Chapter No. 1
"Nmap Fundamentals"
In this package, you will find:
A Biography of the author of the book
A preview chapter from the book, Chapter NO.1 "Nmap Fundamentals"
A synopsis of the book’s content
Information on where to buy this book

About the Author

Paulino Calderón Pale (@calderpwn) is a very passionate software developer and penetration tester from a Caribbean island in México called Cozumel. He learned to write code and administer IT infrastructures early in his life—skills that came handy when he joined the information security industry. Today, he loves learning new technologies, penetration testing, conducting data-gathering experiments, developing software, and contributing to the open source community. He maintains a blog of his public work at http://calderonpale.com.

In the summer of 2011, he joined Google’s Summer of Code program to work on the Nmap project as an NSE (Nmap Scripting Engine) developer. He focused on improving the web scanning capabilities of Nmap and has produced over 20 scripts for gathering information, and detecting and exploiting security vulnerabilities since then.

He is the cofounder of Websec, an information security company focused on web security operation in México (http://websec.mx) and Canada (http://websec.ca), where they help companies in different industries secure their IT infrastructures.

For More Information:
Nmap 6: Network Exploration and Security Auditing Cookbook

Nmap 6: Network Exploration and Security Auditing Cookbook is a 100 percent practical book that follows a cookbook's style. Each recipe focuses on a single task and contains command line examples, sample output, a detailed explanation, and additional tips that could come in handy.

Nmap's vast functionality is explored through nine chapters covering 100 different tasks for penetration testers and system administrators. Unlike Nmap's official book, this cookbook focuses on tasks that you can do with the Nmap Scripting Engine, without forgetting to cover the core functionality of Nmap.

There were many great NSE scripts I wish I had more space to include in this book and many more that will be created after its publication. Luis Martin Garcia recently posted an interesting video that shows how much Nmap has grown over the years at http://www.youtube.com/watch?v=7r1F1MSAbXk. I invite you to register for the development mailing list and stay up-to-date with Nmap's latest features and NSE scripts.

I hope that you not only enjoy reading this cookbook, but also that, as you master the Nmap Scripting Engine, you come up with new ideas to create and contribute to this amazing project.

Finally, don't forget that you can send me your questions and I'll do my best to help you out.

What This Book Covers

Chapter 1, Nmap Fundamentals, covers the most common tasks performed with Nmap. Additionally, it briefly introduces Ndiff, Nping, and Zenmap.

Chapter 2, Network Exploration, covers host discovery techniques supported by Nmap, and other useful tricks with the Nmap Scripting Engine.

Chapter 3, Gathering Additional Host Information, covers interesting information gathering tasks with Nmap and its scripting engine.

Chapter 4, Auditing Web Servers, covers tasks related to web security auditing.

Chapter 5, Auditing Databases, covers security auditing tasks for MongoDB, MySQL, MS SQL, and CouchDB databases.

For More Information:
Chapter 6, Auditing Mail Servers, covers tasks for IMAP, POP3, and SMTP servers.

Chapter 7, Scanning Large Networks, covers tasks that are useful when scanning large networks ranging from scan optimization to distributing scans among several clients.

Chapter 8, Generating Scan Reports, covers the output options supported by Nmap.

Chapter 9, Writing Your Own NSE Scripts, covers the fundamentals of NSE development. It includes specific examples for handling sockets, output, libraries, and parallelism.

Appendix, References, covers references and official documentation used throughout this book.
Nmap Fundamentals

In this chapter we will cover:

- Downloading Nmap from the official source code repository
- Compiling Nmap from source code
- Listing open ports on a remote host
- Fingerprinting services of a remote host
- Finding live hosts in your network
- Scanning using specific port ranges
- Running NSE scripts
- Scanning using a specified network interface
- Comparing scan results with Ndiff
- Managing multiple scanning profiles with Zenmap
- Detecting NAT with Nping
- Monitoring servers remotely with Nmap and Ndiff

For More Information:
Nmap Fundamentals

Introduction

Nmap (Network Mapper) is an open-source tool specialized in network exploration and security auditing, originally published by Gordon "Fyodor" Lyon. The official website (http://nmap.org) describes it 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.

There are many other port scanners out there, but none of them even comes close to offering the flexibility and advanced options of Nmap.

The Nmap Scripting Engine (NSE) has revolutionized the possibilities of a port scanner by allowing users to write scripts that perform custom tasks using the host information collected by Nmap.

Additionally, the Nmap Project includes other great tools:

- Zenmap: A graphical interface for Nmap
- Ndiff: A tool for scan result comparison
- Nping: An excellent tool for packet generation and traffic analysis
- Ncrack: An Nmap-compatible tool for brute forcing network logins
- Ncat: A debugging utility to read and write data across networks

Needless to say, it is essential that every security professional and network administrator master this tool to conduct security assessments, monitor, and administer networks efficiently.

Nmap's community is very active, and new features are added every week. I encourage you to always keep an updated copy in your arsenal, if you haven't done this already; and even better, to subscribe to the development mailing list at http://cgi.insecure.org/mailman/listinfo/nmap-dev.

This chapter describes how to do some of the most common tasks with Nmap, including port scanning and target enumeration. It also includes recipes that illustrate how handy Zenmap's profiles are, how to use Nping for NAT detection, and different applications of Ndiff, including how to set up a remote monitoring system with some help of bash scripting and cron. I've added as many reference links with additional material as possible; I recommend you visit them to learn more about the inner workings of the advanced scanning techniques performed by Nmap.

For More Information:
I've also created the website http://nmap-cookbook.com to post new, related material and additional recipes, so make sure you stop by from time to time.

