

```
typedef struct _IMAGE_FILE_HEADER {
    USHORT  Machine;
    USHORT  NumberOfSections;
    ULONG   TimeDateStamp;
    ULONG   PointerToSymbolTable;
    ULONG   NumberOfSymbols;
    USHORT  SizeOfOptionalHeader;
    USHORT  Characteristics;
} IMAGE_FILE_HEADER, *PIMAGE_FILE_HEADER;

#define IMAGE_SIZEOF_FILE_HEADER            20
```

Informasi pada pe header berguna sistem memperlakukan file ini

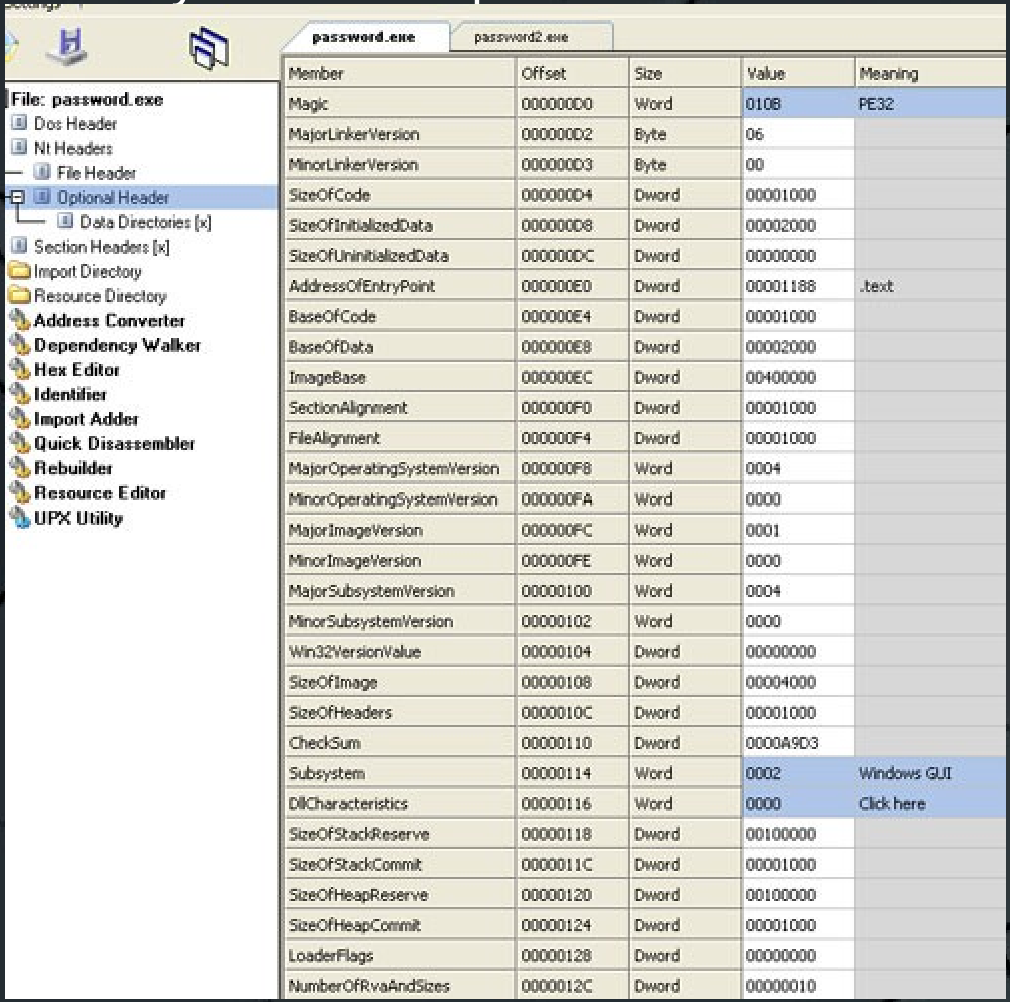
PE File Header

- Bagaimana Offsetnya ditentukan ?

```
#define NTSIGNATURE(a) ((LPVOID)((BYTE *)a +    \
   ((PIMAGE_DOS_HEADER)a)->e_lfanew))
```

