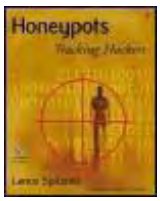



**Honeypots: Tracking Hackers**

By Lance Spitzner

Publisher : Addison Wesley
Pub Date : September 13, 2002
ISBN : 0-321-10895-7
Pages : 480

## RIPPED BY “BUSTER”

## Foreword: Giving the Hackers a Kick Where It Hurts

I'm an unabashed Lance Spitzner fan. This is the guy whose cell phone voice message says, *"I'm busy geeking out right now, but leave a message, and I'll get back to you as soon as I can.*" I don't know when he actually stops geeking out long enough to sleep. I sometimes wonder if there are actually two of him. His enthusiasm for what he's doing bleeds over into all aspects of his life. Ideas for cool stuff erupt from him like a volcano and swirl around him, sucking in casual bystanders and students alike. It's somewhat intimidating to share a stage with him at a conference. He makes just about everyone else look uninteresting and tepid by comparison. Lance is a man who loves what he's doing, and what he loves doing is tracking hackers, sharing that information, and making a difference.

A lot of people like to reserve the term "hacker" for the techno-elite computer hobbyist—those media darlings often described as "misunderstood whiz-kids" or similar nonsense. One of the great by-products of Lance's work with honeypots and honeynets is that he's helped give us a much clearer picture of the hacker in action: often technically unsophisticated kids playing around with technologies they barely understand. In *Know Your Enemy* the Honeynet Project demonstrated just how active and unskilled most hackers are. What's that—you don't believe it? Set up your own honeypot or honeynet and see for yourself. This book gives you the necessary tools and concepts to do it!

I think it's a great thing for the security community that Lance has written this book. In the past, the hackers roamed our networks with supreme confidence in their anonymity. They take advantage of systems they've compromised to chat with their buddies safely or to launch attacks against other systems and sites without fear of detection. Now, however, they may pause to wonder if their bases of operation are safe—whether they're actually planning their attacks and deploying their tricks under a microscope.

Honeypots are going to become a critical weapon in the good guys' arsenals. They don't catch only the lame hackers. Sometimes they catch the new tools and are able to reduce their effectiveness in the wild by letting security practitioners quickly react before they become widespread. They don't catch just the script kiddies outside your firewall but the hackers who work for your own company. They don't catch just unimportant stuff; sometimes they catch industrial spies. They can be time- and effort-consuming to set up and operate, but they're fun, instructive, and a terrific way for a good guy to gain an education on computer forensics in a real-world, low-risk environment.

Right now there are about a half-dozen commercial honeypot products on the market. Lance covers several of them in this book, as well as "homemade" honeypots and honeynets, focusing on how they operate, their value, how to implement them, and their respective advantages. I predict that within one year, there will be dozens of commercial honeypots. Within two years, there will be a hundred. This is all good news for the good guys because it'll make it easier for us to deploy honeypots and harder for the bad guys to recognize and avoid them all. When you're trying to defend against an unknown new form of attack, the best defense is an unknown new form of defense. Honeypots will keep the hackers on their toes and, I predict, will do a lot to shatter their sense of invulnerability. This book is a great place to start learning about the currently available solutions.

In this book Lance also tackles the confusion surrounding the legality of honeypots. Lots of practitioners I've talked to are scared to dabble in honeypots because they're afraid it may be considered entrapment or somehow

illegal. It's probably a good idea to read the chapter on legal issues twice. It may suprise you. Welcome to the cutting edge of technology, where innovation happens and the law is slow to catch up to new concepts. Meanwhile, you can bet that with renewed concerns about state-sponsored industrial espionage and terrorism the "big boys" will be setting up honeypots of their own. I'd hate to be a script kiddy who chose to launch his next attack from a CIA honeypot system! When the big boys come into the honeypot arena, you can bet that they'll make sure it's legal.

The sheer variety and options for mischief with honeypots are staggering. (There is even a honeypot for spam e-mails.) You can use the concepts in this book to deploy just about any kind of honeypot you can imagine. Would you like to build a honeypot for collecting software pirates? I don't think that's been done yet. How about a honeypot that measures which hacking tools are most popular by tracking hits against an index page? I don't think that's been done yet, either. The possibilities are endless, and I found it difficult to read this book without thinking, "What if . . . ?" over and over again.

I hope you enjoy this book and I hope it inspires you to exercise your own creativity and learn what the bad guys are up to and then share it with the security community. Then follow Lance's lead, and make a difference.

## Preface

It began as an innocent probe. A strange IP address was examining an unused service on my system. In this case, a computer based in Korea was attempting to connect to a rpc service on my computer. There is no reason why anyone would want to access this service, especially someone in Korea. Something was definitely up. Immediately following the probe, my Intrusion Detection System screamed an alert: An exploit had just been launched. My system was under assault! Seconds after the attack, an intruder broke into my computer, executed several commands, and took total control of the system. My computer had just been hacked! I was elated! I could not have been happier.

Welcome to the exciting world of honeypots, where we turn the tables on the bad guys. Most of the security books you read today cover a variety of concepts and technologies, but almost all of them are about keeping blackhats out. This book is different: It is about keeping the bad guys in—about building computers you *want* to be hacked. Traditionally, security has been purely defensive. There has been little an organization could do to take the initiative and challenge the bad guys. Honeypots change the rules. They are a technology that allows organizations to take the offensive.

Honeypots come in a variety of shapes and sizes—everything from a simple Windows system emulating a few services to an entire network of productions systems waiting to be hacked. Honeypots also have a variety of values—everything from a burglar alarm that detects an intruder to a research tool that can be used to study the motives of the blackhat community. Honeypots are unique in that they are not a single tool that solves a specific problem. Instead, they are a highly flexible technology that can fulfill a variety of different roles. It is up to you how you want to use and deploy these technologies.

In this book, we explain what a honeypot is, how it works, and the different values this unique technology can have. We then go into detail on six different honeypot technologies. We explain one step at a time how these honeypot solutions work, discuss their advantages and disadvantages, and show you what a real attack looks like to each honeypot. Finally, we cover deployment and maintenance issues of honeypots. The goal of this book is not to just give you an understanding of honeypot concepts and architecture but to provide you with the skills and experience to deploy the best honeypot solutions for your environment. The examples in the book are based on real-world experiences, and almost all of the attacks discussed actually happened. You will see the blackhat community at their best, and some of them at their worst. Best of all, you will arm yourself with the skills and knowledge to track these attackers and learn about them on your own.

I have been using honeypots for many years, and I find them absolutely fascinating. They are an exciting technology that not only teaches you a great deal about blackhats but also teaches you about yourself and security in general. I hope you enjoy this book as much as I have enjoyed writing and learning about honeypot technologies.

## Audience

This book is intended for the security professional. Anyone involved in protecting or securing computer resources will find this resource valuable. It is the first publication dedicated to honeypot technologies, a tool that more and more computer security professionals will want to take advantage of once they understand its power and flexibility.

Due to honeypots' unique capabilities, other individuals and organizations will be extremely interested in this book. Military organizations can apply these technologies to Cyberwarfare. Universities and security research organizations will find tremendous value in the material concerning research honeypots. Intelligence organizations can apply this book to intelligence and counterintelligence activities. Members of law enforcement can use this material for the capturing of criminal activities. Legal professionals will find Chapter 15 to be one of the first definitive resources concerning the legal issues of honeypots.

## Web Site

This book has a Web site dedicated to it. The purpose of the Web site is to keep this material updated. If any discrepancies or mistakes are found in the book, the Web site will have updates and corrections. For example, if any of the URLs in the book have been changed or removed, the Web site will provide the updated links. Also, new technologies are always being developed and deployed. You should periodically visit the Web site to stay current with the latest in honeypot technologies.

http://www.tracking-hackers.com/book/

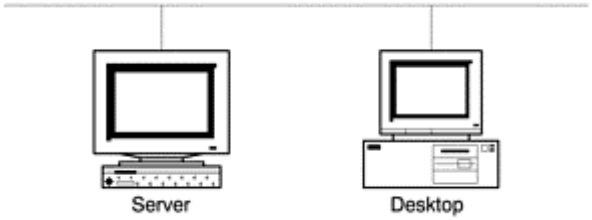
## References

Each chapter ends with a references section. The purpose is to provide you with resources to gain additional information about topics discussed in the book. Examples of references include Web sites that focus on securing operating systems and books that specialize in forensic analysis.

## Network Diagrams

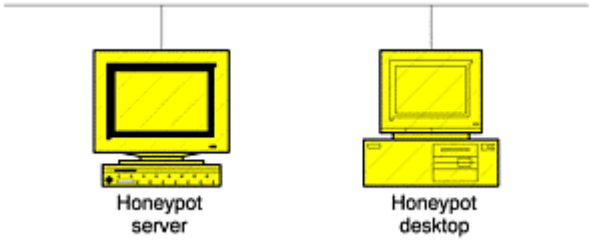
This book contains network diagrams demonstrating the deployment of honeypots. These diagrams show both production systems and honeypots deployed together within a networked environment. All production systems and honeypots are standardized, so you can easily tell them apart. All production systems are simple black-and-white computer objects, as in Figure A. These are systems you do not want to be hacked.

**Figure A. Two production systems deployed on a network**




In contrast, all honeypots can easily be identified by shading and the lines going through the system, as in Figure B.

**Figure B. Two honeypots deployed on a network**




# Chapter 1. The Sting: My Fascination with Honeypots

J4ck was excited. He had just gained access to a powerful new hacking tool. With this he would be able to attack and compromise hundreds, if not thousands, of systems all over the world. It was going to be a very exciting night indeed. If he could hack enough computers tonight, he could prove just how "elite" he was. J4ck knew that most underground hackers considered him just a beginner. By hacking into as many computers as possible, he would prove to them all that he was much better than they thought.

J4ck had just gotten off IRC (Internet Relay Chat) with J1ll, a member of his hacking group. IRC is an online chat mechanism that allowed J4ck to instantly talk on the Internet to any of his hacking buddies anywhere in the world. It is similar to the telephone conference calls that most corporations use today, but IRC is free. All you need is a connection to the Internet. IRC also allows people to exchange files, such as exploits or attack code, over the Internet. This is where J4ck started and where he learned almost all of his hacking skills. He could join different chat rooms, called *channels*, to meet different people and discover the latest exploits. New attacks and vulnerabilities were always being discovered, and IRC was an effective way to instantly distribute that information.

