foreword

If Miko didn't write this book, someone else would have to. That said, it would be difficult to find someone with Miko's history and experience with chaos engineering to put such a practical approach into writing. His background with distributed systems and particularly the critical and complex systems at Bloomberg, combined with his years of work on PowerfulSeal, give him a unique perspective. Not many people have the time and skill of working in the trenches on chaos engineering at an enterprise level.

This perspective is apparent in Miko's pragmatic approach. Throughout the chapters, we see a recurring theme that ties back to the value proposition of doing chaos engineering in the first place: risk and contract verification, holistic assessment of an entire system, and discovery of emergent properties.

One of the most common questions we hear with respect to chaos engineering is "Is it safe?" The second question is usually "How do I get started with chaos engineering?" Miko brilliantly answers both by including a virtual machine (VM) with all the examples and code used in the book. Anyone with basic knowledge of running an application can ease into common and then more advanced chaos engineering scenarios. Mess something up? No worries! Just turn off the VM and reload a new copy. You can now get started with chaos engineering, and do so safely, as Miko facilitates your learning journey from basic service outages (killing processes) to cache and database issues through OS- and application-level experiments, being mindful of the blast radius all the while.

Along the way, you'll get introduced to more advanced topics in system analysis, like the sections on Berkeley Packet Filter (BPF), `sar`, `strace`, and `tcptop`—even virtual

machines and containers. Beyond just chaos engineering, this book is a broad education in SRE and DevOps practices.

The book provides examples of chaos engineering experiments across the application layer, at the operating system level, into containers, on hardware resources, on the network, and even in a web browser. Each of these areas alone is worthy of an entire chapter, if not book; you get the benefit of exploring the full breadth of possible experiments with an experienced facilitator to guide you through. Miko hits different ways each area can be affected in just the right level of detail to give you confidence to try it yourself in your own stack.

It's all very practical, without glossing over the nuances of understanding technical trade-offs; for example, in chapter 8 Miko weighs the pros and cons of modifying application code directly to enable an experiment (easier, more versatile) versus using another layer of abstraction such as a third-party tool (safer, scales better across contexts). These are the appropriate considerations for a pragmatic and tactical approach to implementing chaos engineering. I can say with confidence that this balance has not been struck in the literature on this subject prior to this book, making it an instant addition to the canon.

If you are chaos-curious, or even if you are well-versed in the history and benefits of chaos engineering, this book will take you step-by-step, safely, into the practice. Following along with the exercises will give you practical experience under your belt, and examples and pop quizzes included in the VM reinforce the takeaway learning. You will emerge with a better understanding of complex systems, how they work, and how they fail. This will, of course, allow you to build, operate, and maintain systems that are less likely to fail. The safest systems are, after all, the most complex ones.

—Casey Rosenthal
Former manager of the Chaos Engineering Team at Netflix
CEO and cofounder of Verica.io