1

GETTING STARTED WITH A BASE LINUX SYSTEM AND NETWORK MAP

This chapter presents two fundamental projects: setting up a basic Ubuntu system that you'll use throughout the book and creating a network map. You'll use this system as a base on top of which you will install and run various security tools, and the network map will provide a visual overview of all the devices in your network and how they interrelate and communicate.

We'll start with a definition and overview of common Linux operating systems and then go through the steps to install a version of Linux (specifically, Ubuntu) in a virtual machine (VM), on a physical computer, and in the cloud. Regardless of where it's installed, I'll show you how to make Ubuntu more secure and then add it to your network map. Every time a new endpoint is added to your network, you must update your network map to ensure it's always up-to-date. An out-of-date network map is no use to anyone.

Linux Operating Systems

Linux is the operating system of choice, as Linux systems are open source and therefore very extensible, especially when compared to Windows or macOS. The level of control you have over the operating system and the applications that run on top of it is very granular, enabling you to have far better control over the security of your endpoints and your network.

Several Linux operating systems (or *distributions*) are available. Each distribution uses a different set of basic utilities and graphical user interfaces (GUIs), and each one looks and functions in a slightly different way. For example, Kali Linux is a distribution geared toward offensive operations and is commonly used by penetration testers to perform network assessments. Red Hat Linux is probably the most used enterprise distribution, and several other distributions are based on Red Hat, such as Fedora and CentOS. If you're interested in Linux, try various distributions to find the one you like the most.

In this book, we'll primarily use Ubuntu, which is one of the most user-friendly and among the easiest to use for beginners or those new to Linux in general. Ubuntu is available in three editions: Desktop, Server, and Core. For our purposes, the Desktop edition is sufficient. If you plan to use your Ubuntu servers for additional network services, such as a file or Dynamic Host Control Protocol (DHCP) server, the Server edition would be appropriate. Ubuntu Core is specifically for resource-limited applications, like internet of things (IoT) implementations.

The most recent versions of the Ubuntu operating system are available from ***https://ubuntu.com/download/***. These downloads will be in ISO file format, meaning the file extension will be *.iso*. ISO files are logical images or containers that can be used to emulate physical media such as CDs or DVDs.

The following sections walk through installing Ubuntu, either as a physical device or as a virtual machine on either macOS or Windows,

as well as in the cloud. Using a physical device allows you to take advantage of all of a system's resources, such as CPU and RAM, but it requires that you have a physical system available onto which you can install Ubuntu. Using virtual machines provides several useful features, like the ability to take snapshots (this will be discussed later in this chapter). Creating a virtual machine in the cloud provides additional capabilities, like easy access to your system from any location, but often comes with additional security considerations. Once you're finished with the platform-specific instructions for your setup, jump to "**Finalizing the Linux Installation**" on **page 8**.

#1: Creating an Ubuntu Virtual Machine

Throughout this book, you'll create Ubuntu systems for various purposes. Each of them will be based on the system we'll create now, which will act as a standard base operating system, on top of which you can add tools and applications necessary for securing your network.

Hypervisor Options

A *hypervisor* is software that allows you to create and run virtual machines using a guest operating system. For this initial project, you can create an Ubuntu VM using an inexpensive commercial hypervisor from VMware. Multiple editions of VMware Workstation are available from ***https://www.vmware.com/***. VMware Player (for Windows) and VMware Fusion Player (for Mac) are free for personal use, but they don't have some of the more advanced features we'll want to take advantage of in later chapters. I recommend using VMware Workstation Pro and VMware Fusion Pro. The commercial license for either of these is relatively inexpensive. An alternative solution is to use the free Workstation Player to begin with and then upgrade to the commercial license if you need to. The step-by-step instructions for Workstation and Player editions are mostly the same, with some slight differences between Workstation and Fusion.

Another option is to use VirtualBox, a free solution for creating and managing VMs maintained by Oracle. VirtualBox is available for all major operating systems, and you can download it from ***https://www.virtualbox.org/wiki/Downloads/***.