**Downloading Nmap from the official source code repository**

This section describes how to download Nmap's source code from the official subversion repository. By doing so, users can compile the latest version of Nmap and keep up with the daily updates that are committed to the subversion repository.

**Getting ready**

Before continuing, you need to have a working Internet connection and access to a subversion client. Unix-based platforms come with a command-line client called subversion (svn). To check if it's already installed in your system, just open a terminal and type:

$ svn

If it tells you that the command was not found, install svn using your favorite package manager or build it from source code. The instructions for building 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.

If you would rather work with a graphical user interface, RapidSVN is a very popular, cross-platform alternative. You can download and install RapidSVN from http://rapidsvn.tigris.org/.

**How to do it...**

Open your terminal and enter the following command:

$ svn co --username guest https://svn.nmap.org/nmap/

---

**For More Information:**

Wait until svn downloads all the files stored in the repository. You should see the list of the added files as it finishes, as shown in the following screenshot:

When the program returns/exits, you will have Nmap's source code in your current directory.

**How it works...**

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

This command downloads a copy of the remote repository located at `https://svn.nmap.org/nmap/`. This repository has world read access to the latest stable build, allowing svn to download your local working copy.

**There's more...**

If you are using RapidSVN then follow these steps:

1. Right-click on **Bookmarks**.
2. Click on **Checkout New Working Copy**.
3. Type `https://svn.nmap.org/nmap/` in the URL field.
4. Select your local working directory.
5. Click on **OK** to start downloading your new working copy.

---

For More Information:

Experimenting with development branches

If you want to try the latest creations of the development team, there is a folder named nmap-exp that contains different experimental branches of the project. Code stored there is not guaranteed to work all the time, as the developers use it as a sandbox until it is ready to be merged into the stable branch. The full subversion URL of this folder is https://svn.nmap.org/nmap-exp/.

Keeping your source code up-to-date

To update a previously-downloaded copy of Nmap, use the following command inside your working directory:

```
$ svn update
```

You should see the list of files that have been updated, as well as some revision information.

See also

- The Compiling Nmap from source code recipe
- The Listing open ports on a remote host recipe
- The Fingerprinting services of a remote host recipe
- The Running NSE scripts recipe
- The Comparing scan results with Ndiff recipe
- The Managing multiple scanning profiles with Zenmap recipe
- The Generating a network topology graph with Zenmap recipe in Chapter 8, Generating Scan Reports
- The Saving scan results in normal format recipe in Chapter 8, Generating Scan Reports

Compiling Nmap from source code

Precompiled packages always take time to prepare and test, causing delays between releases. If you want to stay up-to-date with the latest additions, compiling Nmap's source code is highly recommended.

This recipe describes how to compile Nmap's source code in the Unix environment.

Getting ready

Make sure the following packages are installed in your system:

- gcc
- openssl
- make

For More Information:

Install the missing software using your favorite package manager or build it from source code. Instructions to build these packages from source code are out of the scope of this book but are available online.

How to do it...

1. Open your terminal and go into the directory where Nmap’s source code is stored.
2. Configure it according to your system:
   
   ```bash
   ./configure
   ```
   
   An ASCII dragon warning you about the power of Nmap will be displayed (as shown in the following screenshot) if successful, otherwise lines specifying an error will be displayed.

For More Information:

3. Build Nmap using the following command:

    $ make

If you don't see any errors, you have built the latest version of Nmap successfully. You can check this by looking for the compiled binary `Nmap` in your current directory.

If you want to make Nmap available for all the users in the system, enter the following command:

    # make install

**How it works...**

We used the script `configure` to set up the different parameters and environmental variables affecting your system and desired configuration. Afterwards, GNUs `make` generated the binary files by compiling the source code.

**There's more...**

If you only need the Nmap binary, you can use the following configure directives to avoid installing Ndiff, Nping, and Zenmap:

- Skip the installation of Ndiff by using `--without-ndiff`
- Skip the installation of Zenmap by using `--without-zenmap`
- Skip the installation of Nping by using `--without-nping`

**OpenSSL development libraries**

OpenSSL is optional when building Nmap. Enabling it allows Nmap to access the functions of this library related to multiprecision integers, hashing, and encoding/decoding for service detection and Nmap NSE scripts.

The name of the OpenSSL development package in Debian systems is `libssl-dev`.

**Configure directives**

There are several configure directives that can be used when building Nmap. For a complete list of directives, use the following command:

    $ ./configure --help

---

For More Information:

Precompiled packages

There are several precompiled packages available online (http://nmap.org/download.html) for those who don't have access to a compiler, but unfortunately, it's very likely you will be missing features unless it's a very recent build. Nmap is continuously evolving. If you are serious about harnessing the power of Nmap, keep your local copy up-to-date with the official repository.

See also

- The Downloading Nmap from the official source code repository recipe
- The Listing open ports on a remote host recipe
- The Fingerprinting services of a remote host recipe
- The Comparing scan results with Ndiff recipe
- The Managing multiple scanning profiles with Zenmap recipe
- The Running NSE scripts recipe
- The Scanning using a specified network interface recipe
- The Saving scan results in normal format recipe in Chapter 8, Generating Scan Reports
- The Generating a network topology graph with Zenmap recipe in Chapter 8, Generating Scan Reports

### Listing open ports on a remote host

This recipe describes the simplest way of using Nmap to determine the port states on a remote host, a process used to identify running services commonly referred as port scanning.

#### How to do it...

1. Open a terminal.
2. Type the following command:

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

The scan results should appear on the screen, showing the interesting ports and their states. The ports marked as open are of special interest as they represent services running on the target host.
Chapter 1

How it works...

