Preface

This book is about understanding computer security through experiment. Before now, you probably thought that if your computer was compromised, it was the end of the world. But we are going to show you how to look at the bright side of break-ins and teach you to appreciate the insights to be gained from botnets, worms, and malware. In every incident there is a lesson to be learned. Once you know about the many different kinds of honeypots, you can turn the tables on Internet-born attackers. This book discusses a vast range of deployment scenarios for honeypots, ranging from tracking botnets to capturing malware. We also encourage you to take the perspective of adversaries by analyzing how attackers might go about detecting your countermeasures. But first let us set the context appropriately.

Computer networks connect hundreds of thousands of computer systems across the world. We know the sum of all these networks as the Internet. Originally designed for research and military use, the Internet became enormously popular after Tim Berners-Lee invented the HyperText Transfer Protocol (HTTP) in 1990 and created the World Wide Web as we know it. As more of us started using the Net, almost all of our social problems transferred into the electronic realm as well. For example, it was human curiosity that created the first Internet worm.[1] Scanning networks for the number of installed computers or their respective configuration is another sign of our curiosity. In fact, receiving a constant stream of network probes is nowadays considered normal and expected. Unfortunately, many of these activities are no longer benign. Darker elements of society have figured out that the Internet provides new opportunities to turn a quick profit. Underground activities

1. Technically, the first network worm was created in 1982 by Shoch and Hupp of Xerox's PARC, who developed worms such as the *Vampire* worm, which would seek out underutilized computers and have them solve complex computing tasks [81]. However, in most minds, Internet worms started with Morris, who, among many other contributions, also invented the *buffer overflow*.

range from sending millions of spam e-mails, identity theft, and credit card fraud to extortion via distributed denial of service attacks.

As the Internet becomes increasingly popular, its security is also more important for keeping our electronic world healthy and functioning. Yet, despite decades of research and experience, we are still unable to make secure computer systems or even measure their security. Exploitation of newly discovered vulnerabilities often catches us by surprise. Exploit automation and massively global scanning for vulnerabilities make it easy for adversaries to compromise computer systems as soon as they can locate its weaknesses [91].

To learn which vulnerabilities are being used by adversaries (and they might even be some of which we are unaware), we could install a computer systems on a network and then observe what happens to it. If the system serves no other purpose, then every attempt to contact it seems suspect. If the system is attacked, we have learned something new. We call such a system a *honeypot*. Its compromise allows us to study which vulnerability was used to break into it or what an adversary does once he gained complete control over it. A honeypot can be any kind of computing system. It may run any operating system and any number of services. The services we configure determine the attack vectors open to an adversary.

In this book, we often talk about nefarious computer users who want to break into our honeypots. Many readers might expect that we would call these computer users *hackers*, a term adapted and distorted beyond recognition by the press. However, the authors prefer the traditional definition of the word: *A hacker is a person who finds clever technical solutions to problems*. Although there is no shortage of good hackers out there, the supply of people who attempt and succeed to break into computer systems is much larger. We refer to them as *attackers* or *adversaries*.

So far, we have claimed that honeypots allow us to study adversaries and gain insight into their motivations and techniques, but now we will prove it to you with a real case study.

A Real Case

This case tells the story of an actual compromise and what we learned from the adversaries. Our honeypot was closely monitored, and we could observe every single step the adversary took on our system. This incident started on April 3, when our Red Hat 8.0-based honeypot was compromised due to weak SSH passwords. The adversary got access to both a user and the root account. She probably considered herself very lucky to have gained access to a high-speed university network. What she did not know was that we had intentionally installed guessable passwords.

(*Evil grin*.) Actually, this kind of attack is quite common. If you run an SSH server yourself, just take a look at its log files.

Using our log files and other information gathered on the honeypot, it was easy to reconstruct the series of events that took place. As in many movies, the attack took place in the middle of the night. Originating from a university host in Norway, the adversary initiated an attack against the honeypot's SSH server shortly after midnight. Her automatic tools cycled through many thousand different user names and passwords before she got lucky and guessed the *root* password. With complete and unlimited access to our system, the adversary, arriving from an Italian IP address this time, downloaded several tools from different web servers to facilitate her malicious actions. Among these tools was an SSH scanner, an IRC client, and a root kit. Not surprisingly, our adversary used the SSH scanner to find more Internet systems with weak passwords. In addition to the root kit, a back door was installed to allow the adversary to come back at any time without anyone noticing. When the adversary was downloading the movie *Get Rich Or Die Tryin'* (Spanish), we decided that things had gone on long enough, and we shut down the honeypot.

