REbejs

2. workshop (draft0)

Opakování 1. workshopu

- Virtuální adresa
- Opkód instrukce, primární opkód, REX prefixy
- Instrukce, mnemonic, operandy
- Všeobecné registry, EIP, RIP, EFlags
- Volání funkce
- Základy OllyDbg, x64dbg
- Základní instrukce
- Hlavní volací konvence

Podklady

- wiki.base48.cz/REbejs
- Nebojte se reverzního inženýrství I.
- Nebojte se reverzního inženýrství II.
- Nebojte se reverzního inženýrství III.
- x86asm.net/links
- ref.x86asm.net/coder32.html
- ref.x86asm.net/coder64.html

Domácí úkoly

- Cracknutí crackme1 podle návodu
- Cracknutí jiným způsobem
- Změna CMP na SUB a její důsledky
- Přepsání crackme1 do vyššího jazyka
- Nejkratší kód na vyčerpání paměti zásobníku

2. workshop - témata

- Crackme #2 by RubberDuck
- + opkódy instrukcí
- + x64
- + x64 verze crackme #2: crackme2_x64.zip
- Paměťový model aplikace na Windows
- Módy, v kterých procesor může běžet (TODO SMM rootkit zjistit, co a jak)
- Ochrana paměti, virtuální paměť
- Stack, heap, entry point, kód, data, TEB (TIB)
- Binární formát instrukce
- Orientace v manuálech procesorů
- Vyhledávání na MSDN