IRC was another reason why he wanted to hack into as many systems as possible tonight. The more systems he controlled, the more BOTs he could have. BOTs, short for robots, were automated programs installed on the hacked computers. These BOTs maintained a virtual presence on IRC channels, acting out any instructions programmed into them by the attacker. The more computers J4ck hacked into, the more BOTs he could deploy. The more BOTs he deployed, the greater his control on IRC channels. Attackers were always trying to knock each other off IRC—for example, by launching Denial of Service attacks against each other. J4ck had to protect himself against these attacks and then be prepared to launch them himself. It was a cyberwar on the Internet, and the more computers he hacked tonight, the greater his arsenal.

J1ll had just explained to him how to use the new rpc.statd exploit, giving him access to any unpatched Linux server running rpc.statd. Linux is an extremely common form of Unix operating system used around the world. This was incredible! There had to be thousands of such computers out there ready to be exploited. J1ll had also transferred the new exploit to him already compiled. J4ck's only concern was that he had received a precompiled version of the tool and not the source code. That meant he could not review the binary for any malicious code. He did not trust most of the other blackhats out in cyberspace: They would just as happily Trojan a fellow blackhat as anyone else on the Internet.

That reminded him: It was a good thing he had the latest Linux rootkit. *Rootkits* are toolkits designed to reprogram a computer once it is hacked. They execute a variety of functions, including wiping system log files, implementing backdoors, and Trojaning various files or even the running kernel. Once reprogrammed, the computer will lie to the administrator. The system admin may "ask" the computer if it has been attacked or compromised, and the computer will lie and say it hasn't. Also, the reprogrammed computer would now have backdoors implanted on the system, giving the attacker a variety of ways to get back into the system.

J4ck, like many blackhats, had tools to patch and secure a system once it was hacked. J4ck knew that if he did not secure the computer he had just compromised, his buddies or other attackers would also find the system and exploit it. They were out there just as actively scanning and attacking systems as he was. J4ck could not trust his buddies, so he secured the system to keep them out.

J4ck had no real idea of how the new rpc.statd exploit worked—only what commands he had to use to run the tool. But that was all he needed. He already had an older exploit script that he could easily modify. When run, the script would take the input of what networks were to be attacked and then pass that information to the actual exploit. To make the new exploit work, all J4ck had to do was take the script and replace the name of the old

exploit with the name of his new rpc.statd exploit. He repeated this process every time a new exploit was released. The script did all the other work for him, such as scanning for and hacking into vulnerable systems, uploading the rootkit when the system was compromised, reprogramming the attacked system, and maintaining a log file of all the systems successfully compromised.

Once his script was updated, the process was very simple. Since the tool was fully automated, all he had to do was upload the tool set to a computer he had already hacked, launch the tool, and then come back several hours later to see what systems the tool had broken into. In fact, J4ck never used his own computer for attacking other computers or communicating with his h4x0r buddies (h4x0r is a term hackers use for a skilled hacker; it is slang for "hacker"). J4ck thanked J1ll for the new tool, signed off, and with a sinister grin began launching his new weapon against an entire country.

What you have just read is not fiction; it really happened. Every action and conversation of these two attackers was captured and recorded. Their names and identities have been sanitized, but their activities are real. What J4ck and J1ll did not realize is that everything they had just accomplished was watched and captured by a group of security professionals because one of the computers they had hacked into was a honeypot. The Sting had begun[1]!

## The Lure of Honeypots

Since I first heard about the concept, I've been fascinated with honeypot technologies. A honeypot is very different from most traditional security mechanisms. It's a security resource whose value lies in being probed, attacked, or compromised. The idea of building and deploying a computer meant to be hacked has an aura of mystery and excitement to it. The first time I learned about honeypots I was working as a system administrator for a Chicago-based consulting company. Late one night I was working with several fellow employees on rebuilding our file server, which kept crashing due to hard drive failure. Making conversation at that late hour, one of our senior administrators mentioned the idea of honeypots. New to the world of security, I had never heard of this before—a computer you *wanted* to be hacked. How exciting! It sounded almost like some type of spy movie, where the CIA agent infiltrates a foreign country, learning the enemy's greatest and darkest secrets. I was instantly hooked and just had to learn more.

What I find so exciting about the concept is we are turning the tables on the bad guys. The underground world of hacking, of breaking into and taking over a computer, has typically been a difficult one to understand. Similar to other forms of crime, little has been known about how the attackers operate, what tools they use, how they learn to hack, and what motivates them to attack. Honeypots give us an opportunity to peer into this world. By watching attackers break into and control a honeypot, we learn how these individuals operate and why.

Honeypots give us another advantage: the ability to take the offensive. Traditionally, the attacker has always had the initiative. They control whom they attack, when, and how. All we can do in the security community is defend: build security measures, prevent the bad guy from getting in, and then detect whenever those preventive measures fail. As any good military strategist will tell you, the secret to a good defense is a good offense. But how do the good guys take the initiative in cyberspace? Security administrators can't go randomly attacking every system that probes them. We would end up taking down the Internet, not to mention the liability issues involved. Organizations have always been limited on how they can take the battle to the attacker. It is because of this problem that I am so excited about honeypots. They give us the advantage by giving us control: we allow the bad guys to attack them. It is because of issues like these that I could not wait to jump in and start working with honeypots.

I began playing with honeypots in 1999 and soon discovered that there was little information on how to build and use them. In contrast to security tools such as encryption, firewalls, and intrusion detection systems—about which much was being or had already been written—there was little documentation that defined honeypots. So, I decided the best way to learn what a honeypot is was to build one, make a lot of mistakes, and learn from those mistakes. (Making mistakes is something I'm very good at.)

## How I Got Started with Honeypots

So, how do you build a honeypot? One advantage to having no documentation was at least I couldn't do it wrong. Since there were no rules on what a honeypot should be or should look like, whatever I tried was a step in the right direction.

My research began with the only publicly available honeypot at that time: Fred Cohen's The Deception Toolkit[2]. This a suite of tools written in PERL and C that emulate a variety of services. Installed on a Unix system, DTK, as it is commonly called, is used to both detect attacks and deceive the attacker. I tried out the DTK and found it extremely useful for a first crack at a honeypot. However, I felt limited by the fact that it emulated known vulnerabilities, and supplied only a limited amount of information. There is no operating system for the attacker to work with when dealing with DTK—only emulated services. This restricts the information you can gain about the bad guys. Nevertheless, DTK had sparked my interest.

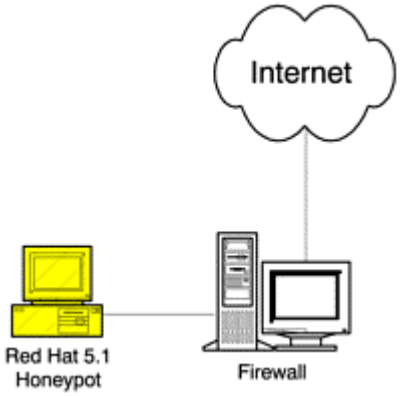
Next I tried another honeypot solution: CyberCop Sting, one of the first commercial honeypots developed for organizations. I was fortunate to know the original developer of the product, Alfred Huger, and I obtained an evaluation copy. What impressed me about this solution was that it was easy to deploy and could emulate a variety of systems at the same time. You simply installed CyberCop Sting on a Windows NT system, configured a few options in a simple file, and let it go. You instantly had a network with various Linux, Solaris, and NT systems. A single honeypot could emulate an entire network of computers. Once again, however, I found CyberCop Sting to be limited, since attackers could only connect to certain ports and read the banners. For example, they could Telnet to an emulated system, such as Solaris, and receive a login banner. The attacker could then repeatedly attempt to login, and each attempt was logged by the honeypot. However, the attacker would never get access because there was no real operating system to access. The emulated service only allowed for attempted logins, so the interaction was severely limited. I was interested in learning how the bad guys operate, so I needed a honeypot with which they could truly interact.

Thus began the idea of developing my own honeypot that could emulate a variety of services. It would be similar to DTK, where the honeypot could interact with the attacker, but also similar to CyberCop Sting, where different honeypots existed simultaneously. I was looking for a system that combined the best of both worlds: a high level of interaction with multiple systems.

Unfortunately, I had two problems: (1) I am extremely impatient, and (2) I dislike writing code. If I were to code my own honeypot that emulated various services, it would take a great deal of effort and most likely be a miserable failure. The combination of my impatience and lack of coding skills forced me to consider another solution. Rather than emulate systems, I would just build standard systems, put them behind a firewall, and see what happened. I loved this idea for several reasons. First, the solution was instant, satisfying my incredible impatience. Second, the solution involved no coding but instead used a firewall to control outbound connections. I'm horrible with coding, but I feel very comfortable with firewalls. Third, and best of all, this solution gave the bad guys a complete operating system to interact with. I could learn a great deal more about the bad guys without merely emulating services. Once they hacked the honeypot, they could make accounts, upload rootkits, and do any other activity attackers commonly do. This was the perfect honeypot for me.

In February 1999, I put the plan in motion. I had a dedicated Internet connection, a simple ISDN line, sitting on my wife's dining room table (much to her chagrin). I then built a default installation of Red Hat 5.1 (a commercial version of Linux), put it behind a firewall, sat back, and waited (see Figure 1-1). The firewall would let anybody in but would let nobody out (kind of like a roach motel). This was the opposite of what a firewall was designed to do, but it satisfied my purposes. The intent was to let anyone hack into the computer, but once they were in, they could not use it to go back out and hack other systems. Systems administrators tend to frown on such activity.

**Figure 1-1. Diagram of the very first honeypot I ever deployed: A default installation of Red Hat Linux 5.1 behind a firewall.**




On February 28, 1999, at 20:15 I put the honeypot online, wondering if anyone would even find the system, let alone attack it. The results were astonishing: The honeypot was a smashing success—and a dismal failure.

Within 15 minutes of my connecting the honeypot to the Internet, an attacker identified, probed, and exploited it. I barely had time to turn around before an attacker had complete access to my computer. I was astonished. How did an attacker know I was putting this system online, and how did he get in so quickly? The only reason I caught the attack was that the lights on my ISDN router suddenly began flashing, indicating extensive network activity. Seeing this activity, I quickly checked my system logs, and sure enough, there was a bad guy on my honeypot. I was thrilled, but I had no idea what to do next.

My wife, on the other hand, knew exactly what to do. "Pull the plug! Pull the plug!" she yelled in horror when I told her a hacker had penetrated our network. While I calmly reassured her that I knew what I was doing, our attacker quickly discovered that something was not right about his newly compromised system. He could easily connect to it *from* the Internet, but he could not connect from the system back out *to* the Internet. The firewall was blocking all outbound connection, protecting the honeypot from attacking other people. The attacker was not happy about this and soon guessed he was on a honeypot. He then proceeded to wipe the entire hard drive clean, deleting every file on the system. I lost all of his keystrokes, system activity, and even his rootkit, which he had just uploaded to the honeypot. He wiped away all traces of his activity, getting clean away. Upon realizing this, I turned to my wife and told her what happened. She smiled at me and replied, "I told you you should have pulled the plug!"