The following command checks the state of the most popular ports on the host `scanme.nmap.org` by launching a TCP port scan:

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

The results contain host information such as the IPv4 address and PTR record, and port information such as a service name and port state.

There's more...

Even for this simplest port scan, Nmap does a lot of things in the background, and these can be configured as well.

Nmap begins by converting the hostname to an IPv4 address using DNS. 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
```

Afterwards, it pings the target address to check if the host is alive. To skip this step use `-PN` as follows:

```
$ nmap -PN scanme.nmap.org
```
Nmap then converts the IPv4 address back to a hostname by using a reverse DNS call. Use -n to skip this step as follows:

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

Finally, it launches a TCP port scan. To specify a different port range, use `\-p[1-65535]`, or \-p- for all possible TCP ports, as shown in the following command:

```bash
$ nmap -p1-30 scanme.nmap.org
```

**Privileged versus unprivileged**

Running `nmap <TARGET>` as a privileged user launches the **SYN Stealth Scan**. For unprivileged accounts that can't create raw packets, the **TCP Connect Scan** is used.

The difference between these two is that a TCP Connect Scan uses the high-level system call \texttt{connect} to obtain information about the port state. This means that each TCP connection is fully completed and, therefore, is slower and more likely to be detected and recorded in system logs. SYN Stealth Scans use raw packets to send specially-crafted TCP packets that detect port states more reliably.

**Port states**

Nmap categorizes ports into the following states:

- **Open**: This indicates that an application is listening for connections on this port.
- **Closed**: This indicates that the probes were received but there is no application listening on this port.
- **Filtered**: This indicates that the probes were not received and the state could not be established. It also indicates that the probes are being dropped by some kind of filtering.
- **Unfiltered**: This 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 Nmap couldn't establish the state.
- **Closed/Filtered**: This indicates that the port was filtered or closed but Nmap couldn't establish the state.

For More Information:

Port scanning techniques supported by Nmap

We showed the simplest way of performing a port scan, but Nmap has a vast number of advanced scanning techniques available. Use `nmap -h` or visit [http://nmap.org/book/man-port-scanning-techniques.html](http://nmap.org/book/man-port-scanning-techniques.html) to learn more about them.

See also

- The Fingerprinting services of a remote host recipe
- The Finding live hosts in your network recipe
- The Scanning using specific port ranges recipe
- The Scanning using a specified network interface recipe
- The Manage different scanning profiles with Zenmap recipe
- The Monitoring servers remotely with Nmap and Ndiff recipe
- The Excluding hosts from your scans recipe in Chapter 2, Network Exploration
- The Scanning IPv6 addresses recipe in Chapter 2, Network Exploration
- The Fingerprinting the operative system of a host recipe in Chapter 3, Gathering Additional Host Information
- The Discovering UDP services recipe in Chapter 3, Gathering Additional Host Information
- The Listing protocols supported by a remote host recipe in Chapter 3, Gathering Additional Host Information

Fingerprinting services of a remote host

Version detection is one of the most popular features of Nmap. Knowing the exact version of a service is highly valuable for penetration testers who use this service to look for security vulnerabilities, and for system administrators who wish to monitor their networks for any unauthorized changes. Fingerprinting a service may also reveal additional information about a target, such as available modules and specific protocol information.

This recipe describes how to fingerprint the services of a remote host by using Nmap.

For More Information:
How to do it...

Open a terminal and type the following command:

```
$ nmap -sV scanme.nmap.org
```

The result of this command is a table containing an additional column named **VERSION**, displaying the specific service version, if identified. Additional information will be enclosed in parenthesis. Refer to the following screenshot:

![Screenshot of Nmap output]

How it works...

The flag `-sV` enables service detection, which returns additional service and version information.

**Service detection** is one of the most loved features of Nmap, as it's very useful in many situations such as identifying security vulnerabilities or making sure a service is running on a given port.

This feature basically works by sending different probes from `nmap-service-probes` to the list of suspected open ports. The probes are selected based on how likely it is that they can be used to identify a service.

There is very detailed documentation on how the service detection mode works, and the file formats used, at [http://nmap.org/book/vscan.html](http://nmap.org/book/vscan.html).
There's more...

You can set the amount of probes to use by changing the intensity level of the scan with the argument `--version-intensity [0-9]`, as follows:

```
# nmap -sV --version-intensity 9
```

**Aggressive detection**

Nmap has a special flag to activate aggressive detection, namely `-A`. **Aggressive mode** enables OS detection (`-O`), version detection (`-sV`), script scanning (`-sC`), and traceroute (`--traceroute`). Needless to say this mode sends a lot more probes and it is more likely to be detected, but provides a lot of valuable host information. You can see this by using one of the following commands:

```
# nmap -A <target>
```

Or

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

For More Information:

Nmap Fundamentals

Submitting service fingerprints

Nmap's 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. If Nmap does not identify the service correctly, please submit your new service fingerprint or correction to http://insecure.org/cgi-bin/submit.cgi.

See also

- The Listing open ports on a remote host recipe
- The Finding live hosts in your network recipe
- The Scanning using specific port ranges recipe
- The Scanning using a specified network interface recipe
- The Managing multiple scanning profiles with Zenmap recipe
- The Monitoring servers remotely with Nmap and Ndiff recipe
- The Hiding our traffic with additional random data recipe in Chapter 2, Network Exploration
- The Scanning IPv6 addresses recipe in Chapter 2, Network Exploration
- The Getting information from WHOIS records recipe in Chapter 3, Gathering Additional Host Information
- The Brute forcing DNS records recipe in Chapter 3, Gathering Additional Host Information
- The Fingerprinting the operative system of a host recipe in Chapter 3, Gathering Additional Host Information

Finding live hosts in your network

Finding live hosts in a network is often used by penetration testers to enumerate active targets, and by system administrators to count or monitor the number of active hosts.

This recipe describes how to perform a ping scan, to find live hosts in a network by using Nmap.

How to do it...

Open your terminal and enter the following command:

