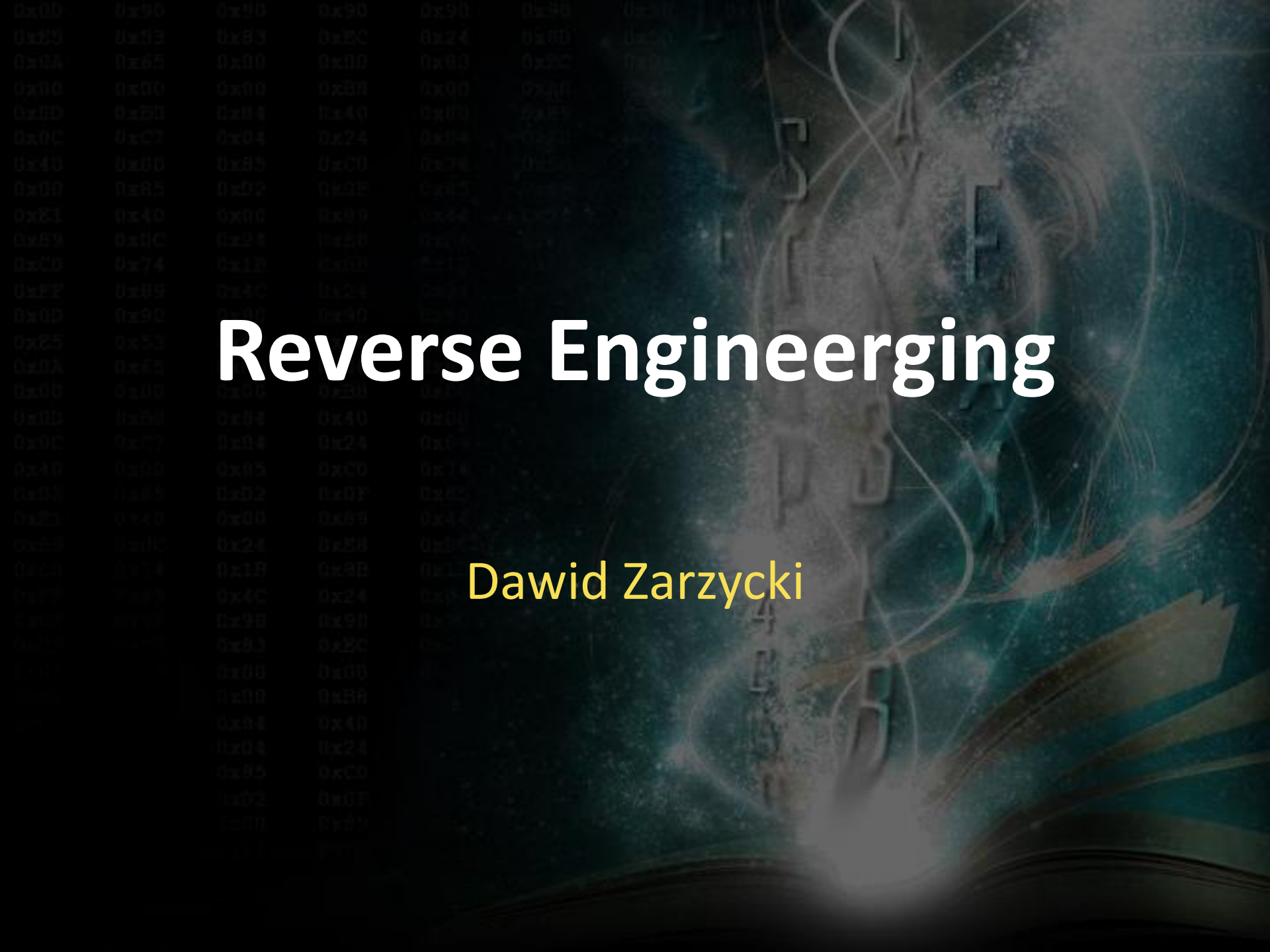

Reverse Engineerging
Dawid Zarzycki

Plan prezentacji

- Definicja Reverse Engineering
- Zastosowanie RE
- Pojęcia
- Podstawy budowy pliku Portable Executable
- Pamięć, proces i PE
- Język programowania Asembler
- Etapy analizy aplikacji
- Zadanie 1 – crack, keygen
- Zadanie 2 – modyfikacja aplikacji

Zanim zaczniemy 0x01...

- Narzędzia:
  - `http://fizyka.umk.pl/~236436/re.rar`

RE

Reverse Engineering (inżynieria wsteczna) – to analiza gotowego oprogramowania lub urządzenia i próba odpowiedzenia na pytanie jak przebiegał proces jego tworzenia.