As I said, this first attempt at a honeypot was both a smashing success and a dismal failure. It was a success because the concept worked. One can put a system online, protect it with a firewall, and sit back and wait. Sooner or later the bad guys will find and attack these systems. You can then use this to capture their activity and learn about the blackhat community. However, the honeypot was also a failure: the attacker discovered the honeypot's true identity and wiped the hard drive clean. This was because the firewall was too constrictive. I should have allowed some type of outbound activity. That way, the attacker might not have discovered he was on a honeypot, allowing me to capture a great deal more data. The concept worked, but I failed to capture any information on the attack. No matter—I was hooked. I decided that honeypots were definitely a cool concept with incredible potential, one that I wanted to invest a great deal more time and effort researching.

Over the next several years I spent extensive time and effort researching, developing, and testing a variety of honeypot solutions. This research developed my skills in a variety of technologies, including logging, packet analysis, firewall design, and intrusion detection systems. To correctly implement a honeypot you must have a solid understanding of a variety of technologies. What was fascinating was not only how these technologies worked but how they could fail. For example, Intrusion Detection Systems may fail to detect an attack, computers may not log attacks when they should, and misconfigured firewalls may permit traffic that should have been denied. By thoroughly understanding these failures, one can better prevent them from happening again.

Besides developing my security skills, an even greater benefit has come from researching and deploying honeypots: my increased knowledge about the enemy. Every time an attacker compromised a honeypot, I learned something new. One time I learned how they implemented and used rootkits; another time I learned how they communicated among each other. Honeypots have taught me what some of the most common threats are, how they operate, and why. These experiences have played a valuable role as I try to help others secure their own

environments. I truly hope that as you read and learn about honeypots, you will not only gain a better understanding of security technologies but about the enemy as well.

## Perceptions and Misconceptions of Honeypots

Historically, honeypots have had a clouded reputation, but undeservedly so. Although the concept of honeypots is more than a decade old, honeypots have not been adopted until recently. Many misconceptions may explain this delay between concept and implementation. People feel that if an attacker is lured into a honeypot, the attacker will only be infuriated by the deception and retaliate against the organization. Others feel that honeypots require too much work: building advanced jail environments, recoding binaries, or developing sophisticated kernel module. Or they fear that if the honeypot is misconfigured or not maintained properly, attackers will have access to resources they otherwise would not have. Many doubt that honeypots have any true value to security. To add to the confusion, security administrators have received many different definitions of what a honeypot is or what it can do.

I believe a large part of this confusion, misunderstanding, and doubt about honeypot technologies comes from a lack of research and documentation. Everyone has his own interpretation of what a honeypot is and its value, and this makes it difficult for people to agree on what they can and cannot do with it. In addition, sometimes honeypots are deployed incorrectly or for the wrong reasons, which diminishes their value and reputation.

Recently, however, there has been an increased appreciation for and understanding of honeypots. There is a perceived value in honeypots—what they can and, just as importantly, cannot do. This can be seen in the more frequent whitepapers on honeypots, the commercial honeypots being released (discussed later in this book), and discussions about honeypots on public mail lists. There is a new appreciation for honeypots as a valuable security tool. I personally believe honeypots are an excellent tool in the security arsenal, that add value to the security community.

## Summary

There is still a great deal of confusion about what a honeypot is and its impact on the security community. I hope this book changes that. This book defines honeypots, their value, the different types, and their deployment. Honeypots also have some unique issues, such as risks and legal issues. It is my hope to piece together all these issues and give you a comprehensive resource on honeypots and honeypot-related technologies.

As I said in the beginning of this chapter, I have always been and continue to be a big fan of honeypots. I love the idea of turning the tables on the bad guys and building a system you invite them to attack. The excitement of not knowing what will happen next or what I will learn keeps me motivated. I also believe honeypots are an incredible tool that can teach you not only about security technologies but also about the enemy. I hope you find this just as exciting as I do.

# Chapter 2. The Threat: Tools, Tactics, and Motives of Attackers

Before we start talking about how honeypots work and the problems they solve, we should first examine the problem: the attacker. By understanding who our threat is and how he operates, we will better understand the value and function of honeypots. The type of attacker you are attempting to identify, detect, or capture will also dictate the type of honeypot you build and how you deploy it. This chapter helps you recognize the enemies you are up against. Most of the information discussed in this chapter was obtained using honeypot technologies.

## Script Kiddies and Advanced Blackhats

In general, there are two types of attackers: the kind who want to compromise as many systems as possible and the kind who want to compromise a specific system or systems of high value. It does not matter if these threats are coming from the outside, such as the Internet, or from the inside, such as a disgruntled employee. Most threats tend to fall into one of these two categories.

The first type doesn't care if the computer belongs to a major organization or the average homeowner. His goal is to hack as many systems as possible with as little effort as possible. These attackers focus on targets of

opportunity—the easy kill. Often they are called script kiddies, since they usually depend on scripted attacks. Sometimes these attackers have certain requirements, such as hacking systems with a fast connection to the Internet or a large hard drive for storing files. In general, however, all they care about are numbers. They tend to be less sophisticated, but they are far more numerous, representing the vast majority of probes, scans, and attacks you see today.

The second type of attacker focuses on a few systems of high value. These individuals are most likely highly experienced and knowledgeable attackers—the advanced blackhat. Their attack is usually financially or nationally motivated, such as state-sponsored terrorism. They have a specific target they want to compromise, and they focus only on that one. Though less common and fewer in number, these attackers are far more dangerous due to their advanced skill level. Not only can they penetrate highly secured systems, their actions are difficult to detect and trace. Advanced blackhats make little "noise" when attacking systems, and they excel at covering their tracks. Even if you have been successfully attacked by such a skilled blackhat, you may never even be aware of it.

## Everyone Is a Target

Many people have the misconception that they are not a target, that the bad guys will never find them, let alone attack them. They believe that since their system has no perceived value, no one would want to hack into it. Or they believe that since they have a dynamically assigned IP address, no one would find or attack them. Nothing could be farther from the truth. You may feel that your Windows 95 desktop has no value, but attackers can find great benefit in your system, such as using your computer to attack another system or using your hard drive to store all of the stolen credit card information the hacker has collected.

Research and statistics from a variety of organizations prove how active the threat is in cyberspace. The following statistics were gathered through the use of honeypots [1].

- At the end of the year 2000, the life expectancy of a default installation of Red Hat 6.2, a commonly used version of Linux, was less than 72 hours.
- One of the fastest recorded times a honeypot was compromised was 15 minutes. This means that within 15 minutes of being connected to the Internet, the system was found, probed, attacked, and successfully exploited by an attacker. The record for capturing a worm was under 90 seconds.
- During an 11-month period (April 2000–March 2001), there was a 100 percent increase in unique scans and an almost 900 percent increase in Intrusion Detection Alerts, based on Snort [2].
- In the beginning of 2002, a home network was scanned on average by 31 different systems a day.

What makes these statistics so frightening is that they were captured with honeypots using simple network connections. Nothing was done to advertise these honeypots or lure attackers. These honeypots represented the most basic systems, those that few people consider valuable. This activity is not just limited to attacks captured by honeypots. Every year CERT, a federally funded security research institute, releases statistics on incidents. The year 2001 saw a 100 percent increase in reported incidents, from 21,756 to 52,658 reported attacks[3]. Unfortunately, the situation is only becoming worse with the availability of far more powerful and fully automated attacking tools, described later in this chapter. Lets take a look at some of the more common methodologies and tools the bad guys use. This will help us understand why this threat is so active and why almost any system is a target.

## Methods of Attackers

As we discussed earlier, there are two general categories of attackers. Each group has their own method: The first type focuses on targets of opportunity, and the second focuses on targets of choice.

Both threats are extremely dangerous. Highly skilled blackhats focus on high-value targets. Because of their high skill level, they often are successful in compromising their targets. However, do not discount the threat of the unskilled attackers, those who concentrate on targets of opportunity. What these individuals lack in skill or finesse, they more than make up for in numbers. While there are no statistics to determine specific percentages, I

would estimate that 80 to 90 percent of attacks today are accomplished by the "easy kill" variety. Since these attackers are by far the more common threat, we will discuss them first.

## Targets of Opportunity

Much of the blackhat community is lazy. Their goal is to hack into as many computers as possible, with the least effort on their part. Their motives may vary, but the goal is the same: to own as many systems as possible. As we mentioned earlier, these tend to be the less sophisticated attackers, often called script kiddies. Their method is simple: focus on a single vulnerability, then scan as many systems as possible for that vulnerability. Persistence, not advanced technical skills, is how these attackers successfully break into a system.

Years ago, attacking computers was difficult and time consuming. An attacker had to go through a series of complex and technically challenging steps. First, they had to identify a vulnerability within an operating system or application. This is not an easy task. It requires extensive knowledge of how operating systems work, such as memory management, kernel mechanisms, and file systems functionality. To identify vulnerabilities in an application, an attacker would have to learn how an application operated and interacted with both the input and output of information. It could take days, weeks, even months to identify vulnerabilities.

After a vulnerability was identified, an attacker would have to develop a tool to exploit it. This requires extensive coding skills, potentially in several different computer languages. After the exploit was developed, the attacker had to find vulnerable systems. Often one scanning tool was used to find systems that were accessible on the Internet, using such functionality as an ICMP ping or a full TCP connection. These tools were used to develop a database of systems that were accessible. Then the attacker had to determine what services existed on the reachable systems—that is, what was actually running on the targets. Further, the attacker had to determine if any of these services were vulnerable. The next step was launching the exploit against the victim, hacking into and gaining control of the system. Finally, various other tools (often called rootkits) were used to take over and maintain control of a compromised system.

