Nmap Network Exploration and Security Auditing Cookbook

Third Edition

Network discovery and security scanning at your fingertips

Paulino Calderon

Nmap Network Exploration and Security Auditing Cookbook

Third Edition

Network discovery and security scanning at your fingertips

Paulino Calderon

Packt

BIRMINGHAM—MUMBAI

Nmap Network Exploration and Security Auditing Cookbook

Third Edition

Group Product Manager: Wilson D'souza
Publishing Product Manager: Rahul Nair
Senior Editor: Arun Nadar
Content Development Editor: Mrudgandha Kulkarni
Technical Editor: Shruthi Shetty
Copy Editor: Safis Editing
Project Coordinator: Ajesh Devavaram
Proofreader: Safis Editing
Indexer: Rekha Nair
Production Designer: Vijay Kamble

First published: November 2012
Second edition: May 2017
Third edition: August 2021

Production reference: 1200721

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

ISBN 978-1-83864-935-7

`www.packt.com`

Contributors

About the author

Paulino Calderon (`@calderpwn` on Twitter) is a published author and international speaker with over 10 years of professional experience in network and application security. He cofounded Websec in 2011, a consulting firm securing applications, networks, and digital assets operating in North America. When he isn't traveling to security conferences or consulting for Fortune 500 companies with Websec, he spends peaceful days enjoying the beach in Cozumel, Mexico. His contributions have reached millions of users through Nmap, Metasploit, OWASP Mobile Security Testing Guide (MSTG), OWASP Juice Shop, and OWASP IoT Goat.

To my father, Dr. Paulino Calderon Medina, who taught me that our only limitations are the ones we set up in our minds, and my mother, Edith Pale Perez, who supported me unconditionally and always believed in me.

About the reviewer

Nikhil Kumar has more than 7 years of experience in cyber security with national and multinational companies. His core expertise and passions are information security, vulnerability assessment, penetration testing on network/infrastructure, and DAST/SAST/IAST on web and mobile applications.

He is an avid blogger and regular speaker on cyber-related topics at many colleges and private and government firms.

To reach his blogs or LinkedIn, visit the following sites:

`https://www.linkedin.com/in/nikhil-kumar-bb7a0590`

`https://blogs4all2017.blogspot.com`

`https://iot4all2017.blogspot.com`

He is a postgraduate in computer science and holds numerous cyber certifications, including Certified Ethical Hacker from the EC Council, ISO 27001 Lead Auditor from the IRCA, Certified 365 Security Administrator from Microsoft, Certified Azure Security Engineer Associate from Microsoft, Cyber Crime Intervention Officer from ISAC India, and Network Security Expert from FORTINET.

I would like to thank my family, who have always motivated me to grow in my life and career. I would like to thank my friends and employers, who have always stood by me. My friends, Aphin Alexander, Rajdeep Gogoi, Prafull Kurekar, and Kanchan Jhangiani, have always been there for me. I would also like to thank Anubhav Kumar Lal and Ravali Vangala for giving me a reason to continue learning and growing.

1
Nmap Fundamentals

Network Mapper (**Nmap**) was originally released by *Gordon Lyon*, known on the internet as *Fyodor*, in the infamous *Phrack magazine Vol. 7 Issue 51* (`https://nmap.org/p51-11.html`). It is still acclaimed today as one of the best tools for network reconnaissance and security auditing in cybersecurity. The first public version was introduced as an advanced port scanner along with a paper describing research on novel techniques for port discovery, but since then, it has gone down a long road and become so much more. The Nmap project itself evolved into a family of advanced networking tools that includes amazing projects such as Ncrack, Ncat, Nping, Zenmap, and, built into Nmap itself, the **Nmap Scripting Engine** (**NSE**). Fyodor's own description on the official website is as follows:

"Nmap (Network Mapper) is a free and open source (license) utility for network discovery and security auditing. Many systems and network administrators also find it useful for tasks such as network inventory, managing service upgrade schedules, and monitoring host or service uptime. Nmap uses raw IP packets in novel ways to determine what hosts are available on the network, what services (application name and version) those hosts are offering, what operating systems (and OS versions) they are running, what type of packet filters/firewalls are in use, and dozens of other characteristics. It was designed to rapidly scan large networks, but works fine against single hosts. Nmap runs on all major computer operating systems, and official binary packages are available for Linux, Windows, and Mac OS X."

Nmap's community is very active, so I encourage you to always keep up with the latest stable releases and patches. Announcements and discussions take place on the development mailing list, so if you would like to contribute to the project, I recommend you subscribe to the mailing list at `https://nmap.org/mailman/listinfo/dev`. These days, you will also find a GitHub repository serving as the official mirror from the Subversion code repository. For issues and pull requests, it is recommended to create them on GitHub and send a friendly reminder to the mailing list so they are easier to track and to avoid them getting lost in all the noise.

This first chapter is for newcomers to Nmap and its projects. It aims to give you a general overview of the main capabilities of the Nmap project. Starting with building Nmap projects from source code, you will become familiar with all the tools of the Nmap project. In just the initial recipes, you will learn how flexible and powerful the Nmap tools are, but as we move through the chapters, you will go deep into the internals to learn how to not only use the tools for a wide range of tasks useful in the cybersecurity field but also extend them and create new functionality by writing your own modules in Lua or C. The practical tasks chosen for this chapter will get you started with Nmap and the most common options and features to start scanning targets and customizing scans.

In this chapter, we will cover the following recipes:

- Building Nmap's source code
- Finding online hosts
- Listing open ports on a target
- Fingerprinting OSes and services running on a target
- Using NSE scripts against a target host
- Scanning random targets on the internet
- Collecting signatures of web servers
- Scanning with Rainmap Lite

Technical requirements

The following tools are officially part of the Nmap project and were created to accomplish common tasks for network diagnostics and security scanning:

- **Nping** (`https://nmap.org/nping/`) specializes in custom network packet crafting for diagnostics and troubleshooting.
- **Ncrack** (`https://nmap.org/ncrack/`) focuses on network authentication cracking, supporting the most popular applications and protocols.

- **Ncat** (`https://nmap.org/ncat/`) is an enhanced version of Netcat that supports encryption out of the box and is extensible using Lua scripts.
- **Zenmap** (`https://nmap.org/zenmap/`) is a cross-platform GUI for Nmap focused on usability.
- **NSE** (`https://nmap.org/book/nse.html`) takes information obtained from scanned targets and provides an interface for users to script additional tasks using Lua.

Building Nmap's source code

Throughout this book, you will use all the tools from the Nmap project, so it is a good idea to start by installing the latest versions now. We will not work with pre-built binaries as mere mortals but build them from the latest source code available in the official repository. This recipe will show how to download the latest copy of the source code from the development repositories and compile and install Nmap and related tools in your Unix-based system.

We always prefer working with the very latest snapshot of the repository because precompiled packages take time to prepare and we will often miss important patches or new NSE scripts. The following recipe will show the process of downloading the source code and configuring, building, installing, and maintaining an up-to-date copy of the Nmap project in your arsenal.

Getting ready

Before continuing, you need to have installed the Subversion client. Unix-based platforms come with a command-line client named **Subversion** (**svn**). To check whether it's already installed on your system, just open a terminal and type the following command:

```
$ svn
```

If the command was not found, install `svn` using your favorite package manager or build it from source code. The instructions to build `svn` from source code are out of the scope of this book, but they are widely documented online. Use your favorite search engine to find specific instructions for your system.

