



Beginning Ethical Hacking with Kali Linux

Computational Techniques for
Resolving Security Issues

—
Sanjib Sinha

Apress®

Beginning Ethical Hacking with Kali Linux

**Computational Techniques for
Resolving Security Issues**

Sanjib Sinha

Apress®

Beginning Ethical Hacking with Kali Linux

Sanjib Sinha
Howrah, West Bengal, India

ISBN-13 (pbk): 978-1-4842-3890-5

ISBN-13 (electronic): 978-1-4842-3891-2

<https://doi.org/10.1007/978-1-4842-3891-2>

Library of Congress Control Number: 2018963849

Copyright © 2018 by Sanjib Sinha

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

Trademarked names, logos, and images may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, logo, or image we use the names, logos, and images only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Managing Director, Apress Media LLC: Welmoed Spahr

Acquisitions Editor: Nikhil Karkal

Development Editor: Matthew Moodie

Coordinating Editor: Divya Modi

Cover designed by eStudioCalamar

Cover image designed by Freepik (www.freepik.com)

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail rights@apress.com, or visit www.apress.com/rights-permissions.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/978-1-4842-3890-5. For more detailed information, please visit www.apress.com/source-code.

Printed on acid-free paper

Disclaimer

This book is intended to be used only in an ethical manner. Performing any illegal actions using the information available in this book may invite legal trouble, and if you have any bad intentions, you will likely be arrested and prosecuted to the full extent of the law.

The publisher Apress and the author take no responsibility if you misuse any information available in this book.

Tip Always use VirtualBox or a virtual machine to experiment with your hacking skills.

You must use this information in a testing environment as shown in this book and, in some special cases, get proper permission from the appropriate authorities.

This book is dedicated to my parents: my late mother, Krishna Deb, and my late father, Sushil Kumar Sinha.

You taught me to appreciate the value of reading and lifelong learning. I hope death has not defeated your longing for happiness.

May you rest in peace.

Table of Contents

About the Author	xiii
About the Technical Reviewer	xv
Acknowledgments	xvii
Introduction	xix
Chapter 1: Security Trends	1
Nature and Perspective.....	3
Before and After the Digital Transformation.....	6
The OSI Security Architecture	6
Security Attacks, Services, and Mechanisms	10
Timeline of Hacking.....	14
How to Use Google Hacking Techniques	15
Further Reading	17
Chapter 2: Setting Up a Penetration Testing and Network Security Lab	19
Why Virtualization?	20
Installing VirtualBox	21
Installing Appliances on VirtualBox.....	23
Installing VirtualBox Guest Addition.....	29
Installing Metasploitable	31
Installing Windows	33
Installing Kali in VMware.....	36

TABLE OF CONTENTS

- Chapter 3: Elementary Linux Commands.....41**
 - Finding the Kali Terminal42
 - Navigating the File System44
 - Working with Text Files48
 - Searching Files49
 - Writing to the Terminal.....51
 - Working with Directories.....52
 - Setting File Permissions53

- Chapter 4: Know Your Network61**
 - Networking Layers61
 - Internetworking Models65
 - OSI65
 - TCP/IP68
 - Further Reading69

- Chapter 5: How to Build a Kali Web Server71**
 - Why Do You Need a Web Server?72
 - Introducing Sockets73
 - Beginning the Web Server.....73
 - Diving into Sockets76
 - Installing PyCharm and the Wing IDE Editor84
 - How to Stay Anonymous86
 - Changing Your Proxy Chain.....88
 - Working with DNS Settings92
 - Using a VPN94
 - Changing Your MAC Address100

Chapter 6: Kali Linux from the Inside Out.....	105
More About Kali Linux Tools	106
Information Gathering.....	107
Vulnerability Analysis.....	108
Wireless Attacks	109
Web Applications	109
WPS Tools	110
Exploitation Tools.....	111
Forensic Tools.....	111
Sniffing and Spoofing	112
Password Attacks	112
Maintaining Access	113
Reverse Engineering.....	113
Hardware Hacking	114
Exploring Kali Linux from the Inside	114
Machine Language	114
Registers	115
Why Is Understanding Memory So Important?	116
Editors	117
Hacking Tools	121
Staying Updated with SSH	124
Getting Started	125
Working with Blacklists and Whitelists.....	128
Securing SSH.....	130
Connecting to Kali Linux Over SSH.....	134