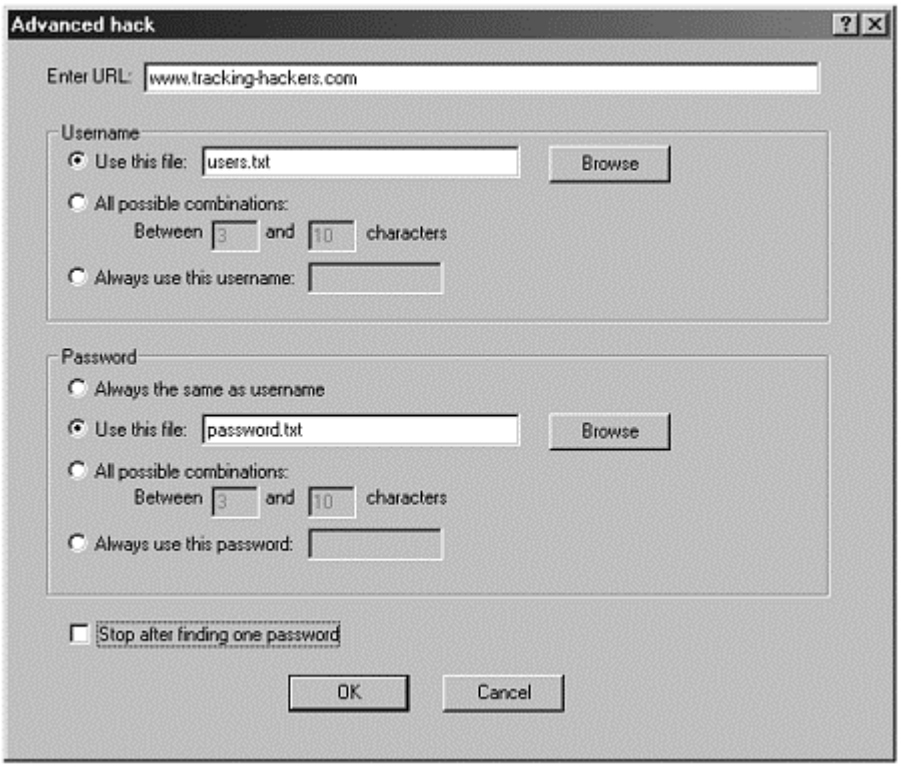
Each of the steps just described required the development of a unique tool, and using all those tools took a lot of time and resources. Once launched, the tools were often manually intensive, requiring a great deal of work from an experienced attacker.

Today, the story is dramatically different. With almost no technical skills or knowledge, anyone can simply download tools from the Internet that do all the work for them. Sometimes these tools combine all of the activity just described into a fully automated weapon that only needs to be pointed at certain systems, or even entire networks, and then launched with the click of a button. An attacker simply downloads these tools, follows the instructions, launches the attacks, and happily hacks her way into hundreds or even thousands of systems. These tools are rapidly spreading across the Internet, giving access to thousands of attackers. What used to be a highly complex development process is now extremely simple.

Attackers can download the automated tools from a variety of resources or exchange them with their friends. IRC (Internet Relay Chat) and the World Wide Web enable blackhats to instantly share new attack tools around the world. Then they simply learn the command line syntax for the tool. For attackers who are unfamiliar with command line syntax, a variety of tools have been designed for Windows with point-and-click capabilities. Some of the exploits even come with well-written, step-by-step instructions.

An example is the point-and-click tool wwwhack, which uses brute force to access password-protected Web sites (Figure 2-1). Without such a tool, an attacker would have to guess someone's password. With enough guesses, sooner or later the attacker will find someone's account and get into the password-protected site. However, it takes a great deal of time and effort to repeatedly enter different names and passwords and then track each attempt. It is much easier and more efficient to automate the attack process and wrap a GUI around the tool. This way anyone familiar with a mouse can use the tool to attack Web sites. With wwwhack, an attacker merely has to launch the GUI, enter the name of the site they want to force their way into, and launch the attack. Tools like this make it easy for even your grandmother to hack into sites.

**Figure 2-1. The hacking tool wwwhack has a simple point-n-click interface.**




An even greater threat is the automation of the attacker's arsenal. These tools, often called auto-rooters, work by focusing on a single vulnerability, usually one that has become common knowledge. The tool scans for any vulnerable system and then automatically hacks into it. It is unlikely that the vulnerability was discovered by the attacker, but was probably identified by a highly experienced individual, such as an advanced blackhat or perhaps a legitimate security professional researching vulnerabilities. Only a small percentage of individuals can identify and develop exploit code, but once the code is accessible on the Internet, anyone can apply it. Auto-rooters simply take the exploit code and automate the entire scan, probe, and attack sequence. What used to be an intensively manual process is now simplified to executing a single tool. Their effectiveness is based on their persistence. Even though only 1 percent of the Internet may be vulnerable, statistics are in the attackers' favor. For every one million systems they scan, they can potentially compromise 10,000 systems. And there are hundreds of millions of systems on the Internet for the attackers to probe. All our friend J4ck needs to do is launch his automated weapon before he goes to bed and wake up the next morning to see how many computers he has broken into.

The very randomness of these tools is what makes them so dangerous. An attacker launches the tool, designating a network, and the tool then randomly probes and attacks every system in the entire network. Even if your system has no value, these tools will find and take it over. If you have a dynamically allocated IP address that changes every five minutes, these tools can scan entire network blocks and find and attack you. Even dial-up users who are connected for only 20 minutes a day will be attacked sooner or later.

### An Example Auto-rooter: Luckroot

To better understand these automated tools, lets take a look at one. In February 2001, the automated tool *luckroot* was captured in the wild by a honeypot. Luckroot takes advantage of the Linux rpc.statdx vulnerability (the same exploit J4ck was using in Chapter 1). The tool was designed to probe and attack any system running rpc.statd. A group of Romanian hackers compromised a Linux honeypot using this tool. Then they uploaded the tool and tried to use it against other networks, attempting to scan almost a million systems in several hours. Fortunately, the honeypot successfully blocked these scan attempts.

Luckroot is an auto-rooter, an automated tool designed to attack Unix computers. Not only does this tool automatically find vulnerable systems, it will attack them and take them over. The entire hacking process is automated for the attacker. This toolkit consists of three separate tools: luckscan, luckstatdx, and luckgo.

*Luckscan* is the scanner; it probes a given IP address range and determines if the remote system is running rpc. The scanner makes no attempt to identify the remote operating system, determine the version of rpc.statd, or discover if it is even vulnerable. The tool only determines if the service is running. Any additional functionality would require additional coding and time for scanning, which serves no real purpose. If any system is running the service, vulnerable or not, the exploit will be launched, which will determine if the system is vulnerable anyway.

Any system identified as running rpc.statd is then attacked by *luckstatdx*, the actual exploit. If the exploit is successful, it will download a rootkit from a predetermined Web site to take control and reprogram the compromised system.

The third tool, *luckgo*, is a shell script that calls on the scanning tool luckscan and exploiting tool luckstatdx, automating the entire process. All an attacker has to do is download this toolkit, and launch the shell script luckgo.

Figure 2-2 shows an actual attacker's keystrokes (highlighted in bold) captured by the honeypot. After gaining access to the system, the attacker first downloads the toolkit luckroot. Once downloaded, he opens the package and immediately begins attacking networks. The numbers after the command luckgo are the first two octets of a class B network that have over 64,000 systems in each network. Also, note how simple the tool is, requiring only the launch of the tool and the network designator. Very little skill is required to launch this very powerful weapon. It took this attacker only 60 seconds to download the auto-rooter, unpackage the arsenal, and begin attacking millions of systems.

**Figure 2-2 Attacker's keystrokes captured by a honeypot. Here we see an attacker launch an auto-rooter against entire class B networks, each network having more than 64,000 IP addresses.**

```
Jan 8 18:48:12 PID=1246 UID=0 lynx www.becys.org/LUCKROOT.TAR
Jan 8 18:48:31 PID=1246 UID=0 y
Jan 8 18:48:45 PID=1246 UID=0 tar -xvfz LUCKROOT.TAR
Jan 8 18:48:59 PID=1246 UID=0 tar -xzvf Lu
Jan 8 18:49:01 PID=1246 UID=0 tar -xzvf L
Jan 8 18:49:03 PID=1246 UID=0 tar -xzvf LUCKROOT.TAR
Jan 8 18:49:06 PID=1246 UID=0 cd luckroot
Jan 8 18:49:13 PID=1246 UID=0 ./luckgo 216 210
Jan 8 18:51:07 PID=1246 UID=0 ./luckgo 200 120
Jan 8 18:51:43 PID=1246 UID=0 ./luckgo 64 120
Jan 8 18:52:00 PID=1246 UID=0 ./luckgo 216 200
Jan 8 18:52:06 PID=1246 UID=0 ./luckgo 216 200
Jan 8 18:54:37 PID=1246 UID=0 ./luckgo 200 120
Jan 8 18:55:26 PID=1246 UID=0 ./luckgo 63 1
Jan 8 18:56:06 PID=1246 UID=0 ./luckgo 216 10
Jan 8 19:06:04 PID=1246 UID=0 ./luckgo 210 120
Jan 8 19:07:03 PID=1246 UID=0 ./luckgo 64 1
```

If one of your IP addresses falls into the network range being scanned, you will be probed and most likely attacked. In this specific case, the attacker attempted to probe 16 different Class B networks, totaling over one million systems. For a detailed analysis of the tool and the actual source code, refer to "Scan of the Month 13" on the CD-ROM.

What makes an auto-rooter so dangerous is that exploits can be interchanged. The auto-rooter we just saw had three components, one of which is the actual exploit luckstatdx. When a new exploit is discovered and a tool developed to attack the vulnerability, existing auto-rooters merely have to be modified for the new attack, which generally does not take a great deal of work. The attack process is much simpler now; the automated weapons are already available on the Internet for anyone to use. An individual does not need to know why a new attack works. She only needs to know the command line syntax of a new attack tool, modify one or two lines of script within her auto-rooter, and they have an updated tool that can potentially compromise thousands of systems a night.

In 2002 a new variant of auto-rooters was discovered in the wild: mass-rooters. These tools operate on the same principle as auto-rooters. They automatically probe for, exploit, and take over compromised systems. But whereas auto-rooters focus on a single vulnerability, mass-rooters focus on multiple vulnerabilities. For example, a mass-rooter could scan for and exploit systems running vulnerable versions of FTP, rpc.statd, SSH, and BIND. It could also have the knowledge to compromise different versions of the same service on different operating systems. This exponentially increases the tool's capabilities and the attacker's chances of compromising systems. Mass-rooters not only demonstrate how easy it is for an attacker to compromise a system, but also verify how attackers are continually advancing their arsenal.

Figure 2-3 shows the output of a mass-rooter developed by a group called TESO. This tool can break into 34 different versions and distributions of wu-ftpd, expotentially increasing its effectiveness over auto-rooters.