RCE – Reverse Code Engineering

Zastosowanie

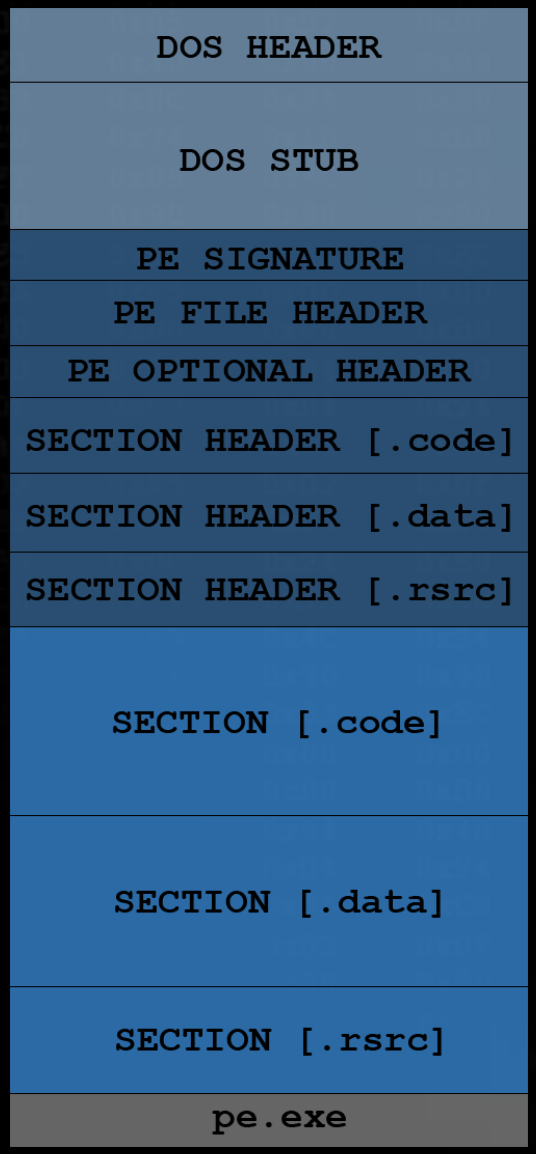
- Modyfikacja aplikacji:
  - Zmiana wersji językowej.
  - Odblokowanie dodatkowych funkcji.
  - Zmodyfikowanie, usunięcie pewnych funkcji.
  - Zmiana API, platformy (Syndicate Wars)
- Analiza złośliwych aplikacji (malware etc.)
- Cracking, cheating at games, fixing vulns
- Vulnerability discover, analysis
- Analiza i modyfikacja sprzętu (Play Station 3)
- Patch analysis: Black Tuesday / Exploit Wednesday

Pojęcia

- **Relative Virtual Address** (RVA) – adres pamięci, relatywny, od początku segmentu (np. ImageBase).
- **Virtual Address** (VA) – bezwzględny adres pamięci.
- **Offset** – przesunięcie względem danego segmentu.
- **Raw Address** (RA) – offset w pliku.

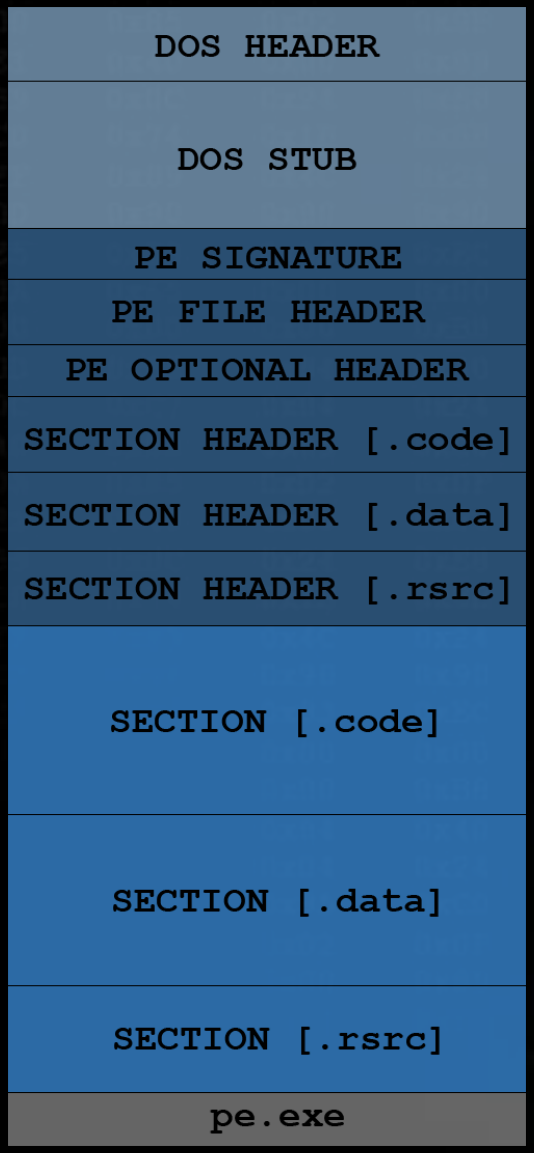
```
VA(RVA) = ImageBase + RVA;
RVA(VA) = VA - ImageBase;
RA(RVA) = Section[RVA].PointerToRawData + (RVA -
   Section[RVA].VirtualAddress);
```

Budowa PE



Specyfikacja Microsoftu:
http://www.microsoft.com/whdc/system/platform/firmware/PECOFF.mspx