```
$ nmap -sP 192.168.1.1/24
```
The result shows hosts that are online and responded to the ping sweep.

Nmap scan report for 192.168.1.102
Host is up.
Nmap scan report for 192.168.1.254
Host is up (0.0027s latency).
MAC Address: 5C:4C:A9:F2:DC:7C (Huawei Device Co.)
Nmap done: 256 IP addresses (2 hosts up) scanned in 10.18 seconds

In this case, we found two live hosts in the network. Nmap has also found the MAC address, and it identified the vendor of a home router.

**How it works...**

Nmap uses the `-sP` flag for ping scanning. This type of scan is very useful for enumerating the hosts in a network. It uses a TCP ACK packet and an ICMP echo request if executed as a privileged user, or a SYN packet sent via `connect()` `syscall` if run by users who can't send raw packets.

CIDR /24 in 192.168.1.1/24 is used to indicate that we want to scan all the 256 IPs in our network.

**There's more...**

ARP requests are used when scanning a local Ethernet network as a privileged user, but you can override this behavior by including the flag `--send-ip`.

```
# nmap -sP --send-ip 192.168.1.1/24
```

**Traceroute**

Use `--traceroute` to include a path between your machine and each host that was found.

Nmap scan report for 192.168.1.101
Host is up (0.062s latency).
MAC Address: 00:23:76:CD:C5:BE (HTC)

```
TRACEROUTE
HOP RTT ADDRESS
1  61.70 ms 192.168.1.101
```
Nmap Fundamentals

Nmap scan report for 192.168.1.102
Host is up.

Nmap scan report for 192.168.1.254
Host is up (0.0044s latency).
MAC Address: 5C:4C:A9:F2:DC:7C (Huawei Device Co.)

TRACEROUTE

<table>
<thead>
<tr>
<th>HOP</th>
<th>RTT</th>
<th>ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4.40 ms</td>
<td>192.168.1.254</td>
</tr>
</tbody>
</table>

Nmap done: 256 IP addresses (3 hosts up) scanned in 10.03 seconds

NSE scripts

Ping scanning does not perform port scanning or service detection, but the Nmap Scripting Engine can be enabled for scripts depending on host rules, such as the cases of sniffer-detect and dns-brute.

```bash
# nmap -sP --script discovery 192.168.1.1/24
```

Pre-scan script results:

```
<table>
<thead>
<tr>
<th>broadcast-ping:</th>
</tr>
</thead>
</table>
| Use the newtargets script-arg to add the results as targets
```

Nmap scan report for 192.168.1.102
Host is up.

Host script results:

```
| dns-brute: Can't guess domain of "192.168.1.102"; use dns-brute.domain script argument.
```

Nmap scan report for 192.168.1.254
Host is up (0.0023s latency).
MAC Address: 5C:4C:A9:F2:DC:7C (Huawei Device Co.)

Host script results:

For More Information:
Chapter 1

| _dns-brute: Can't guess domain of "192.168.1.254"; use dns-brute.domain script argument. |
| _sniffer-detect: Likely in promiscuous mode (tests: "11111111") |

Nmap done: 256 IP addresses (2 hosts up) scanned in 14.11 seconds

See also

- The Running NSE scripts recipe
- The Discovering hosts using broadcast pings recipe in Chapter 2, Network Exploration
- The Discovering hosts with TCP SYN ping scans recipe in Chapter 2, Network Exploration
- The Discovering hosts with TCP ACK ping scans recipe in Chapter 2, Network Exploration
- The Discovering hosts with ICMP ping scans recipe in Chapter 2, Network Exploration
- The Gathering network information with broadcast scripts recipe in Chapter 2, Network Exploration
- The Discovering hostnames pointing to the same IP recipe in Chapter 3, Gathering Additional Host Information
- The Brute forcing DNS records recipe in Chapter 3, Gathering Additional Host Information
- The Spoofing the origin IP of a port scan recipe in Chapter 3, Gathering Additional Host Information

Scanning using specific port ranges

There are situations when a system administrator is looking for infected machines that use a specific port to communicate, or when users are only looking for a specific service or open port and don't really care about the rest. Narrowing down the port ranges used also optimizes performance, which is very important when scanning multiple targets.

This recipe describes how to use port ranges when performing Nmap scans.

How to do it...

Open your terminal and enter the following command:

```bash
# nmap -p80 192.168.1.1/24
```

For More Information:

A list of hosts with the state of port 80 will appear in the results.

**Nmap scan report for 192.168.1.102**
Host is up (0.000079s latency).

<table>
<thead>
<tr>
<th>PORT</th>
<th>STATE</th>
<th>SERVICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>80/tcp</td>
<td>closed</td>
<td>http</td>
</tr>
</tbody>
</table>

**Nmap scan report for 192.168.1.103**
Host is up (0.016s latency).

<table>
<thead>
<tr>
<th>PORT</th>
<th>STATE</th>
<th>SERVICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>80/tcp</td>
<td>open</td>
<td>http</td>
</tr>
</tbody>
</table>

MAC Address: 00:16:6F:7E:E0:B6 (Intel)

**Nmap scan report for 192.168.1.254**
Host is up (0.0065s latency).

<table>
<thead>
<tr>
<th>PORT</th>
<th>STATE</th>
<th>SERVICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>80/tcp</td>
<td>open</td>
<td>http</td>
</tr>
</tbody>
</table>

MAC Address: 5C:4C:A9:F2:DC:7C (Huawei Device Co.)

Nmap done: 256 IP addresses (3 hosts up) scanned in 8.93 seconds

**How it works...**

Nmap uses the flag `-p` for setting the port ranges to be scanned. This flag can be combined with any scanning method. In the previous example, we used the argument `-p80` to indicate to Nmap that we are only interested in port 80.

The CIDR `/24` in `192.168.1.1/24` is used to indicate that we want to scan all of the 256 IPs in our network.

**There's more...**

There are several accepted formats for the argument `-p`:

- Port list:

  ```
  # nmap -p 80,443 localhost
  ```
Port range:
# nmap -p 1-100 localhost

All ports:
# nmap -p * localhost

Specific ports by protocols:
# nmap -pT:25,U:53 <target>

Service name:
# nmap -p smtp <target>

Service name wildcards:
# nmap -p smtp* <target>

Only ports registered in Nmap services:
# nmap -p [1-65535] <target>

See also
- The Finding live hosts in your network recipe
- The Listing open ports on a remote host recipe
- The Scanning using a specified network interface recipe
- The Running NSE scripts recipe
- The Hiding our traffic with additional random data recipe in Chapter 2, Network Exploration
- The Forcing DNS resolution recipe in Chapter 2, Network Exploration
- The Excluding hosts from your scans recipe in Chapter 2, Network Exploration
- The Scanning IPv6 addresses recipe in Chapter 2, Network Exploration
- The Listing protocols supported by a remote host recipe in Chapter 3, Gathering Additional Host Information

Running NSE scripts

NSE scripts are very powerful and have become one of Nmap's main strengths, performing tasks from advanced version detection to vulnerability exploitation.

The following recipe describes how to run NSE scripts, and the different options available for this engine.

For More Information:
**How to do it...**

To include the title of the index document of a web server in your scan results, open your terminal and type the following command:

```
$ nmap -sV --script http-title scanme.nmap.org
```

**How it works...**

The argument `--script` sets which NSE scripts should be run with the scan. In this case, when the service scan detects the web server, a parallel thread is initialized for the selected NSE script.

There are more than 230 scripts available, which perform a wide variety of tasks. The NSE script `http-title` returns the title of the root document if a web server is detected.

**There's more...**

You can run multiple scripts at once:

```
$ 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

<table>
<thead>
<tr>
<th>PORT</th>
<th>STATE</th>
<th>SERVICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>22/tcp</td>
<td>open</td>
<td>ssh</td>
</tr>
<tr>
<td>25/tcp</td>
<td>filtered</td>
<td>smtp</td>
</tr>
<tr>
<td>80/tcp</td>
<td>open</td>
<td>http</td>
</tr>
</tbody>
</table>

For More Information:

Additionally, NSE scripts can be selected by category, expression, or folder:

- Run all the scripts in the vuln category:
  
  $ nmap -sV --script vuln <target>

- Run the scripts in the categories version or discovery:
  
  $ nmap -sV --script="version,discovery" <target>

- Run all the scripts except for the ones in the exploit category:
  
  $ nmap -sV --script "not exploit" <target>

- Run all HTTP scripts except http-brute and http-slowloris:
  
  $ nmap -sV --script "(http-*) and not(http-slowloris or http-brute)" <target>

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

$ nmap -sV --script exploit -d3 --script-trace 192.168.1.1

**NSE script arguments**

The flag --script-args is used to set arguments of NSE scripts. For example, if you would like to set the HTTP library argument useragent, you would use:

$ nmap -sV --script http-title --script-args http.useragent="Mozilla 999" <target>

You can also use aliases when setting the arguments for NSE scripts. For example, you could use:

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

---

For More Information:

Instead of:

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

Adding new scripts

To test new scripts, you simply need to copy them to your /scripts directory and run the following command to update the script database:

# nmap --script-update-db

NSE script 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 help conduct brute-force password auditing.
- **default**: This category is for scripts that are executed when a script scan is executed (-sC).
- **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 advanced versioning.
- **vuln**: This category is for scripts related to security vulnerabilities.

See also

- The Managing different scanning profiles with Zenmap recipe
- The Monitoring servers remotely with Nmap and Ndiff recipe
- The Fingerprinting services of a remote host recipe
- The Finding live hosts in your network recipe
- The Gathering network information with broadcast scripts recipe in Chapter 2, Network Exploration
- The Collecting valid e-mail accounts recipe in Chapter 3, Gathering Additional Host Information

For More Information:

The Discovering hostnames pointing to the same IP recipe in Chapter 3, Gathering Additional Host Information

- The Brute forcing DNS records recipe in Chapter 3, Gathering Additional Host Information

## Scanning using a specified network interface

Nmap is known for its flexibility, and allows users to specify the network interface used when scanning. This is very handy when running some of the sniffer NSE scripts, discovering whether your interface supports the promiscuous mode, or when testing a network connection with routing problems.

The following recipe describes how to force Nmap to scan using a specified network interface.

### How to do it...

Open your terminal and enter the following command:

```bash
$ nmap -e <INTERFACE> scanme.nmap.org
```

This will force Nmap to perform a TCP scan of `scanme.nmap.org` using the interface `<INTERFACE>`.

For More Information:

How it works...

The flag `-e` is used to set a specific network interface when Nmap is unable to select one automatically. The existence of this flag allows Nmap to send and receive packets through an alternate interface.

There's more...

If you need to select your interface manually, you will see the following message:

WARNING: Unable to find appropriate interface for system route to ...