Attack Timeline

Our in-depth investigation produced the following timeline of events:

- `00:23:07` AM: After several minutes of scanning, the adversary manages to log in for the first time, utilizing the *guest* account. Not satisfied, the adversary continued to guess passwords for further accounts.
- `00:35:53` AM: Jackpot! Successful login in as *root*. However, despite getting *root*, the password guessing continues — a strong indicator that we are looking at a completely automated attack.
- `00:51:24` AM: The user *guest* logs in but logs off a few seconds later. We assume that the adversary manually verified the correctness of the automatically guessed user names and passwords.
- `00:52:44` AM: The user *root* logs in, but this time from the IP 83.103.xxx.xxx. While logged in, three new users are created. All of them with group and uid 0, the identity of the system administrator.
- `00:54:08` AM: The intruder logs in using the *guest* account and changes the password for this account. She then starts downloading a file with her tools of trade from a remote web server.
- `00:54:29` AM: The file completes downloading. It contains an SSH scanner, shell scripts to start it, and two dictionary files to generate user names and

passwords. Ten seconds later, files *xyz* and *1* are downloaded as well. File *xyz* is another dictionary file for the previously mentioned SSH scanner. File *1* is a simple shell script, which facilitates the proper execution of the SSH scanner.

- `00:54:53` AM: The adversary initiates an SSH scan against the IP range 66.252.*. The scan finishes after about three minutes. Don't worry: Our control mechanisms prevented any harm to other machines.
- `00:58:18` AM: The *guest*, *george*, and *root* users log out.
- `01:24:34` PM: User *george* logs back in, this time from IP address 151.81.xxx.xxx. The adversary switches to the *root* account and starts downloading a file called *90*. A quick analysis reveals that it is some kind of kernel modifying program, probably a root kit.
- `02:22:43` PM: Another file is downloaded, and the adversary also changes the *root* password. The new file contains a modified SSH server that listens on port 3209 and another SSH scanner. From now on, all connections to the honeypot were made through the freshly installed back door.
- `02:23:32` PM: The adversary establishes a connection to the mail server `mta238.mail.re2.yahoo.com` but fails to send an e-mail due to improper formatting of the `MAIL FROM` header.
- `02:31:17` PM: The adversary downloads `mirkforce.tgz`, which contains a modified IRC client. A moment later, she executes the IRC client and connects to an IRC server running at 194.109.xxx.xxx.
- `02:58:04` PM: The adversary attempts to download the movie *Get Rich Or Die Tryin'* via HTTP.
- `03:02:05` PM: A `whois` query is executed for the domains `bogdan.mine.nu` and `pytycu.ro`.
- `04:46:49` PM: The adversary starts scanning the IP range 125.240.* for more machines with weak SSH passwords. She stops scanning at about 05:01:16 PM.
- `04:58:37` PM: She downloads the compressed file `scanjapan.tar` to the `/tmp` directory. The file contains another SSH scanner with Japanese user name and password dictionaries.
- `05:30:29` PM: It was time to go home and have a beer, so we shut down the honeypot.

Once the incident was over, we had plenty of time to analyze what really happened. We saved copies of all tools involved and were able to determine their purpose in detail. For example, the installed root kit was called *SucKIT* and has

been described in detail in *Phrack*, issue 58 [78]. SucKIT is installed by modifying kernel memory directly via `/dev/kmem` and does not require any support for loadable kernel modules. Among other things, SucKIT provides a password-protected remote access shell capable of bypassing firewall rules. It supports process, file, and connection hiding, and survives across reboots as well.

There is much more to be learned, and we have dedicated an entire chapter to case studies like this.

Target Audience

We wrote this book to appeal to a broad spectrum of readers. For the less experienced who are seeking an introduction to the world of honeypots, this book provides sufficient background and examples to set up and deploy honeypots even if you have never done so before. For the experienced reader, this book functions as a reference but should still reveal new aspects of honeypots and their deployment. Besides providing solid foundations for a wide range of honeypot technologies, we are looking at the future of honeypots and hope to stimulate you with new ideas that will still be useful years from now.