PE Optional Header

- 224 bytes setelah pe header



Member	Offset	Size	Value	Meaning
Magic	000000D0	Word	010B	PE32
MajorLinkerVersion	000000D2	Byte	06	
MinorLinkerVersion	000000D3	Byte	00	
SizeOfCode	000000D4	Dword	00001000	
SizeOfInitializedData	000000D8	Dword	00002000	
SizeOfUninitializedData	000000DC	Dword	00000000	
AddressOfEntryPoint	000000E0	Dword	00001188	.text
BaseOfCode	000000E4	Dword	00001000	
BaseOfData	000000E8	Dword	00002000	
ImageBase	000000EC	Dword	00400000	
SectionAlignment	000000F0	Dword	00001000	
FileAlignment	000000F4	Dword	00001000	
MajorOperatingSystemVersion	000000F8	Word	0004	
MinorOperatingSystemVersion	000000FA	Word	0000	
MajorImageVersion	000000FC	Word	0001	
MinorImageVersion	000000FE	Word	0000	
MajorSubsystemVersion	00000100	Word	0004	
MinorSubsystemVersion	00000102	Word	0000	
Win32VersionValue	00000104	Dword	00000000	
SizeOfImage	00000108	Dword	00004000	
SizeOfHeaders	0000010C	Dword	00001000	
CheckSum	00000110	Dword	0000A9D3	
Subsystem	00000114	Word	0002	Windows GUI
DllCharacteristics	00000116	Word	0000	Click here
SizeOfStackReserve	00000118	Dword	00100000	
SizeOfStackCommit	0000011C	Dword	00001000	
SizeOfHeapReserve	00000120	Dword	00100000	
SizeOfHeapCommit	00000124	Dword	00001000	
LoaderFlags	00000128	Dword	00000000	
NumberOfRvaAndSizes	0000012C	Dword	00000010	

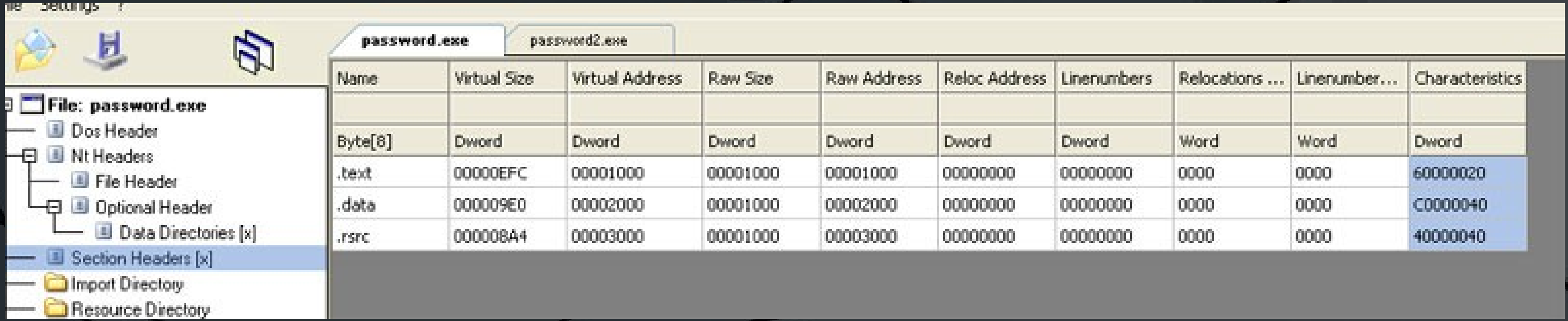
PE Optional Header

- 010b menandakan sebagai PE untuk 32 bit

- Offset ditentukan dengan makro:

```
#define OPTHDROFFSET(a) ((LPVOID)((BYTE *)a                + \
    ((PIMAGE_DOS_HEADER)a)->e_lfanew + SIZE_OF_NT_SIGNATURE + \
    sizeof (IMAGE_FILE_HEADER)))
```

Section Headers



Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations ...	Linenumber...	Characteristics
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word	Word	Dword
.text	00000EFC	00001000	00001000	00001000	00000000	00000000	0000	0000	60000020
.data	000009E0	00002000	00001000	00002000	00000000	00000000	0000	0000	C0000040
.rsrc	000008A4	00003000	00001000	00003000	00000000	00000000	0000	0000	40000040

```
typedef struct _IMAGE_SECTION_HEADER {
    UCHAR   Name[IMAGE_SIZEOF_SHORT_NAME];
    union {
            ULONG   PhysicalAddress;
            ULONG   VirtualSize;
    } Misc;
    ULONG   VirtualAddress;
    ULONG   SizeOfRawData;
    ULONG   PointerToRawData;
    ULONG   PointerToRelocations;
    ULONG   PointerToLinenumbers;
    USHORT  NumberOfRelocations;
    USHORT  NumberOfLinenumbers;
    ULONG   Characteristics;
} IMAGE_SECTION_HEADER,
*PIMAGE_SECTION_HEADER;
```

