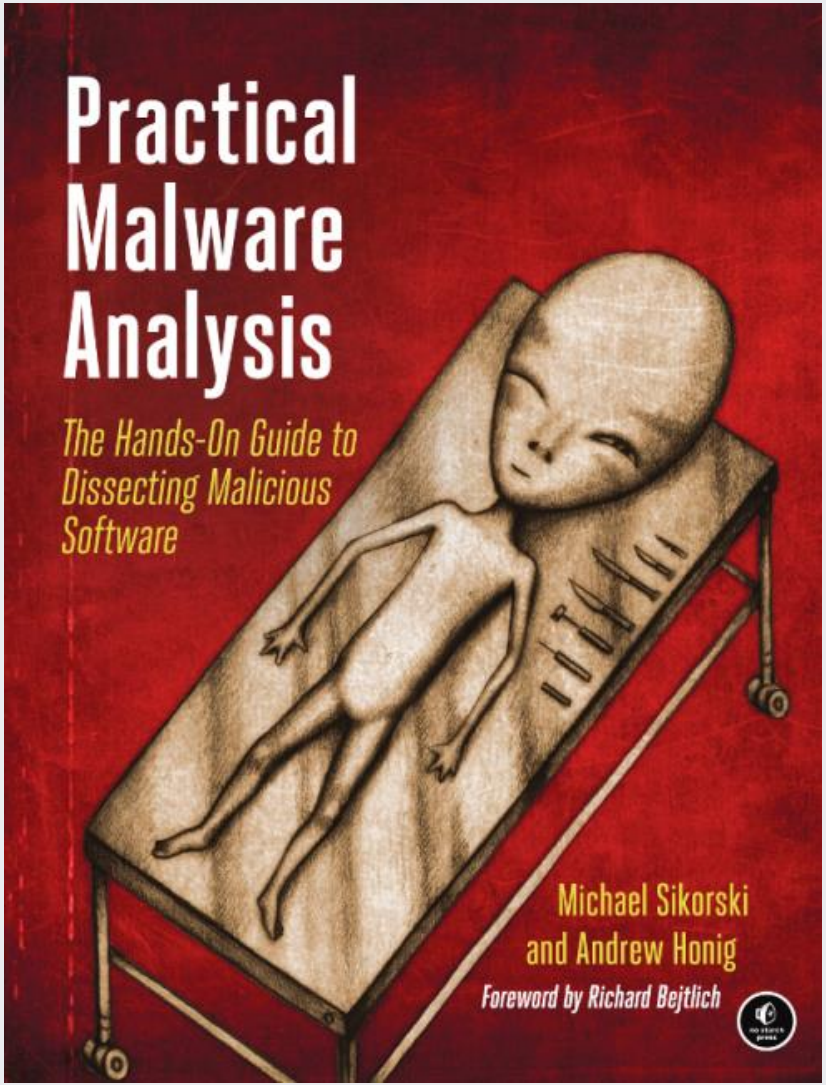
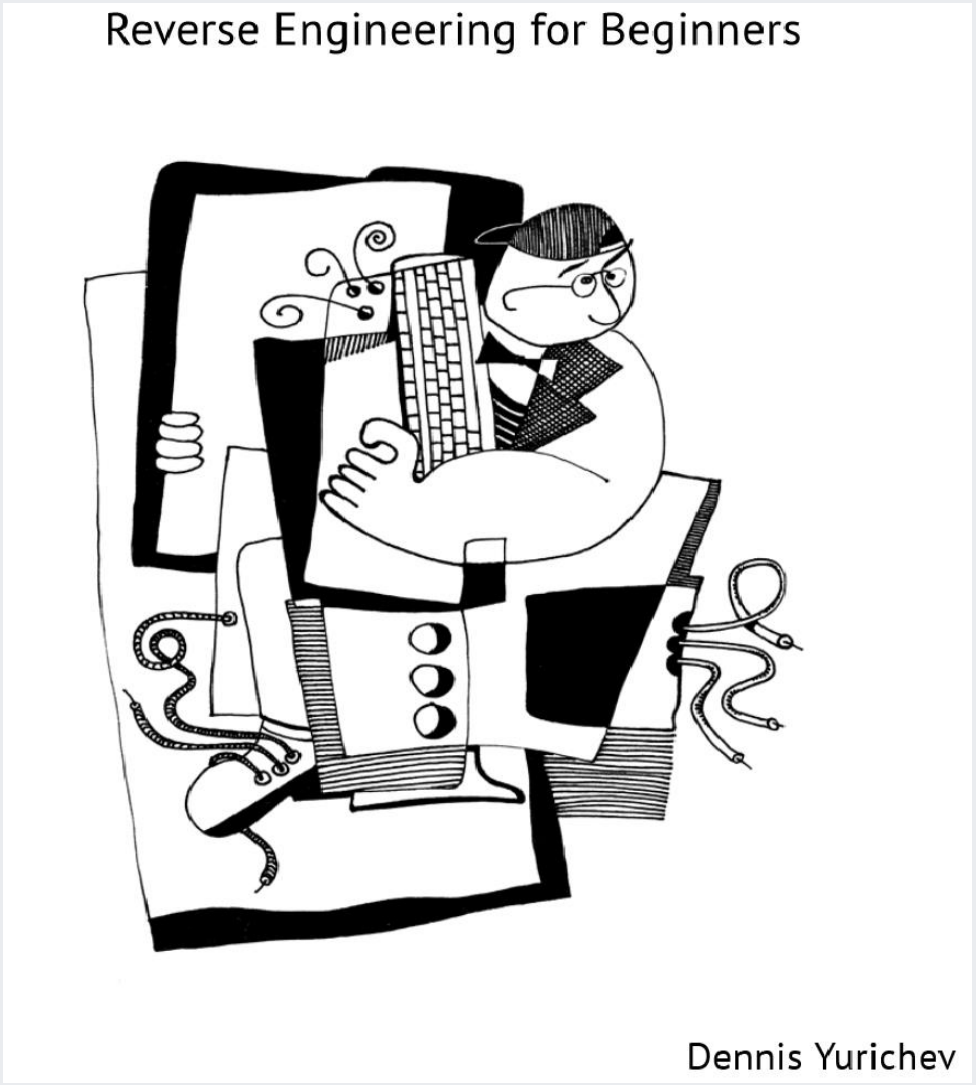
RE Windows