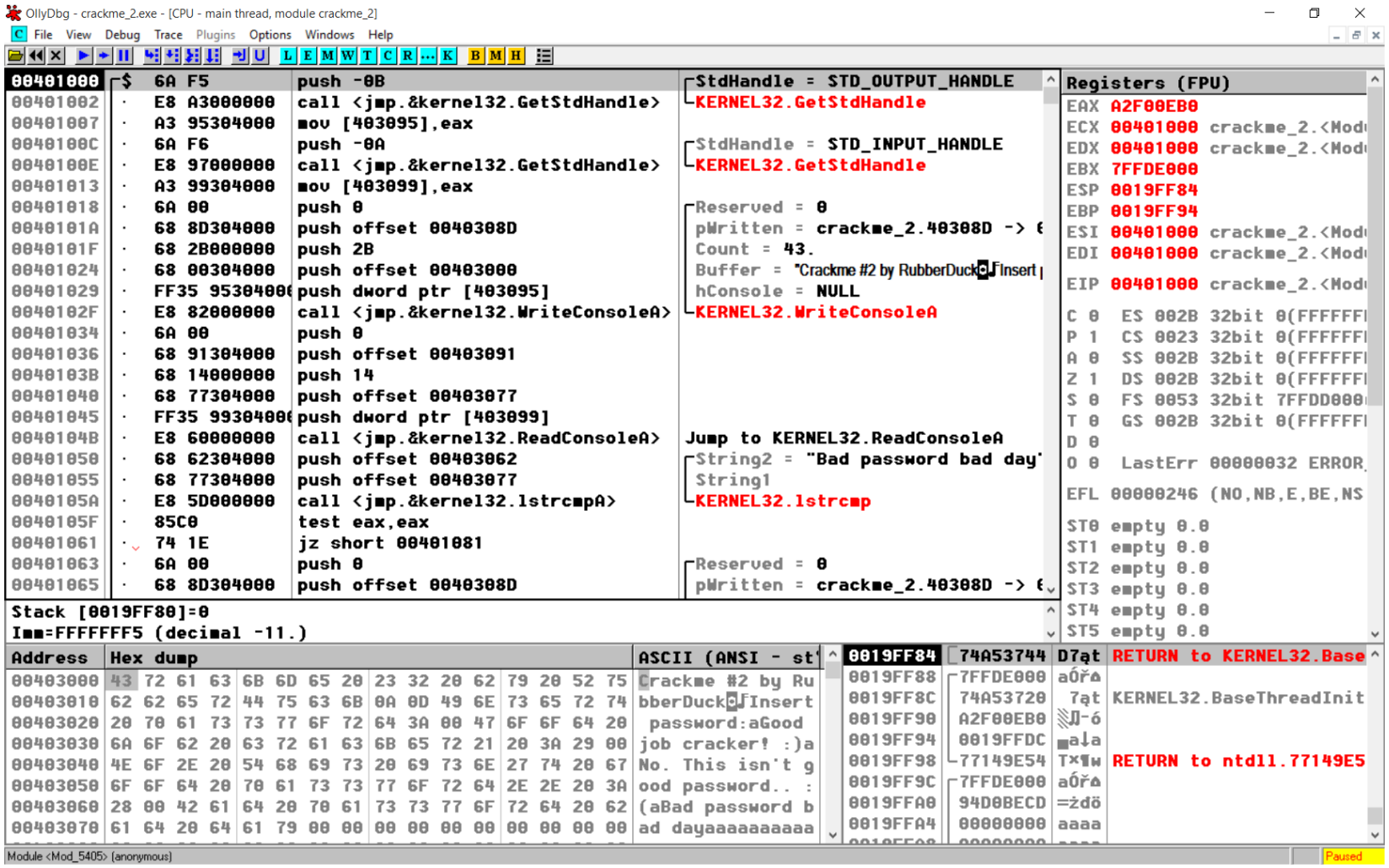
1. workshop - nástroje

- 32bitový debugger OllyDbg
  - OllyDbg 2.01
- 64bitový debugger x64dbg
  - x64dbg snapshot_2016-01-21_02-44

Crackme #2 by RubberDuck

```
C:\>crackme_2.exe
Crackme #2 by RubberDuck
Insert password: dunno
No. This isn't good password.. :(
C:\>_
```

OllyDbg – crackme2.exe



crackme2.exe – 1. instrukce PUSH

- **`00401000  6A F5  PUSH -0B`**
- ref.x86asm.net/coder32.html#x6A
- **`6A PUSH (sign extended) imm8`**
- **`imm8`**: 8bitová immediate value
- Takže jinak (velikost položky na zásobníku je DWORD):
- PUSH *FFFFFF*F5
- Je to první parametr volání WinAPI funkce GetStdHandle()

GetStdHandle()

- MSDN: Retrieves a handle to the specified standard device (standard input, standard output, or standard error).

```
HANDLE WINAPI GetStdHandle(
  _In_ DWORD nStdHandle
);
```

- Potřebujeme návratovou hodnotu `HANDLE`
- `winnt.h` (Microsoft SDKs):
- `typedef void *HANDLE;`
- Volací konvence stdcall: návratová hodnota je v EAX, ta je uložená pomocí následující instrukce na později

crackme2.exe – 3. instrukce MOV

- **`00401007  A3 95304000  MOV [403095], EAX`**
- ref.x86asm.net/coder32.html#xA3
- **`A3 MOV moffs16/32, eAX`**
- Opkódy **`A0-A3`**
- speciální forma MOV mezi AL/AX/EAX a pamětí
- Registr eAX (tzn. AX nebo EAX):
- *accumulator*, historicky optimalizovaný pro častý přístup
- Spousta opkódů podporujících accumulator, umožňujících kratší kódování instrukcí (viz třeba 0x04, 0x05, 0x0C, ...)
- Dědictví historie, dnes nemá podstatný význam
- *moffs* - *memory offset*: konstatní hodnota adresuje paměť

Crackme2.exe – 3. instrukce MOV

- **`MOV [403095], EAX`**
- 0x403095 **=** 4206741 = nezarovnaná paměť při ukládání DWORD hodnoty
- Architektura x86 nevynucuje zarovnání paměti na určitou hodnotu
- Připomenutí: 64-bit Windows vynucuje zarovnání zásobníku na 16 bytů

crackme2.exe – 10. a 11. instr. PUSH

- `00401024  68 00304000`
  `PUSH OFFSET 00403000`

- `00401029  FF35 95304000`
  `PUSH DWORD PTR [403095]`

- ref.x86asm.net/coder32.html#xFF
- Skupina různých instrukcí
- **`FF /6 PUSH r/m16/32`**
- *ModR/M byte* **`0x35`**
- **`DWORD PTR`**: označení velikosti odkazované hodnoty

crackme2.exe – porovnání hesel

- `WriteConsoleA()`: vypíše výzvu k zadání hesla
- `ReadConsoleA()`: přečte zadané heslo
- `lstrcmp()`: WinAPI funkce porovnávající řetězce, podobná funkci ze standardní C knihovny

```
int WINAPI lstrcmp(
  _In_ LPCTSTR lpString1,
  _In_ LPCTSTR lpString2
);
```