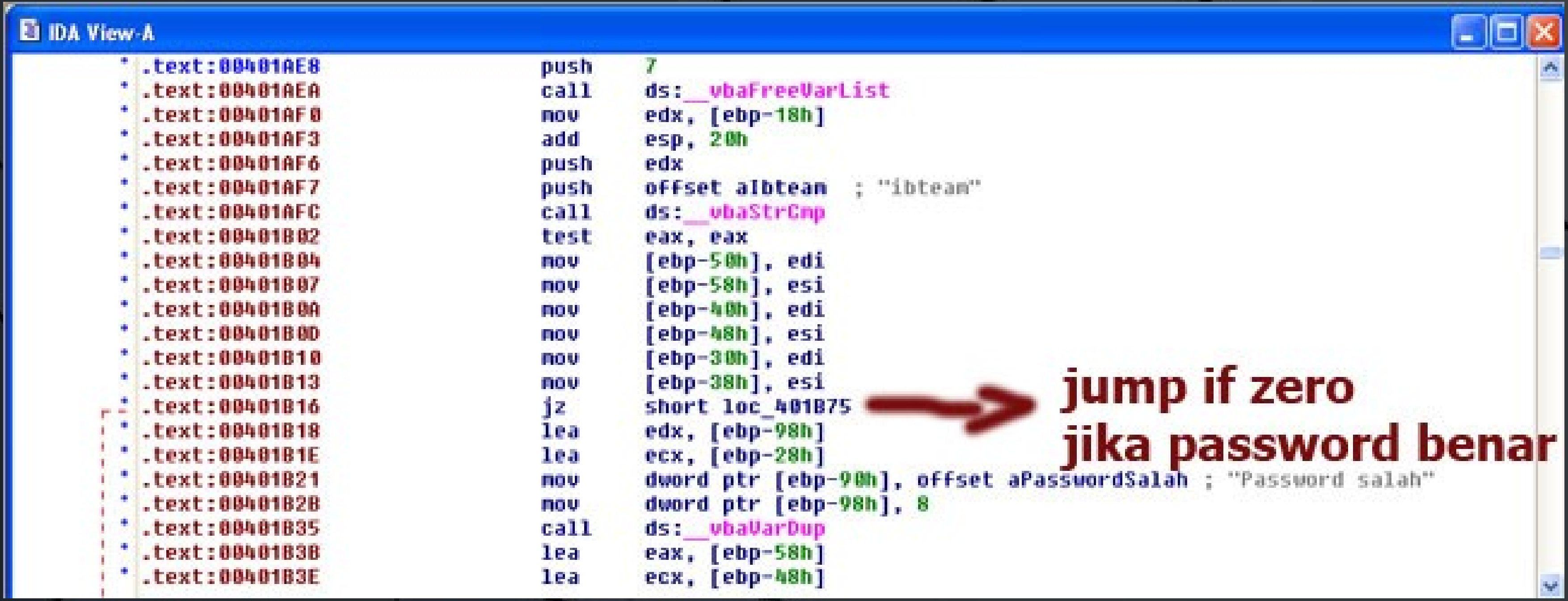
Contoh Reverse Engineering PE (1)

- Contoh PE : password.exe
- Compiler : MS VB 6.0

X86 Assembly Instruction: test

- Most Significant Bit dari logika and
- Jika hasil 0 maka carry flag 1 (kondisi tidak terpenuhi)
- Jika hasil 1 maka carry flag 0 (kondisi terpenuhi)

