FOREWORD

It is an undeniable fact that malware usage is a growing threat to computer security. We see alarming statistics everywhere demonstrating the increase in malware's financial impact, its complexity, and the sheer number of malicious samples. More security researchers than ever, in both industry and academia, are studying malware and publishing research across a wide spectrum of venues, from blogs and industry conferences to academic settings and books dedicated to the subject. These publications cover all kinds of angles: reverse engineering, best practices, methodology, and best-of-breed toolsets.

Thus, a lot of discussions on malware analysis and automation tooling are already taking place, and every day brings more. So you might be wondering: Why another book on the subject? What does this book bring to the table that others haven't?

First and foremost, while this book is about the reverse engineering of advanced—by which I mean *innovative*—malware, it covers all the foundational knowledge about why that piece of code in the malware was possible in the first place. This book explains the inner workings of the different components affected—from the platform's bootup, through the operating system loading to different kernel components, and to the application layer operation, which flows back down into the kernel.

I have found myself more than once explaining that *foundational* coverage is not the same as *basic*—although it does need to extend down to the base, the essential building blocks of computing. And by that measure, this book is about more than just malware. It is a discussion of how computers work, how the modern software stack uses both the basic machine capabilities and the user interfaces. Once you know all that, you start *automagically* understanding how and why things break and how and why they can be abused.

Who better to provide this guidance than authors with a track record of unveiling—on multiple occasions—truly advanced malicious code that pushed the envelope on the state of the art in every case? Add to that the deliberate and laborious effort to connect that experience back to the foundations of computers and the bigger picture, such as how to analyze and understand different problems with similar conceptual characteristics, and it's a no-brainer why this book should be at the top of your reading list.

If the content and methodology chosen more than justify the need for such a book, the next question is why no one took on the challenge of writing one before. I've seen (and had the honor of actively participating in and hopefully contributing to) the evolution of this book, which took several years of constant effort, even with all the raw materials the authors already had. Through that experience, it became clear to me why no one else had tried it before: not only is it hard, but it also requires the right mix of skills (which, given the authors' background, they clearly possess), the right support from the editors (which No Starch offered, working patiently through the editing process and accepting the unavoidable mid-project delays due to the shifting realities of offensive security work), and, last but not least, the enthusiasm of early access readers (who were essential for driving this work toward the finish line).

A lot of this book's focus is on building an understanding of how trust (or lack thereof) is achieved in a modern computer, and how the different layers and transitions between them can be abused to break the assumptions made by the next layer. This highlights, in a unique way, two major problems in implementing security: composition (multiple layers each depending on another's correct behavior to properly function) and assumptions (because the layers must inherently assume the previous one behaves correctly). The authors also share their expertise in the toolsets and approaches used for the uniquely challenging analysis of early boot components and the deeper layers of an operating system. This cross-layer approach alone is worth a book of its own, making this a book within a book. As a reader, I love this two-for-one deal, one that few authors offer to their readers.

My belief about the nature of knowledge is that if you really know something, you can hack it. Using reverse engineering to understand code that hacks a system's usual behavior is an amazing technical feat that often uncovers a lot of knowledge. Being able to learn from professionals with a successful track record in performing this feat—leveraging their understanding, methods, recommendations, and overall expertise—while following along yourself is a unique opportunity. Do not miss it! Go deep; use the supporting materials; practice; engage the community, friends, and even professors (who, I hope, see the value this book brings to the classroom). This is not a book just for reading—it is a book worth studying.

Rodrigo Rubira Branco
(BSDaemon)