When building Nmap, we will also need additional libraries such as the development definitions from **OpenSSL** or the `make` command. In Debian-based systems, try the following command to install the missing dependencies:

```
#apt-get install libssl-dev autoconf make g++ subversion
```

Note that OpenSSL is optional, and Nmap can be built without it; however, without it, Nmap will be crippled as it uses it for functions related to integers, hashing, and encoding/decoding SSL requests for service detection and NSE.

How to do it...

1. Start by grabbing a copy of the source code from the official Subversion repository. To download the latest development branch, use the `svn checkout` command. This command can also be used through the `co` alias:

   ```
   $svn co https://svn.nmap.org/nmap
   ```

2. This command will start downloading and listing the files and when it finishes, the **Checked out revision <Revision number>** message will be shown. A new directory containing the source code is now available in your current working directory. At this point, you should have installed all the required dependencies and you will be ready to compile Nmap with the standard Unix compilation procedure by running `configure`, `make`, and `make install`. Enter the directory containing the source code and start with the `configure` command:

   ```
   $./configure
   ```

3. If the configuration process completes successfully, you should also see the configuration options applied:

   ```
   Configured with: ndiff zenmap nping openssl zlib libssh2
   lua ncat
   Configured without: localdirs nmap-update
   Type make (or gmake on some *BSD machines) to compile.
   ```

4. Compile Nmap with `make`:

   ```
   $make
   ```

5. When it finishes building Nmap and the other tools, you will be able to find the `nmap` binary in your current working directory. Finally, make it available system-wide by installing Nmap on the system:

   ```
   #make install
   ```

After installing the application, you should see the **NMAP SUCCESSFULLY INSTALLED** message and now you can run Nmap from any path on the system. Test your Nmap installation and learn about the supported scanning techniques and options with the help command:

```
$nmap -h
```

How it works...

The `svn` repository, hosted at `https://svn.nmap.org/nmap`, contains the latest development version of Nmap and has world read access that allows anyone to grab a copy of the source code. We built the project from scratch to get the latest patches and features. The installation process described in this recipe also installed Ncat, Zenmap, Ndiff, and Nping.

There's more...

The process of compiling Nmap is similar to compiling other Unix-based applications, but there are several compile-time variables that can be adjusted to configure the installation. Precompiled binaries are recommended for users who can't compile Nmap from source code. Unix-based systems are recommended because of some Windows limitations that affect performance, described at `https://nmap.org/book/inst-windows.html`.

Experimental branches

If you want to try the latest creations of the development team, there is a folder named `nmap-exp` that contains several experimental branches of the project. The code stored in this folder is not guaranteed to work all the time as it is used as a sandbox by developers, although some hidden gems can be found there from time to time. These branches are located at `https://svn.nmap.org/nmap-exp/`.

Updating your local working copy

The Nmap project is quite active, especially during summer because of Google Summer of Code, so do not forget to update your installed copy regularly. If you keep a working copy of the svn repository, `https://svn.nmap.org/nmap`, you could update it with the following commands inside your `svn` working directory:

```
$svn up
$make -j4
#make install
```

Customizing the building process

If you do not need the other Nmap utilities, such as Nping, Ncat, Ndiff, or Zenmap, you may use different configure directives to omit their installation during the configuration step:

```
./configure --without-ndiff
./configure –without-ncat
./configure --without-zenmap
./configure --without-nping
```

For a complete list of configuration directives, use the `--help` command argument:

```
$./configure --help
```

Precompiled packages

Precompiled Nmap packages can be found for all major platforms at `https://nmap.org/download.html` for those who do not feel like setting up the build environment. When working with precompiled packages, just make sure that you grab the latest version to avoid missing important fixes or enhancements. This is especially important with Windows and the Npcap driver, which has gone through some serious improvements.

Finding online hosts

Finding online hosts in networks or on the internet is a common task among penetration testers and system administrators. Nmap offers better host detection as it sends more probes than the ICMP echo request sent by the traditional `ping` utility.

This recipe describes how to determine whether a host is online with Nmap.

How to do it...

Launch a `ping` scan against a target to determine whether it is online using the following command:

```
#nmap -sn <target>
```

The results will include all hosts that responded to any of the packets sent by Nmap during the ping scan, that is, the active machines on the target network segment or the internet. Nmap takes as a target any option not recognized and it supports IPv4/IPv6 addresses, hostnames, and network ranges that can be defined using wildcards and **Classless Inter-Domain Routing** (**CIDR**) notation. For example, to scan the local network, `192.168.0.1/24`, you can run the following command:

```
#nmap -sn 192.168.0.1/24
Nmap scan report for 192.168.0.1 Host is up (0.0025s latency).
MAC Address: F4:B7:E2:0A:DA:18 (Hon Hai Precision Ind.) Nmap
scan report for 192.168.0.2
Host is up (0.0065s latency).
MAC Address: 00:18:F5:0F:AD:01 (Shenzhen Streaming Video
Technology Company Limited)
Nmap scan report for 192.168.0.3 Host is up (0.00015s latency).
MAC Address: 9C:2A:70:10:84:BF (Hon Hai Precision Ind.) Nmap
scan report for 192.168.0.8
Host is up (0.029s latency).
MAC Address: C8:02:10:39:54:D2 (LG Innotek) Nmap scan report
for 192.168.0.10
Host is up (0.0072s latency).
MAC Address: 90:F6:52:EE:77:E9 (Tp-link Technologies) Nmap scan
report for 192.168.0.11
Host is up (0.030s latency).
MAC Address: 80:D2:1D:2C:20:55 (AzureWave Technology) Nmap scan
report for 192.168.0.18
Host is up (-0.054s latency).
MAC Address: 78:31:C1:C1:9C:0A (Apple)
Nmap scan report for 192.168.0.22 Host is up (0.030s latency).
MAC Address: F0:25:B7:EB:DD:21 (Samsung Electro Mechanics) Nmap
scan report for 192.168.0.5
Host is up.
Nmap done: 256 IP addresses (9 hosts up) scanned in 27.86
seconds
```

Ping scans in Nmap may also identify MAC addresses and vendors based on the MAC address identifier if executed as a privileged user on local Ethernet networks.

How it works...

The Nmap `-sn` option disables port scanning, leaving only the host discovery phase enabled, which makes Nmap perform a **ping scan** or **ping sweep**. Depending on the privileges, Nmap by default uses different techniques: sending a `TCP SYN` packet to port `443`, a `TCP ACK` packet to port `80`, and an ICMP echo and timestamp requests if executed as a privileged user. If the user running Nmap can't send raw packets, it sends a `SYN` packet to ports `80` and `443` via `connect() syscall`. **ARP/Neighbor Discovery** is also enabled when scanning local Ethernet networks as privileged users. MAC addresses and vendors are identified from the ARP requests sent during the ARP/Neighbor Discovery phase.

There's more...

Nmap supports several host and port discovery techniques, and probes can be customized to scan hosts effectively even in the most restricted environments. It is important that we grasp how these network scanning techniques work. Let's learn more about host discovery with Nmap.