Budowa PE

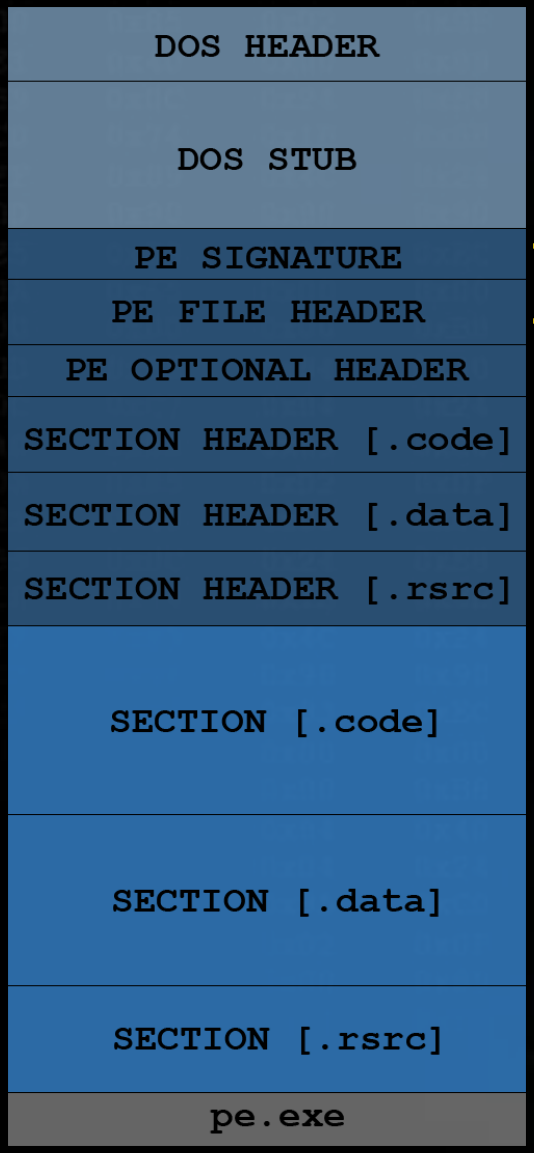


Kompatybilność wsteczna dla MS-DOS.

Struktura: `IMAGE_DOS_HEADER`

`e_magic`: MZ (Mark Zbikowski)
`e_lfanew`: adres nowego nagłówka PE

Budowa PE



Signature: `0x50 0x45 0x00 0x00`

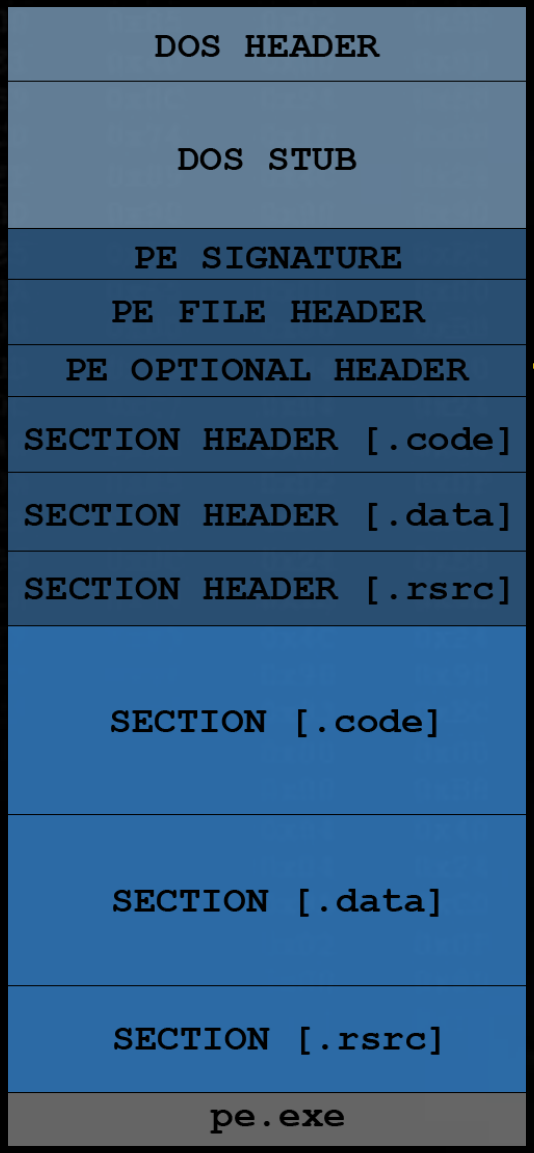
Struktura: `IMAGE_FILE_HEADER`

```
WORD    Machine;
```
IMAGE_FILE_MACHINE_I386
IMAGE_FILE_MACHINE_AMD64

```
WORD    NumberOfSections;

WORD    Characteristics;
```
IMAGE_FILE_EXECUTABLE_IMAGE
IMAGE_FILE_DLL