Road Map to the Book

Although you are more than welcome to read the chapters in almost any order, here is a chapter overview and some suggestions about the order that you may find helpful.

- Chapter 1 provides a background on Internet protocols, honeypots in general, and useful networking tools. This chapter is intended as a starting point for readers who are just learning about this topic.
- Chapters 2 and 3 present honeypot fundamentals important for understanding the rest of the book. We introduce the two prevalent honeypot types: *high-interaction* and *low-interaction*. Low-interaction honeypots emulate services or operating systems, whereas high-interaction honeypots provide real systems and services for an adversary to interact with.
- Chapters 4 and 5 focus on *Honeyd*, a popular open source honeypot framework that allows you to set up and run hundreds of virtual honeypots on just a single physical machine. The virtual honeypots can be configured to mimic many different operating systems and services, allowing you to simulate arbitrary network configurations.

- Chapter 6 presents different approaches for capturing malware, such as worms and bots, using honeypots. Because botnets and worms are significant risks to today's Internet, the honeypots presented in this chapter will help you learn more about these threats.
- Chapter 7 discusses different approaches for creating high-performance honeypots that combine technologies from both low- and high-interaction honeypots. These *hybrid* systems are capable of running honeypots on over 60,000 different IP addresses.
- In Chapter 8, we turn the tables, and instead of waiting to be attacked, we present the concept of *client honeypots* that actively seek out dangerous places on the Internet to be compromised.
- Taking the viewpoint of an attacker, Chapter 9 discusses how to detect the presence of honeypots and circumvent logging. This is what adversaries do to make the life of honeypot operators harder. By understanding their technologies, we are better prepared to defend against them.
- In Chapter 10, we present several case studies and discuss what we learned from deploying virtual honeypots in the real world. For each honeypot that was compromised, we present a detailed analysis of the attackers' steps and their tools.
- *Botnets*, networks of compromised machines under remote control of an attacker, are one of the biggest threats on the Internet today. Chapter 11 presents details on botnets and shows what kind of information can be learned about them with the help of honeypots.
- Because honeypots often capture malware, Chapter 12 introduces *CWSandbox*, a tool that helps you to automatically analyze these binaries by creating behavior profiles for each of them. We provide an overview of CWSandbox and examine a sample malware report in great detail.

If you are unfamiliar with honeypots and want to learn the basics before delving into more complex topics, we strongly encourage you to start with Chapters 1–3. These chapters will help you get an understanding of what the methodology is about and what results you can expect from deploying honeypots.

Once you know the basics, you can dive right into the more advanced topics of Honeyd in Chapters 4 and 5. Chapter 6 discusses capturing autonomously spreading malware like worms and bots. Closely related to Chapter 6 are Chapter 11 on botnets and Chapter 12 on malware analysis. But you can also learn more about hybrid approaches in Chapter 7 and the new concept of

client-side honeypots in Chapter 8. Chapters 9 and 10 are also rather independent: The former introduces several ways to detect the presence of honeypots, a risk you should always have in mind. The latter presents several case studies that show you which kind of information you can learn with honeypots based on real-world examples.

Although the chapters are organized to build on each other and can be read in their original order, most chapters can be understood by themselves once you are familiar with the basics concepts. If any chapter looks particularly interesting to you, don't hesitate to skip forward and read it.

Prerequisites

When reading this book, familiarity with the basic concepts of network security will prove helpful. We expect you to be familiar with the terms *firewall* and *intrusion detection system* (IDS), but it is not necessary for you to have extensive knowledge in any of these areas. Our first chapter lays the basic background for most of what is required to understand the rest of the book. We also make extensive use of references for anyone who would like to get more details on topics we discuss.

Since many honeypot solutions are designed to run on Linux or BSD variants, it is helpful to have some basic understanding of these operating systems. However, even if you are an avid Windows user, you can install a virtual machine to experiment with these operating systems. Doing so by itself teaches many of the principles that underly honeypot technologies. That way, you can better understand the tools we introduce and also experiment with them yourself. We often give step-by-step guidance on how to install and configure a specific solution and point you to further references. So even with only some background, you should be able to learn more about the fascinating topic of virtual honeypots.