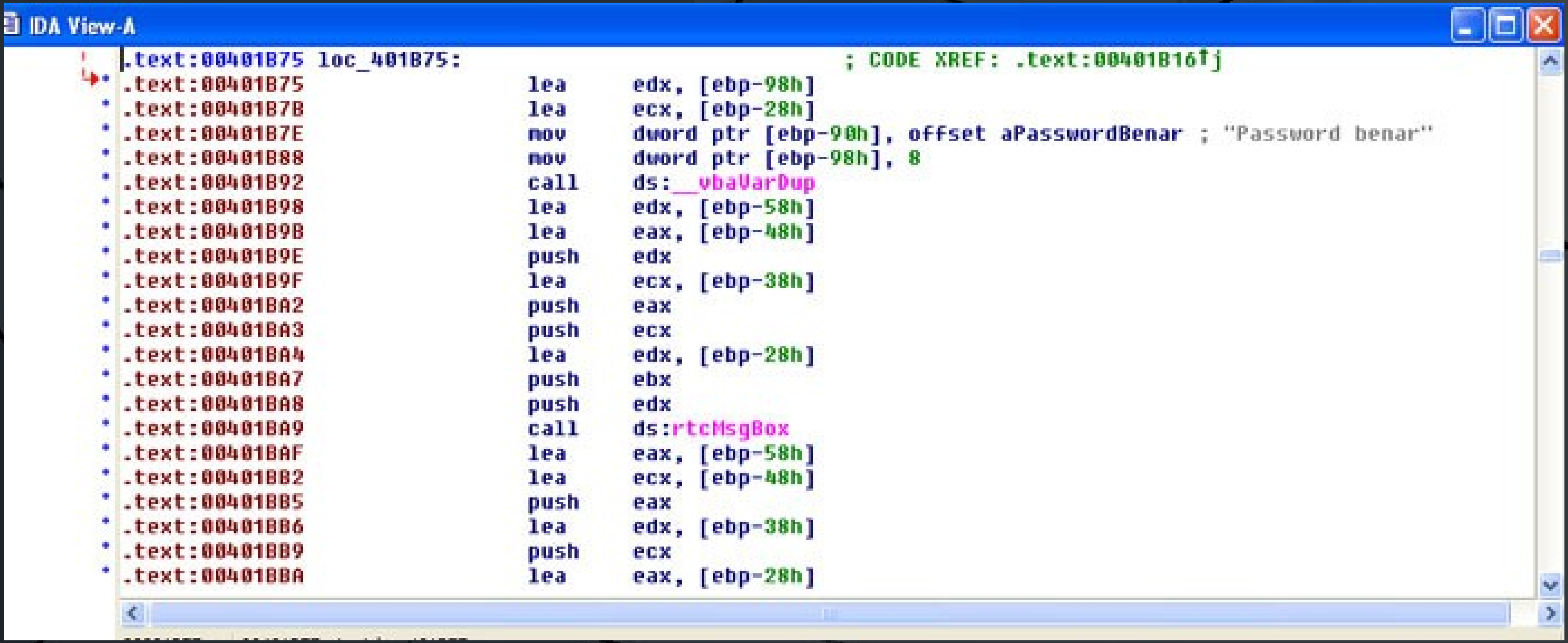
Contoh Reverse Engineering PE (1)



```
.text:00401AE8                 push    7
.text:00401AEA                 call    ds:__vbaFreeVarList
.text:00401AF0                 mov     edx, [ebp-18h]
.text:00401AF3                 add     esp, 20h
.text:00401AF6                 push    edx
.text:00401AF7                 push    offset aIbteam  ; "ibteam"
.text:00401AFC                 call    ds:__vbaStrCmp
.text:00401B02                 test    eax, eax
.text:00401B04                 mov     [ebp-50h], edi
.text:00401B07                 mov     [ebp-58h], esi
.text:00401B0A                 mov     [ebp-40h], edi
.text:00401B0D                 mov     [ebp-48h], esi
.text:00401B10                 mov     [ebp-30h], edi
.text:00401B13                 mov     [ebp-38h], esi
.text:00401B16                 jz      short loc_401B75
.text:00401B18                 lea     edx, [ebp-98h]
.text:00401B1E                 lea     ecx, [ebp-28h]
.text:00401B21                 mov     dword ptr [ebp-90h], offset aPasswordSalah ; "Password salah"
.text:00401B2B                 mov     dword ptr [ebp-98h], 8
.text:00401B35                 call    ds:__vbaVarDup
.text:00401B3B                 lea     eax, [ebp-58h]
.text:00401B3E                 lea     ecx, [ebp-48h]
```