Tracing routes

Ping scans allow including traceroute information of the targets. Use the Nmap `--traceroute` option to trace the route from the scanning machine to the target host:

```
$ nmap -sn --traceroute google.com microsoft.com
Nmap scan report for google.com (216.58.193.46) Host is up
(0.16s latency).
Other addresses for google.com (not scanned):
2607:f8b0:4012:805::200e
rDNS record for 216.58.193.46: qro01s13-in-f14.1e100.net

TRACEROUTE (using port 443/tcp) HOP RTT     ADDRESS
1     1.28 ms  192.168.0.1
2     ...
3     158.85 ms 10.165.1.9
4     ... 5
6     165.50 ms 10.244.158.13
7     171.18 ms 10.162.0.254
8     175.33 ms 200.79.231.81.static.cableonline.com.mx
(200.79.231.81)
9     183.16 ms 10.19.132.97
10    218.60 ms 72.14.203.70
11    223.35 ms 209.85.240.177
```

```
12    242.60 ms 209.85.142.47
13    ...
14    234.79 ms 72.14.233.237
15    235.17 ms qro01s13-in-f14.1e100.net (216.58.193.46)
Nmap scan report for microsoft.com (23.96.52.53) Host is up
(0.27s latency).
Other addresses for microsoft.com (not scanned): 23.100.122.175
104.40.211.35 104.43.195.251 191.239.213.197
TRACEROUTE (using port 443/tcp) HOP RTT    ADDRESS
-    Hops 1-9 are the same as for 216.58.193.46 10
183.27 ms 10.19.132.30
11   231.26 ms 206.41.108.25
12   236.77 ms ae5-0.atb-96cbe-1c.ntwk.msn.net (104.44.224.230)
13   226.22 ms be-3-0.ibr01.bn1.ntwk.msn.net (104.44.4.49)
14   226.89 ms be-1-0.ibr02.bn1.ntwk.msn.net (104.44.4.63)
15   213.92 ms be-3-0.ibr02.was05.ntwk.msn.net (104.44.4.26)
16   251.91 ms ae71-0.bl2-96c-1b.ntwk.msn.net (104.44.8.173)
17   ... 19
20   220.70 ms 23.96.52.53
Nmap done: 2 IP addresses (2 hosts up) scanned in 67.85 seconds
```

Running NSE during host discovery

NSE can be enabled during the host discovery phase to obtain additional information about a target. As with any other NSE script, its execution will depend on the **hostrule** specified. To execute an NSE script without port scanning our targets, we skip port scanning with `-sn` and use `--script <file,folder,category>` to select the desired script:

```
$ nmap -sn --script dns-brute websec.mx
Nmap scan report for websec.mx (54.210.49.18) Host is up.
rDNS record for 54.210.49.18: ec2-54-210-49-18.compute-
1.amazonaws.com

Host script results:
| dns-brute:
|     DNS Brute-force hostnames:
|     ipv6.websec.mx - 54.210.49.18
|     web.websec.mx - 198.58.116.134
|     www.websec.mx - 54.210.49.18
|_    beta.websec.mx - 54.210.49.18
```

An interesting NSE script to try when discovering online hosts in networks is the `broadcast-ping` script, which uses a broadcast ping request to attempt to discover online hosts:

```
$ nmap -sn --script broadcast-ping 192.168.0.1/24
Pre-scan script results:
| broadcast-ping:
|     IP: 192.168.0.11    MAC: 80:d2:1d:2c:20:55
|     IP: 192.168.0.18    MAC: 78:31:c1:c1:9c:0a
|_    Use --script-args=newtargets to add the results as
targets
```

Exploring more host discovery scanning techniques

Nmap supports several host discovery scanning techniques using different protocols. By default, the host discovery phase (`nmap -sn <target>`) only scans as a privileged user internally executes Nmap with the `-PS443 -PA80 -PE -PP` options corresponding to `TCP SYN` to port `443`, `TCP ACK` to port `80`, and ICMP echo and timestamps requests.

In *Chapter 3*, *Network Scanning*, you will learn more about the following ping scanning techniques supported by Nmap:

- `-PS/PA/PU/PY [portlist]`: TCP SYN/ACK, UDP, or SCTP discovery to given ports
- `-PE/PP/PM`: ICMP echo, timestamp, and netmask request discovery probes
- `-PO [protocol list]`: IP protocol ping

Listing open ports on a target

This recipe describes how to use Nmap to determine the port states of a target, a process used to identify running services commonly referred to as port scanning. This is one of the tasks Nmap excels at, so it is important to learn about the essential Nmap options related to port scanning.

How to do it...

To launch a default scan, the bare minimum you need is a target. A target can be an IP address, a hostname, or a network range:

```
$ nmap scanme.nmap.org
```

The scan results will show all the host information obtained, such as the IPv4 (and IPv6 if available) address, reverse DNS name, and interesting ports with service names. All listed ports have a state. Ports marked as open or filtered are of special interest as they represent services running on the target host:

```
Nmap scan report for scanme.nmap.org (45.33.32.156)
Host is up (0.16s latency).
Other addresses for scanme.nmap.org (not scanned):
2600:3c01::f03c:91ff:fe18:bb2f
Not shown: 995 closed ports PORT STATE SERVICE
22/tcp    open  ssh 25/tcp filtered smtp 80/tcp open http
9929/tcp  open nping-echo 31337/tcp open   Elite
Nmap done: 1 IP address (1 host up) scanned in 333.35 seconds
```

How it works...

The default Nmap scan returns a list of ports. In addition, it returns a service name from a database distributed with Nmap and the port state for each of the listed ports.

Nmap categorizes ports into the following states:

- **Open**: Open indicates that a service is listening for connections on this port.
- **Closed**: Closed indicates that the probes were received, but it was concluded that there was no service running on this port.
- **Filtered**: Filtered indicates that there were no signs that the probes were received and the state could not be established. This could indicate that the probes are being dropped by some kind of filtering.
- **Unfiltered**: Unfiltered indicates that the probes were received but a state could not be established.
- **Open/Filtered**: This indicates that the port was filtered or open but the state could not be established.
- **Closed/Filtered**: This indicates that the port was filtered or closed but the state could not be established.

Even for this simple port scan, Nmap does many things in the background that can be configured as well. Nmap begins by converting the hostname to an IPv4 address using DNS name resolution. If you wish to use a different DNS server, use `--dns-servers <serv1[,serv2],...>`, or use `-n` if you wish to skip this step, as follows:

```
$ nmap --dns-servers 8.8.8.8,8.8.4.4 scanme.nmap.org
```

Afterward, it performs the host discovery process to check whether the target is online (see the *Finding online hosts* recipe). To skip this step, use the no ping option, `-Pn`:

```
$ nmap -Pn scanme.nmap.org
```

Nmap then converts the IPv4 or IPv6 address back to a hostname using a reverse DNS query. Use `-n` to skip this step as well if you do not need that information:

```
$ nmap -n scanme.nmap.org
```

The previous command will launch either a SYN stealth scan or a TCP connect scan depending on the privileges of the user running Nmap.

There's more...

Port scanning is one of the most powerful features available, and it is important that we understand the different techniques and options that affect the scan behavior of Nmap.

Privileged versus unprivileged

Running the simplest port scan command, `nmap <target>`, as a privileged user by default launches a **SYN stealth scan**, whereas unprivileged users that cannot create raw packets use the **TCP connect scan** technique. The difference between these two techniques is that a TCP connect scan uses the high-level `connect()` system call to obtain the port state information, meaning that each TCP connection is fully completed and therefore slower. SYN stealth scans use raw packets to send specially crafted TCP packets to detect port states with a technique known as **half-open**.

Scanning specific port ranges

Setting port ranges correctly during your scans is a task you often need to do when running Nmap scans. You can also use this to filter machines that run a service on a specific port, for example, finding all the SMB servers open in port `445`. Narrowing down the port list also optimizes performance, which is very important when scanning multiple targets.