VMware Workstation and VMware Player for Windows

To create your VM in VMware Workstation or VMware Player, follow these steps:

1. Click **File ▸ New Virtual Machine** in VMware.
2. On the New Virtual Machine screen that opens, choose **Typical (recommended)** and click **Next**.
3. Select **Installer Disc Image File (iso)**.
4. Using the **Browse** button, navigate to and select the Ubuntu ISO you downloaded earlier; then click **Next**.
5. The Easy Install wizard will ask for the user details for your VM; fill out the Full Name, User Name, and Password fields, and click **Next**.
6. Give your VM a meaningful name indicating its role on your network when asked.
7. Save the VM to the default location (or anywhere you desire) and click **Next**.
8. Set the virtual disk size to 40GB if your host machine has enough disk space; otherwise, accept the default 20GB.
9. Store the virtual disk as a single file, rather than split into multiple files and click **Next**.
10. Click **Customize Hardware**.
11. If your host has enough RAM, increase the RAM of the VM from 2GB to 4GB.
12. Set Processors to 1.
13. Under Network Adapter, select **Bridged** mode to give your VM its own independent IP address and network connection.
14. Click **Sound Card ▸ Remove**.
15. Click **Printer ▸ Remove**.

16. Click **Finish**.

Your virtual machine will be created, and the operating system will begin installing.

VMware Fusion and VMware Fusion Player for macOS

Once you've installed VMware Fusion or VMware Fusion Player, follow these steps to create your first VM:

1. Click **File ▸ New ▸ Continue** in VMware.
2. Drag and drop your ISO file onto the VMware Fusion window, or click the **Use Another Disc or Disc Image** button to locate the file in your filesystem; then click **Continue**.
3. The Easy Install wizard will ask you for the user details for your VM; fill out the Display Name, Account Name, and Password fields.
4. Ensure the Make your home folder accessible to the virtual machine checkbox is not ticked and click **Continue**.
5. Click **Customize Settings**.
6. Save the VM to the default location (or anywhere you desire).
7. Set the virtual disk size to 40GB if your host machine has enough disk space; otherwise, accept the default 20GB.
8. In the **Processors and Memory** menu, if your host has enough RAM, increase the RAM of the VM from 2GB to 4GB, and set Processors to 1.
9. Untick the **Connect** checkbox to either add or disconnect the following peripherals within their context menus: sound card, floppy, printer, and camera. (Disconnecting unused or superfluous peripherals from your virtual machines removes potential attack vectors.)

Click the **Play** button to start your VM, and the operating system installation will begin.

VirtualBox

The steps for creating a VM in VirtualBox are the same whether you're using a Windows PC or Mac as the host system. Once you've downloaded and installed VirtualBox, follow these steps to create a VM:

1. Click the **New** button at the top of the VirtualBox window.
2. Provide a relevant name for your VM, specify the location to save the files (the default folder is usually fine), and select the correct operating system from the drop-down menus: **Linux ▸ Ubuntu (64-bit)**; then click **Continue**.
3. If your host has enough RAM, increase the RAM of the VM from 2GB to 4GB, and click **Continue**.
4. Select **Create a New Virtual Hard Disk Now** and then click **Create**.
5. Select **VMDK** as the hard disk format and click **Continue**.
6. Select **Dynamically Allocated** and click **Continue** or **Next** (depending on your OS).
7. Set the virtual disk size to 40GB if your host machine has enough disk space; otherwise, accept the default 32GB and click **Create**.
8. Select the VM in VirtualBox and click **Settings**.
9. Go to **Settings ▸ System ▸ Motherboard**.
10. Under Boot Order, untick the **Floppy** checkbox.
11. Go to **Settings ▸ System ▸ Storage**.
12. Select the CD drive (it'll be listed as **Controller: IDE** and have a CD icon next to it).
13. In the attributes pane, click the **CD icon** to choose a disk file, and point it at your Ubuntu ISO file.
14. Under **Settings ▸ Audio**, untick the **Enable Audio** checkbox.
15. Under **Settings ▸ Network ▸ Adapter 1**, switch the **Attached to** drop-down to **Bridged Adapter** so your VM will be assigned its own IP address and be logically separate from the host system's network settings.
16. Click **OK**.