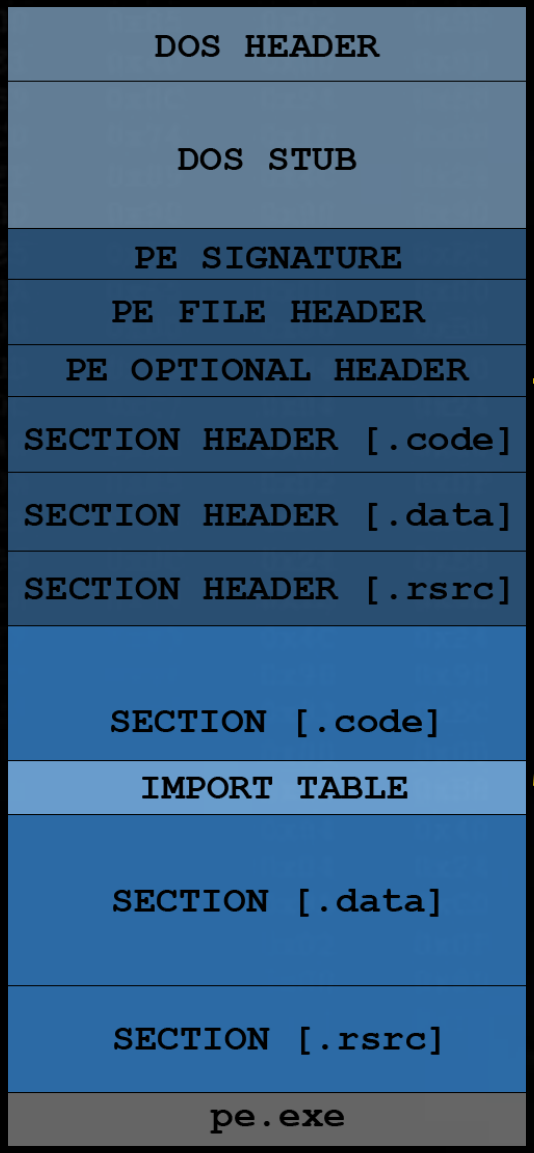
Budowa PE



Struktura: IMAGE_OPTIONAL_HEADER

```
DWORD  AddressOfEntryPoint;
DWORD  ImageBase;
```
DLL: 0x10000000 EXE: 0x00400000
```
DWORD  SectionAlignment;
```
0x1000 = 4096 = 1 strona
```
DWORD  FileAlignment;
```
0x200 = 512
```
DWORD  SizeOfImage;
WORD   Subsystem;
```
IMAGE_SUBSYSTEM_WINDOWS_GUI
IMAGE_SUBSYSTEM_WINDOWS_CUI

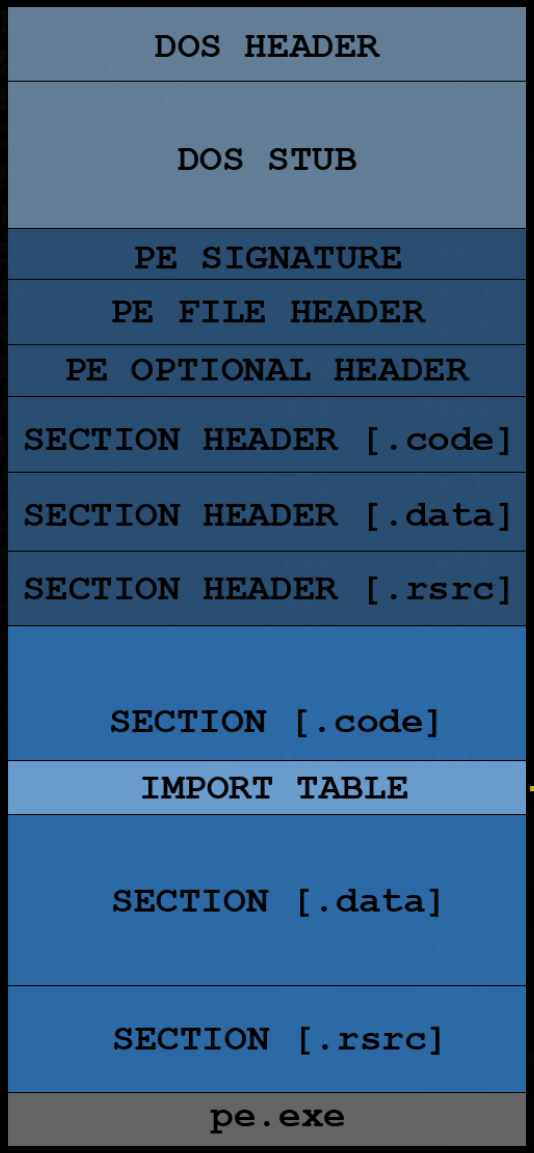
Budowa PE




```
IMAGE_DATA_DIRECTORY
DataDirectory[16];
```

0 EXPORT
1 IMPORT
2 RESOURCE
3 EXCEPTION
4 SECURITY
5 BASERELOC
6 DEBUG
7 ARCHITECURE
8 GLOBALPTR
9 TLS
10 LOAD_CONFIG
11 BOUND_IMPORT
12 IAT
13 DELAY_IMPORT
14 COM_DESCRIPTOR