**Figure 2-3 The output of a mass-rooter, demonstrating all the different systems it can break into**

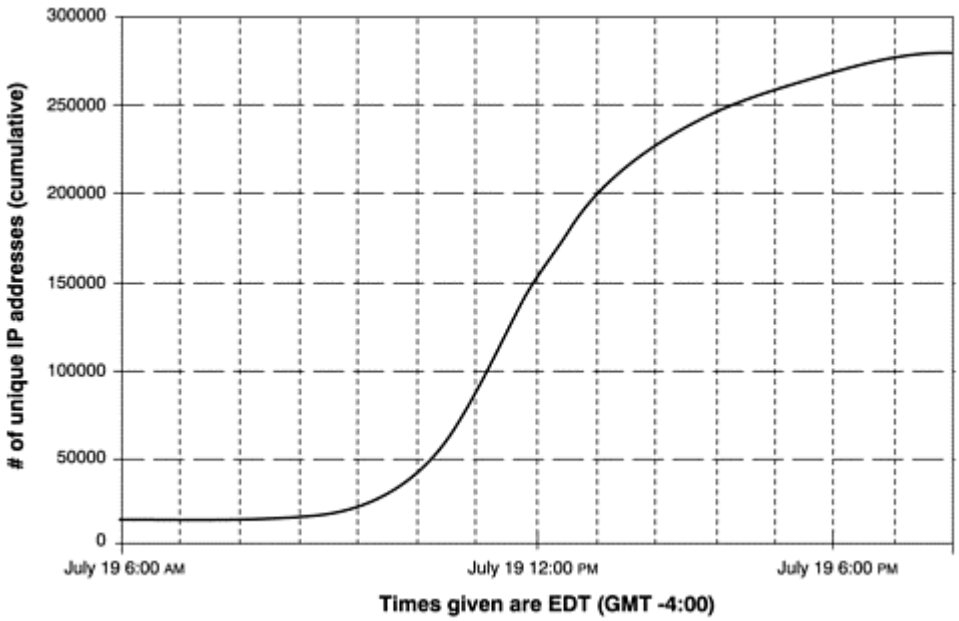
```
[system]$ ./wu-sploit -t0
7350wurm - x86/linux wuftpd <= 2.6.1 remote root (version 0.2.2)
team teso (thx bnuts, tomas, synnergy.net !).
Compiled for MnM 01/12/2001..pr0t!
num . description
----+-----------------------------------------------------
  1 | Caldera eDesktop|eServer|OpenLinux 2.3 update
      [wu-ftpd-2.6.1-13OL.i386.rpm]
  2 | Debian potato [wu-ftpd_2.6.0-3.deb]
  3 | Debian potato [wu-ftpd_2.6.0-5.1.deb]
  4 | Debian potato [wu-ftpd_2.6.0-5.3.deb]
  5 | Debian sid [wu-ftpd_2.6.1-5_i386.deb]
  6 | Immunix 6.2 (Cartman) [wu-ftpd-2.6.0-3_StackGuard.rpm]
  7 | Immunix 7.0 (Stolichnaya) [wu-ftpd-2.6.1-6_imnx_2.rpm]
  8 | Mandrake 6.0|6.1|7.0|7.1 update
      [wu-ftpd-2.6.1-8.6mdk.i586.rpm]
  9 | Mandrake 7.2 update [wu-ftpd-2.6.1-8.3mdk.i586.rpm]
 10 | Mandrake 8.1 [wu-ftpd-2.6.1-11mdk.i586.rpm]
 11 | RedHat 5.0|5.1 update [wu-ftpd-2.4.2b18-2.1.i386.rpm]
 12 | RedHat 5.2 (Apollo) [wu-ftpd-2.4.2b18-2.i386.rpm]
 13 | RedHat 5.2 update [wu-ftpd-2.6.0-2.5.x.i386.rpm]
 14 | RedHat 6.? [wu-ftpd-2.6.0-1.i386.rpm]
 15 | RedHat 6.0|6.1|6.2 update [wu-ftpd-2.6.0-14.6x.i386.rpm]
 16 | RedHat 6.1 (Cartman) [wu-ftpd-2.5.0-9.rpm]
 17 | RedHat 6.2 (Zoot) [wu-ftpd-2.6.0-3.i386.rpm]
 18 | RedHat 7.0 (Guinness) [wu-ftpd-2.6.1-6.i386.rpm]
 19 | RedHat 7.1 (Seawolf) [wu-ftpd-2.6.1-16.rpm]
 20 | RedHat 7.2 (Enigma) [wu-ftpd-2.6.1-18.i386.rpm]
 21 | SuSE 6.0|6.1 update [wuftpd-2.6.0-151.i386.rpm]
 22 | SuSE 6.0|6.1 update wu-2.4.2 [wuftpd-2.6.0-151.i386.rpm]
 23 | SuSE 6.2 update [wu-ftpd-2.6.0-1.i386.rpm]
 24 | SuSE 6.2 update [wuftpd-2.6.0-121.i386.rpm]
 25 | SuSE 6.2 update wu-2.4.2 [wuftpd-2.6.0-121.i386.rpm]
 26 | SuSE 7.0 [wuftpd.rpm]
 27 | SuSE 7.0 wu-2.4.2 [wuftpd.rpm]
 28 | SuSE 7.1 [wuftpd.rpm]
 29 | SuSE 7.1 wu-2.4.2 [wuftpd.rpm]
 30 | SuSE 7.2 [wuftpd.rpm]
 31 | SuSE 7.2 wu-2.4.2 [wuftpd.rpm]
 32 | SuSE 7.3 [wuftpd.rpm]
 33 | SuSE 7.3 wu-2.4.2 [wuftpd.rpm]
 34 | Slackware 7.1
```

### Worms: CodeRed

Along with auto-rooters and mass-rooters comes an even more dangerous weapon, the worm: Worms are similar to auto-rooters in that they automatically search out vulnerable systems, exploit them, and take over them. However, worms take the attack process one step further: They self-replicate. Once a worm has compromised and taken over a system, it then begins scanning again, looking for new victims. This means a single system can compromise one hundred systems, those one hundred systems can each compromise one hundred more systems, and so on, exponentially growing and attacking systems. This propagation method can spread extremely fast, giving administrators little time to react and ravaging entire organizations.

One of the fastest worms to have ever spread was the CodeRed worm, released in July 2001. CodeRed targeted Microsoft systems, specifically the Internet Information Server (IIS) Web server, susceptible to a specific buffer overflow. This worm was able to successfully compromise over 250,000 systems in less than nine hours [4]. These attacks were so devastating that entire organizations were unable to function. Figure 2-4 shows its incredible growth.

**Figure 2-4. Graph from CERT of IP addresses compromised by the CodeRed worm (Data for July 13, 2001 as represented to the CERT/CC; from: Incident data for CERT #36881. Used with permission.)**

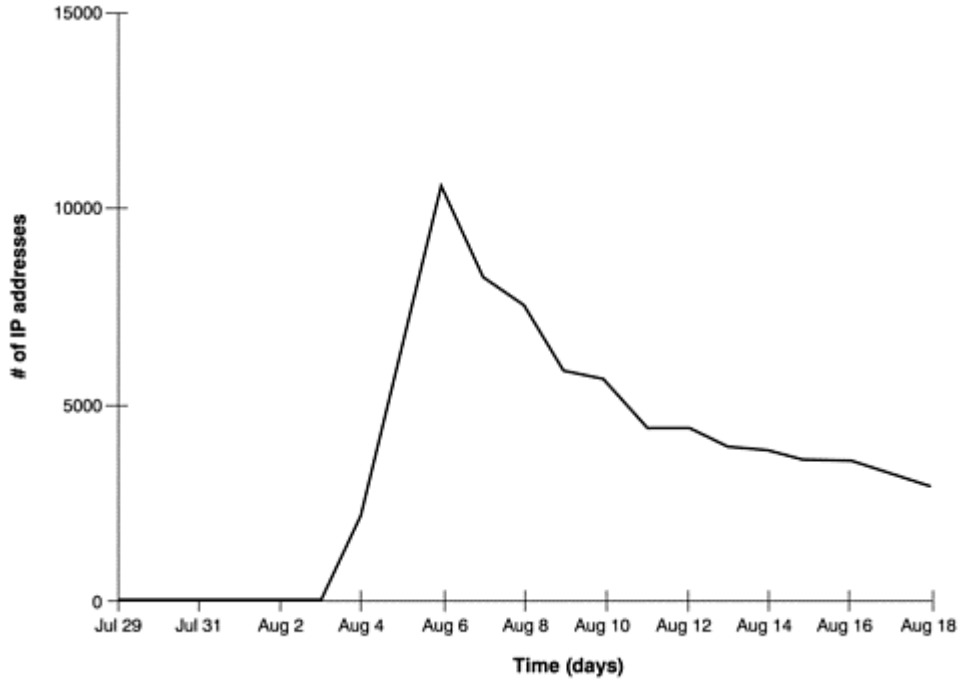



Several weeks after the CodeRed attacks, a new worm was released: CodeRed II. This worm targeted the same vulnerabilities as the original CodeRed, but it had a vastly improved propagation method. Traditionally, worms such as CodeRed randomly select and scan networks on the Internet. This method is not effective because there are large IP ranges that are currently not used, whereas other networks have high concentrations of system. A worm may spend hours scanning large networks that have no active systems, wasting time and resources. CodeRed II changed all that with a far more effective scanning mechanism.

Instead of randomly selecting networks to scan, CodeRed II first targeted systems on the local networks. Based on its scanning algorithim, there is a far greater chance that the worm will find vulnerable systems on the same networks it hacked into, as opposed to randomly selecting a network that may not have any systems. This increases the chance it will find vulnerable systems and thus increases its infection rate. Securityfocus.com[5] and eEye Digital Security [6] did an excellent analysis of the CodeRed II worm. They captured the worm using a honeypot and then completed a forensic analysis of the worm, including identifying its propagation method. You can find the complete analysis by securityfocus.com in the CD-ROM.

Figure 2-5 demonstrates just how fast the CodeRed II worm can spread. CodeRed II did not infect as many systems as the CodeRed worm because it targeted the same vulnerability, and the majority of systems were already compromised. However, CodeRed II was extremely devastating once it infiltrated an organization, since it quickly compromised all the systems of the same local network.

**Figure 2-5. This diagram, taken from securityfocus.com's analysis, illustrates how the CodeRed II worms improved propagation algorithim dramatically increases its infection rate.**




Worms will only continue to mutate and improve. One of the most interseting worms released is the Nimda worm. Most worms, such as the CodeRed or CodeRed II worm, scan for and exploit a single vulnerability, but Nimda (Admin spelled backwards) uses at least five different methods for propagation. Increasing the number of vulnerabilities the worm probed for and exploited resulted in increased propagation capabilities, making it a far more dangerous weapon. This tactic is similar to that of a mass-rooter. CERT [7] identified the following five propagation methods used by Nimda.