NOTE *The options for hard disk format are VDI, VHD, or VMDK. VDI is VirtualBox's proprietary format. VHD was developed by Microsoft, is*

compatible with Windows, and can be easily mounted under the Windows operating system as a virtual disk. VMware originally developed VMDK, but it's now an open file format. VMDK is compatible with both VirtualBox and VMware, so if you choose to switch from one to the other, your virtual hard disks shouldn't cause any challenges.

#2: Creating a Physical Linux System

Instead of creating a virtual machine, you might want to use a physical computer and install Ubuntu the same way you'd install Windows or macOS directly onto the hardware. Using a physical system has benefits like increased performance or reduced resource requirements in terms of memory and processing power. The main drawback is that physical systems usually aren't as flexible as virtual machines. As you progress through this book, you'll be asked to create multiple Linux systems, so we'll assume that you'll use mostly virtual machines. However, should you choose to use physical systems for each of these projects, you should still be able to follow along.

To create a physical Ubuntu system, you need a *bootable USB drive*, which means you'll install Ubuntu on a USB that you can plug into any computer and install it from there.

Bootable USB on Windows

On a Windows computer, the simplest way to create a bootable Ubuntu USB is with Rufus, a small utility specifically for creating bootable media. Download the latest version from ***https://rufus.ie/***. Rufus is a *portable executable*, which means you don't need to install it; just download and run it. Once downloaded, follow these steps:

1. Plug in a USB thumb drive at least 16GB in size. Rufus will format this USB drive, so make sure it doesn't contain anything you want to keep.
2. Run the Rufus executable.

3. Once Rufus is open, ensure the Device drop-down menu indicates that the correct USB drive is selected. It's often easiest to plug in only your target USB device and unplug any others.
4. Under Boot Selection, choose **Disk or ISO Image**.
5. Click **Select**.
6. Navigate to your Ubuntu ISO file and select it.
7. Once selected, Rufus will load a set of default settings for the bootable USB; accept them and click **Start**.
8. Rufus might display a pop-up asking whether you want to write the media in ISO or DD image mode; choose **ISO Mode** and click **OK**. When installing Ubuntu later, if you aren't able to proceed with the installation or it seems to hang, repeat this process and select **DD Mode** instead.
9. Rufus will display a pop-up to inform you that it will format the USB drive; click **OK** to proceed.

Bootable USB on macOS

Etcher is an open source utility for macOS used for writing operating system images to removable media such as USB drives and SD cards. Download the latest version from ***https://www.balena.io/etcher/***. Once it's downloaded and installed, follow these steps:

1. Plug in a USB thumb drive at least 16GB in size. Etcher will format your USB drive, so make sure it doesn't contain anything you want to keep.
2. Run Etcher.
3. Once Etcher is open, click **Flash from File**, and select your Ubuntu ISO file.
4. Click **Select Target** and select your USB drive.
5. Click **Flash** to create your bootable Ubuntu USB (you might be asked to enter your computer password to allow Etcher to make changes to the USB).
6. The flashing process will begin, and a progress bar will appear. Once the process completes, you may be informed that "The disk

you inserted was not readable by this computer." If so, just eject the USB; don't choose Initialize.

Using the Bootable USB

When the process completes, you'll have a bootable Ubuntu Linux USB drive. Plug it into the computer on which you want to install Ubuntu and boot or reboot it. You might have to change the system's boot order so it boots from the USB instead of the internal hard drive. To do that, you need to interrupt the boot sequence, which is typically done by pressing ESC, F8, F10, or F12. Do an internet search to find the correct interrupt key for your computer, or reboot the system and press each of those keys until you successfully enter the computer's basic input/output system (BIOS).