Budowa PE

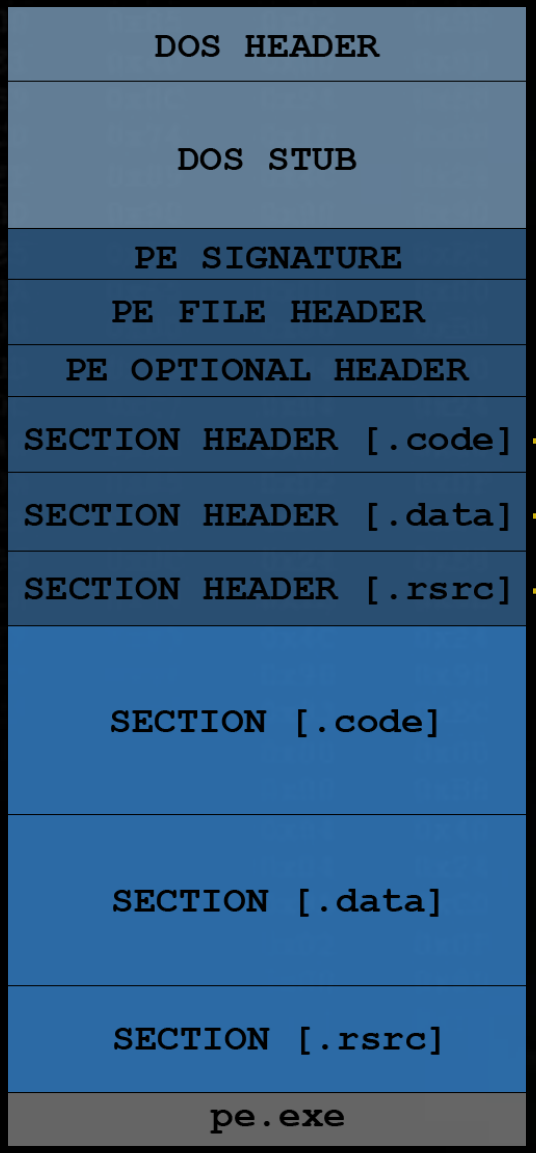


kernel32.dll
- CreateFileA
- LoadLibraryA
- WriteFile
- ExitProcess

user32.dll
- MessageBoxA

msvcrt.dll
- puts
- printf
- scanf

Budowa PE

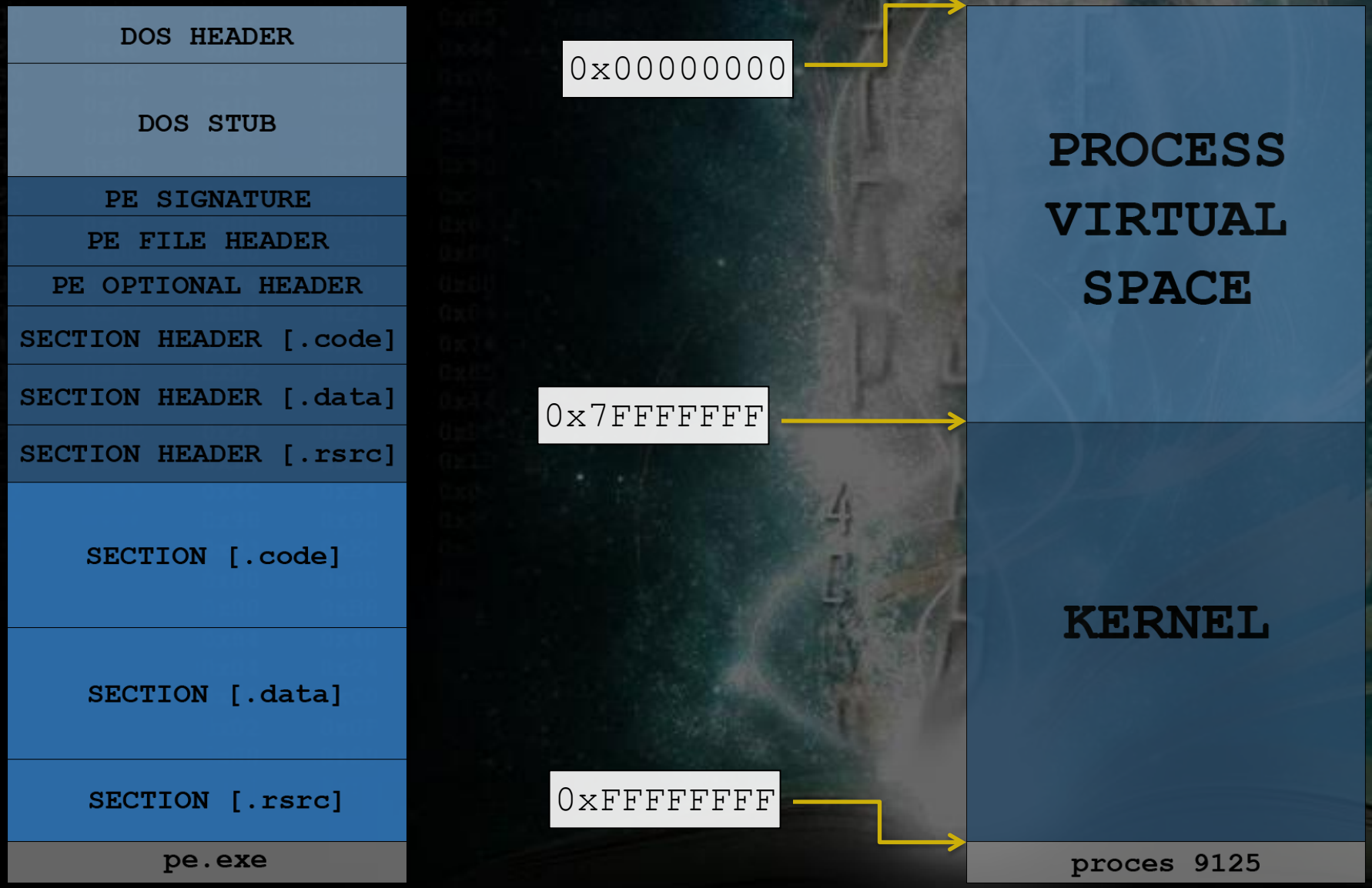


Struktura `IMAGE_SECTION_HEADER`.