TABLE OF CONTENTS

Chapter 7: Kali Linux and Python137

- What Is Penetration Testing? 137
- First Penetration Using Python..... 139
 - Whois Searches for More Information..... 142
 - Finding Secret Directories 152
 - Top-Level Domain Scanning..... 158
 - Obtaining a Web Site’s IP Address 161
- TCP Client in Python and Services 164
- Capturing Raw Binary Packets..... 170
- Port Scanning Using Nmap 174
 - Importing the Nmap Module..... 175
 - What Does Nmap Do? 180
 - Nmap Network Scanner 183

Chapter 8: Information Gathering189

- Python Virtual Environment..... 190
- Reconnaissance Tools..... 197
 - Know the Domain and Hostname 198
 - E-mail Tracking Made Easy 200
 - Searching the Internet Archive 202
 - Passive Information 204
 - Web Spiders Are Crawling 205
 - More About Scanning 206
 - You Can Find Location Too!..... 213
- DMitry, Maltego, and Other Tools..... 214
- Summarizing the Five Phases of Penetration 220

Chapter 9: SQL Mapping	221
Sniffing and Spoofing.....	221
Packing and Unpacking with Python	223
Why Wireless Media Is Vulnerable.....	227
ARP Poisoning Is a Threat.....	228
SQL Injection.....	241
Detecting SQL Injection Vulnerabilities.....	242
How to Use sqlmap.....	243
Brute-Force or Password Attacks	253
Chapter 10: Vulnerability Analysis	259
Overview of Vulnerability Analysis Tools	259
How to Use OpenVas.....	260
How to Use Nikto.....	268
How to Use Vega	270
How to Use Burp Suite	276
Chapter 11: Information Assurance Model.....	283
What the AI Model Is All About	284
How to Tie the Elements Together?.....	285
How the AI Model Works	287
Why Is the AI Model Important?	289
Further Reading	290
Chapter 12: Introducing Metasploit in Kali Linux	291
Understanding the Metasploit Architecture	292
Summarizing Modules	295
Mixins and Plugins in Ruby.....	302
Metasploit Console or Interface.....	304

TABLE OF CONTENTS

- Exploits and Payloads in Metasploit 308
 - How to Use Exploit and Payloads 309
 - How to Start Exploits 315
- Chapter 13: Hashes and Passwords 323**
 - Hashes and Encryption 324
 - Password Testing Tools 327
 - John the Ripper and Johnny 338
 - How to Use RainbowCrack 342
- Chapter 14: Classic and Modern Encryption 347**
 - Nature and Perspective 348
 - Models of the Cryptography System 352
 - Types of Attacks on Encrypted Messages 354
- Chapter 15: Exploiting Targets 357**
 - Exploiting Linux with Metasploit 358
 - Exploiting Samba 359
 - Exploiting IRC 371
 - Exploiting Windows with Armitage 380
- Index 405**

About the Author



Sanjib Sinha is a certified .NET Windows and web developer, specializing in Python, security programming, and PHP; he won Microsoft's Community Contributor Award in 2011. Sanjib Sinha has also written *Beginning Ethical Hacking with Python* and *Beginning Laravel* for Apress.

About the Technical Reviewer

Vaibhav Chavan holds a certification in ethical hacking and has worked as a security analyst in the IT world as well as in the banking, insurance, and e-commerce industries. He now works as a security analyst in Mumbai and has more than five years of experience in the IT industry. He has hands-on experience in Kali Linux and other tools such as the Metasploit Framework, Burp Suite, Nessus, and more.

Acknowledgments

I wish to record my gratitude to my wife, Kaberi, for her unwavering support and encouragement in the preparation of this book.

I am extremely grateful to Mr. Matthew Moodie, lead development editor, for his numerous valuable suggestions, complementary opinions, and thorough thumbing; as well as editor Nikhil Karkal, coordinating editor Divya Modi, and the whole Apress team for their persistent support and help. I also wish to thank Vaibhav Chavan, the technical reviewer, for their valued suggestions.

