INTRODUCTION

Welcome to *Designing BSD Rootkits*! This book will introduce you to the fundamentals of programming and developing kernel-mode rootkits under the FreeBSD operating system. Through the "learn by example" method, I'll detail the different techniques that a rootkit can employ so that you can learn what makes up rootkit code at its simplest level. It should be noted that this book does not contain or diagnose any "full-fledged" rootkit code. In fact, most of this book concentrates on *how* to employ a technique, rather than *what* to do with it.

Note that this book has nothing to do with exploit writing or how to gain root access to a system; rather, it is about maintaining root access long after a successful break-in.

What Is a Rootkit?

While there are a few varied definitions of what constitutes a rootkit, for the purpose of this book, a *rootkit* is a set of code that allows someone to control certain aspects of the host operating system without revealing his or her presence. Fundamentally, that's what makes a rootkit—evasion of end user knowledge.

Put more simply, a rootkit is a "kit" that allows a user to maintain "root" access.

Why FreeBSD?

FreeBSD is an advanced, open source operating system; with FreeBSD, you have full, uninhibited access to the kernel source, making it easier to learn systems programming—which is, essentially, what you'll be doing throughout this book.

The Goals of This Book

The primary goal of this book is to expose you to rootkits and rootkit writing. By the time you finish this book, you should "theoretically" be able to rewrite the entire operating system, on the fly. You should also understand the theory and practicality behind rootkit detection and removal.

The secondary goal of this book is to provide you with a practical, hands-on look at parts of the FreeBSD kernel, with the extended goal of inspiring you to explore and hack the rest of it on your own. After all, getting your hands dirty is always the best way to learn.

Who Should Read This Book?

This book is aimed at programmers with an interest in introductory kernel hacking. As such, experience writing kernel code is not required or expected.

To get the most out of this book, you should have a good grasp of the C programming language (i.e., you understand pointers) as well as *x*86 Assembly (AT&T Syntax). You'll also need to have a decent understanding of operating system theory (i.e., you know the difference between a process and a thread).

Contents Overview

This book is (unofficially) divided into three sections. The first section (Chapter 1) is essentially a whirlwind tour of kernel hacking, designed to bring a novice up to speed. The next section (Chapters 2 through 6) covers the gamut of current, popular rootkit techniques (i.e., what you would find in "the wild"); while the last section (Chapter 7) focuses on rootkit detection and removal.

Conventions Used in This Book

Throughout this book, I have used a boldface font in code listings to indicate commands or other text that I have typed in, unless otherwise specifically noted.

Concluding Remarks

Although this book concentrates on the FreeBSD operating system, most (if not all) of the concepts can be applied to other OSes, such as Linux or Windows. In fact, I learned half of the techniques in this book on those very systems.

NOTE *All of the code examples in this book were tested on an IA-32–based computer running FreeBSD 6.0-STABLE.*

1

LOADABLE KERNEL MODULES

The simplest way to introduce code into a running kernel is through a *loadable kernel module (LKM)*, which is a kernel subsystem that can be loaded and unloaded after bootup, allowing a system administrator to dynamically add and remove functionality from a live system. This makes LKMs an ideal platform for kernel-mode rootkits. In fact, the vast majority of modern rootkits are simply LKMs.

NOTE *In FreeBSD 3.0, substantial changes were made to the kernel module subsystem, and the LKM Facility was renamed the Dynamic Kernel Linker (KLD) Facility. Subsequently, the term KLD is commonly used to describe LKMs under FreeBSD.*

In this chapter we'll discuss LKM (that is, KLD) programming within FreeBSD for programmers new to kernel hacking.