- „If the strings are equal, the return value is zero.“

crackme2.exe – vyhodnocení hesla

- **`85C0 TEST EAX, EAX`**
- Zvláštní případ na vyhodnocení nulové hodnoty v registru
- Opkódy `84` a `85`: TEST r/m, r
- ModR/M byte **`C0`**
- TEST: logical compare
- TEST nemění žádný operand
- Operace TEST = operace AND
- TEST nastavuje šest stavových příznaků, viz:

85		r						TEST	r/m16/32	r16/32					o..szapc

- Smysl dávají tři: SF, ZF a PF
- `0 AND 0` = 0 → nastaví se ZF

crackme1.exe – vyhodnocení CMP

- crackme1:

```
83F8 01   CMP EAX, 1
74 0C     JE SHORT 00401016
```

- crackme2:

```
85C0      TEST EAX, EAX
74 1E     JZ SHORT 00401081
```

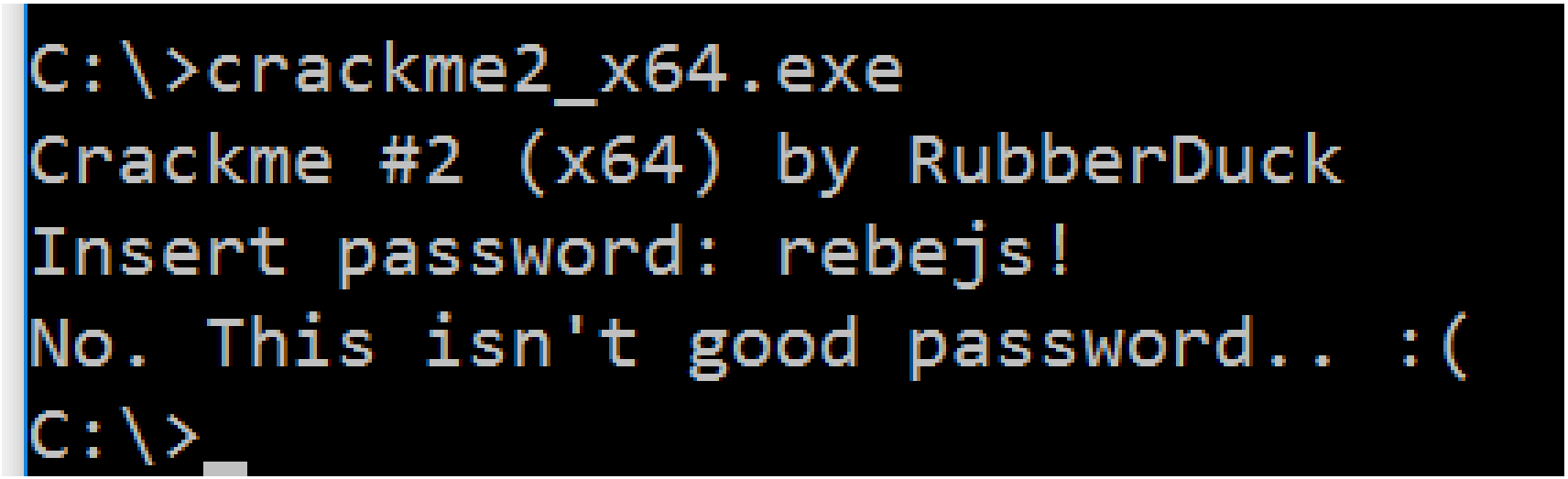
- `TEST EAX, EAX` má kratší opkód než `CMP EAX, 0`