filip.kafka@eset.sk

Základy reverzného inžinierstva





http://beginners.re/Reverse_Engineering_for_Beginners-en.pdf

Čo nám pomáha pochopiť a analyzovať kód

- Dobrá znalosť ASM
- Reverzovať si vlastný kód
- Prax, prax, prax
- Automatizácia (skripty + pluginy), dobrá znalosť nástrojov
- Read the documentation in other than Slovak language
- Znalosť operačného systému a jeho štruktúr – procesy, vlákna, PE súbory
- Poznať nedokumentované časti systému

Reverzné metódy:

Základné (za krátky čas získať odhad o činnosti softwaru):

- Statické – importy (funkcie), exporty, stringy, sekcie, entropia (packer)...
- Dynamické - API funkcie, sieťová komunikácia...

Pokročilé (detailné):

- Statické (disassembler) - všetky inštrukcie, importy
- Dynamické (debugger) - krokovanie, pozeranie obashu pamäte / registrov / argumentov

Základné (za krátky čas získať odhad o činnosti softwaru):

- Statické: Hiew, PEView, PEiD CFF Explorer, IDR, DeDe
- Dynamické: Windows Sysinternals Tools (process monitor (procmon) , process explorer (procexp)), dependency walker, Regshot; faking network: fakenet, ApateDNS, Netcat, Wireshark, Network monitor, API Rohitab

Pokročilé (detailné):

- Statické (disassembler): IDA Pro, Bokken, Radare
- Dynamické (debugger): OllyDbg, WinDbg, IDA Debugger

Level abstrakcie

Hardware – Jediný fyzický level pozostávajúci z elektrických okruhov, ktoré implementujú komplexnú kombináciu logických operácii (XOR, AND, OR, NOT...) - ’digitálna logika ’.

Microcode (firmware) – Operuje len na tom okruhu, pre ktorý bol navrhnutý. Interaguje s hardwarom – hovorí, čo treba robiť keď prídu jednotlivé opkódy.

Machine code (opcodes) – Bajty, ktoré hovoria procesoru čo chcete robiť.

Low-level language – Ľudsky čitateľná verzia súboru inštrukcii.

Hight-level language – C, C++... Kód premenený do machine kódu procesom (kompiláciou), ktorú vykonáva kompilátor.

Interpreted language – Java, C#... Kód nie je kompilovaný do machine kódu ale do ‘bytecode’. Bytecode vykonáva ’interpreter ’, ktorý ho za behu kompiluje (a skompilovanú verziu vykonáva).

Reverzné inžinierstvo

Reverzovaný objekt (software) väčšinou uložený na disku v binárnej podobe na úrovni machine kódu. Pri disassemblovaní zoberieme binárnu podobu ako vstup a snažíme sa vygenervať jazyk assembler ako výstup disassemblerom.