There are several ways of using the Nmap `-p` option:

- Port list separated by commas: `$ nmap -p80,443 localhost`
- Port range denoted with hyphens: `$ nmap -p1-100 localhost`
- Alias for all ports from `1` to `65535`: `# nmap -p- localhost`
- Specific ports by protocol: `# nmap -pT:25,U:53 <target>`

- Service name: `# nmap -p smtp <target>`
- Service name with wildcards: `# nmap -p smtp* <target>`
- Only ports registered in the Nmap services database: `# nmap -p[1-65535] <target>`

Selecting a network interface

Nmap attempts to automatically detect your active network interface; however, there are some situations where it will fail or perhaps you will need to select a different interface in order to test networking issues. To force Nmap to scan using a different network interface, use the `-e` argument:

```
#nmap -e <interface> <target>
#nmap -e eth2 scanme.nmap.org
```

This is only necessary if you have problems with broadcast scripts or see the **WARNING: Unable to find appropriate interface for system route to** message.

More port scanning techniques

In this recipe, we talked about the two default scanning methods used in Nmap: SYN stealth scan and TCP connect scan. However, Nmap supports several *more advanced port scanning* techniques. Use `nmap -h` or visit `https://nmap.org/book/man-port-scanning-techniques.html` to learn more about them as Fyodor has done a fantastic job describing how they work in depth.

Target specification

Nmap supports several target formats that allow users to work with **IP address ranges**. The most common type is when we specify the target's IP or host, but it also supports the reading of targets from files and ranges, and we can even generate a list of random targets as we will see later.

Any arguments that are not valid options are read as targets by Nmap. This means that we can tell Nmap to scan more than one range in a single command, as shown in the following command:

```
# nmap -p25,80 -O -T4 192.168.1.1/24 scanme.nmap.org/24
```

There are several ways that we can handle IP ranges in Nmap:

- Multiple host specification
- Octet range addressing (they also support wildcards)
- CIDR notation

To scan the `192.168.1.1`, `192.168.1.2`, and `192.168.1.3` IP addresses, the following command can be used:

```
$ nmap 192.168.1.1 192.168.1.2 192.168.1.3
```

We can also specify octet ranges using `-`. For example, to scan hosts `192.168.1.1`, `192.168.1.2`, and `192.168.1.3`, we could use the expression `192.168.1.1-3`, as shown in the following command:

```
$ nmap 192.168.1.1-3
```

Octet range notation also supports wildcards, so we could scan from `192.168.1.0` to `192.168.1.255` with the expression `192.168.1.*`:

```
$ nmap 192.168.1.*
```

Excluding hosts from scans

In addition, you may exclude hosts from the ranges by specifying the `--exclude` option, as shown next:

```
$ nmap 192.168.1.1-255 --exclude 192.168.1.1
$ nmap 192.168.1.1-255 --exclude 192.168.1.1,192.168.1.2
```

Otherwise, you can write your exclusion list in a file using the `--exclude-file` option:

```
$ cat dontscan.txt
192.168.1.1
192.168.1.254
$ nmap --exclude-file dontscan.txt 192.168.1.1-255
```

CIDR notation for targets

The CIDR notation (pronounced *cider*) is a compact method for specifying IP addresses and their routing suffixes. This notation gained popularity due to its granularity when compared with classful addressing because it allows subnet masks of variable length.

The CIDR notation is specified by an IP address and network suffix. The network or IP suffix represents the number of network bits. IPv4 addresses are 32-bit, so the network can be between 0 and 32. The most common suffixes are `/8`, `/16`, `/24`, and `/32`.

To visualize it, take a look at the following CIDR-to-netmask conversions:

- `/8`: `255.0.0.0`
- `/16`: `255.255.0.0`
- `/24`: `255.255.255.0`
- `/32`: `255.255.255.255`

For example, `192.168.1.0/24` represents the 256 IP addresses from `192.168.1.0` to `192.168.1.255`. `50.116.1.121/8` represents all the IP addresses between `50.0-255.0-255.0-255`. The `/32` network suffix is also valid and represents a single IP address.

The CIDR notation can also be used when specifying targets. To scan the 256 hosts in `192.168.1.0-255` using the CIDR notation, you will need the `/24` suffix:

```
$ nmap 192.168.1.0/24
```

Working with target lists

Many times, we will need to work with multiple targets, but having to type a list of targets in the command line is not very practical. Fortunately, Nmap supports the loading of targets from an external file. Enter the list of targets into a file, each separated by a new line, tab, or space(s):

```
$cat targets.txt
192.168.1.23
192.168.1.12
```

To load the targets from the `targets.txt` file, use the Nmap `-iL <filename>` option:

```
$ nmap -iL targets.txt
```

Important note

This feature can be combined with any scan option or method, except for exclusion rules set by `--exclude` or `--exclude-file`. The `--exclude` and `--exclude-file` options will be ignored when `-iL` is used.

You can also use different target formats in the same file. In the following file, we specify an IP address and an IP range inside the same file:

```
$ cat targets.txt
192.168.1.1
192.168.1.20-30
```

You can enter comments in your target list by starting the new line with the # character:

```
$ cat targets.txt
# FTP servers 192.168.10.3
192.168.10.7
192.168.10.11
```

Fingerprinting OSes and services running on a target

Version detection and **OS detection** are two of the most important features of Nmap. Nmap is known for having the most comprehensive OS and service fingerprint databases, contributed to over the years by millions of users. Knowing the OS and the exact software version of a service is highly valuable for people looking for security vulnerabilities or monitoring their networks for any unauthorized changes. Fingerprinting services may also reveal additional information about a target, such as available modules, last time of update, database version, and sometimes additional protocol information.

This recipe shows how to fingerprint the OS and running services of a remote host using Nmap.

How to do it...

1. To enable service detection, add the Nmap `-sV` option to your port scan command:

   ```
   $ nmap -sV <target>
   ```

2. The `-sV` option adds an additional column named `VERSION` that displays the specific software version. Additional information can be found enclosed in parentheses:

   ```
   $ nmap -sV scanme.nmap.org
   Nmap scan report for scanme.nmap.org (45.33.32.156) Host
   is up (1.4s latency).
   Other addresses for scanme.nmap.org (not scanned):
   2600:3c01::f03c:91ff:fe18:bb2f
   ```

```
Not shown: 994 closed ports
PORT     STATE      SERVICE  VERSION
22/tcp    open      ssh    OpenSSH 6.6.1p1 Ubuntu
2ubuntu2.3
(Ubuntu Linux; protocol 2.0) 25/tcp   filtered smtp
80/tcp    open  http  Apache httpd 2.4.7 ((Ubuntu)) 514/
tcp   filtered shell
9929/tcp   open  nping-echo Nping echo 31337/tcp open
tcpwrapped
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect
results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 137.71
seconds
```

3. To enable OS detection, add the Nmap `-O` option to your scan command. Note that OS detection requires Nmap to be run as a privileged user:

```
# nmap -O <target>
```

4. The result will now include OS information at the bottom of the port list:

```
# nmap -O scanme.nmap.org
Nmap scan report for scanme.nmap.org (45.33.32.156)
Host is up (0.25s latency).
Other addresses for scanme.nmap.org (not scanned):
2600:3c01::f03c:91ff:fe18:bb2f
Not shown: 994 closed ports
PORT      STATE   SERVICE
22/tcp   open     ssh
25/tcp   filtered smtp
80/tcp   open     http
514/tcp  filtered shell
9929/tcp open    nping-echo
31337/tcp open  Elite
Device type: WAP|general purpose|storage-misc
Running (JUST GUESSING): Actiontec embedded (99%), Linux
2.4.X|3.X (99%), Microsoft Windows 7|2012|XP (96%),
BlueArc embedded (91%)
OS CPE: cpe:/h:actiontec:mi424wr-gen3i cpe:/
o:linux:linux_kernel cpe:/o:linux:linux_kernel:2.4.37
cpe:/o:linux:linux_kernel:3.2 cpe:/o:microsoft:windows_7
cpe:/o:microsoft:windows_server_2012
```

```
cpe:/o:microsoft:windows_xp::sp3 cpe:/
h:bluearc:titan_2100 Aggressive OS guesses: Actiontec
MI424WR-GEN3I WAP (99%), DD-WRT v24-sp2 (Linux 2.4.37)
(98%), Linux 3.2 (98%), Microsoft Windows
7 or Windows Server 2012 (96%), Microsoft Windows XP SP3
(96%),
BlueArc Titan 2100 NAS device (91%)
No exact OS matches for host (test conditions non-ideal).
OS detection performed. Please report any incorrect
results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 114.03
seconds
```

How it works...

