Review of the book
"Identifying Malicious Code Through Reverse Engineering"
by Abhishek Singh and Baibhav Singh
Springer
ISBN: 978-0-387-09824-1

Jannik Pewny

2009-11-03

1 What the book is about

Contrary to its title, the book does not include a manual on how to identify malicious code through reverse engineering.

It does include an introduction to the x86 *Assembly language*.
The table of contents lists sub-chapters about registers, instruction-format, instructions, stack, calling conventions, data constructs (including the thread local storage), representation of arithmetic operations in assembly, representation of data structure and virtual function calls in assembly.

It follows a chapter on the *Fundamental[s] of Windows-programming*, including memory management, virtual address descriptors, processes and threads with mentioning of synchronization objects, process initialization sequence, reversing Windows NT and security features of Vista.

There is also a chapter called *Portable-Executable-File-Format* (the file format for .exe and .dll and a few others; it is the format Windows stores executable code), which shows how a Portable Executable (PE) is constructed.

Reversing Binaries for Identifying Vulnerabilities is the title of the next chapter. It handles common programming errors like stack-overflows, heap-overflows, off-by-one-errors, integer overflows, format string-vulnerabilities and such along with their exploitation. This includes rather extraordinary (compared to the standard stack-overflow-exploit) things like exploitation via the structure exception handler (SEH). The chapter closes with general concepts on how to write exploits and the localization of the payload.

The last chapter is called *Fundamental[s] of Reverse Engineering* and starts with a sub-chapter called "Anti-Reversing method[s]", continues with anti-debugging techniques, virtual machine detection and unpacking (reversing the actions of so called packers).

Each chapter ends with a conclusion, in which the authors sum up a chapter's content.

The book ends with a list of 18 pages of file-signatures which could help you to identify the file-type of a file. The range goes from standard MIDI-Files over dBase III files to MP3- or MPEG-files and even Amiga disk files. So it pretty much covers a lot of common and uncommon files. But not a single virus, worm or Trojan horse (except those, who were found and quarantined and thereby marked with a specific signature by Norton Anti-Virus).

2 What the book is like

While reading the first text-page of the book, I discovered the first spelling mistake. Since I read this book with writing this review in mind, I thought I better keep track of them and be a helpful reader, for that the publisher can correct such mistakes.
I stopped counting when I had more than 15 spelling mistakes on the first 20 pages. During the whole book, one can see that the book was written by two authors and it seems, that their knowledge about the English language differs. The writing style of one author is quite okay, while the other prefers very short sentences. The latter are not so nice to read, since they seem choppy and interrupt the train of thoughts a subordinate clause starting with "because" lets just happen.
But there were also a lot of mistakes that are not due to a lack of knowledge of a certain language, but made me think that the book was not too carefully checked after the initial writing.

Some of the representative mistakes are those on page 7, talking about the status flags, when the authors state that there is a "Piraty Flag" instead of a parity flag, page 15, talking about C-Data types and a "shirt" instead of a short [int].
One can also read something about "device divers" and the "widows kernel space" on page 37.
There is e.g. also a "Transletor" mentioned on page 140.
On page 136 the authors talk about a function called "Check Number". In the source-code it is called "chceknumber".
Such mistakes are not harmful themselves, they are only disturbing. Much worse is, if such bad spelling leads to factual mistakes: Due to one of such uncorrected mistakes, one can read on page 106 that a 16-bit-signed-short-int takes values from "-32767 to 32767" (instead of the correct "-32768 to 32767", stated only one page later).

You can also read about "Format specifiers in C function[s]" on page 112. There is stated that "%r - String (converts any python object using repr()". Apparently the whole table was copied from a python-manual.

There are also a lot of format inconsistencies. I found the ways "ebp", "Ebp" and "EBP" to name the same register.
The table of file-signatures in the appendix also shows this. Vertical lines vary in their alignment and some of the rows are broken in parts along with the page-breaks.

One last annoying thing is to be found on page 160: The authors missed to insert several links to programs the readers could use in such a way that you can read "(available at)" in the text.

It is also in doubt, whether such a strong focus on Windows-systems as this book has, should not be mentioned in the title or at least the blurb. Side note: One of the authors works for Microsoft.
In this context it would have been desirable to name the chapter about PE-Files "Executable File-Formats" with sub-chapters about PE and ELF (Executable and Linking Format-Files, the format, Linux stores its executable code. It is not a derivate of the PE-format), instead of making ELF a sub-chapter of PE.

All the chapters seem rather short and incomplete.

3 Recommendation

To be fair, a later edition of this book, which was carefully checked again by the authors and a lector, could be quite useful.
The contents of the book are very interesting. Showing at the same time how something works and how people try do break it apart, is a very interesting and straight-forward approach.
The book provides a lot of code samples and explains them pretty good.

But as you could read above, the book is **not** about identifying malicious code. It will also not explain fully to you how to write assembly code or how a PE looks like — if you seek this, you will find better

sources for free on the internet.
The chapter *Reversing Binaries for Identifying Vulnerabilities* might give you a delightful insight on how some attacks and vulnerabilities you should already know, look like in assembly language.
When you have written some assembly code and maybe have tried to reverse-engineer some programs the chapter *Fundamental[s] of Reverse Engineering* might also be interesting, since it shows some anti-debugging and anti-disassembly techniques.
I cannot say that those two chapters are exhausting their topics, but they might give you a starting point for further research.

To conclude: If you already know assembly code, how a PE looks like, how buffer-overflows are exploited and how to disassemble a piece of code, you can read this book. But in that case I doubt that you will learn anything from it. The catchwords from the index should be enough to let you find good information on the internet.

The reviewer is a student of IT-Security at the Ruhr-University of Bochum (Horst Görtz Institute).