Volacie konvencie v MSVC

- Spôsob zadania argumentov do funkcie a upratania argumentov
- Najznámejšie formy volacích konvencií:
  - __cdecl
  - __stdcall
  - __fastcall
  - __thiscall
- Existujú aj ďalšie (__clrcall, __vectorcall)

__cdecl

- Väčšina C/C++ kompilátorov používa ako štandard pre vytvorené funkcie
- Nie je vopred známy počet argumentov
- Argumenty vkladá na stack v opačnom poradí ←
- Upratuje po sebe ten, kto volal funkciu

```
char* const_string = "this is ";
cdecl_Example(const_string ,"cdecl\n");
```

```
mov     [ebp+const_string], offset aThisIs ; "this is "
push    offset aCdecl   ; "cdecl\n"
mov     eax, [ebp+const_string]
push    eax
call    cdecl_Example
add     esp, 8
```

__stdcall

- Štandard pre Win32 API funkcie
- Vopred známy počet argumentov
- Argumenty vkladá na stack v opačnom poradí ←
- Stack čistí volaná funkcia – ESP sa posunie po zavolaní ret

#define WINAPI __stdcall // Windows.h

```
stdcall_Example(const_string ,"stdcall\n");
```

```
push    offset aStdcall ; "stdcall\n"
mov     ecx, [ebp+const_string]
push    ecx
call    stdcall_Example
```

__fastcall

- Prvé 2 argumenty sa vkladajú do ECX, EDX
- Ostatné argumenty vkladá na stack v opačnom poradí ←
- Stack čistí volaná funkcia

```
fastcall_example(const_string ,"fastcall\n");
```

```
mov     edx, offset aFastcall ; "fastcall\n"
mov     ecx, [ebp+const_string]
call    fastcall_example
```

__thiscall

- Používa sa v C++ pri objektoch
- Do ECX sa vkladá inštancia objektu – this
- Argumenty vkladá na stack v opačnom poradí ←
- Stack čistí volaná funkcia

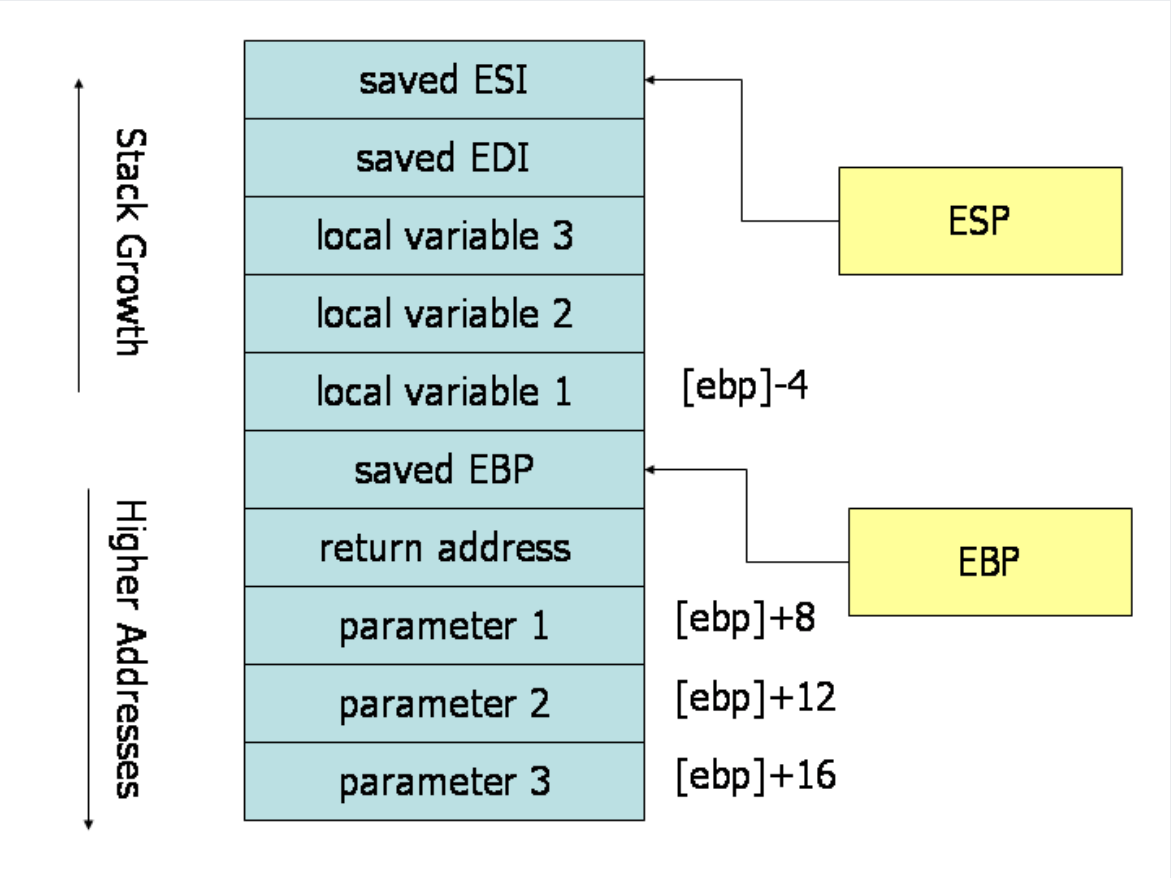
```
class_example->thiscall_example(const_string ,"thiscall\n");
```

```
push    offset aThiscall ; "thiscall\n"
mov     ecx, [ebp+const_string]
push    ecx
mov     ecx, [ebp+Example_class]
call    Example4__thiscall_example
```