jump if zero
jika password benar

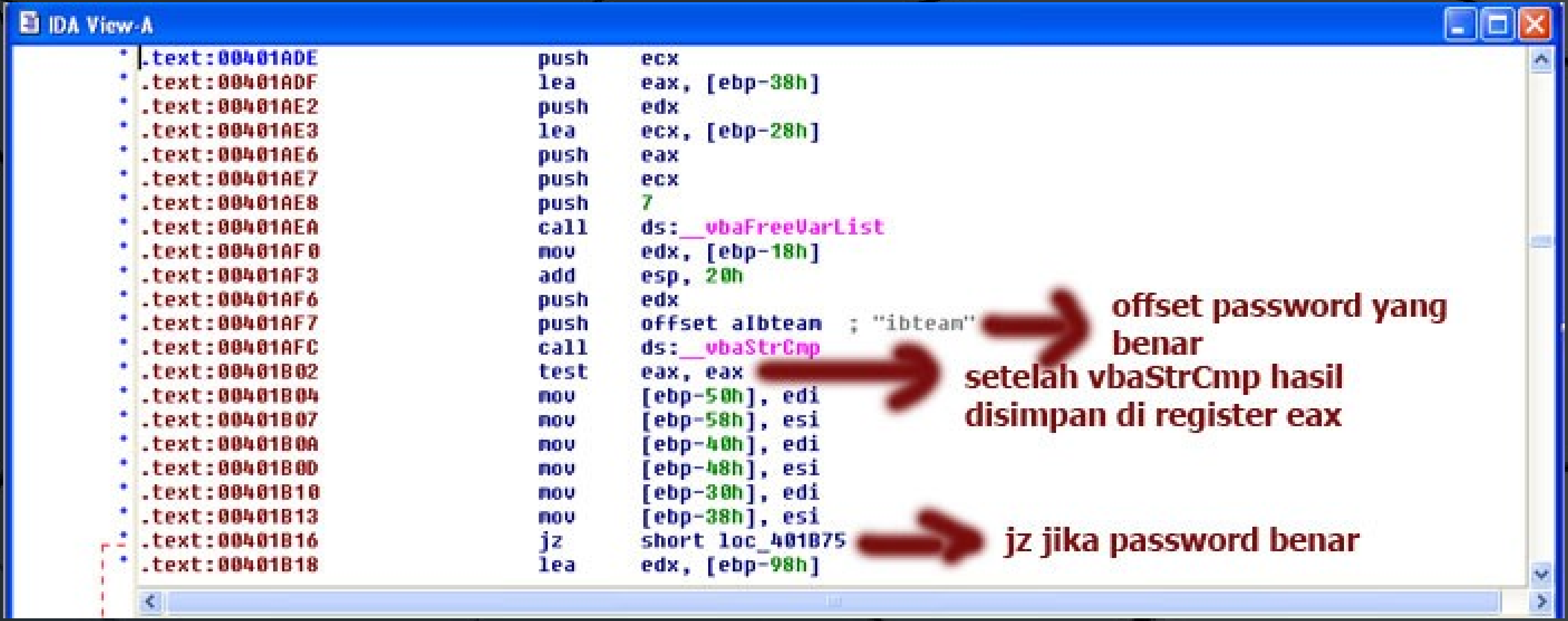
Contoh Reverse Engineering PE (1)



```
.text:00401B75 loc_401B75:                                  ; CODE XREF: .text:00401B16↑j
.text:00401B75                 lea     edx, [ebp-98h]
.text:00401B7B                 lea     ecx, [ebp-28h]
.text:00401B7E                 mov     dword ptr [ebp-90h], offset aPasswordBenar ; "Password benar"
.text:00401B88                 mov     dword ptr [ebp-98h], 8
.text:00401B92                 call    ds:__vbaVarDup
.text:00401B98                 lea     edx, [ebp-58h]
.text:00401B9B                 lea     eax, [ebp-48h]
.text:00401B9E                 push    edx
.text:00401B9F                 lea     ecx, [ebp-38h]
.text:00401BA2                 push    eax
.text:00401BA3                 push    ecx
.text:00401BA4                 lea     edx, [ebp-28h]
.text:00401BA7                 push    ebx
.text:00401BA8                 push    edx
.text:00401BA9                 call    ds:rtcMsgBox
.text:00401BAF                 lea     eax, [ebp-58h]
.text:00401BB2                 lea     ecx, [ebp-48h]
.text:00401BB5                 push    eax
.text:00401BB6                 lea     edx, [ebp-38h]
.text:00401BB9                 push    ecx
.text:00401BBA                 lea     eax, [ebp-28h]
```

loc_401B75 merupakan rutin prosedur yang dieksekusi jika inputan password benar

Contoh Reverse Engineering PE (1)