The Nmap `-sV` option enables service detection, which returns additional service and version information. Service detection is one of the most loved features of Nmap because it is very useful in many situations, such as when identifying security vulnerabilities, making sure a service is running on a given port, or checking whether a patch or update pack has been applied successfully.

This feature works by sending several probes defined in the `nmap-service-probes` file to the list of detected open ports. The probes are selected based on how likely it is they can be used to identify a service based on the port number and a score that determines the rareness of the service.

Important note

If you are interested in the inner workings, you can find very detailed documentation on how service detection mode works and how the file formats are used at `https://nmap.org/book/vscan.html`.

Similarly, the `-O` option tells Nmap to attempt OS detection by sending several probes to the TCP, UDP, and ICMP protocols against opened and closed ports. OS detection mode is very powerful due to Nmap's user community, which contributes fingerprints that identify a wide variety of systems, including residential routers, IP webcams, OSes, and many other hardware devices. It is important to note that OS detection requires raw packets, so Nmap needs to be run in privileged mode.

Important note

The complete documentation of the tests and probes sent during OS detection can be found at `https://nmap.org/book/osdetect-methods.html`.

Nmap uses **Common Platform Enumeration** (**CPE**) as the naming scheme for service and OS detection. This convention is used in the information security industry to identify packages, platforms, and systems.

There's more...

OS and version detection scan options can be customized thoroughly and are very powerful when tuning performance. Let's learn about some additional Nmap options related to these scan modes.

Increasing version detection intensity to detect odd services

You can increase or decrease the probes that get sent during version detection by changing the version detection intensity level of the scan with the `–version-intensity <level from 0 to 9>` parameter:

```
$ nmap -sV --version-intensity 9 <target>
```

This Nmap option is incredibly effective against services running on non-default ports due to configuration changes or services that are very rare and are likely to be skipped during a scan.

Aggressive detection mode

Nmap has the special `-A` parameter to activate aggressive detection mode. Aggressive mode enables OS detection (`-O`), version detection (`-sV`), script scanning (`-sC`), and traceroute (`--traceroute`). This mode sends a lot of specially crafted probes, and it is more likely to be detected, but provides a lot of valuable target information. You can try aggressive detection with the following command:

```
# nmap -A <target>
Nmap scan report for scanme.nmap.org (45.33.32.156) Host is up
(0.071s latency).
Other addresses for scanme.nmap.org (not scanned):
2600:3c01::f03c:91ff:fe18:bb2f
Not shown: 994 closed ports
PORT STATE SERVICE    VERSION
```

```
22/tcp    open  ssh   OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.3 (Ubuntu
Linux; protocol 2.0)
| ssh-hostkey:
|    1024 ac:00:a0:1a:82:ff:cc:55:99:dc:67:2b:34:97:6b:75 (DSA)
|    2048 20:3d:2d:44:62:2a:b0:5a:9d:b5:b3:05:14:c2:a6:b2 (RSA)
|_   256 96:02:bb:5e:57:54:1c:4e:45:2f:56:4c:4a:24:b2:57
(ECDSA)
25/tcp     filtered smtp
80/tcp     open  http      Apache httpd 2.4.7 ((Ubuntu))
|_http-server-header: Apache/2.4.7 (Ubuntu)
|_http-title: Go ahead and ScanMe! 514/tcp  filtered shell
9929/tcp  open nping-echo Nping echo 31337/tcp open
tcpwrapped
Device type: WAP|general purpose|storage-misc
Running (JUST GUESSING): Actiontec embedded (98%), Linux
2.4.X|3.X (98%), Microsoft Windows 7|2012|XP (96%), BlueArc
embedded (91%) OS CPE: cpe:/h:actiontec:mi424wr-gen3i cpe:/
o:linux:linux_kernel cpe:/o:linux:linux_kernel:2.4.37 cpe:/
o:linux:linux_kernel:3.2 cpe:/o:microsoft:windows_7 cpe:/
o:microsoft:windows_server_2012
cpe:/o:microsoft:windows_xp::sp3 cpe:/h:bluearc:titan_2100
Aggressive OS guesses: Actiontec MI424WR-GEN3I WAP (98%),
DD-WRT v24-sp2 (Linux 2.4.37) (98%), Linux 3.2 (98%), Microsoft
Windows 7 or Windows Server 2012 (96%), Microsoft Windows XP
SP3 (96%), BlueArc Titan 2100 NAS device (91%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 2 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
TRACEROUTE (using port 80/tcp)
HOP RTT    ADDRESS
1     0.08 ms 192.168.254.2
2     0.03 ms scanme.nmap.org (45.33.32.156)
OS and Service detection performed. Please report any incorrect
results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 208.05 seconds
```

Configuring OS detection

If OS detection fails, you can use `--osscan-guess` to force Nmap to guess the OS:

```
# nmap -O --osscan-guess <target>
```

To launch OS detection only when the scan conditions are ideal, use `--osscan-limit`:

```
# nmap -O --osscan-limit <target>
```

OS detection in verbose mode

Try OS detection in verbose mode to see additional target information, such as the TCP and IP ID sequence number values:

```
# nmap -O -v <target>
```

The IP ID sequence number can be found under the **IP ID Sequence Generation** label. Note that incremental IP ID sequence numbers can be abused by port scanning techniques such as **idle scan**, which will use this value to predict whether a service is open or not when spoofing the real connection origin:

```
# nmap -O -v 192.168.0.1
Initiating Ping Scan at 11:14 Scanning 192.168.0.1 [4 ports]
Completed Ping Scan at 11:14, 0.00s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 11:14
Completed Parallel DNS resolution of 1 host. at 11:14, 0.02s
elapsed
Initiating SYN Stealth Scan at 11:14 Scanning 192.168.0.1 [1000
ports] Discovered open port 80/tcp on 192.168.0.1
Completed SYN Stealth Scan at 11:14, 13.80s elapsed (1000 total
ports)
Initiating OS detection (try #1) against 192.168.0.1 Retrying
OS detection (try #2) against 192.168.0.1 Nmap scan report for
192.168.0.1
Host is up (0.11s latency). Not shown: 998 closed ports PORT
STATE   SERVICE
80/tcp     open  http 514/tcp filtered shell
Device type: WAP|general purpose|storage-misc
Running (JUST GUESSING): Actiontec embedded (99%), Linux
2.4.X|3.X (99%), Microsoft Windows 7|2012|XP (96%), BlueArc
embedded (91%) OS CPE: cpe:/h:actiontec:mi424wr-gen3i
cpe:/o:linux:linux_kernel cpe:/o:linux:linux_kernel:2.4.37
cpe:/o:linux:linux_kernel:3.2 cpe:/o:microsoft:windows_7
cpe:/o:microsoft:windows_server_2012 cpe:/o:microsoft:windows_
xp::sp3 cpe:/h:bluearc:titan_2100 Aggressive OS guesses:
Actiontec MI424WR-GEN3I WAP (99%), DD-WRT v24-sp2 (Linux
2.4.37) (98%), Linux 3.2 (97%),
Microsoft Windows 7 or Windows Server 2012 (96%), Microsoft
Windows XP SP3 (96%),
BlueArc Titan 2100 NAS device (91%)
No exact OS matches for host (test conditions non-ideal). TCP
Sequence Prediction: Difficulty=259 (Good luck!)
IP ID Sequence Generation: Incremental

Read data files from: /usr/local/bin/../share/nmap
OS detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 32.40 seconds
Raw packets sent: 1281 (59.676KB) | Rcvd: 1249 (50.520KB)
```