1. From client to client via e-mail
2. From client to client via open network shares
3. From Web server to client via browsing of compromised Web sites
4. From client to Web server via active scanning for and exploitation of various Microsoft IIS 4.0/5.0 directory traversal vulnerabilities (VU#111677 and CA-2001-12)
5. From client to Web server via scanning for the back doors left behind by the "CodeRed II" (IN-2001-09) and "sadmind/IIS" (CA-2001-11) worms

Months after the CodeRed, CodeRed II, and Nimda worms were released, they are still actively scanning the Internet. My collection of honeypots from a home network picked up the activity shown in Figure 2-6 on a single day. Of the 39 different systems that scanned my honeypots in a single day, 32 of them, or 82 percent, were scanning for http-related vulnerabilities.

**Figure 2-6 Scans detected by a honeypot in a single day. Each entry represents a different system probing the honeypot.**

| Date | Time | IP of Attacker | Proto | Src Pt | Dst Pt |
|---|---|---|---|---|---|
| 01/11/05 | 00:12:03 | 216.232.130.32 | tcp | 3765 | http |
| 01/11/05 | 00:25:13 | 216.140.131.115 | tcp | 1202 | http |
| 01/11/05 | 00:46:23 | 202.85.165.108 | tcp | 2046 | sunrpc |
| 01/11/05 | 01:19:09 | 216.161.236.121 | tcp | 1278 | http |

```
01/11/05  01:47:58      216.78.152.78      tcp     3729    http
01/11/05  01:53:26      216.79.219.137     tcp     3033    http
01/11/05  02:28:09      216.160.69.157     tcp     3144    http
01/11/05  02:35:12      216.191.169.200    tcp     1477    http
01/11/05  02:45:11      216.231.43.128     tcp     4852    http
01/11/05  03:33:24      216.187.238.114    tcp     4519    http
01/11/05  04:07:04      216.80.147.224     tcp     1346    Http
01/11/05  05:17:23      216.151.90.150     tcp     2400    http
01/11/05  05:48:24      216.205.68.202     tcp     1801    http
01/11/05  06:38:44      216.102.57.26      tcp     2663    http
01/11/05  07:55:21      216.221.101.90     tcp     3335    http
01/11/05  08:53:45      80.11.196.88       tcp     3627    ftp
01/11/05  10:02:55      216.210.187.196    tcp     2893    http
01/11/05  11:06:06      216.237.145.204    tcp     1823    http
01/11/05  11:26:41      216.160.123.225    tcp     3224    http
01/11/05  11:30:56      24.129.104.196     tcp     2983    http
01/11/05  11:30:57      216.54.180.59      tcp     4118    http
01/11/05  12:17:18      216.96.96.189      tcp     3559    http
01/11/05  12:35:27      216.138.83.234     tcp     4559    http
01/11/05  12:54:19      216.126.66.46      tcp     3982    http
01/11/05  13:36:35      216.236.145.171    tcp     4046    http
01/11/05  14:48:20      216.35.163.169     tcp     3112    http
01/11/05  15:32:30      216.102.228.12     tcp     4151    http
01/11/05  15:53:48      216.80.74.144      tcp     4862    http
01/11/05  15:56:15      216.143.129.235    tcp     3946    http
01/11/05  15:58:42      63.50.219.123      udp     smb     smb
01/11/05  16:38:50      213.93.128.42      tcp     SSH     SSH
01/11/05  17:12:44      216.17.4.68        tcp     63304   http
01/11/05  18:10:33      216.232.179.242    udp     smb     smb
01/11/05  18:18:31      216.0.53.250       tcp     1241    http
01/11/05  18:55:32      216.31.154.133     tcp     3236    http
01/11/05  20:15:25      216.80.152.231     tcp     1355    http
01/11/05  20:32:49      216.244.164.138    tcp     1708    domain
01/11/05  20:33:08      216.80.74.190      tcp     1901    http
```

The development of multiple propagation methods demonstrates that worms will continue to be a growing and constantly changing threat that can strike and propagate quickly, infecting a large percentage of the Internet within hours.

Of even greater concern is the fact that the worm code has already spread among the blackhat community. To release a new worm, all an attacker has to do to is modify the exploit mechanism and insert it into an existing worm code, similar to what we have already seen with auto-rooters.

While they are extremely destructive, worms like CodeRed, CodeRed II, and Nimda attacked known vulnerabilities. Patches existed for months before the worms came out, and the only systems successfully attacked were those that were not patched. Once the worms were identified, vulnerable systems only needed to download and install the existing patches to protect against the worm. But consider the damage one of these worms could accomplish if it used an unknown attack for which no patch existed. By the time the patch was released, most of the Internet would most likely be compromised, with devastating results. The code is already out on the Internet, so it is only a matter of time.

## Targets of Choice

While script kiddies and automated attacks represent the largest percentage of attackers, the smaller, more dangerous percentage of attackers are the skilled ones that don't want anyone to know about their existence. These advanced blackhats do not release their tools. They only attack and compromise systems of high value, systems of choice. When these attackers are successful, they do not tell the world about it. Instead, they silently infiltrate organizations, collecting information, users accounts, and access to critical resources. Often

organizations have no idea that they have been compromised. Advanced attackers can spend months, even years, within a compromised organization without anyone finding out.

These attackers are interested in a variety of targets. It could be an online banking system, where the attacker is after the database containing millions of credit cards. It could be a case of corporate espionage, where the attacker is attempting to infiltrate a car manufacturer and obtain research designs of future cars. Or it can be as sinister as a foreign government attempting to access highly confidential government secrets, potentially compromising the security of a country.

These individuals are highly trained and experienced and they are far more difficult to detect than script kiddies. Even after they have successfully penetrated an organization, they will take advanced steps to ensure that their presence or activity cannot be detected. Very little is known about these attackers. Unlike unskilled attackers, advanced blackhats do not share the same tools or techniques. Each one tends to develop his own skills, methods, and tool sets specialized for specific activities. As such, when the tools and methods of one advanced attacker are discovered, the information gained may not apply to other advanced blackhats.

But this is not to say that advanced attackers never use the tools or techniques used by unskilled attackers. For example, advanced blackhats may hide their attacks by attacking multiple systems, creating multiple hopping points, and hiding their tracks. Let's say an attacker based in the United States, specifically Chicago, wanted to attack a computer in the Federal Reserve in Washington, D.C. The goal is to gain access to a critical banking source and potentially transfer funds from one account to another. Such an attack is extremely dangerous, so an attacker would naturally want to hide his tracks. A skilled hacker would not log in on a local computer in Chicago and directly attack the system in D.C. That trail would be far too easy to follow. So a skilled attacker deliberately creates a long and sordid trail.

For example, our attacker in Chicago could first attack a system in Brazil. From there he compromises a system in Korea, then Romania, and finally attacks the system in Washington, D.C. This distorted trail helps conceal the attacker. To follow the trail from the hacked D.C computer, investigators must first trace the attack to Romania, then Korea, then Brazil, then finally Chicago. Not only is this technically very difficult but there are other challenges. First, notice how every computer in this series of hops represents a country with a different language. To trace this attacker you have to find someone who speaks Romanian, Korean, and Portuguese. Also, each country has a different and unique legal system, requiring different rules and regulations to follow such an individual. Last, each country represents a different time zone, making communications and coordination that much more difficult.

Also, the attacker may have targeted home systems, as opposed to major servers belonging to big corporations. Think about it—how much logging does your Windows 98 desktop do? Almost none. If an attacker uses such a system as a hopping point, his activity will most likely not be logged, making it extremely difficult to trace him.

All of these issues demonstrate the fact that regardless of who you are and where you are located, any system can be a target. Almost everyone knows that major corporations with systems of great value are constantly under attack. What most people do not realize is that to attackers, every system has value, including a desktop at home. It's a war going on in cyberspace, and that war is affecting everyone, including the computer in your living room.

## Motives of Attackers

Recognizing attackers motivations can be of considerable assistance in understanding threats. To help understand motivational issues, counterintelligence agencies used the acronym MICE, for Money, Ideology, Compromise (caught doing something you don't want to be caught doing), and Ego. An adaptation of this acronym, developed by Dr. Max Kilger, MEECES, can be applied to the motivations driving computer attackers. In short, MEECES stands for Money, Ego, Entertainment, Cause (basic ideology), Entrance to a social group, and Status. Following are just some of the many different reasons why an attacker would target and attempt to compromise your systems. Almost all of these motives have been confirmed using honeypots.

### Denial of Service

DoS attacks are those designed to take out the computer systems or networks of a victim. This is commonly done by flooding the intended target (such as a Web server) with a barrage of network traffic. The more traffic that is thrown at a victim, the more effective the attack. Attackers will often compromise hundreds, if not thousands, of

systems to be used for attacks. The more computers they own, the more traffic they can launch at a target. Many blackhats use Denial of Service attacks to take out other blackhats. One example is IRC wars, where one individual attempts to knock out another individual from an IRC channel, using DoS attacks.

### BOTs

BOTs are automated robots that act on behalf of an individual in a preprogrammed fashion. They are most commonly used to maintain control of IRC. The more computers one hacks into, the more BOTs one can launch, and the more one can control specific IRC channels. Using many BOTs protects individuals from losing control of an IRC from Denial of Service attacks.

### Credit Cards

Hacked computers have become a form of currency. Blackhats will trade their hacked accounts for stolen credit cards. The more computers one hacks into, the more money-making potential. This behavior was documented by the Honeynet Project in their paper "Know Your Enemy: Motives." [8]

### Bragging Rights

Many blackhat organizations are meritocracies: groups where your status is based on your merits, your skill. To elevate your status, you must prove your technical skills, often by breaking into other sites. The more sites you can break into, the greater your status. We often see individuals modifying Web sites after compromising systems as a way of bragging to the world about their technical skills and attempting to improve their status.

### CPU Cycles

Some worms have been developed to take over the CPU cycles of client systems to win contests. The more computers the worm infects, the greater the use of combined CPU cycles. The more computers and processing power the attacker takes over, the greater the chances of winning the contest. This motivation is documented in the paper "Know Your Enemy: Worms at War." [9]

### Corporate Espionage

Organizations may attempt to breach the security of their competitors to gain a competitive advantage. This is a common motive of the more advanced blackhats because it involves financial gain.

### Political Motives

Attacks can be politically motivated. One such example was GFORCE following the terrorist attacks of September 11, 2001. This Pakistani-based hacker group targeted the United States and Great Britain by hacking into government computers and posting messages threatening to "hit major U.S. military and major British Web sites" and "very high confidential U.S. data that will be given to the right authorities of Al-Qaeda." [10]

Often these motives can be combined. In the following conversation, captured by a honeypot, we see two hackers (J4ck and J1LL) discussing how they can make money by "packeting," (slang for Denial of Service attacks).

```
J4ck: why don't you start charging for packet attacks?
J4ck: give me x amount and I'll take blah blah offline for this amount
of
      time.
J1LL:  it was illegal last I checked.
J4ck: heh, then everything you do is illegal. Why not make money off of
it?
J4ck: I know plenty of people that'd pay exorbitent amounts for
packeting.
```

## Adapting and Changing Threats

The attackers' abilities to constantly change and improve have become truly dangerous. The blackhats themselves may not be getting better, but their tools definitely are improving.