- Zajímavost: OllyDbg disassembler vyhodnotil, že vhodnější je mnemonic **`JZ`**

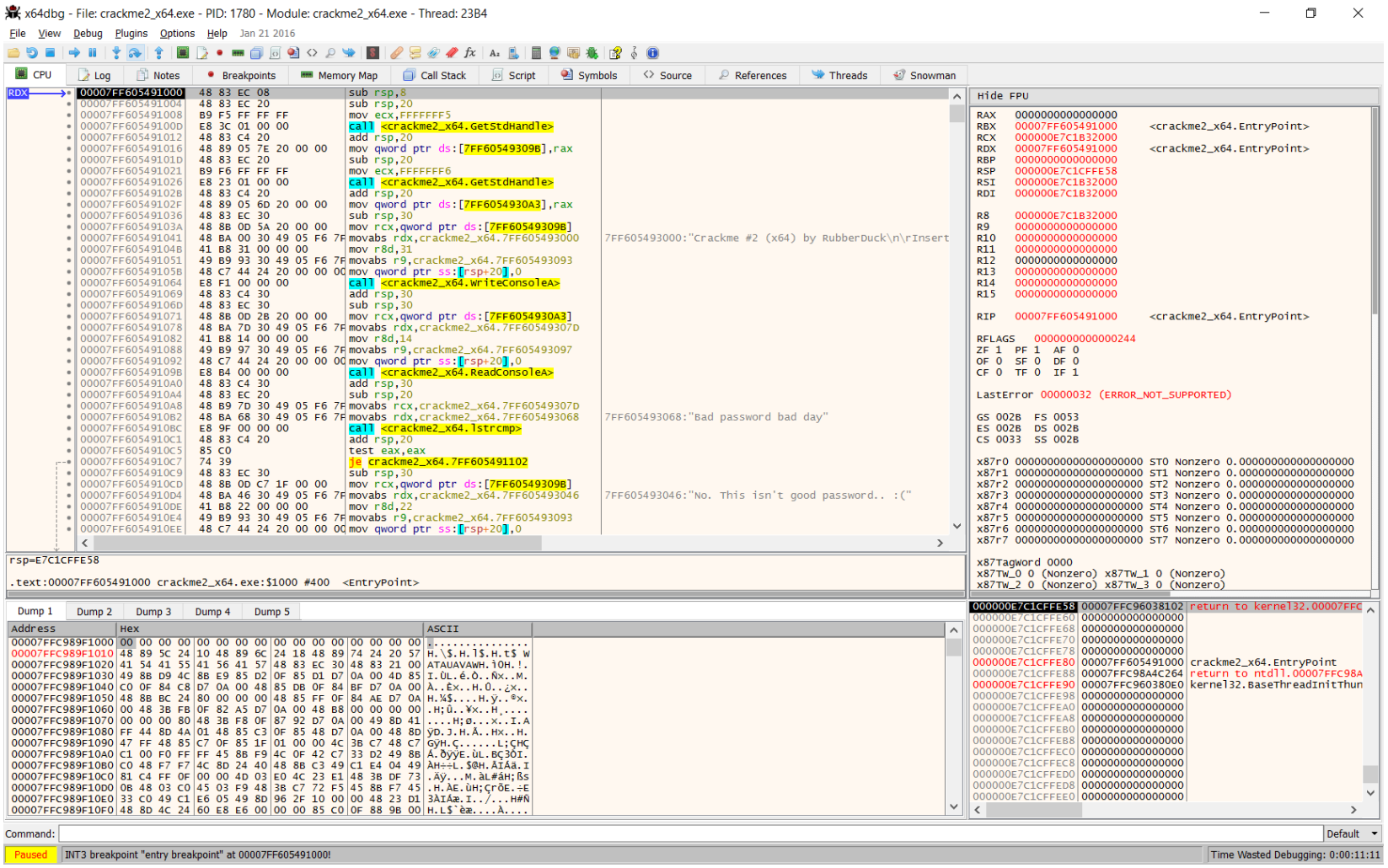
Shrnutí instrukcí

- MOV
- CMP, SUB
- **TEST**, **AND**
- JE, Jcc
- JMP
- PUSH
- CALL
- **Adresace paměti pomocí `[moffs]` a `[disp32]`**
- INT3
- NOP
- ADD [EAX], AL