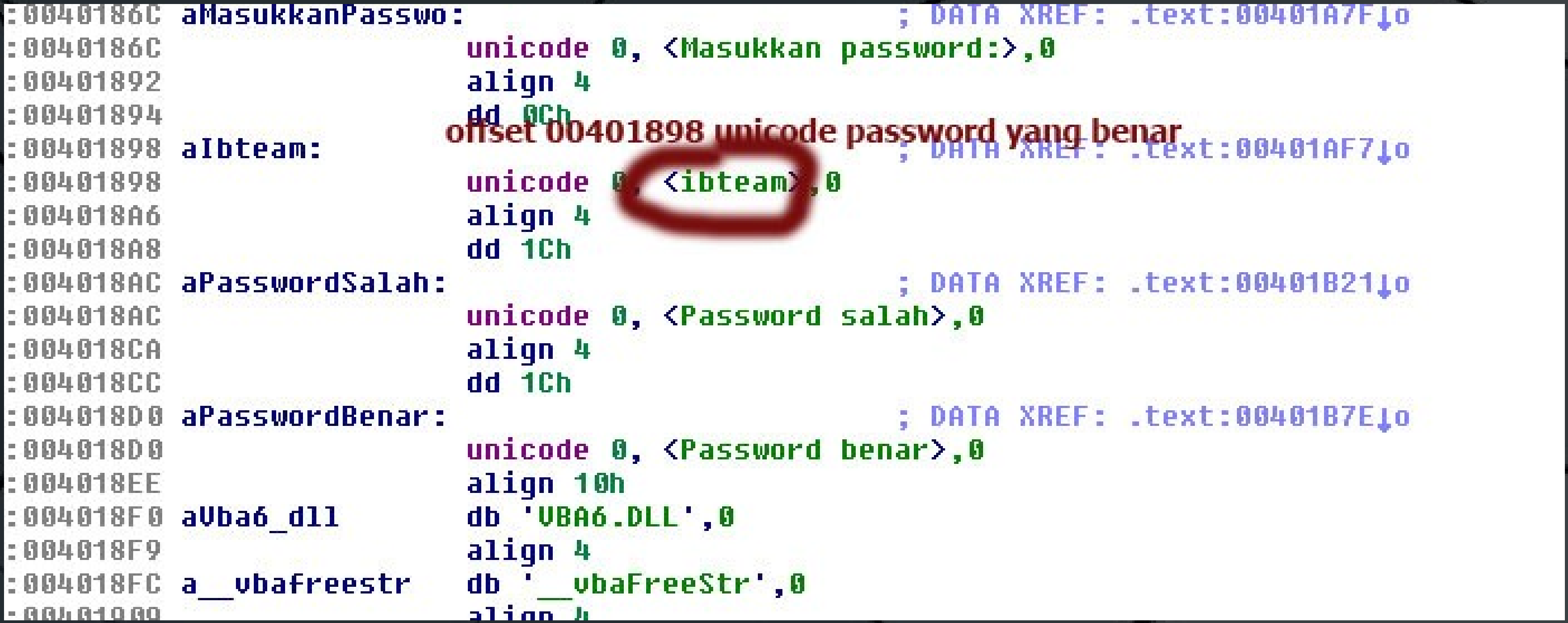
```
.text:00401ADE                push    ecx
.text:00401ADF                lea     eax, [ebp-38h]
.text:00401AE2                push    edx
.text:00401AE3                lea     ecx, [ebp-28h]
.text:00401AE6                push    eax
.text:00401AE7                push    ecx
.text:00401AE8                push    7
.text:00401AEA                call    ds:__vbaFreeVarList
.text:00401AF0                mov     edx, [ebp-18h]
.text:00401AF3                add     esp, 20h
.text:00401AF6                push    edx
.text:00401AF7                push    offset aIbteam  ; "ibteam"
.text:00401AFC                call    ds:__vbaStrCmp
.text:00401B02                test    eax, eax
.text:00401B04                mov     [ebp-50h], edi
.text:00401B07                mov     [ebp-58h], esi
.text:00401B0A                mov     [ebp-40h], edi
.text:00401B0D                mov     [ebp-48h], esi
.text:00401B10                mov     [ebp-30h], edi
.text:00401B13                mov     [ebp-38h], esi
.text:00401B16                jz      short loc_401B75
.text:00401B18                lea     edx, [ebp-98h]
```

offset password yang benar

setelah vbaStrCmp hasil disimpan di register eax

jz jika password benar

Contoh Reverse Engineering PE (1)