Submitting new fingerprints for OS and service detection

Nmap's result accuracy comes from a database that has been collected over the years through user submissions. It is very important that we help keep this database up to date. Nmap will let you know when it encounters an unknown signature and will kindly ask you to contribute to the project by submitting an unidentified OS, device, or service.

Please take the time to submit your contributions, as Nmap's detection capabilities come directly from these databases. Visit `https://nmap.org/cgi-bin/submit.cgi?` to submit new fingerprints or corrections.

Using NSE scripts against a target host

The Nmap project introduced a feature named the Nmap Scripting Engine that allows users to extend the capabilities of Nmap via Lua scripts. NSE scripts are very powerful and have become one of Nmap's main strengths, performing tasks from advanced version detection to vulnerability exploitation. At the moment, there are more than 600 scripts helping users perform a wide range of tasks using the target information obtained from the executed scan. Using host and port rules, they can even be configured to run without port scanning a target, something that is really useful during reconnaissance tasks.

This recipe describes how to run NSE scripts, and the different options available to configure their execution.

How to do it...

Enable **script scan** using the Nmap `-sC` option. This mode will select all NSE scripts belonging to the default category and execute them against our targets based on their host and port rules:

```
$ nmap -sC <target>
$ nmap -sC scanme.nmap.org
Nmap scan report for scanme.nmap.org (45.33.32.156)
Host is up (0.14s latency).
Other addresses for scanme.nmap.org (not scanned):
2600:3c01::f03c:91ff:fe18:bb2f
Not shown: 995 closed ports
PORT       STATE      SERVICE
22/tcp     open       ssh
| ssh-hostkey:
|     1024 ac:00:a0:1a:82:ff:cc:55:99:dc:67:2b:34:97:6b:75
(DSA)
|     2048 20:3d:2d:44:62:2a:b0:5a:9d:b5:b3:05:14:c2:a6:b2
(RSA)
|_    256 96:02:bb:5e:57:54:1c:4e:45:2f:56:4c:4a:24:b2:57
(ECDSA)
25/tcp     filtered smtp 80/tcp  open  http
|_http-title: Go ahead and ScanMe! 9929/tcp open  nping-echo
31337/tcp open  Elite
Nmap done: 1 IP address (1 host up) scanned in 24.42 seconds
```

In this case, the results included the output of the `ssh-hostkey` and `http-title` scripts. When the script runs and finds additional information, it will include the results in the output. Nmap will likely run more scripts than the output shows but NSE scripts are only shown when they obtain results.

How it works...

The Nmap `-sC` option enables script scan mode, which tells Nmap to select the default scripts and execute them if the host or port rule matches.

NSE scripts are divided into the following categories:

- **auth**: This category is for scripts related to user authentication.
- **broadcast**: This is a very interesting category of scripts that use broadcast petitions to gather information.

- **brute**: This category is for scripts that conduct brute-force password auditing attacks.
- **default**: This category is for scripts that are executed when a script scan is executed (`-sC`). Scripts in this category are considered safe and non-intrusive.
- **discovery**: This category is for scripts related to host and service discovery.
- **dos**: This category is for scripts related to denial-of-service attacks.
- **exploit**: This category is for scripts that exploit security vulnerabilities.
- **external**: This category is for scripts that depend on a third-party service.
- **fuzzer**: This category is for NSE scripts that are focused on fuzzing.
- **intrusive**: This category is for scripts that might crash something or generate a lot of network noise; scripts that system administrators may consider intrusive belong to this category.
- **malware**: This category is for scripts related to malware detection.
- **safe**: This category is for scripts that are considered safe in all situations.
- **version**: This category is for scripts that are used for the advanced versioning of services.
- **vuln**: This category is for scripts related to security vulnerabilities.

There's more...

Let's learn about some Nmap options that are required to customize NSE. Some scripts require being configured correctly, so it is important that we are familiar with all the NSE options.

NSE script arguments

The Nmap `--script-args` parameter is used to set the arguments of NSE scripts. For example, if you would like to set the `useragent` HTTP library argument, add the following argument:

```
$ nmap --script http-title --script-args http.
useragent="Mozilla 4.20" <target>
```

A feature that is not very well known is script argument aliases. You can use aliases when setting the arguments for NSE scripts. For example, say you were setting the script argument path as follows:

```
$ nmap -p80 --script http-trace --script-args http-trace.path
<target>
```

You could instead just write the following:

```
$ nmap -p80 --script http-trace --script-args path <target>
```

While in this particular example you don't save yourself from typing that much, it really helps in longer and more complex commands.

Script selection

Users may select specific scripts when scanning using the Nmap `--script <filename or path/folder/category/expression>` option:

```
$nmap --script <filename or path/folder/category/expression>
<target>
```

For example, the command to run the `dns-brute` NSE script is as follows:

```
$nmap --script dns-brute <target>
```

NSE also supports the execution of multiple scripts simultaneously simply by separating them with commas:

```
$ nmap --script http-headers,http-title scanme.nmap.org
Nmap scan report for scanme.nmap.org (74.207.244.221) Host is
up (0.096s latency).
Not shown: 995 closed ports PORT STATE SERVICE
22/tcp    open  ssh 25/tcp filtered smtp 80/tcp  open  http
| http-headers:
|     Date: Mon, 24 Oct 2011 07:12:09 GMT
|     Server: Apache/2.2.14 (Ubuntu)
|     Accept-Ranges: bytes
|     Vary: Accept-Encoding
|     Connection: close
|     Content-Type: text/html
|
|_    (Request type: HEAD)
|_http-title: Go ahead and ScanMe! 646/tcp  filtered ldp
9929/tcp open   nping-echo
```

In addition, NSE scripts can be selected by category, expression, or folder. For example, you can do the following:

- Run all the scripts in the `vuln` category with the following command:

```
$ nmap -sV --script vuln <target>
```

- Run all scripts in the `version` or `discovery` categories with a category list separated by commas, as follows:

```
$ nmap -sV --script="version,discovery" <target>
```

- You can also apply negative selections such as running all the scripts except for the ones in the exploit category with the `not` expression:

```
$ nmap -sV --script "not exploit" <target>
```

- Run all HTTP scripts that are named `http-<something>` except `http-brute` and `http-slowloris` with the help of the `*` wildcard character and the `and`, `or`, and `not` expressions:

```
$ nmap -sV --script "(http-*) and not(http-slowloris or
http- brute)" <target>
```

Expressions are very handy as they allow fine-grained script selection, as shown in the preceding example.