The greatest risk the security community has identified is the growing trend of automated tools. These tools allow untrained individuals around the world to probe and hack into literally thousands of computers in a single night. This threat is exponentially dangerous as thousands of attackers may be launching the same tools at the same time from locations around the world. If you put an IP stack on a coffeepot, these automated tools will find and attack them. Once released, worms can mutate faster than the security community can react. It may take organizations several days to identify a worm and the vulnerability it targets and then protect against that attack. However, during that same time frame, a new variant of the worm can be released, focusing on new vulnerabilities or propagation methods.

Blackhats have also adapted their tools to use encryption. Until late 1990s, most tools used by attackers were cleartext based, such as Telnet, FTP, http, and cleartext configuration files. This helped organizations to monitor their network traffic and detect attacks. Times have changed. A large percentage of attackers are now using encryption, such as SSH or encrypted configuration files. Now organizations are blind at the network level. They cannot monitor their network traffic. Attackers often even install encryption functionality in systems that do not already have that capability. This limits an organization's ability to monitor traffic at a network level.

Another area of change has been the use of kernel modification. After compromising a system, attackers usually reprogram the computer to lie to the system administrator. This reprogramming hides the attackers activities and gives the attacker unrestricted access to the system, regardless of the administrators actions. In the past, such modifications were easy to detect, since they involved mainly modifying system binaries or log files. Today the attackers have changed their tools, making it far more difficult to detect their activities. Instead of modifying system binaries, the actual kernel of the system can be modified in real time. This means that no matter what interaction is made with the compromised system, the results cannot be trusted. The system must be taken offline and analyzed with a trusted system before any output can be considered valid.

These are just some examples of the attackers' abilities to change and improve their tools and tactics. The threat will always be adapting, improving, and changing. In many ways it is a cyberspace arms race, and unfortunately, in many ways we are losing.

## Summary

This chapter is an overview of the types of threats you or your organization may face. In general, there are two types of attackers: those with basic skill sets focused on attacking as many systems as possible and those with more advanced skill sets focusing on a few systems of high value. Both of these threats are constantly evolving and improving their capabilities. By understanding these threats, you will better appreciate the value of honeypots. As you read the following chapters, understanding how threats develop and operate will help you decide what type of honeypots you want to use and how to build and deploy them.

If you want to learn more about the different types of tools and tactics attackers use, I highly recommend these books: *Hacking Exposed* [11], *Hackers Beware* [12], and *Counter Hack* [13]. These three books focus on the tools and methods of the hacking community.

## References


[1] *"Know Your Enemy: Statistics"* http://project.honeynet.org/papers/stats/ and book CD-ROM

[2] Snort is an Open Source Intrusion Detection System, http://www.snort.org

[3] CERT Statistics http://www.cert.org/stats/cert_stats.html

[4] CERT Advisory CA-2001-23 Continued Threat of the "CodeRed" Worm http://www.cert.org/advisories/CA-2001-23.html

[5] http://www.securityfocus.com

[6] http://www.eyee.com

[7] CERT Advisory CA-2001-26 Nimda Worm http://www.cert.org/advisories/CA-2001-26.html

[8] *"Know Your Enemy: Motives"* http://project.honeynet.org/papers/motives/ and book CD-ROM

[9] *"Know Your Enemy: Worms at War"* http://project.honeynet.org/papers/worm/ and book CD-ROM

[10] Hacker group claiming cyber-jihad http://www.vnunet.com/News/1126240

[11] McClure, Stuart, Joel Scambray, and George Kurtz. 2001. *Hacking Exposed, Third Edition*. Berkeley, California: Osborne/McGraw-Hill. http://www.hackingexposed.com

[12] Cole, Eric. 2001. *Hackers Beware*. Indianapolis, Indiana: New Riders. http://www.securityhaven.com/

[13] Skoudis, Ed. 2001. *Counter Hack*. Upper Saddle River, New Jersey: Prentice Hall PTR. http://www.counterhack.net

# Chapter 3. History and Definition of Honeypots

Up to this point, we've been working with a pretty simple definition of a honeypot. It's time to get more precise. But first, it will be beneficial to review the history of honeypots. Examining history and developing a more precise definition will prepare us for details about honeypot technologies that we will see in upcoming chapters. Finally, we will discuss how honeypots work.

## The History of Honeypots

Honeypots have a unique history. Even though the concepts have been around for more than a decade, only recently have commercial products been developed or papers published on the concept. The following list summarizes some key events in the history of honeypots. We will discuss each of them in more detail in subsequent sections.

- **1990/1991—** First public works documenting honeypot concepts—Clifford Stoll's *The Cuckoo's Egg* and Bill Cheswick's "An Evening With Berferd."
- **1997—** Version 0.1 of Fred Cohen's Deception Toolkit was released, one of the first honeypot solutions available to the security community.
- **1998—** Development began on CyberCop Sting, one of the first commercial honeypots sold to the public. CyberCop Sting introduces the concept of multiple, virtual systems bound to a single honeypot.
- **1998—** Marty Roesch and GTE Internetworking begin development on a honeypot solution that eventually becomes NetFacade. This work also begins the concept of Snort.
- **1998—** BackOfficer Friendly is released—a free, simple-to-use Windows-based honeypot that introduced many people, including me, to honeypot concepts.
- **1999—** Formation of the Honeynet Project and publication of the "Know Your Enemy" series of papers. This work helped increase awareness and validate the value of honeypots and honeypot technologies.
- **2000/2001—** Use of honeypots to capture and study worm activity. More organizations adopting honeypots for both detecting attacks and for researching new threats.
- **2002—** A honeypot is used to detect and capture in the wild a new and unknown attack, specifically the Solaris dtspcd exploit.

### Early Publications

Surprisingly little if any material can be found before 1990 concerning honeypot concepts. The first resource was a book written by Clifford Stoll titled *The Cuckoo's Egg* [1]. The second is the whitepaper "An Evening with

Berferd in Which a Cracker Is Lured, Endured, and Studied" [2], by the security icon Bill Cheswick. This does not mean that honeypots were not invented until 1990; they were undoubtedly developed and used by a variety of organizations well before then. A great deal of research and deployment has occurred within military, government, and commercial organizations, but very little of it is public knowledge before 1990.

In *The Cuckoo's Egg*, Clifford Stoll discusses a series of true events that occurred over a ten-month period in 1986 and 1987. Stoll was an astronomer at Lawrence Berkeley Lab who worked with and helped administer a variety of computer systems used by the astronomer community. A 75-cent accounting error led him to discover that an attacker, code named "Hunter," had infiltrated one of his systems. Instead of disabling the attacker's accounts and locking him out of the system, Stoll decided to allow the attacker to stay on his system. His motives were to learn more about the attacker and hunt him down. Over the following months he attempted to discover the attacker's identity while at the same time protecting the various government and military computers the attacker was targeting. Stoll's computers were not honeypots; they were production systems used by the academic and research communities. However, he used the compromised systems to track the attacker in a manner very similar to the concept of honeypots and honeypot technologies. Stoll's book is not technical; it reads more like a Tom Clancy spy novel. What makes the book unique and important for the history of honeypots are the concepts Stoll discusses in it.

The most fascinating thing in the book is Stoll's approach to gaining information without the attacker realizing it. For example, he creates a bogus directory on the compromised system called SDINET, for Strategic Defense Initiative Network. He wanted to create material that would attract the attention of the attacker. He then filled the directory with a variety of interesting-sounding files. The goal was to waste the attacker's time by compelling him to look through a lot of files. The more time he spent on the system, the more time authorities had to track down the attacker. Stoll also included documents with different values. By observing which particular documents the attacker copied, he could identify the attacker's motives. For example, Stoll provided documents that included those that appeared to have financial value and those that had government secrets. The attacker bypassed the financial documents and focused on materials about national security. This indicated that the attackers motives were not financial gain but access to highly secret documents.

Bill Cheswick's paper "An Evening with Berferd in Which a Cracker Is Lured, Endured, and Studied" was released in 1990. This paper is more technical than *The Cuckoo's Egg*. It was written by security professionals for the security community. Like *The Cuckoo's Egg*, everything in Cheswick's paper is nonfiction. However, unlike the book, Cheswick builds a system that he wants to be compromised—our first documented case of a true honeypot. In the paper, he discusses not only how the honeypot was built and used but how a Dutch hacker was studied as he attacked and compromised a variety of systems.

Cheswick initially built a system with several vulnerabilities (including Sendmail) to determine what threats existed and how they operated. His goal was not to capture someone specific but rather to learn what threatening activity was happening on his networks and systems.

> Our secure Internet gateway was firmly in place by the spring of 1990. With the castle gate in place, I wondered how often the lock was tried. I knew there were barbarians out there. Who were they? Where did they attack from and how often? What security holes did they try? [2]

Cheswick's paper explains not only the different methodologies he used in building his system (he never calls it a honeypot) but also how these methodologies were used. In addition to a variety of services that appeared vulnerable, he created a controlled environment called a "jail," which contained the activities of the attacker. He takes us step by step how an intruder (called Berferd) attempts to infiltrate the system and what Cheswick was able to learn from the attacker. We see how Berferd infiltrated a system using a Sendmail vulnerability and then gained control of the system. Cheswick describes the advantages and disadvantages of his approach. (This paper is on the CD-ROM that accompanies this book.)

Both Stoll's book and Cheswick's paper make for fascinating reading, and I highly recommend them for anyone interested in honeypots. However, neither resource describes how to design and deploy honeypots in detail. And neither provides a precise definition of honeypots or explores the value of honeypot technologies.

## Early Products

The first public honeypot solution, called Deception Toolkit (DTK) [3], was developed by Fred Cohen. Version 0.1 was released in November 1997, seven years after *The Cuckoo's Egg* and "An Evening with Berferd." DTK is one of the first free honeypot solutions you could download, install, and try out on your own. It is a collection of PERL scripts and C code that is compiled and installed on a Unix system. DTK is similar to Bill Cheswick's Berferd system in that it emulates a variety of known Unix vulnerabilities. When attacked, these emulated vulnerabilities log the attacker's behavior and actions and reveal information about the attacker. The goal of DTK is not only to gain information but to deceive the attacker and psychologically confuse her. DTK introduced honeypot solutions to the security community.

Following DTK, in 1998, development began on the first commercial honeypot product, CyberCop Sting. Originally developed by Alfred Huger at Secure Networks Inc., it was purchased by NAI in 1998. This honeypot had several features different from DTK. First, it ran on Windows NT systems and not Unix. Second, it could emulate different systems at the same time, specifically a Cisco router, a Solaris server, and an NT system.

