Reverse Engineering for Beginners

Dennis Yurichev

Reverse Engineering for Beginners

Dennis Yurichev
<dennis(a)yurichev.com>

©2013-2015, Dennis Yurichev.

This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-nd/3.0/.

Text version (November 25, 2015).

The latest version (and Russian edition) of this text accessible at beginners.re. An A4-format version is also available.

You can also follow me on twitter to get information about updates of this text: @yurichev[1] or to subscribe to the mailing list[2].

The cover was made by Andy Nechaevsky: facebook.

[1] twitter.com/yurichev
[2] yurichev.com

Warning: this is a shortened LITE-version!

It is approximately 6 times shorter than full version (~150 pages) and intended to those who wants for very quick introduction to reverse engineering basics. There are nothing about MIPS, ARM, OllyDBG, GCC, GDB, IDA, there are no exercises, examples, etc.

If you still interesting in reverse engineering, full version of the book is always available on my website: beginners.re.

Contents

I Code patterns **1**

1 A short introduction to the CPU **3**

2 The simplest Function **5**
2.1 x86 5

3 Hello, world! **7**
3.1 x86 7
3.1.1 MSVC 7
3.2 x86-64 9
3.2.1 MSVC—x86-64 9
3.3 Conclusion 10

4 Function prologue and epilogue **11**
4.1 Recursion 11

5 Stack **13**
5.1 Why does the stack grow backwards? 13
5.2 What is the stack used for? 14
5.2.1 Save the function's return address 14
5.2.2 Passing function arguments 16
5.2.3 Local variable storage 17
5.2.4 x86: alloca() function 17
5.2.5 (Windows) SEH 19
5.2.6 Buffer overflow protection 19
5.2.7 Automatic deallocation of data in stack 19
5.3 A typical stack layout 19

6 `printf()` with several arguments **21**
6.1 x86 21

6.1.1 x86: 3 arguments . . . 21
6.1.2 x64: 8 arguments . . . 23
6.2 Conclusion . . . 24
6.3 By the way . . . 25

7 scanf() 26
7.1 Simple example . . . 26
7.1.1 About pointers . . . 26
7.1.2 x86 . . . 27
7.1.3 x64 . . . 29
7.2 Global variables . . . 30
7.2.1 MSVC: x86 . . . 31
7.2.2 MSVC: x64 . . . 33
7.3 scanf() result checking . . . 34
7.3.1 MSVC: x86 . . . 34
7.3.2 MSVC: x86 + Hiew . . . 37
7.3.3 MSVC: x64 . . . 38
7.4 Exercise . . . 39

8 Accessing passed arguments 40
8.1 x86 . . . 40
8.1.1 MSVC . . . 40
8.2 x64 . . . 42
8.2.1 MSVC . . . 42

9 More about results returning 45
9.1 Attempt to use the result of a function returning *void* . . . 45
9.2 What if we do not use the function result? . . . 47

10 GOTO operator 48
10.1 Dead code . . . 49

11 Conditional jumps 50
11.1 Simple example . . . 50
11.1.1 x86 . . . 51
11.2 Calculating absolute value . . . 56
11.2.1 Optimizing MSVC . . . 56
11.3 Ternary conditional operator . . . 57
11.3.1 x86 . . . 57
11.3.2 Let's rewrite it in an `if/else` way . . . 59
11.4 Getting minimal and maximal values . . . 60
11.4.1 32-bit . . . 60
11.5 Conclusion . . . 61
11.5.1 x86 . . . 61

Click here to download full PDF material