PART I

DATA COLLECTION

Malware detection begins with data collection. All malicious code performs actions on an infected system that deviate from the norm. Therefore, by collecting sufficient data, you can uncover any infection.

Symptoms of digital pathogens often reflect the malware's goals or capabilities. For example, if a computer is infected with adware, you'll likely see browser subversions or hijacked search pages. In the case of a stealthy backdoor, you may observe a listening socket that allows an attacker to remotely control the infected system or its unauthorized network traffic. And any malware that wants to survive a reboot will have to persist, resulting in noticeable filesystem modifications.

In Part I, I discuss how security software could programmatically collect data from a macOS system to detect any digital infections, just as a doctor might when checking whether a human patient is sick. Most malicious code on macOS systems runs as a stand-alone process, so I'll start this section by discussing programmatic methods of querying the system to retrieve a snapshot of all running processes. Then we'll extract information about each process, such as their arguments, hierarchies, loaded libraries, and much more. If any running process is indeed malware, the information we extract here should readily reveal this fact.

Subsequent chapters will bolster our malware detection capabilities by illustrating how to extract other types of data, either from specific items or from the system as a whole. I'll discuss code signing by delving into

mechanisms and APIs to obtain and validate cryptographic code signing signatures. This information can further uncover malware, but equally importantly, it also allows us to ignore trusted items in our hunt for malicious code. I'll also show how to glean important data from Mach-O binaries, the network, and Apple's proprietary Background Task Management database used to manage persistent items.