In the preparation of this book, I consulted open source documentation and numerous textbooks on a variety of subjects related to ethical hacking and want to thank the countless authors who wrote them. I hereby acknowledge my special indebtedness to Nmap original author Gordon Lyon (Fyodor) and the developer of the Metasploit Framework, H.D. Moore. I am also thankful for the ever-helpful open source community.

Introduction

You can get started in white-hat ethical hacking using Kali Linux, and this book starts you on that road by giving you an overview of security trends, where you will learn about the OSI security architecture. This will form the foundation for the rest of *Beginning Ethical Hacking with Kali Linux*.

With the theory out of the way, you'll move on to an introduction to VirtualBox, networking terminologies, and common Linux commands, followed by the step-by-step procedures to build your own web server and acquire the skill to be anonymous. When you have finished the examples in the first part of your book, you will have all you need to carry out safe and ethical hacking experiments.

After an introduction to Kali Linux, you will carry out your first penetration tests with Python and code raw binary packets for use in those tests. You will learn how to find secret directories of a target system, how to use a TCP client in Python and services, and how to do port scanning using Nmap. Along the way, you will learn how to collect important information; how to track e-mail; and how to use important tools such as DMitry, Maltego, and others. You'll also take a look at the five phases of penetration testing.

After that, this book will cover SQL mapping and vulnerability analysis where you will learn about sniffing and spoofing, why ARP poisoning is a threat, how SniffJoke prevents poisoning, how to analyze protocols with Wireshark, and how to use sniffing packets with Scapy. Then, you will learn how to detect SQL injection vulnerabilities, how to use Sqlmap, and how to do brute-force or password attacks. In addition, you will learn how to use important hacking tools such as OpenVas, Nikto, Vega, and Burp Suite.

INTRODUCTION

The book will also explain the information assurance model and the hacking framework Metasploit, taking you through important commands, exploits, and payload basics. Moving on to hashes and passwords, you will learn password testing and hacking techniques with John the Ripper and Rainbow. You will then dive into classic and modern encryption techniques where you will learn to work with the conventional cryptosystem.

In the final chapter, you will use all the skills of hacking to exploit a remote Windows and Linux system, and you will learn how to “own” a remote target entirely.

Who This Book Is For

This book is primarily for information security professionals. However, security enthusiasts and absolute beginners will also find this book helpful. For absolute beginners, knowledge of high school algebra, the number system, and the Python programming language is a plus. However, this book provides an explanation of the foundational rules so you can understand the relationship between them and ethical hacking, information security, and the hacking-related tools of Kali Linux.

For more advanced professionals, the book also includes in-depth analysis.

Whether you are new to ethical hacking or a seasoned veteran, this book will help you understand and master many of the powerful and useful hacking-related tools of Kali Linux and the techniques that are widely used in the industry today.

To start with, you need a virtual box or virtual machine, so proceed to [Chapter 1](#).

CHAPTER 1

Security Trends

Computer security depends on the following:

- Confidentiality
- Integrity
- Availability

To understand security trends, you need to understand what these three terms actually mean in the context of your project because you may interpret these three terms in a different way depending on the environment. The needs of individuals vary, and customs differ between countries.

The Computer Emergency Readiness Team (CERT) was formed by the Defense Advanced Research Projects Agency (DARPA) in November 1988 because of a security breach issue. An unknown virus was detected by some Cornell University researchers. It spread to other universities, national laboratories, and other hosts on interconnected networks. A warning message took 26 hours to reach MIT from Harvard University.

A request for comment (RFC) is a formal document from the Internet Engineering Task Force (IETF). Interested parties may ask for it and review it. Some requests for comments are informational in nature.

The next development was that from RFC 1636, the Internet Architecture Board (IAB) issued a report on security in 1994 titled “Security in the Internet

Architecture.” There was a consensus that the Internet should be more secure. The IAB identified a few key areas that were vulnerable.

SECURITY IN THE INTERNET ARCHITECTURE ABSTRACT

The report was presented at an Internet architecture workshop, initiated by the IAB and held at the SC Information Sciences Institute in February 1994. This workshop generally focused on security issues in the Internet architecture.

This document should be regarded as a set of working notes containing ideas about security that were developed by Internet experts in a broad spectrum of areas, including routing, mobility, real-time services, and provider requirements. It contains some significant diversity of opinions on some important issues. This report is offered as one input in the process of developing viable security mechanisms and procedures for the Internet.