Checking a TCP connection

To check if a network interface can communicate with your network, you could try a ping scan that forces Nmap to use a specified interface:

```
$ nmap -sP -e INTERFACE 192.168.1.254
```

Timing report

```
--------------- Timing report ---------------
  hostgroups: min 1, max 100000
  rtt-timeouts: init 1000, min 100, max 10000
  max-scan-delay: TCP 1000, UDP 1000, SCTP 1000
  parallelism: min 0, max 0
  max-retries: 10, host-timeout: 0
  min-rate: 0, max-rate: 0

Initiating ARP Ping Scan at 02:46
Scanning 192.168.1.254 [1 port]
Packet capture filter (device wlan2): arp and arp[18:4] = 0x00C0CA50 and arp[22:2] = 0xE567
Completed ARP Ping Scan at 02:46, 0.06s elapsed (1 total hosts)
Overall sending rates: 16.76 packets / s, 704.05 bytes / s.
mass_rdns: Using DNS server 192.168.1.254
Initiating Parallel DNS resolution of 1 host. at 02:46
mass_rdns: 0.03s 0/1 [#: 1, OK: 0, NX: 0, DR: 0, SF: 0, TR: 1]
Completed Parallel DNS resolution of 1 host. at 02:46, 0.03s elapsed
DNS resolution of 1 IPs took 0.03s. Mode: Async [#: 1, OK: 0, NX: 1, DR: 0, SF: 0, TR: 1, CN: 0]
```

Nmap scan report for 192.168.1.254

For More Information:

Host is up, received arp-response (0.0017s latency).
MAC Address: 5C:4C:A9:F2:DC:7C (Huawei Device Co.)
Final times for host: srtt: 1731 rttvar: 5000 to: 100000
Read from /usr/local/bin/../share/nmap: nmap-mac-prefixes nmap-payloads.
Nmap done: 1 IP address (1 host up) scanned in 0.17 seconds
Raw packets sent: 1 (28B) | Rcvd: 1 (28B)

See also
- The Running NSE scripts recipe
- The Scanning using specific port ranges recipe
- The Hiding our traffic with additional random data recipe in Chapter 2, Network Exploration
- The Forcing DNS resolution recipe in Chapter 2, Network Exploration
- The Excluding hosts from your scans recipe in Chapter 2, Network Exploration
- The Brute forcing DNS records recipe in Chapter 3, Gathering Additional Host Information
- The Fingerprinting the operative system of a host recipe in Chapter 3, Gathering Additional Host Information
- The Discovering UDP services recipe in Chapter 3, Gathering Additional Host Information
- The Listing the protocols supported by a remote host recipe in Chapter 3, Gathering Additional Host Information

Comparing scan results with Ndiff

Ndiff was designed to address the issues of using diff with two XML scan results. It compares files by removing false positives and producing a more readable output, which is perfect for anyone who needs to keep a track of the scan results.

This recipe describes how to compare two Nmap scans to detect the changes in a host.

Getting ready

Ndiff requires two Nmap XML files to work, so make sure you have previously saved the scan results of the same host. If you haven't, you can always scan your own network, deactivate a service, and scan again to get these two test files. To save the results of an Nmap scan into an XML file use -oX <filename>.

For More Information:
**How to do it...**

1. Open your terminal.
2. Enter the following command:
   
   ```bash
   $ ndiff FILE1 FILE2
   ```

3. The output returns all the differences between FILE1 and FILE2. New lines are shown after a plus sign. The lines that were removed on FILE2 are displayed after a negative sign.

**How it works...**

Ndifff uses the first file as a base to compare against the second one. It displays the state differences for host, port, services, and OS detection.

**There's more...**

If you prefer Zenmap, you can use the following steps instead:

1. Launch Zenmap.
2. Click on **Tools** on the main toolbar.
3. Click on **Compare Results** (Ctrl + D).
4. Select the first file by clicking on **Open** in the section named **A scan**.

---

**For More Information:**

5. Select the second file by clicking on Open in the section named B scan.

![Comparison of scan files](image)

**Output format**

A human readable format is returned by default. However, Ndiff can return the differences in XML format, if preferred, by using the flag `--xml`.

**Verbose mode**

Verbose mode includes all of the information including hosts and ports that haven't changed. To use it, enter the following commands:

```
$ ndiff -v FILE1 FILE2
$ ndiff --verbose FILE1 FILE2
```

**See also**

- The Monitoring servers remotely with Nmap and Ndiff recipe
- The Managing multiple scanning profiles with Zenmap recipe
- The Geo-locating an IP address recipe in Chapter 3, Gathering Additional Host Information
- The Getting information from WHOIS records recipe in Chapter 3, Gathering Additional Host Information

---

**For More Information:**

Nmap Fundamentals

- The Fingerprinting the operative system of a host recipe in Chapter 3, Gathering Additional Host Information
- The Discovering UDP services recipe in Chapter 3, Gathering Additional Host Information
- The Detecting possible XST vulnerabilities recipe in Chapter 4, Auditing Web Servers

Managing multiple scanning profiles with Zenmap

Scanning profiles are a combination of Nmap arguments that can be used to save time and the need to remember argument names when launching an Nmap scan.

This recipe is about adding, editing, and deleting a scanning profile in Zenmap.

How to do it...

Let's add a new profile for scanning web servers:

1. Launch Zenmap.
2. Click on Profile on the main toolbar.
3. Click on New Profile or Command (Ctrl + P). The Profile Editor will be launched.
4. Enter a profile name and a description on the Profile tab.
5. Enable Version detection and disable reverse DNS resolution on the Scan tab.
6. Enable the following scripts on the Scripting tab:
   - hostmap
   - http-default-accounts
   - http-enum
   - http-favicon
   - http-headers
   - http-methods
   - http-trace
   - http-php-version
   - http-robots.txt
   - http-title

For More Information:
7. Next, go to the **Target** tab and click on **Ports** to scan and enter 80, 443.
8. Save your changes by clicking on **Save Changes**.