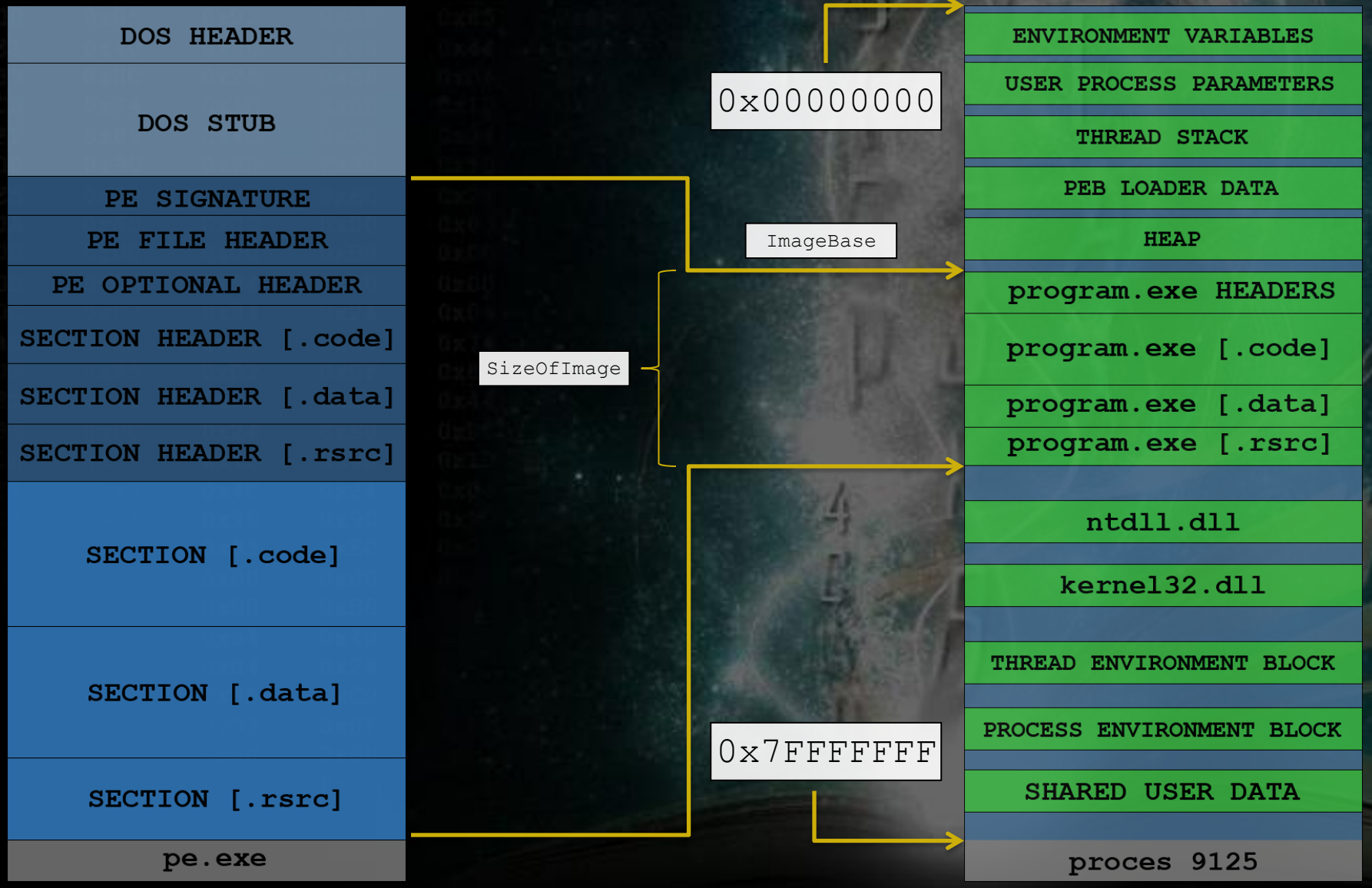
```
BYTE    Name[8];
DWORD   VirtualSize;
DWORD   VirtualAddress;
DWORD   SizeOfRawData;
DWORD   PointerToRawData;
…
DWORD   Characteristics;
```