There were many suggestions about how best to tackle a modern monster: the security breach. The report states that the network infrastructure is one of the main points of vulnerabilities. One needs to control traffic, keeping a close watch on unauthorized monitoring. The user-to-user communications should also go through authentication and encryption mechanisms.

The concern that the Internet needed more security was true. A number of security-related complaints started popping up at about that same time. If you look over a ten-year period, there were security-related vulnerabilities in the operating systems of the computers attached to the Internet, and at the same time, there were weaknesses in Internet routers and other network devices.

The report showed one particular thing. Authentication based on IP addresses was violated; intruders created packets with false IP addresses and exploited the applications. As time goes by, the problems seem to outpace the actual growth. This happens for one reason: hackers now depend more on automated attacks. So, the frequency of them has increased.

Since most attacks are automated, they have become more sophisticated. The knowledge of the intruders might have declined, but the frequency of automated attacks has increased. The increased use of the Internet combined with the increase of the complexities of protocols and applications, makes things more difficult to tackle. The CERT report shows the complexities are ever-increasing.

Nature and Perspective

Let's try to understand what confidentiality means. The literal meaning applies to the computer world also. It starts with the concealment of information. The information may be linked to particular protected resources.

Consider sensitive fields such as government or industrial zones; in these industries you need to keep the information secret. In that sense, a military zone has a close relationship with industrial companies that want to keep their own strategies and designs secret.

The need to keep personal information secret is more important too. The recent debate over how to keep a user's personal data secure will continue for years to come.

These aspects are interdependent. Confidentiality and integrity have a direct relationship with availability and accountability. As a matter of principle, confidentiality depends on integrity, and vice versa. Integrity ensures authorized modifications. It includes correctness and trustworthiness. It usually refers to data integrity and origin integrity. Data integrity refers to the content of the information, and the origin integrity is actually the source of the data often called authentication.

Several mechanisms support confidentiality such as access control mechanisms.

For example, cryptography runs on a “lock and key” algorithm. Cryptography scrambles data, and it makes the data not perceivable. You virtually lock the data, and after that, you have the cryptographic key to unlock that data. However, the need to lock the cryptographic key itself becomes a necessity after that. If you try to write that algorithm, you need to decide how deep you need to go. Basically, you have to handle two separate layers of data.

Sometimes the mere fact that data exists needs to be protected. Think of two secrets associated with each other. For example, a social media company commercially uses a user’s data without their permission. Here the mere existence of such information needs to be concealed by the social media company by using access control mechanisms because the existence itself may reveal more sensitive data.

Resource hiding is another important aspect. A company doesn’t want others to know their trade secrets such as a formula, a specific strategy, or what equipment they have. Here authorization is important, which is not same as authentication; an access control mechanism can provide these capabilities as the authorized users only can access classified data. However, there is a hidden difficulty or disadvantage. You need to assume that the system should give the proper supporting services so that all these mechanisms could be enforced. Therefore, there are two more layers of assumptions and trust; you need to trust all the authorized users. You will get a detailed explanation about this in Chapter 11 where I discuss the information assurance model.

The advantage of system-dependent mechanisms over cryptographic is not foolproof. It can more completely protect data than cryptographic mechanisms, but the system-related controls could fail or someone could bypass them to get access to the data.

Working with integrity is not the same as working with confidentiality.

In general, two classes—prevention mechanisms and detection mechanisms—broadly define integration. In any case, if someone wants to make any unauthorized attempt to change the data, the prevention mechanism starts working. The unauthorized attempt may be classified into two broad categories. In the first case, the user is unauthorized, but he has bypassed the authentication and tries to change the data; in the second case, the user has authorization but he is not supposed to change the data.

When a violation of integrity happens, the detection mechanism does not try to prevent it. Its job is reporting the fact. It reports that the data's integrity has been violated, and it has no longer trustworthiness. Confidentiality checks whether the data has been compromised or not. Integrity encompasses a wide range of responsibility. It first checks the origin of the data and how well the data was protected before it comes to the current machine. The next step is checking the current state where the data belongs. The evaluation of integrity is more difficult as it deals with the assumptions and trust at the same time.