### How it works...

After using the editor to create our profile, we are left with the following Nmap command:

```
```

Using the **Profile** wizard, we have enabled service scanning (-sV), set the scanning ports to 80 and 443, set the **Timing** template to 4, and selected a bunch of HTTP-related scripts to gather as much information as possible from this web server. And we now have this profile saved for some quick scanning without having to type all these flags and options again.

---

For More Information:

There's more...

Zenmap includes 10 predefined scan profiles to help newcomers familiarize themselves with Nmap. I recommend that you to analyze them in order to understand the additional scanning techniques that are available to Nmap, along with some of the more useful combinations of its options.

- Intense scan: `nmap -T4 -A -v`
- Intense scan plus UDP: `nmap -sS -sU -T4 -A -v`
- Intense scan, all TCP ports: `nmap -p 1-65535 -T4 -A -v`
- Intense scan, no ping: `nmap -T4 -A -v -Pn`
- Ping scan: `nmap -sn`
- Quick scan: `nmap -T4 -F`
- Quick scan plus: `nmap -sV -T4 -O -F -version-light`
- Quick traceroute: `nmap -sn -traceroute`
- Regular scan: `nmap`
- Slow comprehensive scan: `nmap -sS -sU -T4 -A -v -PE -PP -PS80,443 -PA3389 -PU40125 -PY -g 53 --script default or discovery and safe`

Editing and deleting a scan profile

To edit or delete a scan profile, you need to select the entry you wish to modify from the Profile drop-down menu. Click on Profile on the main toolbar and select Edit Selected Profile (Ctrl + E).

The editor will be launched allowing you to edit or delete the selected profile.

See also

- The Listing open ports on a remote host recipe
- The Fingerprinting server of a remote host recipe
- The Finding live hosts in your network recipe
- The Scanning using specific port ranges recipe
- The Running NSE scripts recipe
- The Scanning IPv6 addresses recipe in Chapter 2, Network Exploration
- The Gathering network information with broadcast scripts recipe in Chapter 2, Network Exploration
- The Discovering UDP services recipe in Chapter 3, Gathering Additional Host Information

For More Information:

Detecting NAT with Nping

Nping was designed for packet crafting and traffic analysis and is perfect for a variety of networking tasks.

The following recipe will introduce Nping by showing how to perform NAT detection with some help of the Nping Echo protocol.

How to do it...

Open a terminal and enter the following command:

```
# nping --ec "public" -c 1 echo.nmap.org
```

This will result in an output stream similar to the following example:

Nping will return the packet traffic between the client and the Nping echo server:

```
Starting Nping 0.5.59BETA1 ( http://nmap.org/nping ) at 2011-10-27 16:59 PDT
SENT (1.1453s) ICMP 192.168.1.102 > 74.207.244.221 Echo request (type=8/ code=0) ttl=64 id=47754 iplen=28
CAPT (1.1929s) ICMP 187.136.56.27 > 74.207.244.221 Echo request (type=8/ code=0) ttl=57 id=47754 iplen=28
RCVD (1.2361s) ICMP 74.207.244.221 > 192.168.1.102 Echo reply (type=0/ code=0) ttl=53 id=37482 iplen=28
Max rtt: 90.751ms | Min rtt: 90.751ms | Avg rtt: 90.751ms
Raw packets sent: 1 (28B) | Rcvd: 1 (46B) | Lost: 0 (0.00%) | Echoed: 1 (28B)
Tx time: 0.00120s | Tx bytes/s: 23236.51 | Tx pkts/s: 829.88
Rx time: 1.00130s | Rx bytes/s: 45.94 | Rx pkts/s: 1.00
Nping done: 1 IP address pinged in 2.23 seconds
```

Take note of the source address 192.168.1.102 in the first packet marked as SENT.

```
SENT (1.1453s) ICMP 192.168.1.102 > 74.207.244.221 Echo request (type=8/ code=0) ttl=64 id=47754 iplen=28
```

Compare this address to the source address in the second packet marked as CAPT.

```
CAPT (1.1929s) ICMP 187.136.56.27 > 74.207.244.221 Echo request (type=8/ code=0) ttl=57 id=47754 iplen=28
```

The addresses are different, indicating the presence of NAT.

For More Information:

Nmap Fundamentals

How it works...

Nping's echo mode was designed to help troubleshoot firewall and routing problems. Basically, it returns a copy of the received packet back to the client.

The command is:

```bash
# nping --ec "public" -c 1 echo.nmap.org
```

It uses Nping's echo mode (--ec or --echo-client) to help us analyze the traffic between Nmap's Nping echo server, to determine if there is a NAT device on the network. The argument after --ec corresponds to a secret passphrase known by the server to encrypt and authenticate the session.

The flag -c is used to specify how many iterations of packets must be sent.

There's more...

With Nping it is really simple to generate custom TCP packets. For example, to send a TCP SYN packet to port 80, use the following command:

```bash
# nping --tcp -flags syn -p80 -c 1 192.168.1.254
```

This will result in the following output:

```
SENT (0.0615s) TCP 192.168.1.102:33599 > 192.168.1.254:80 S ttl=64 id=21546 iplen=40 seq=2463610684 win=1480
RCVD (0.0638s) TCP 192.168.1.254:80 > 192.168.1.102:33599 SA ttl=254 id=30048 iplen=44 seq=457728000 win=1536 <mss 768>
```

Max rtt: 2.342ms | Min rtt: 2.342ms | Avg rtt: 2.342ms
Raw packets sent: 1 (40B) | Rcvd: 1 (46B) | Lost: 0 (0.00%)
Tx time: 0.00122s | Tx bytes/s: 32894.74 | Tx pkts/s: 822.37
Rx time: 1.00169s | Rx bytes/s: 45.92 | Rx pkts/s: 1.00
Nping done: 1 IP address pinged in 1.14 seconds

Nping is a very powerful tool for traffic analysis and packet crafting. Take a moment to go through all of its options by using the following command:

```
$ nping -h
```

Nping Echo Protocol

To learn more about the Nping Echo Protocol visit [http://nmap.org/svn/nping/docs/EchoProtoRFC.txt](http://nmap.org/svn/nping/docs/EchoProtoRFC.txt).