IMAGE_SCN_CNT_CODE
IMAGE_SCN_CNT_INITIALIZED_DATA
IMAGE_SCN_MEM_EXECUTE
IMAGE_SCN_MEM_READ
IMAGE_SCN_MEM_WRITE

Proces, pamięć i PE



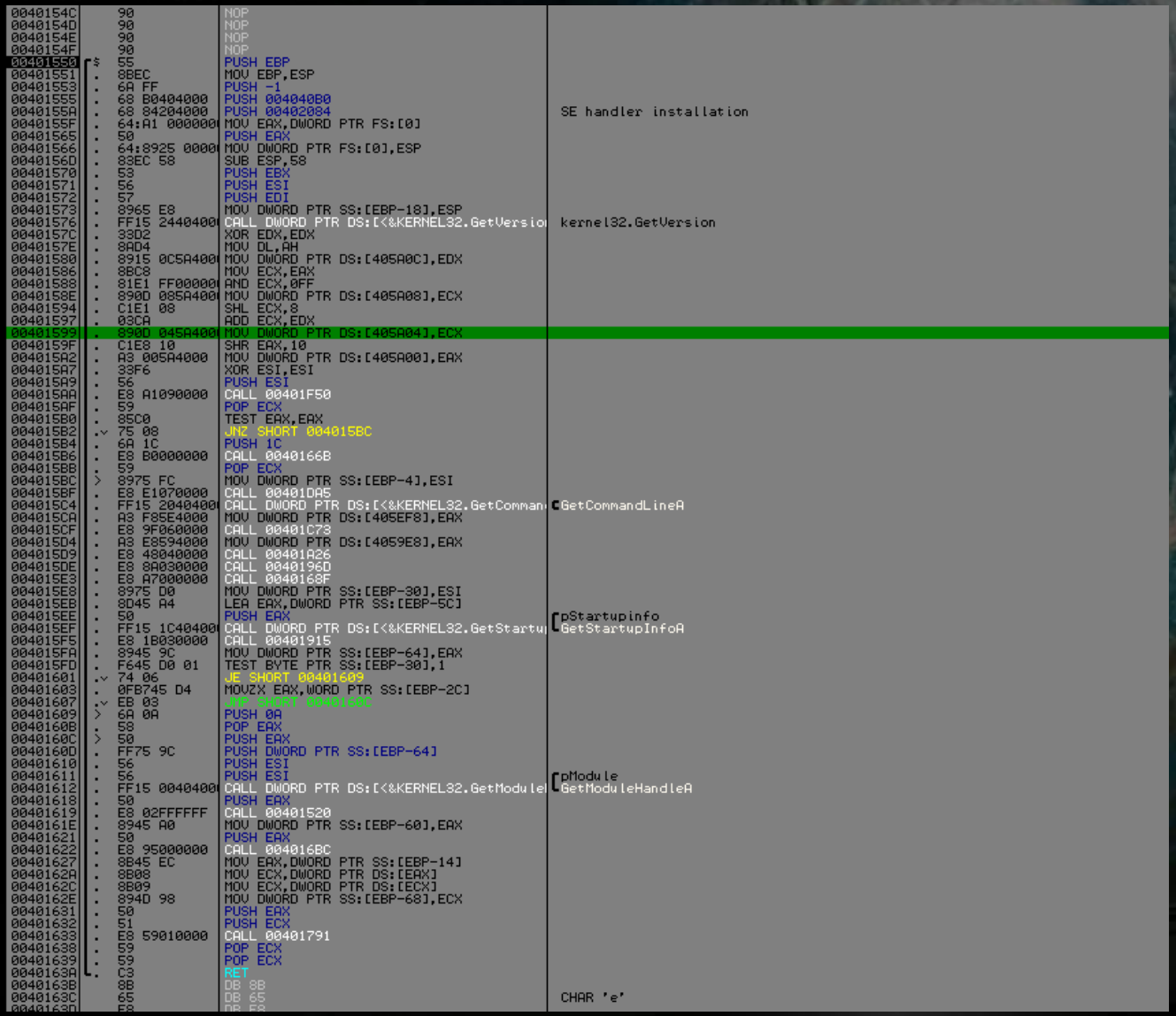
Proces, pamięć i PE



Język Asembler

- Język **asembler** jest znany jako język niskiego poziomu. Na poziomie języka asembler i kodu maszynowego tworzone są polecenia interpretowane **bezpośrednio przez procesor**. Język asembler jest najczęściej używany w czasie komunikacji z systemem operacyjnym i bezpośredniej pracy ze sprzętem.
- Dzięki temu językowi możliwa jest także **optymalizacja** pewnych krytycznych obszarów aplikacji oraz zwiększenie ich **wydajności**.
- Język asembler należy poznawać wraz z **architekturą komputera** i koncepcjami systemu operacyjnego.