NOTE *Technically, most modern computers use the Unified Extensible Firmware Interface (UEFI), which has improved features over the outdated BIOS. We'll use the terms BIOS and UEFI interchangeably.*

From the BIOS, which is responsible for hardware management outside your operating system, change the boot order so that the computer boots from the USB first. Then restart the computer, and it will boot into the Ubuntu installation environment. On a Mac, just hold the OPTION key while the system boots and then choose to boot from USB.

#3: Creating a Cloud-Based Linux System

It's common to move network infrastructure to the cloud, which just means running our services on someone else's computer(s). Websites and the web servers that run them are often easier to access (from anywhere in the world) and manage in the cloud than they would be on our private networks and VPN servers. (We'll cover VPNs in greater detail in **Chapter 5**.) In this section, we'll explain how to create your Linux computer using a cloud service provider. We'll use Vultr for this project as it's relatively inexpensive, it's reliable, and it presents an

easy learning curve if you haven't used a cloud provider before. The steps should be similar regardless of provider, whether you're using Amazon Web Services, Microsoft Azure, or something else.

1. Create an account at ***https://www.vultr.com/***.
2. On the account dashboard, click + ▸ **Deploy New Server**.
3. Choose **Cloud Compute**. The other options (High Frequency, Bare Metal, and so on) are for specialist applications, not suitable for our purposes.
4. Choose a location for your server. Choosing a location geographically close to you can improve access speeds to your VM; however, if you want to obfuscate your location, choose a location in a different country.
5. For Server Type, choose the latest available version of Ubuntu.
6. Choose a server size. The cheapest option is a good place to start; you can always upgrade your VM later if necessary.
7. Supply a hostname for your server.
8. Click **Deploy**.

The service provider will now instantiate your Ubuntu VM, which is the same as creating a VM in VMware or VirtualBox. This process can take some time. Once your VM is confirmed to be up and running, your service provider will supply the IP address, username, and password to access your VM. You'll then be able to complete the steps in the following sections to set up and secure your VM.

Finalizing the Linux Installation

If you created your Linux system in the cloud or using VMware and Easy Install, booting the VM will automatically install Ubuntu, create your user account, and present you with the Ubuntu desktop environment, which will be similar to a Windows or Mac desktop. If you used VirtualBox or are creating a physical Linux system, you'll need to complete some additional steps to get to that stage.

In VirtualBox, follow these steps:

1. Click the **Start** button to boot the VM.
2. Using the Ubuntu installation wizard, select your desired language and click **Install Ubuntu**.
3. Select your keyboard layout and click **Continue**.
4. On the Updates and Software screen, select **Minimal Installation** as you won't need a lot of the additional software that would otherwise be installed with the operating system.
5. Tick both checkboxes to allow software updates to be installed from various sources.
6. Click **Continue**.
7. On the next screen, the wizard will ask if you want to erase the disk and install Ubuntu, with a warning, as shown in **Figure 1-1**. Click the **Advanced Features** button and select **Use LVM with the New Ubuntu Installation**. Using LVM provides greater flexibility and control over your disks and their partitioning. LVM allows for advanced features such as naming logical volumes and dynamically resizing partitions and virtual hard disks when required.

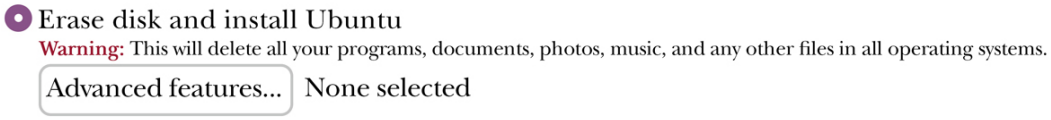

***Figure 1-1**: Ubuntu installation type prompt*

Keep in mind that this installation wizard is referring only to the virtual machine and the virtual hard disk that is attached to it (which we created earlier). It does not affect the physical hard drive of your host system. There is no risk of losing your files or data by proceeding with the installation inside the VM.