C – konštruky v skompilovanej podobe

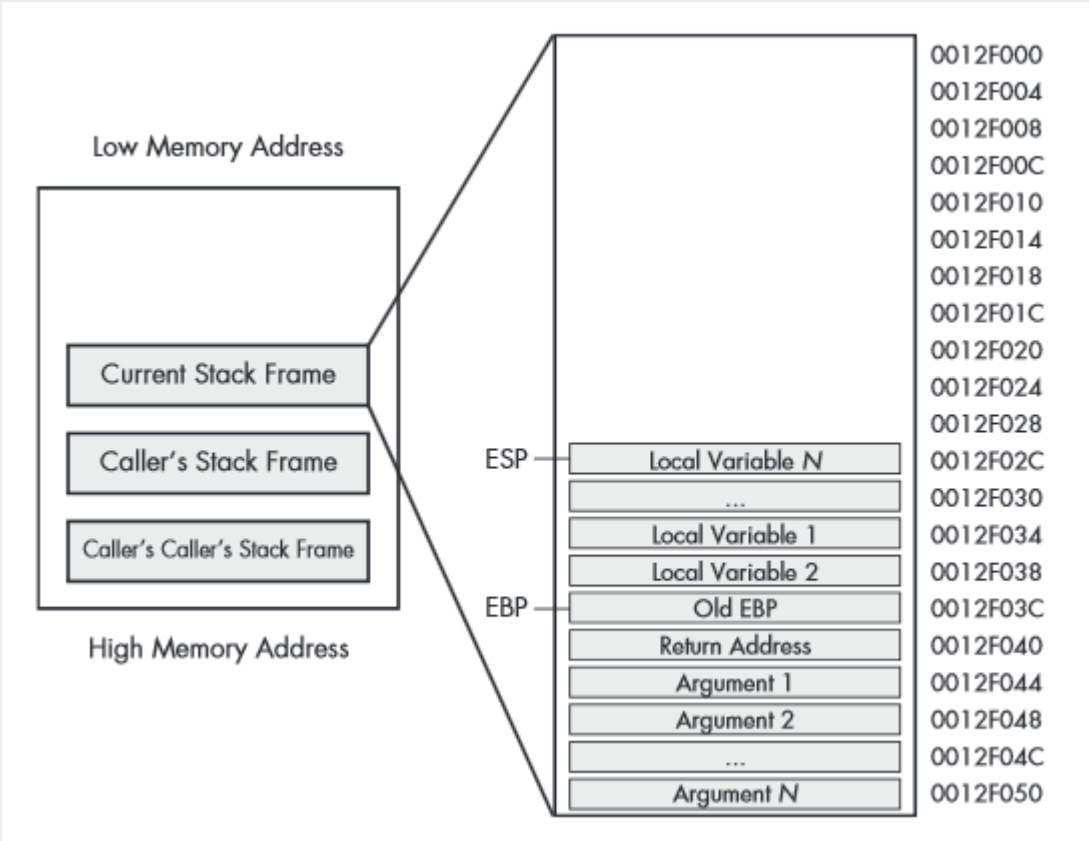
Ako vyzerá cyklus, podmienka, vnorená podmienka, switch-case?

Ako vyzerá stack počas funkcie (stack frame - čo všetko sa v ňom nachádza)



http://www.cs.virginia.edu/~evans/cs216/guides/x86.html

Stack z globálneho pohľadu



Practical Malware Analysis – Michael Sikorski, Andrew Honig