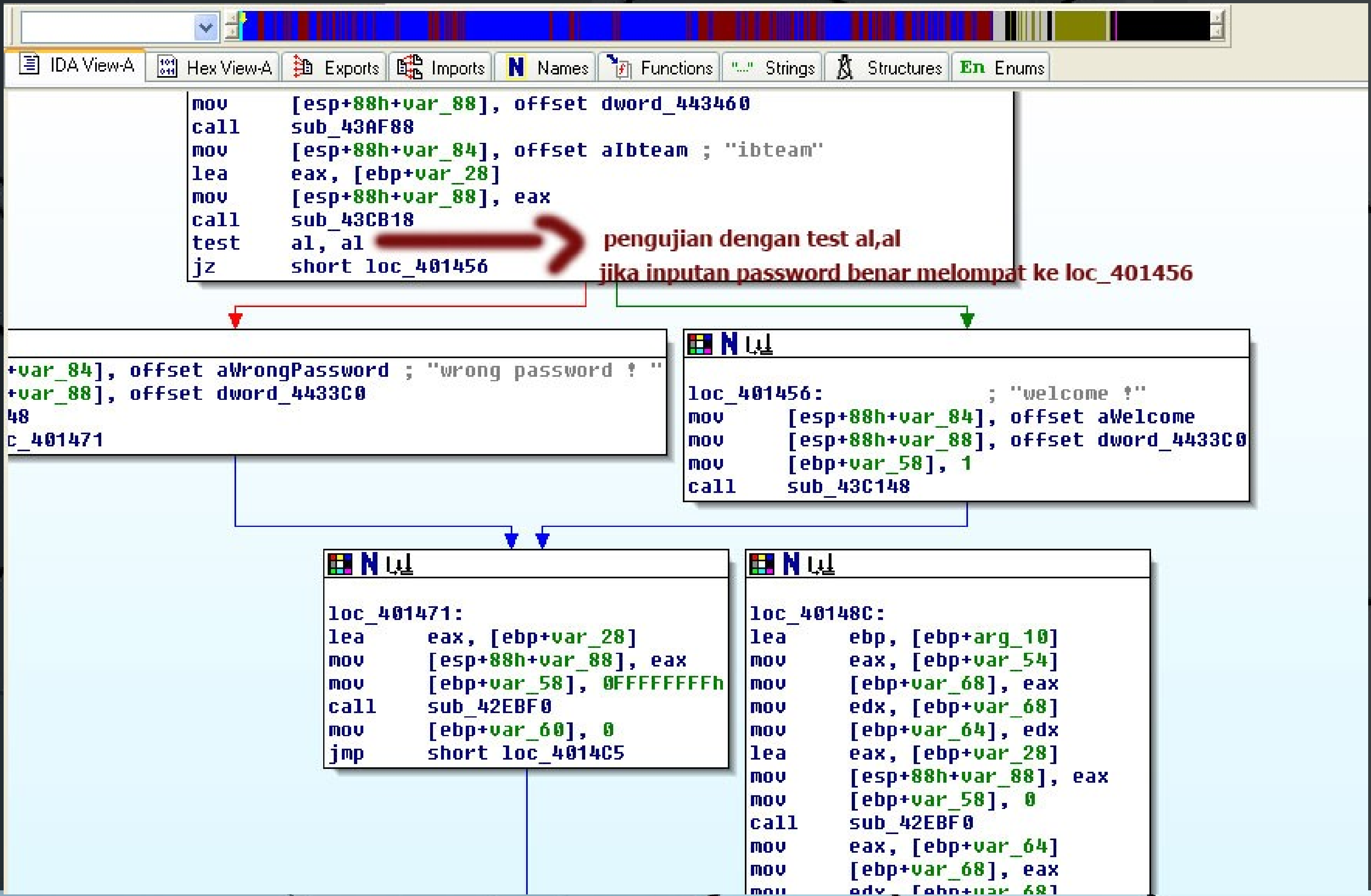
```
:0040186C aMasukkanPasswo:                     ; DATA XREF: .text:00401A7F↓o
:0040186C                unicode 0, <Masukkan password:>,0
:00401892                align 4
:00401894                dd 0Ch
:00401898 aIbteam:                             ; DATA XREF: .text:00401AF7↓o
:00401898                unicode 0, <ibteam>,0
:004018A6                align 4
:004018A8                dd 1Ch
:004018AC aPasswordSalah:                      ; DATA XREF: .text:00401B21↓o
:004018AC                unicode 0, <Password salah>,0
:004018CA                align 4
:004018CC                dd 1Ch
:004018D0 aPasswordBenar:                      ; DATA XREF: .text:00401B7E↓o
:004018D0                unicode 0, <Password benar>,0
:004018EE                align 10h
:004018F0 aVba6_dll      db 'VBA6.DLL',0
:004018F9                align 4
:004018FC a__vbafreestr  db '__vbaFreeStr',0
:00401900                align 4
```

offset 00401898 unicode password yang benar

Contoh Reverse Engineering 2

- Contoh PE : password2.exe
- Compiler : Bloodshed Dev C++

Contoh Reverse Engineering 2



```
mov     [esp+88h+var_88], offset dword_443460
call    sub_43AF88
mov     [esp+88h+var_84], offset aIbteam ; "ibteam"
lea     eax, [ebp+var_28]
mov     [esp+88h+var_88], eax
call    sub_43CB18
test    al, al
jz      short loc_401456
```