Debugging NSE scripts

To debug NSE scripts, use `--script-trace`. This enables a stack trace of the executed script that will help you in debugging. Remember that sometimes you may need to increase the debugging level with the `-d[1-9]` option to get to the bottom of the problem:

```
$ nmap -sC --script-trace <target>
$ nmap --script http-headers --script-trace scanme.nmap.org
NSOCK INFO [18.7370s] nsock_trace_handler_callback(): Callback:
CONNECT SUCCESS for EID 8 [45.33.32.156:80]
NSE: TCP 192.168.0.5:47478 > 45.33.32.156:80 | CONNECT NSE: TCP
192.168.0.5:47478 > 45.33.32.156:80 | 00000000:
48 45 41 44 20 2f 20 48 54 54 50 2f 31 2e 31 0d HEAD / HTTP/1.1
00000010: 0a 43 6f 6e 6e 65 63 74 69 6f 6e 3a 20 63 6c 6f
Connection: clo
00000020: 73 65 0d 0a 55 73 65 72 2d 41 67 65 6e 74 3a 20 se
User- Agent:
```

```
00000030: 4d 6f 7a 69 6c 6c 61 2f 35 2e 30 20 28 63 6f 6d
Mozilla/5.0 (com
00000040: 70 61 74 69 62 6c 65 3b 20 4e 6d 61 70 20 53 63
patible;
Nmap Sc
00000050: 72 69 70 74 69 6e 67 20 45 6e 67 69 6e 65 3b 20
ripting
Engine;
00000060: 68 74 74 70 73 3a 2f 2f 6e 6d 61 70 2e 6f 72 67
https://nmap.org
00000070: 2f 62 6f 6f 6b 2f 6e 73 65 2e 68 74 6d 6c 29 0d
/book/nse.html)
00000080: 0a 48 6f 73 74 3a 20 73 63 61 6e 6d 65 2e 6e 6d
Host:
scanme.nm
00000090: 61 70 2e 6f 72 67 0d 0a 0d 0a     ap.org [Output
removed to save space]Nmap scan report for scanme.nmap.org
(45.33.32.156)
Host is up (0.14s latency).
Other addresses for scanme.nmap.org (not scanned):
2600:3c01::f03c:91ff:fe18:bb2f
Not shown: 995 closed ports PORT STATE SERVICE
22/tcp     open  ssh 25/tcp filtered smtp 80/tcp open  http
| http-headers:
|     Date: Sun, 24 Apr 2016 19:52:13 GMT
|     Server: Apache/2.4.7 (Ubuntu)
|     Accept-Ranges: bytes
|     Vary: Accept-Encoding
|     Connection: close
|     Content-Type: text/html
|
|_    (Request type: HEAD) 9929/tcp   open  nping-echo 31337/
tcp open  Elite

Nmap done: 1 IP address (1 host up) scanned in 18.89 seconds
```

Adding new scripts

Often, you will want to try scripts not included officially with Nmap. To test new scripts, you simply need to copy them to your script folder inside your Nmap directory and run the following command to update the script database:

```
# nmap --script-updatedb
```

After updating the script database, you simply need to select them, as you would normally do, with the `--script` option. In addition, you may execute scripts without including them in the database by setting a relative or absolute script path as the argument:

```
# nmap --script /root/loot/non-official.nse <target>
```

I have created a GitHub repository at `https://github.com/cldrn/nmap-nse-scripts` to attempt to track all unofficial NSE scripts that for different reasons are not included officially with Nmap. There are scripts for all types of software and devices. Having scripts not included officially doesn't necessarily mean they don't work. I highly recommend you grab a copy to keep additional scripts in your arsenal.

Scanning random targets on the internet

Nmap supports a very interesting feature that allows us to run scans against random targets on the internet for research reasons. Although it is not recommended (and not legal in some countries) to do aggressive scans blindly, you could generate a sample of random hosts when conducting research about hosts facing the internet.

This recipe shows you how to generate random hosts as targets for your Nmap scans.

How to do it...

1. To generate a random target list of *n* hosts, use the following Nmap command:

   ```
   $ nmap -iR <n>
   ```

   For example, to generate a list of 100 hosts, use the following:

   ```
   $ nmap -iR 100
   ```

2. Now, let's check how common ICMP is in remote servers. Let's launch host discovery against three random targets:

   ```
   $ nmap -sn -iR 3
   Nmap scan report for host86-190-227-45.wlms-broadband.com
   (86.190.227.45)
   Host is up (0.000072s latency).
   Nmap scan report for 126.182.245.207
   Host is up (0.00023s latency).
   Nmap scan report for 158.sub-75-225-31.myvzw.com
   (75.225.31.158) Host is up (0.00017s latency).
   Nmap done: 3 IP addresses (3 hosts up) scanned in 0.78
   seconds
   ```

How it works...

The `-iR 3` option tells Nmap to generate three external IP addresses and use them as targets in the scan. This target assignment can be used with any combination of the regular scan options.

While this is a useful feature for conducting internet research, I recommend that you be careful with this flag. Nmap does not have control over the external IP addresses it generates; this means that inside the generated list could be a critical machine that is being heavily monitored. To avoid getting into trouble, use this feature wisely.

There's more...

Use Nmap to generate an unlimited number of IPs and run indefinitely with the `-iR 0` option:

```
$ nmap -iR 0
```

For example, to find random NFS shares online, you could use the following command:

```
$ nmap -p2049 --open -iR 0
```

Legal issues with port scanning

Port scanning without permission is not very welcome, and it is even illegal in some countries. I recommend you research your local laws to find out what you are permitted to do and whether port scanning is frowned upon in your country. You also need to consult with your ISP as they may have their own rules on the subject.

The official documentation of Nmap has an amazing write-up about the legal issues involved with port scanning, available at `https://nmap.org/book/legal-issues.html`. I recommend that everyone considering doing internet-wide research scanning reads it. While nowadays it is normal and even expected to be scanned from the internet, you should take all considerations when conducting research.

Collecting signatures of web servers

Nmap is an amazing tool for information gathering, and the variety of tasks that can be done with NSE is simply remarkable. The popular service **ShodanHQ** (`https://www.shodan.io/`) offers a database with a nice GUI of HTTP banners, which is useful for analyzing the impact of vulnerabilities. Its users can find out the number of devices that are online by country, which are identified by their service banners and IP geolocation. ShodanHQ uses its own built-in house tools to gather its data, but Nmap can easily be used for this task and take advantage of NSE.

In this recipe, we will see how to scan indefinitely for web servers and collect their HTTP headers with Nmap.

How to do it...

Open your terminal and enter the following command:

```
$ nmap -p80,443 -Pn -T4 --open --script http-headers,http-
title,ssl-cert --script-args http.useragent="A friendly web
crawler (http://calderonpale.com)",http-headers.useget -oX
random-webservers.xml -iR 0
```

This command will launch an instance of Nmap that will generate targets indefinitely, looking for web servers in ports `80` and `443`, and then it will save the output into the `random-webservers.xml` file. Each host with port `80` or `443` open will return something similar to the following:

```
Nmap scan report for XXXX Host is up (0.23s latency). PORT
STATE SERVICE
80/tcp open      http
|_http-title: Protected Object
| http-headers:
|    WWW-Authenticate: Basic realm="TD-8840T"
|    Content-Type: text/html
|    Transfer-Encoding: chunked
|    Server: RomPager/4.07 UPnP/1.0
|    Connection: close
|    EXT:
|
|_    (Request type: GET)
```

How it works...