8. Click **OK ▸ Install Now**.
9. You'll be asked to write the changes to disk (meaning the virtual hard disk of the VM). Click **Continue** to accept the configuration

you just set for this VM.

As Ubuntu installs, you'll be asked for certain settings for the operating system, such as your location (for time zone settings), your name, your computer or hostname, and your user details such as username and password. Set those as appropriate and continue the installation. Eventually, the operating system installation will complete, and you'll be presented with the Ubuntu desktop environment.

WARNING *Do not set or allow the user to log in automatically, as that configuration isn't secure for any computer. Always use the* ***Require my Password to Log in*** *setting.*

The first time you log in, Ubuntu will ask you to configure online accounts and whether you want to share anonymous statistics with the developer. This system needs to be a secure system and therefore shouldn't be connected to services such as Google or Microsoft cloud services. Skip all of those configuration options and disallow sharing of data wherever possible. This advice is good for life (if you're concerned about privacy), not just the configuration of Ubuntu virtual machines.

You'll follow the same steps to complete Ubuntu installation on a physical system, the only difference being the disk partitioning will affect the physical hard drive within the computer, and not a virtual hard drive. Once you've installed Ubuntu, reset the boot order in the BIOS as you did before so the computer boots from the internal hard drive instead of USB, and also remove the bootable USB from the computer.

Hardening Your Ubuntu System

Now that you've created a base virtual or physical machine, you'll make some initial configuration changes to ensure your system is secure. This process is called *hardening*, which broadly means keeping the system up-to-date with the latest operating system and software

patches, installing some additional management software, and altering configuration files to make the system more secure.

#4: Installing System Packages

In Ubuntu, you'll use the Advanced Package Tool (APT) to ensure the system is up-to-date with all of the latest patches. In Linux, people use the term *packages* to refer to software, and APT is a package management utility used to install, uninstall, update, or otherwise manage the tools and software on your system.

APT is a command line interface (CLI) utility, which means you'll use the Linux Terminal to interact with it, rather than a GUI tool like Windows Update.

NOTE *Most operating systems have a CLI; Windows has Command Prompt and PowerShell, and macOS has its own Terminal. Essentially, a CLI is a more direct way to interact with the operating system using text commands. A CLI will look like a simple text editor with a prompt for your input. Command Prompt, Linux Terminal, and macOS Terminal are all black with white text by default. PowerShell is blue.*

In a cloud deployment, you might have access only to the Linux Terminal by default, with no access to a GUI. If that's the case, you'll be presented with a terminal window immediately upon logging in. Otherwise, to access the terminal in Ubuntu, click the **Activities** menu at the top-left side of the screen in Ubuntu, type **Terminal**, and click it when it appears, the same way you'd search for and open an application in the Windows Start menu.

By default, even as an administrator, you can't run certain commands or perform some actions on a Linux system because you don't have the necessary permissions (called *privileges* in Linux). A lot of commands and actions are reserved for *superusers*, or the *root* user account in Linux. As a non-root (that is, a nonsuperuser) user in Linux,

you need the `sudo` command, which stands for *superuser do*. For example, to use APT to update all the installed packages in your Ubuntu system, use the following commands, pressing ENTER after each command to execute them:

```
$ sudo apt update

$ sudo apt upgrade
```

The first command, `sudo apt update`, retrieves the list of available updates for each application currently installed. The second command, `sudo apt upgrade`, downloads and installs those updates. When prompted, enter your password; making you authenticate to run privileged commands is a `sudo` security feature. Every time you run a command with `sudo`, the action is logged in the */var/log/auth.log* file, so all administrative actions can be audited after the fact. When asked to continue installing packages, enter `Y` (for yes) and press ENTER.