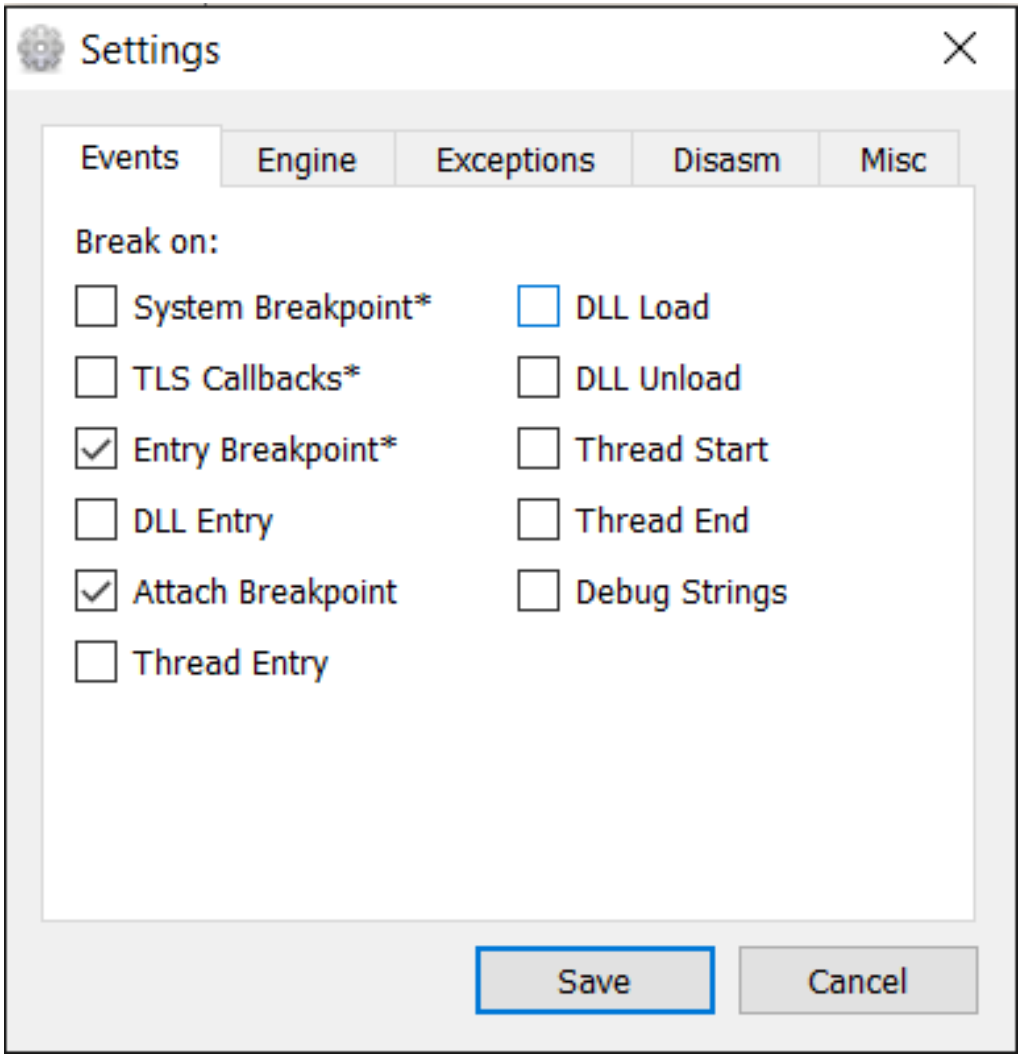
Crackme #2 (x64) by RubberDuck



x64dbg – crackme2_x64.exe



x64dbg – nastavení eventů



Domácí úkoly

1. Crackněte crackme2 (x86 i x64), jakobyste neznali heslo