Thus, CyberCop Sting could emulate an entire network, with each system having its own unique services devoted to the operating system it was emulating. It would be possible for an attacker to scan a network and find a variety of Cisco, Solaris, and NT systems. The attacker could then Telnet to the Cisco router and get a banner saying the system was Cisco, FTP to the Solaris server and get a banner saying the system was Solaris, or make a HTTP connection to the NT server. Even the emulated IP stacks were modified to replicate the proper OS. This way if active fingerprinting measures were used, such as Nmap[4], the detected OS would reflect the services for that IP address. The multiple honeypot images created by a single CyberCop Sting installation greatly increased the chance of the honeypots being found and attacked. This improved detection of and alerting to the attacker's activity.

For its time and development, CyberCop Sting was a cutting edge and advanced honeypot. Also, it was easy to install, configure, and maintain, making it accessible to a large part of the security community. However, as a commercial product it never really took off and has now been discontinued. Since its demise, several excellent commercial honeypot products have been released, including NetSec's Specter[5] and Recourse's Mantrap[6], both of which we will discuss in detail later in the book.

In 1998, Marty Roesch, while working at GTE Internetworking, began work on a honeypot solution for a large government client. Roesch and his colleagues developed a honeypot system that would simulate an entire class C network, up to 254 systems, using a single host to create the entire network. Up to seven different types of operating systems could be emulated with a variety of services. Although the resulting commercial product, NetFacade[7], has seen little public exposure, an important side benefit of this honeypot solution is that Roesch also developed a network-based debugging tool, which eventually led to his OpenSource IDS, Snort [8].

The year 1998 also saw the release of BackOfficer Friendly, a Windows- and Unix-based honeypot developed by Marcus Ranum and released by Network Flight Recorder. What made BOF unique is that it was free, extremely easy to use, and could run on any Windows based desktop system. All you had to do was download the tool, install it on your system, and you instantly had your own personal honeypot. Though limited in its capabilities, BOF was many people's first introduction to the concepts of honeypot technologies.

In 1999 the Honeynet Project was formed [9]. A nonprofit research group of 30 security professionals, this group is dedicated to researching the blackhat community and sharing what they learned. Their primary tool for learning is the Honeynet, an advanced type of honeypot. Over several years the Honeynet Project demonstrated the capabilities and value of honeypots, specifically Honeynets, for detecting and learning about attacks and the attackers themselves. All of the group's research methods, specifically how they designed and deployed honeypots, were publicly documented and released for the security community in a series of papers known as "Know Your Enemy." In 2001 they released the book *Know Your Enemy* [10] that documented their research and findings. This helped develop the awareness, credibility, and value of honeypots.

## Recent History: Honeypots in Action

During 2000 and 2001 there was a sudden growth in both Unix-based and Windows-based worms. These worms proved to be extremely effective. Their ability to exponentially spread across the Internet astounded the Internet community. One of the challenges that various security organizations faced was obtaining a copy of the worm for analysis and understanding how it worked. Obtaining copies of the worm from compromised production systems was difficult because of data pollution or, as in the case of the CodeRed worm [11], because the worms only

resided in the system's memory. Honeypots proved themselves a powerful solution in quickly capturing these worms, once again proving their value to the security community.

One example was the capture and analysis of the Leaves worm by Incidents.org. On June 19, 2001 a sudden rise of scans for the Sub7 Trojan was detected. Sub7 was a Trojan that took over Windows systems, giving an attacker total remote control of the system. The Trojan listened on the default port 27374. The attacker controlled the compromised system by connecting to this port with special client software. A team of security experts from Incidents.org attempted to find the reason for the activity.

On June 21, Johannes Ullrich of the SANS Institute deployed a honeypot he had developed to emulate a Windows system infected with the Sub7 Trojan (the source code is included in the CD-ROM). Within minutes this honeypot captured an attack, giving the Incidents team the ability to analyze it. They discovered that a worm was pretending to be a Sub7 client and attempting to infect systems already infected by the Sub7 Trojan. This saved the attacker the trouble of hacking into systems, since the systems were already attacked and compromised. Matt Fearnow and the Incidents.org team were able to do a full analysis of the worm, which was eventually identified as the W32/Leaves worm, and forward the critical information to the National Infrastructure Protection Center (NIPC). Other organizations also began using honeypots for capturing worms for analysis, such as Ryan Russel at SecurityFocus.com for analysis of the CodeRed II worm. These incidents helped develop awareness of the value of honeypots within the security community and security research.

The first recorded instance involving honeypot technologies in capturing an unknown exploit occurred on January 8, 2002. A Solaris honeypot captured a dtspcd exploit, an attack never seen before. On November 12, 2001, the CERT Coordination Center, a security research organization, had released an advisory for the CDE Subprocess Control Service [12], or, more specifically, dtspcd. The security community was aware that the service was vulnerable. An attacker could theoretically remotely attack and gain access to any Unix system running the dtspcd service. However, no actual exploit was known, and it was believed that there was no exploit being used in the wild. When a honeypot was used to detect and capture a dtspcd attack in the wild, it confirmed that exploit code did exist and was being used by the blackhat community. CERT was able to release an advisory [13] based on this information, warning the security community that the vulnerability was now being actively attacked and exploited. This demonstrated the value of honeypots in not only capturing known attacks, such as worm, but detecting and capturing unknown attacks.

## Definitions of Honeypots

We've been talking about honeypots for several chapters now, with only a simple, rather intuitive definition of the term. It's time to get more precise. There is no way we can discuss the value of honeypots or how they work if we don't agree first on what a honeypot is.

As I mentioned earlier, I believe that the lack of a clear, widely accepted definition of a honeypot is one of the main reasons the security community has been so slow to adopt them. Everyone has his or her own definition of a honeypot. This creates a great deal of confusion and miscommunication. Some think a honeypot is a tool for deception, whereas others consider it a weapon to lure hackers, and still others believe it is simply another intrusion detection tool. Some believe a honeypot should emulate vulnerabilities. Others see it as simply a jail. Some view honeypots as controlled production systems that attackers can break into. These various viewpoints have caused a lot of misunderstandings about what a honeypot is and, thus, its value.

Therefore, for the purposes of this book, we will define a honeypot as follows.

> A honeypot is security resource whose value lies in being probed, attacked, or compromised.

This means that whatever we designate as a honeypot, our expectations and goals are to have the system probed, attacked, and potentially exploited. It does not matter what the resource is (a router, scripts running emulated services, a jail, an actual production system). What does matter is that the resource's value lies in its being attacked. If the system is never probed or attacked, then it has little or no value. This is the exact opposite of most production systems, which you do *not* want to be probed or attacked.

As should be apparent from this definition, honeypots are different from most security tools in that they can take on different manifestations. Most of the security technologies used today were designed to address specific problems. For example, firewalls are a technology that protect your organization by controlling what traffic can

flow where. They are used as an access control device. Firewalls are most commonly deployed around an organization's perimeter to block unauthorized activity. Network Intrusion Detection Systems are designed to detect attacks by monitoring either system or network activity. They are used to identify unauthorized activity.

Honeypots are different in that they aren't limited to solving a single, specific problem. Instead, honeypots are a highly flexible tool that can be applied to a variety of different situations. This is why the definition of honeypots may at first seem vague, because they can be used to achieve so many different goals and come in a variety of different forms. For example, honeypots can be used to deter attacks, a goal shared with firewalls. Honeypots also can be used to detect attacks, similar to the functionality of an Intrusion Detection System. Honeypots can be used to capture and analyze automated attacks, such as worms, or act as early indication and warning sensors. Honeypots also have the capability to research the activities of the blackhat community, capturing the keystrokes or conversations of attackers. How you use honeypots is up to you. It depends on what you are trying to achieve. In Chapter 4 we go into far greater detail on the different goals you can accomplish with a honeypot. However, all the possible manifestations share one common feature: Their value lies in being probed, attacked or compromised.

## How Honeypots Work

Just as the definition of a honeypot is basic, so is the concept of how they work. Honeypots are security resources that have no production value; no person or resource should be communicating with them. As such, any activity sent their way is suspect by nature. Any traffic sent to the honeypot is most likely a probe, scan, or attack. Any traffic initiated by the honeypot means the system has most likely been compromised and the attacker is making outbound connections. Cliff Stoll discussed this concept in *The Cuckoo's Egg*. When attempting to analyze his compromised system, he approached it as follows.

> Collect raw data and throw away what is expected. What remains challenge your theories. (p. 22)

With a honeypot, nothing is expected. This concept is what gives honeypots its unique advantages and disadvantages, which we discuss in the next chapter.

## Two Examples of Honeypots

To better understand the definition and concepts of honeypots, let's take a look at two examples of honeypot deployments. The purpose here is to demonstrate to you that honeypots can come in many different flavors, and they can achieve different things. However, they are both honeypots because they share the same definition and concepts.

For our first example, let us assume your organization is a Microsoft environment. You depend on Microsoft Exchange to handle receiving and sending your organization's e-mail. The system that your organization is currently using is over two years old. It is a production system, since it provides the e-mail functionality for the 500 employees of your organization. Your boss decides to retire the system and upgrade to a new server, leaving the old Exchange server to use as you see fit. You decide to use this retired system as a honeypot, leave it as it is on your DMZ (see Figure 3-1, Honeypot A), and closely watch any traffic to or from the retired Exchange server. Your intent is to use the system as a honeypot, to determine if there is any unauthorized activity happening within your DMZ.

**Figure 3-1. Though vastly different in how they were built and what their purpose is, Honeypots A and B share the definition and concepts of a honeypot.**

sample content of Honeypots: Tracking Hackers

- **click The Way of Science: Finding Truth and Meaning in a Scientific Worldview book**
- click New American Vegan
- A Journey with Two Maps: Becoming a Woman Poet here
- read online Levitate the Primate: Handjobs, Internet Dating, and Other Issues for Men pdf, azw (kindle)
- read Toy Story Screenplay pdf, azw (kindle), epub, doc, mobi

- http://www.1973vision.com/?library/The-End-of-Work-as-You-Know-It--8-Strategies-to-Redefine-Work-on-Your-Own-Terms.pdf
- http://rodrigocaporal.com/library/Only-the-Innocent.pdf
- http://aseasonedman.com/ebooks/A-Journey-with-Two-Maps--Becoming-a-Woman-Poet.pdf
- http://aseasonedman.com/ebooks/Dancing-With-Einstein--A-Novel.pdf
- http://fortune-touko.com/library/Pushed--Sense-Thieves--Book-2-.pdf