pengujian dengan test al,al

jika inputan password benar melompat ke loc_401456

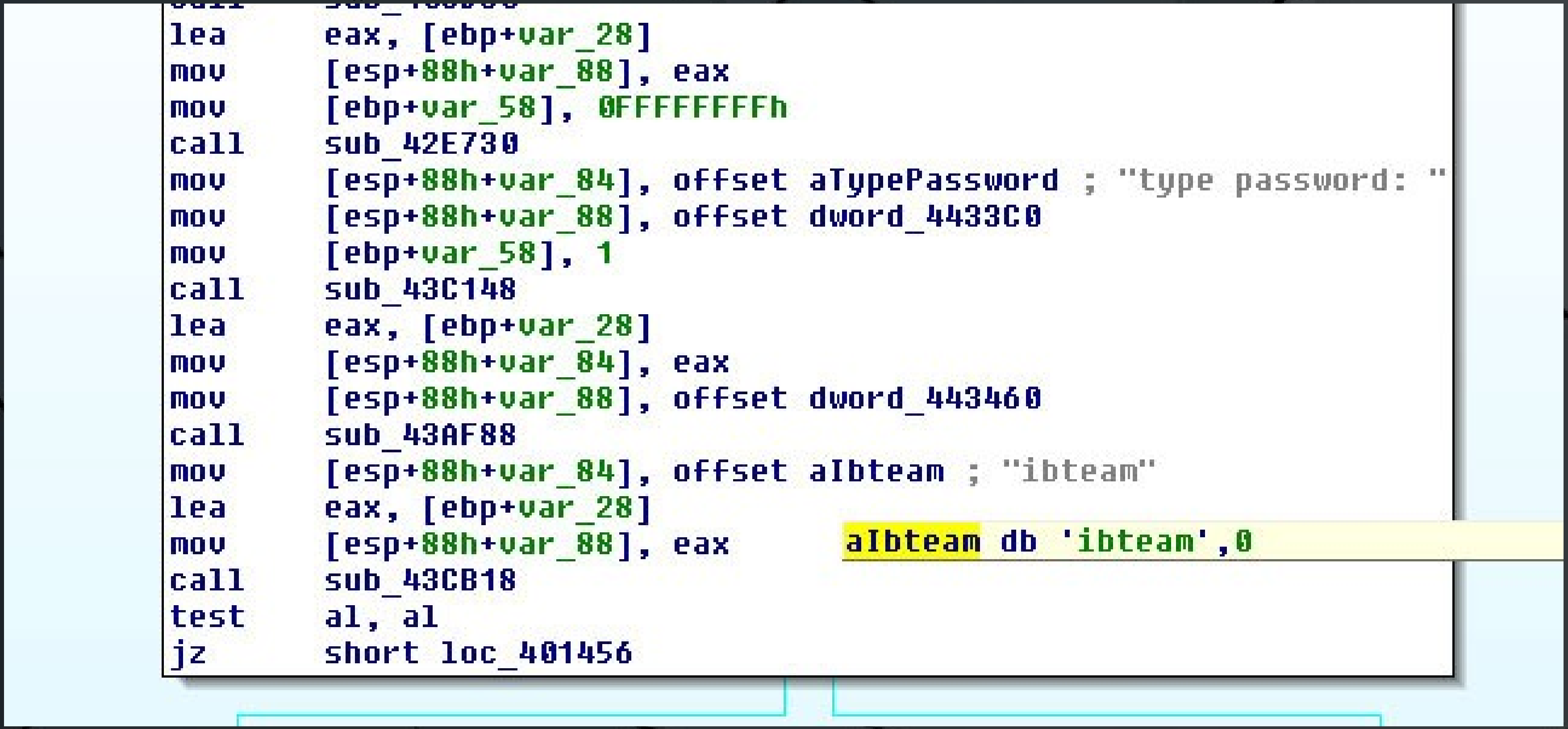
```
+var_84], offset aWrongPassword ; "wrong password ! "
+var_88], offset dword_4433C0
48
c_401471
```

```
loc_401456:             ; "welcome !"
mov     [esp+88h+var_84], offset aWelcome
mov     [esp+88h+var_88], offset dword_4433C0
mov     [ebp+var_58], 1
call    sub_43C148
```

```
loc_401471:
lea     eax, [ebp+var_28]
mov     [esp+88h+var_88], eax
mov     [ebp+var_58], 0FFFFFFFFh
call    sub_42EBF0
mov     [ebp+var_60], 0
jmp     short loc_4014C5
```

```
loc_40148C:
lea     ebp, [ebp+arg_10]
mov     eax, [ebp+var_54]
mov     [ebp+var_68], eax
mov     edx, [ebp+var_68]
mov     [ebp+var_64], edx
lea     eax, [ebp+var_28]
mov     [esp+88h+var_88], eax
mov     [ebp+var_58], 0
call    sub_42EBF0
mov     eax, [ebp+var_64]
mov     [ebp+var_68], eax
mov     edx, [ebp+var_68]
```

Contoh Reverse Engineering 2



```
lea     eax, [ebp+var_28]
mov     [esp+88h+var_88], eax
mov     [ebp+var_58], 0FFFFFFFFh
call    sub_42E730
mov     [esp+88h+var_84], offset aTypePassword ; "type password: "
mov     [esp+88h+var_88], offset dword_4433C0
mov     [ebp+var_58], 1
call    sub_43C148
lea     eax, [ebp+var_28]
mov     [esp+88h+var_84], eax
mov     [esp+88h+var_88], offset dword_443460
call    sub_43AF88
mov     [esp+88h+var_84], offset aIbteam ; "ibteam"
lea     eax, [ebp+var_28]
mov     [esp+88h+var_88], eax       aIbteam db 'ibteam',0
call    sub_43CB18
test    al, al
jz      short loc_401456
```

- pada offset esp + 88h – 84h disimpan offset password
- albteam sebelumnya dideklarasikan dengan define byte : “ibteam”
- hasil call selanjutnya disimpan untuk dilakukan instruksi test
- jump if zero ke loc_401456 (password benar)

Referensi

- **Http://en.wikipedia.org/wiki/Portable_Executable**
- **Http://en.wikipedia.org/wiki/Reverse_engineering**