The following command will tell Nmap to only check port `80` or `443` (`-p80,443`), skip the host discovery phase (`-Pn`), and use the aggressive timing template as we have plenty of servers to scan (`-T4`). If either port `80` or `443` is open, Nmap will run the NSE `http-title`, `http-headers`, and `ssl-cert` (`--script http-headers,http-title,ssl-cert`) scripts to collect server headers and the web server title; if HTTPS is detected, we will also extract information from SSL certificates:

```
$nmap -p80 -Pn -T4 --open --script http-headers,http-title
--script-args http.useragent="A friendly web crawler
(http://calderonpale.com)",http-headers.useget -oX random-
webservers.xml -iR 0
```

The script arguments that are passed set the HTTP user agent in the requests (`--script-args http.useragent="A friendly web crawler (http://calderonpale.com)"`) and use a `GET HTTP` request to retrieve the headers (`--script-args http-headers.useget`).

Finally, the Nmap `-iR 0` argument generates external IP addresses indefinitely and saves the results in a file in XML format (`-oX random-webservers.xml`).

There's more...

Nmap's HTTP library has *cache* support, but if you are planning to scan many hosts, you need to think about its size. The cache is stored in a temporary file that grows with each new request. If this file starts to get too big, cache lookups start to take a considerable amount of time.

You can disable the cache system of the HTTP library by setting the `http-max-cache-size` library argument, as shown in the following command:

```
$ nmap -p80 --script http-headers --script-args http-max-cache-
size=0    -iR 0
```

Important note

The **HTTP NSE library** is highly configurable. Read *Appendix A*, *HTTP, HTTP Pipelining, and Web Crawling Configuration Options*, to learn more about the advanced options available.

Scanning with Rainmap Lite

Rainmap Lite is a web application designed for running Nmap scans from any web browser. It was designed to be light and to depend on as few dependencies as possible. It is perfect for installing on a remote server and then just logging in from your phone and scheduling scans when you are on the road.

In this recipe, you will learn how to launch a Nmap scan using Rainmap Lite.

Getting ready

To run Rainmap Lite, we need to download the code and run the application as follows:

1. Grab the latest stable version of Rainmap Lite:

```
$git clone https://github.com/cldrn/rainmap-lite.git
```

2. Install Django and the only project dependency, `lxml`:

```
$ pip install Django
$ pip install lxml
```

3. Change your working directory to the newly created folder and create the database schema:

```
$python manage.py migrate
```

4. Load the default scanning profiles:

```
$python manage.py loaddata nmapprofiles
```

5. Locate the `nmaper-cronjob.py` file and update it, and also adjust the notification settings. The minimum configuration variables that need to update are `BASE_URL`, `SMTP_SERVER`, `SMTP_USER`, `SMTP_PASS`, and `SMTP_PORT`, if you would like to receive email notifications.
6. Run the application:

```
#python manage.py runserver 127.0.0.1:8080
```

7. Add a `cron` task that executes the agent periodically:

```
*/5 * * * * cd <App path> && /usr/bin/python nmaper-
cronjob.py >> /var/log/nmaper.log 2>&1
```

8. Finally, don't forget to add an administrative user to be able to log in to the web interface to schedule your scans:

```
$ python manage.py createsuperuser
```

How to do it...

Point your favorite web browser to the URL where Rainmap Lite is running. If you followed the steps described previously, it should be running on port `8080`.

The interface was designed to require as little typing as possible. Just fill in the field for the target, select a scan profile from the drop-down list, and enter the email address where you would like to receive the report. Hit **SCAN** when you are ready to add your scan to the queue:

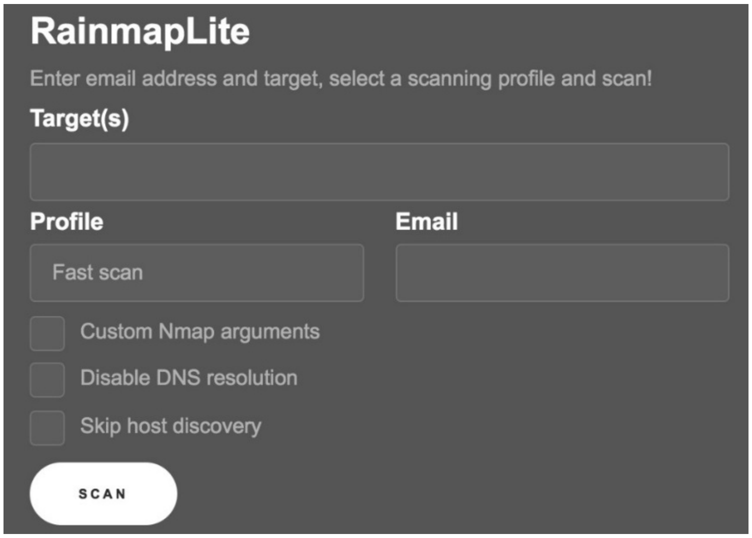


Figure 1.1 – Rainmap Lite's web interface

How it works...

Rainmap Lite is a simple Django application that allows users to schedule and run Nmap scans from any web browser. The application was designed to be easy to install on any server, and it is great for installing on a remote **Virtual Private Server** (**VPS**) and using the interface to schedule scans and share the results with your team easily.

An important aspect is that it is based on a standard cron agent to reduce the number of dependencies.

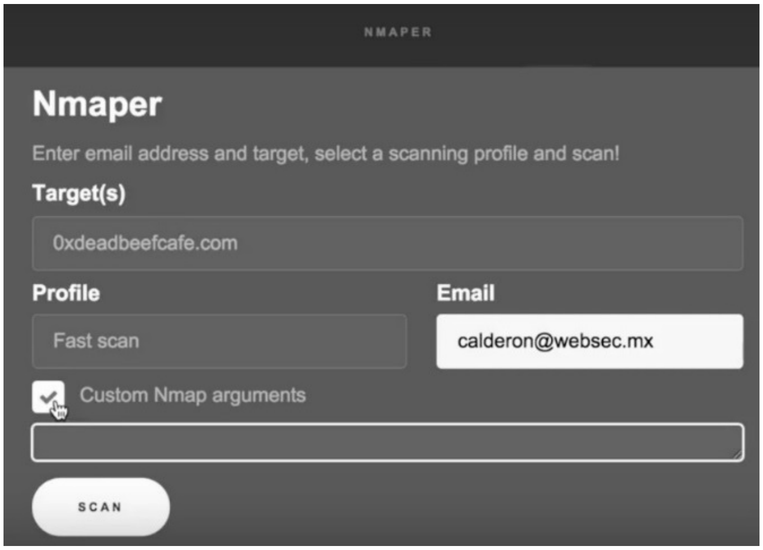
It started as a personal project that I decided to share at Black Hat US Arsenal 2016. Feel free to send any bug reports or suggestions to the project's GitHub page directly: `https://github.com/cldrn/rainmap-lite`.

There's more...

Scan profiles can be customized from the management console. The scanning profiles are updated in every version, and you are invited to contribute your own to the project's wiki at `https://github.com/cldrn/rainmap-lite/wiki/Scanning-profiles`.

Custom arguments

Custom arguments may be added on the fly without accessing the administration console by checking the box with the **Custom Nmap arguments** option:


NMAPER
Nmaper
Enter email address and target, select a scanning profile and scan!
SCAN

Target(s): 0xdeadbeefcafe.com
Profile: Fast scan
Email: calderon@websec.mx
[✓] Custom Nmap arguments

Figure 1.2 – Customizing scan options in Rainmap Lite

There are additional checkboxes in the interface to disable DNS resolution and enable/disable the host discovery phase. Any other combination of options will need to be added using the **Custom Nmap arguments** option.