Wstęp do Reverse Engineeringu

The great thing about knowing asm is –
everything is open source.

Michał „sc0ty” Szymaniak

http://sc0ty.pl/

Plan wykładu

- Co to jest Reverse Engineering?
- Budowa plików PE (EXE, DLL)
  - Sekcje
  - Adresowanie pamięci
  - Import Address Table (IAT)
- Trochę asemblera x86
  - Calling conventions
  - Funkcje WinAPI
- Narzędzia
- Praktyczne przykłady

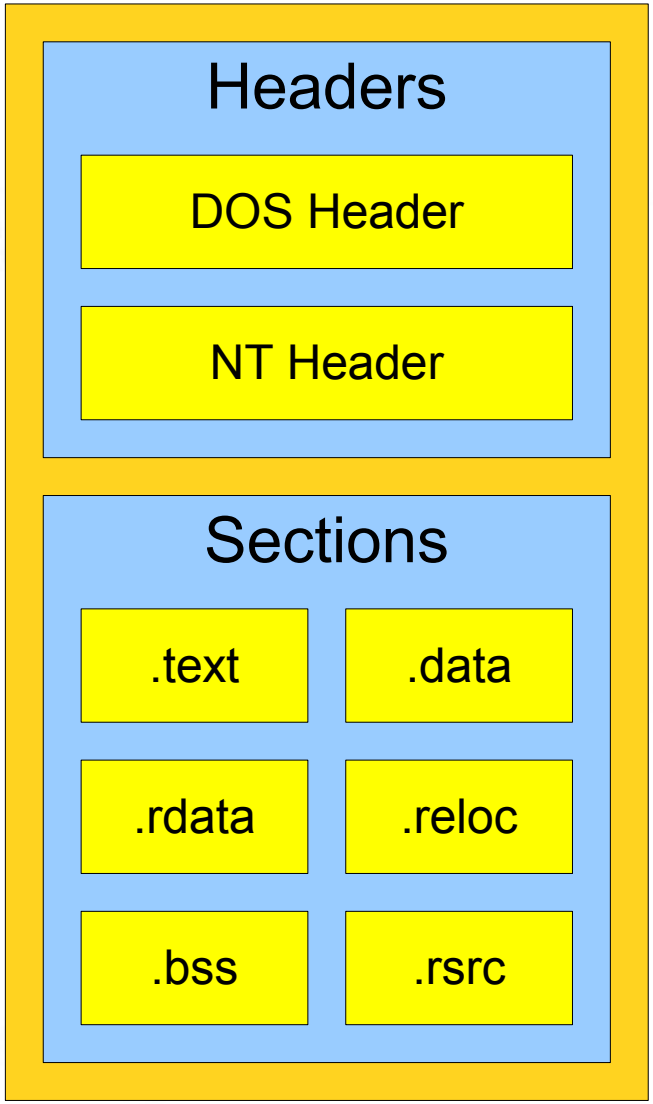
Co to jest Reverse Engineering?

Inżynieria wsteczna – proces badania produktu (programu) w celu ustalenia jak on dokładnie działa oraz w jaki sposób został wykonany.

Zastosowanie:

- analiza malware'u
- omijanie zabezpieczeń
- rozpracowywanie nieudokumentowanych formatów plików, protokołów komunikacyjnych etc.

Budowa plików PE



DOS Header:

- „This program cannot be run in DOS mode.”

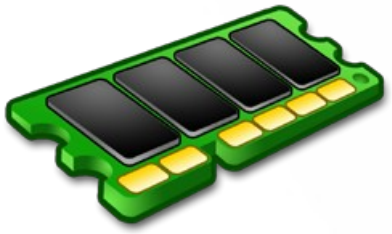
NT Header:

- lista sekcji oraz ich parametry
- adres początkowy kodu (entrypoint)
- adres bazowy
- memory layout

Microsoft PE and COFF Specification:
http://msdn.microsoft.com/en-us/windows/hardware/gg463119.aspx

Adresowanie sekcji

Base Address – adres bazowy aplikacji w pamięci RAM



Raw Address
(offset w pliku dyskowym)

Virtual Address (VA)
(adres w pamięci RAM)

VA = Base Address + Relative Virtual Address (RVA)

RVA – offset względem adresu bazowego

Import Address Table (IAT)

Służy do adresowania funkcji znajdujących się w bibliotekach DLL

Zawiera:

- nazwy plików DLL
- nazwy importowanych funkcji (lub identyfikatory numeryczne)
- miejsce na adres importowanej funkcji

System w czasie ładowania programu uzupełnia adresy funkcji

Calling conventions

Sposoby wywoływania funkcji, przekazywania parametrów oraz rezultatu jej wykonania

- *cdecl*
- *stdcall*
- *pascal*
- *safecall*
- *syscall*
- *fastcall*
- *optlink*

Calling conventions

cdecl

- parametry przekazywane przez stos w odwrotnej kolejności
- wynik przekazywany w rejestrze (EAX)
- stos czyści wywołujący funkcję
- C/C++

stdcall

- parametry przekazywane przez stos w odwrotnej kolejności
- wynik przekazywany w rejestrze (EAX)
- stos czyści sama funkcja
- WinAPI, biblioteki DLL

Calling conventions

Przykład wywołania funkcji

```
funkcja(arg1, arg2);
```

cdecl

```
...
push arg2
push arg1
call funkcja
add esp, 8
...
```

stdcall

```
...
push arg2
push arg1
call funkcja
...
```

Funkcje WinAPI

Funkcje importowane z plików *Kernel32.dll*, *User32.dll, Gdi32.dll*, *Comctl32.dll*, *Ntdll.dll* i innych, np:

C/C++:

```
MessageBox(0, „Hello World!”, 0, 0);
```

ASM:

```
    ...
    push    0
    push    0
    push    offset Text
    push    0
    call    [MessageBoxA]
    ...
Text:
    db "Hello World!", 0
```

Narzędzia

Niezbędnik:

- CFF Explorer - http://ntcore.com/
- HEX Editor
- IDA Pro - http://hex-rays.com/

Dodatkowo:

- Kalkulator
- NASM - http://nasm.us/
- Hex-Rays
- ProcMonitor

Zakończenie

Dziękuję za uwagę