Confidentiality and integrity are directly related to availability and accountability. They constantly interact with the two components.

Availability assures that authorized access to data and resources is always available. However, in the real world, something different may happen. You have probably heard of denial-of-service attacks. Someone may intentionally set up a trap that ensures the denial of service. In general, a statistical model works behind analyzing the expected patterns of a user's actions. There are a few parameters that may indirectly control that use, say, network traffic. Now, someone may be able to manipulate that whole system. In such cases, you will encounter a denial of service.

The accountability ensures that the user's action is traceable. This traceroute should be unique with respect to the user. When these four components work in harmony and the four objects meet all the criteria, you can say that the security is foolproof.

Before and After the Digital Transformation

Let's forget about the old concept of physical security where you might keep paper documents inside a locked container. The world is fast becoming paperless. In the past, people primarily kept information on paper. Lock and key were widely used to keep them secure. The safe transport of secret information was emphasized.

Twenty years ago, administrative security was mainly comprised of three main components: control access to materials, personal screening, and auditing.

Information security today has completely changed. It starts with system complexities that are manifold. As the information becomes digital, the need to keep it secure becomes important. The system of maintaining security is getting complicated with the ubiquitous presence of automated attacks.

The information security breaches have inflated financial losses, which have skyrocketed. As far as national defense is concerned, there are two main points where security should be tightened: the power grid and the air transportation. The interlinked digital system has made them vulnerable now. Twenty years ago, the case was entirely contrasting. Think about the interlinked government agencies.

The OSI Security Architecture

As a penetration tester or ethical hacker, a basic conception of Open Systems Interconnection (OSI) security architecture is necessary because hacking is all about interconnected networks. In the coming chapters where you will learn Kali Linux tools, I will refer to this standard often.

OSI is related to the standards developed by the International Telecommunication Union (ITU) Telecommunication Standardization Sector (ITU-T). The United Nations sponsors the ITU agency, which develops security standards.

First, you need to understand why OSI security architecture is important. It organizes the task of providing security. It was developed as an international standard. For that reason, companies related to computer manufacturers and communication vendors now maintain that standard. They have developed security features according to that standard, and now they maintain it strictly. The OSI security architecture has some guidelines and structured definitions of services and mechanisms.

The ITU-T recommendation X.800 is the OSI security architecture. It defines an organized approach.

X.800 is an extension of recommendation X.200 that originally described the OSI security architecture reference model. As far as the system interconnection is concerned, the act of working together in an efficient and organized way is important. In other words, existing and future recommendations should communicate with each other. The OSI security architectural mechanism establishes a framework for such coordination.

In the OSI reference model, there are seven layers. Each layer is generally called an N layer. Usually, the N + 1 entity asks for the transmission services to the N entity. It is a kind of linear algorithm on which the objective of the OSI reference model is based. It permits the interconnection of disparate computer systems so that communication between application processes may be achieved. You'll learn more about this in [Chapter 4](#).

There is every possibility that when information is exchanged between applications, it could be hijacked. To protect against that, security controls need to be built. However, there is a glitch. In the end, sometimes it turns out that the value of the information is less than the effort to build those controls to obtain data. However, one needs to follow the recommendation because it defines the general security-related architectural elements, and the communication between open system-dependent applications should remain secure. To allow secure communication, you need guidelines. The OSI security reference model guides you to improve the existing recommendations.

Let's try to understand the OSI security reference model in detail. As mentioned, it is composed of seven layers. Let's imagine someone is in charge of each particular layer. Conceptually, each host has a person in charge at each layer who will communicate with a peer on other hosts. It always maintains the same-layer principle that tells us one key thing in general. The person in charge at layers 1, 2, and 3 will communicate only with the similar person in charge at the connecting hosts. At the other end of communication, the person in charge at layers 4, 5, 6, and 7 waits to receive communication from the person in charge at layers 4, 5, 6, and 7 of the connected hosts. This provides the abstract representation of networks suitable for secure communication.

Let's suppose there are three hosts in the neighborhood called A, B, and C. A, the starting point, wants to communicate with C; however, B is in between. In such cases, A will first determine who is the nearest host. Here B is the nearest. Using the appropriate routing protocol, A forwards the messages to B. Next, B determines who is the nearest host. If there was another host named D that was between B and C, then B would pass the messages to D, and the process would continue until C, the end point, gets the messages.