Język Asembler



```
0040154C     90           NOP
0040154D     90           NOP
0040154E     90           NOP
0040154F     90           NOP
00401550  /$ 55           PUSH EBP
00401551  |. 8BEC         MOV EBP,ESP
00401553  |. 6A FF        PUSH -1
00401555  |. 68 B0404000  PUSH 004040B0
0040155A  |. 68 84204000  PUSH 00402084                    SE handler installation
0040155F  |. 64:A1 000000 MOV EAX,DWORD PTR FS:[0]
00401565  |. 50           PUSH EAX
00401566  |. 64:8925 0000 MOV DWORD PTR FS:[0],ESP
0040156D  |. 83EC 58      SUB ESP,58
00401570  |. 53           PUSH EBX
00401571  |. 56           PUSH ESI
00401572  |. 57           PUSH EDI
00401573  |. 8965 E8      MOV DWORD PTR SS:[EBP-18],ESP
00401576  |. FF15 2440400 CALL DWORD PTR DS:[<&KERNEL32.GetVersio kernel32.GetVersion
0040157C  |. 33D2         XOR EDX,EDX
0040157E  |. 8AD4         MOV DL,AH
00401580  |. 8915 0C5A400 MOV DWORD PTR DS:[405A0C],EDX
00401586  |. 8BC8         MOV ECX,EAX
00401588  |. 81E1 FF00000 AND ECX,0FF
0040158E  |. 890D 085A400 MOV DWORD PTR DS:[405A08],ECX
00401594  |. C1E1 08      SHL ECX,8
00401597  |. 03CA         ADD ECX,EDX
00401599  |. 890D 045A400 MOV DWORD PTR DS:[405A04],ECX
0040159F  |. C1E8 10      SHR EAX,10
004015A2  |. A3 005A4000  MOV DWORD PTR DS:[405A00],EAX
004015A7  |. 33F6         XOR ESI,ESI
004015A9  |. 56           PUSH ESI
004015AA  |. E8 A1090000  CALL 00401F50
004015AF  |. 59           POP ECX
004015B0  |. 85C0         TEST EAX,EAX
004015B2  |.v 75 08       JNZ SHORT 004015BC
004015B4  |. 6A 1C        PUSH 1C
004015B6  |. E8 B0000000  CALL 0040166B
004015BB  |. 59           POP ECX
004015BC  |> 8975 FC      MOV DWORD PTR SS:[EBP-4],ESI
004015BF  |. E8 E1070000  CALL 00401DA5
004015C4  |. FF15 2040400 CALL DWORD PTR DS:[<&KERNEL32.GetComman GetCommandLineA
004015CA  |. A3 F85E4000  MOV DWORD PTR DS:[405EF8],EAX
004015CF  |. E8 9F060000  CALL 00401C73
004015D4  |. A3 E8594000  MOV DWORD PTR DS:[4059E8],EAX
004015D9  |. E8 48040000  CALL 00401A26
004015DE  |. E8 8A030000  CALL 0040196D
004015E3  |. E8 A7000000  CALL 0040168F
004015E8  |. 8975 D0      MOV DWORD PTR SS:[EBP-30],ESI
004015EB  |. 8D45 A4      LEA EAX,DWORD PTR SS:[EBP-5C]
004015EE  |. 50           PUSH EAX                         pStartupinfo
004015EF  |. FF15 1C40400 CALL DWORD PTR DS:[<&KERNEL32.GetStartu GetStartupInfoA
004015F5  |. E8 1B030000  CALL 00401915
004015FA  |. 8945 9C      MOV DWORD PTR SS:[EBP-64],EAX
004015FD  |. F645 D0 01   TEST BYTE PTR SS:[EBP-30],1
00401601  |.v 74 06       JE SHORT 00401609
00401603  |. 0FB745 D4    MOVZX EAX,WORD PTR SS:[EBP-2C]
00401607  |.v EB 03       JMP SHORT 0040160C
00401609  |> 6A 0A        PUSH 0A
0040160B  |. 58           POP EAX
0040160C  |> 50           PUSH EAX
0040160D  |. FF75 9C      PUSH DWORD PTR SS:[EBP-64]
00401610  |. 56           PUSH ESI
00401611  |. 56           PUSH ESI                         pModule
00401612  |. FF15 0040400 CALL DWORD PTR DS:[<&KERNEL32.GetModule GetModuleHandleA
00401618  |. 50           PUSH EAX
00401619  |. E8 02FFFFFF  CALL 00401520
0040161E  |. 8945 A0      MOV DWORD PTR SS:[EBP-60],EAX
00401621  |. 50           PUSH EAX
00401622  |. E8 95000000  CALL 004016BC
00401627  |. 8B45 EC      MOV EAX,DWORD PTR SS:[EBP-14]
0040162A  |. 8B08         MOV ECX,DWORD PTR DS:[EAX]
0040162C  |. 8B09         MOV ECX,DWORD PTR DS:[ECX]
0040162E  |. 894D 98      MOV DWORD PTR SS:[EBP-68],ECX
00401631  |. 50           PUSH EAX
00401632  |. 51           PUSH ECX
00401633  |. E8 59010000  CALL 00401791
00401638  |. 59           POP ECX
00401639  |. 59           POP ECX
0040163A  |. C3           RET
0040163B     8B           DB 8B
0040163C     65           DB 65                            CHAR 'e'
0040163D     E8           DB E8
```


Etapy analizy aplikacji

Rekonesans

- **Narzędzia:**
  - PeID, PeView, Skanery AV (VirusTotal)
  - Process Explorer, Process Monitor
  - OllyDump, Import Reconstructor

Właściwa analiza

- **Narzędzia:**
  - Disassembler IDA
  - Debugger OllyDbg

Modyfikacja

- **Narzędzia:**
  - Debuggery OllyDbg
  - Edytor binarny Hexplorer
  - Resource Hacker
  - Własne programy patchujące

Narzędzia

- Disasembler IDA
  - http://www.hex-rays.com/idapro/idadownfreeware.htm
- Debugger OllyDbg
  - http://www.ollydbg.de
- PeID
  - http://www.peid.info/
- Icy's Hexplorer
  - http://artemis.wszib.edu.pl/~mdudek/
- Process Explorer, Process Monitor
  - http://technet.microsoft.com/en-us/sysinternals/bb896653
  - http://technet.microsoft.com/en-us/sysinternals/bb896645

Zanim zaczniemy 0x02...

Metod analizowania aplikacji jest wiele. **Nie ma jednej** i zawsze działającej **metody**. Każdy reverser ma swoje strategie. W zależności od aplikacji będziemy zmuszeni skorzystać z innych metod.

Dlatego aby **skutecznie** i **szybko** analizować różnego rodzaju aplikacje należy **poznać** jak **najwięcej** sztuczek, metod, tricków i podejść.

Zadanie 1

- **Aplikacja:** ConsoleInfinity.exe
- **Autor:** Promix17 (www.crackmes.de)
- **Packer:** ???

- **Zadania:**
  - Zanalizować aplikację, rozpakować (jeśli trzeba), zcrackować.
  - Napisać generator kluczy (aka keygen).

Zadanie 2

- **Aplikacja:** snake.exe
- **Packer:** brak

- **Zadania:**
  - Zanalizować aplikację
  - Załadować własną bibliotekę w pamięć aplikacji
  - Dodać nowe funkcjonalności:
    - Możliwość wybory poziomu gry
    - Obsługa ekstra zwierzaka (dodatkowe punkty, znika po x sek. etc.)
    - …


Dziękuję za uwagę!