For More Information:  
Combining tools from the Nmap project allows us to set up a simple but powerful monitoring system. This can then be used by system administrators monitoring a web server or by penetration testers wanting to surveil a remote system.

This recipe describes how to use bash scripting, cron, Nmap, and Ndiff to set up a monitoring system that alerts the user by an e-mail if changes are detected in a network.

### How to do it...

Create the directory `/usr/local/share/nmap-mon/` to store all the necessary files.

Scan your target host and save the results in the directory that you just created.

```
# nmap -oX base_results.xml -sV -PN <target>
```

The resulting file `base_results.xml` will be used as your base file, meaning that it should reflect the known "good" versions and ports.

Copy the file `nmap-mon.sh` into your working directory.

The output of the scan will be as follows.

```
#!/bin/bash

# Bash script to email admin when changes are detected in a network using Nmap and Ndiff.
```

For More Information:
# Don't forget to adjust the CONFIGURATION variables.
# Paulino Calderon <calderon@websec.mx>

# CONFIGURATION

NETWORK="YOURDOMAIN.COM"
ADMIN=YOUR@EMAIL.COM
NMAP_FLAGS=-sV -Pn -p- -T4
BASE_PATH=/usr/local/share/nmap-mon/
BIN_PATH=/usr/local/bin/
BASE_FILE=base.xml
NDIFF_FILE=ndiff.log
NEW_RESULTS_FILE=newscanresults.xml

BASE_RESULTS="$BASE_PATH$BASE_FILE"
NEW_RESULTS="$BASE_PATH$NEW_RESULTS_FILE"
NDIFF_RESULTS="$BASE_PATH$NDIFF_FILE"

if [ -f $BASE_RESULTS ]
then
  echo "Checking host $NETWORK"
  ${BIN_PATH}nmap -oX $NEW_RESULTS $NMAP_FLAGS $NETWORK
  ${BIN_PATH}ndiff $BASE_RESULTS $NEW_RESULTS > $NDIFF_RESULTS
  if [ $(cat $NDIFF_RESULTS | wc -l) -gt 0 ]
  then
    echo "Network changes detected in $NETWORK"
    cat $NDIFF_RESULTS
    echo "Alerting admin $ADMIN"
    mail -s "Network changes detected in $NETWORK" $ADMIN < $NDIFF_RESULTS
  fi
fi

Update the configuration values according to your system.

NETWORK="YOURDOMAIN.COM"

For More Information:
ADMIN=YOUR@EMAIL.COM
NMAP_FLAGS="-sV -Pn -p- -T4"
BASE_PATH=/usr/local/share/nmap-mon/
BIN_PATH=/usr/local/bin/
BASE_FILE=base.xml
NDIFF_FILE=ndiff.log
NEW_RESULTS_FILE=newscanresults.xml

Make nmap-mon.sh executable by entering the following command:
# chmod +x /usr/local/share/nmap-mon/nmap-mon.sh

You can now run the script nmap-mon.sh to make sure it is working correctly.
# /usr/local/share/nmap-mon/nmap-mon.sh

Launch your crontab editor:
# crontab -e

Add the following command:
0 * * * * /usr/local/share/nmap-mon/nmap-mon.sh

You should now receive e-mail alerts when Ndiff detects a change in your network.

How it works...

Ndiff is a tool for comparing two Nmap scans. With some help from bash and cron, we set up a task that is executed at regular intervals to scan our network and compare our current state with an older state, in order to identify the differences between them.

There's more...

You can adjust the interval between scans by modifying the cron line:
0 * * * * /usr/local/share/nmap-mon/nmap-mon.sh

To update your base file, you simply need to overwrite your base file located at /usr/local/share/nmap-mon/. Remember that when we change the scan parameters to create our base file, we need to update them in nmap-mon.sh too.

Monitoring specific services
To monitor some specific service, you need to update the scan parameters in nmap-mon.sh.
NMAP_FLAGS="-sV -Pn"

For More Information:
For example, if you would like to monitor a web server, you may use the following parameters:

```
NMAP_FLAGS="-sV --script http-google-safe -Pn -p80,443"
```

These parameters set port scanning only to ports 80 and 443, and in addition these parameters include the script `http-google-safe` to check if your web server has been marked as malicious by the Google Safe Browsing service.

### See also

- The Listing open ports on a remote host recipe
- The Fingerprinting services of a remote host recipe
- The Finding live hosts in your network recipe
- The Running NSE scripts recipe
- The Comparing scan results with Ndiff recipe
- The Discovering hosts with ICMP ping scans recipe in Chapter 2, Network Exploration
- The Scanning IPv6 addresses recipe in Chapter 2, Network Exploration
- The Gathering network information with broadcast scripts recipe in Chapter 2, Network Exploration
- The Checking if a host is known for malicious activities recipe in Chapter 3, Gathering Additional Host Information
- The Discovering UDP services recipe in Chapter 3, Gathering Additional Host Information

For More Information:

Where to buy this book


Free shipping to the US, UK, Europe and selected Asian countries. For more information, please read our shipping policy.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.