Let's see a mathematical representation of the same abstract. Let there be n number of hosts. The constant C starts from 0 and extends to n . Two devices, i and $i+1$, are directly connected. In such case, $0 \leq i < n$. When

a communication protocol is established, you can suppose that the two endpoints will be 0 and n . You can call this an *end-to-end protocol*. The two connected applications, that is, i and $i+1$, have a communication protocol between them, which is called a *link-protocol*.

There is a world of difference between an end-to-end protocol and a link-protocol as far as the security threats are concerned. Telnet is an ideal candidate for an end-to-end protocol where the intermediate parts play no part. IP is an ideal candidate for a link-protocol, where the messages move from the host to the immediately available neighbor, and the process continues until the messages reach the recipient.

An end-to-end protocol has other features also. In an end-to-end protocol, the cryptographic processing is done only at the source and at the endpoint or destination. In a link-protocol, the cryptographic processing is done at each spot along the path. Suppose there are four hosts, called A, B, C, and D. Each host gets the encrypted message, and the PPP encryption protocol enciphers the message. This is called *link encryption*. Each host shares the cryptographic key with its neighbor. These keys may be set in two ways, either a per-host basis or a per-host-pair basis.

For A, B, C, and D, let's consider the first basis: per host key. Each host has its own key and three for the others. Therefore, there are four keys.

In the second scenario, per-host-pair basis, there are six possible well-defined and finite sets, as shown here:

$\{A, B\}$, $\{A, C\}$, $\{A, D\}$, $\{B, C\}$, $\{B, D\}$, and $\{C, D\}$

Since each host has one key per possible connection, it has six keys altogether. In the Python programming language, you can have a representation of this set as a list within a list, with each list having one possible key.

Later you will see how this cryptographic mechanism helps in defending against attacks and other security threats.

Security Attacks, Services, and Mechanisms

There are two types of attacks: passive and active. A *passive* attack might make an attempted effort to learn information and make use of that information to learn even more about the system, but it does not affect the system. You have probably heard of *reconnaissance*; this is a passive attack. An *active* attack is directed toward the system to alter the system resources and operations.

In most cases, security breaches start with reconnaissance. Since it takes more time to gather information than a directed attack, reconnaissance (also called *footprinting*) covers almost 90 percent of the hacking activity.

You may compare this to a war strategy. Before any war takes place, army generals sit down and listen to every minute detail about the enemy's strength and weaknesses such as the location, the vulnerable strategic points, the possible backup strength, and more. These processes of collecting the information are also called *information gathering*. Before any attack, the more information you gather, the greater your chance of success. Why? It's because you will learn about your enemy strength, and if you feel that this is not the right moment to proceed, you can defer the operation for the time being.

Any attack involves a risk of losing the war. In the hacking world, the same concept applies when someone plans to attack system resources. Gathering information is important in that sense. In the real world, there are spies, informers, and other techniques involved that gather as much information as possible.

When someone tries to communicate between two open systems, the protocol layer provides some services that are defined as security services to ensure the adequate security of the systems, system-dependent applications, and data transfers.

These services can be divided into five broad categories: authentication, access control, data confidentiality, data integrity, and nonrepudiation. You have already gained some insights about the first four categories.

Nonrepudiation talks about the specifications of the source and the origin. It checks the messages sent by the specific party and received by the specific party. It also provides protection against any kind of denial by one of the entities. You will get more details about this in Chapter 11.

What does *availability* of services mean? Both X.800 and RFC 2828 define it as a property of a system or of system-dependent resources that could be accessible and usable on demand by the authorized entities. The process, leading to a loss or reduction in availability, may trigger a variety of attacks. Authentication and encryption are some automated courses of action. Sometimes, we require some sort of physical actions too.

An attacker often relies on a traceroute. A *traceroute* is the act of tracing the route of the link protocol's journey, that is, how IP addresses change in the course of movement. The Web depends on one principle concept: request and response. When someone requests a web page, the browser goes through many networks and finally pops up as a dead HTML page. There could be a packet filtering router or a firewall there. You may find many more surprising things in the journey of link-protocols. If you find any IP address that is different from those allocated to the target, it could be the end router.

There are many more techniques that help you to learn which machine is running and which is not. There are ping sweeps, port scanning, fingerprinting, footprinting, and many more. These are more or less the same in purpose but different in approach and characteristics.

- Pinging is done by issuing an echo command. If the machine is on, it'll answer. The procedure is not as easy as this, though. It's kind of like sending a request to a remote machine to check whether it's responding. It uses an end-to-end protocol. You can send one request at one time. If you want to send multiple pings, then you must get help from a tool such as `hping` or `fping`. In such cases, the technique used is called a *ping sweep*. Scanning port 80 is much easier because most firewalls allow packets to pass through that port if that network hosts a web server. This is called *TCP scanning*.
- Fingerprinting is gaining knowledge about the target's operating system. It's usually done by gaining information such as the banner, services, ports, and OS used on the target system. I'll discuss it in detail in Chapter 10.
- Nmap does plenty for us such as identifying services and port scanning and much more. Nmap even saves the output in a human-readable format. The most commonly used ports are SSH, FTP, SMTP, NETBIOS, HTTP, and many more.
- Footprinting is another important method to collect information. It's usually done by using tools such as domain lookup, Whois, NSlookup, and Spam Spade. By footprinting the target, a hacker can make decisions earlier. A hacker can eliminate the tools at the beginning that will not work against the target. The main goal of reconnaissance is to collect passive information before launching the actual attack. With the help of a passive attack, you can decide which tool will be more effective and which will not work at all.

In a link-protocol, attackers who are monitoring the whole network medium will not be able to read the messages passed in on a per-host basis, but attackers of the intermediate hosts will be able to do so.

As you have learned already, in end-to-end encryption, each host shares a cryptographic key with each destination. For that reason, neither attackers monitoring the network nor attackers on the intermediate hosts can read the message. However, the routing information is useful here. It is used to forward the message. These differences give you a chance to create a form of cryptanalysis known as *traffic analysis*. Here the messages are not important. The information about the sender and the recipient is crucial.

Several security mechanisms have been defined by X.800. Sometimes they are specific to certain protocols, but most of the time they are not specific to any particular protocol or services.

Let's see some of the specific security mechanisms.

These mechanisms are applied to protect the system from both active and passive attackers. One of them is *notarization*. You have probably seen lots of examples of this in your life. You use third-party software to prove that the signing party is not a robot. *Traffic padding* is a technique where bits are inserted into the gaps in the data stream so that passive attackers get frustrated. By means of exchanging some specific information, you can assure that the identity of an entity is genuine. This is called *authentication exchange*. In addition, *routing control* is a complex mechanism that involves routing changes when breaches of security are suspected.

You have already learned about data integrity and access control. In addition, there are specific security mechanisms like digital signature and encipherment. A *digital signature* is a cryptographic transformation that defends against forgery. Encipherment is based on a mathematical algorithm that enables a digital transformation that is not readily comprehensible. The request and response processes depends on the cryptic algorithm.

There are security mechanisms that are not so specific to OSI security services or protocols. The security label, audit trail, and recovery are part of that mechanism where mainly recovery actions take place. In such mechanisms, the security attributes of the resource are mentioned. Another feature is trusted functionality, which checks some criteria, such as some criteria established by a security policy. This mechanism also includes the detection of security-related events.

Timeline of Hacking

Before jumping to the interesting topic of Google hacking, you will see a list of attacks that has spanned more than 20 years. As I have said, with the passage of time, the knowledge for hacking has been replaced by the automated services, and the nature of the attacks is now more difficult and full of complexities.

The list starts from 1990 and moves toward the present time. I roughly assume the starting point based to be the CERT report. At that time, automated services were not available on such a large scale. So, a high amount of knowledge was the key factor then.

- Internet social engineering, packet spoofing, sniffers, and session hijacking were rampant in the first half of the 1990s. In the middle of the 1990s, we came to know about automated probes and scans. Graphical user interface hacking tools started appearing. Automated widespread attacks like denial-of-service (DoS) attacks came out and were reported from every part of the world. Browsers were targeted by executable code attacks. Widespread attacks on DNS infrastructure were first reported in 1997, and at the same time to distribute such malicious attacks, hackers used the Network News Transfer Protocol (NNTP).