WARNING *On the command line in Linux and macOS, when your prompt displays a dollar sign (`$`), this means you're currently in the context of (that is, operating as) a normal, non-administrative user. If your prompt displays a hash mark (`#`), you're in the context of the root user and have full system access to make changes, move files, and delete files. Be careful if you're operating in the context of root, as it's easy to make mistakes and cause problems with your operating system. It's always best to work primarily in the context of a normal user and use `sudo` when using the command line.*

When you install new packages, APT often also installs any dependencies required for those packages (otherwise, your software would look for things it depends on, not find them, and fail to run successfully). However, when you remove or uninstall software, those dependencies might be left behind. Having unnecessary applications on your systems is insecure, as an attacker might exploit a vulnerability in those

leftover packages to gain access to your network or use them to perform other nefarious activities. Run `sudo apt autoremove` and `sudo apt clean` as shown here to remove any no longer needed dependencies and delete previously downloaded packages, respectively:

```
$ sudo apt autoremove

$ sudo apt clean
```

To install new packages, use `sudo apt install`. A useful package that allows you to access and administer your system remotely via the command line is SSH (for *secure shell*). Run `sudo apt install openssh-server` to install SSH (to install a different package, you would substitute `openssh-server` for the package name).

You can install multiple packages with `apt` at the same time like this:

```
$ sudo apt install openssh-server package_name1 package_name2
```

Again, type your password and enter `Y` if prompted. With SSH installed, you'll be able to configure remote access to your system.

#5: Managing Linux Users

Part of managing your network security is managing the user accounts and hosts within your network. You may need to add new users to your Ubuntu machines, such as a new user account for a new service or application, or to allow others to administer your systems. Adding new users is an administrative function and requires the `sudo` command. Use the `adduser` command to add new users:

```
$ sudo adduser username
```

You'll be asked to specify a password for the user, but using passphrases is better, because they're easier to remember, tend to be

longer, and are harder to crack. (We'll discuss passphrases and creating strong passwords in more detail in **Chapter 11**.)

You also can set names, phone numbers, and other information for your users if you want; otherwise, press ENTER to leave these fields blank.

Deleting a user is just as easy:

```
$ sudo deluser username
```

In addition, you may want to give your new user `sudo` privileges to allow them to administer your system, which you can do with the `usermod` command:

```
$ sudo usermod -aG sudo username
```

The `-aG` (add group) parameter will add the user to the *sudo* group. User groups in Linux are a collection of user accounts, and they're used to assign privileges and permissions to specific user accounts, such as the ability to read and write certain files. Keep in mind, however, that the fewer users with sudo privileges the better. Always practice the principle of least privilege and allow users only as much control as they require on a day-to-day basis. Providing administrator credentials and privileges to more people than necessary will lead to a far less secure network configuration.

Finally, you can reset the password for a given user with the `passwd` command:

```
$ sudo passwd username
```

Managing the users in your network is an important part of keeping your network secure. Having superfluous user accounts, especially if they have more privileges than they require, provides an easy way for adversaries to compromise and gain a foothold inside your network.

This is easily preventable, so always be aware of the risks of additional or unnecessary user accounts.

Besides managing the users in your environment, each of your endpoints has a hostname, which is a friendly name or human-readable name used to identify the host. Often, these are configured as some default value by the operating system when it's installed (like `ubuntu` for Ubuntu systems). It can be beneficial to choose a naming scheme for your hosts and to ensure each host has a different name. In Windows networks, for example, multiple hosts cannot have the same hostname, because this results in conflicts that create administrative problems within the network.

You can check the hostname of your Linux system with the `hostname` command:

```
$ hostname

ubuntu
```

To change the hostname, use the `hostname` command again, but this time use `sudo` and specify the desired hostname:

```
$ sudo hostname your_hostname
```

Run the `hostname` command again to confirm the change has occurred. Reboot your server to make the change permanent.

#6: Securing Remote Access

Now that you've made it possible to access the system remotely with SSH, you need to lock down that capability so only authorized users can log in to this host. Several settings are involved in this process. You'll disable password login in favor of SSH keys, as well as disallow the root account from logging in directly via SSH. Allowing superusers such as root to log in interactively using utilities like SSH is bad prac-

tice because it allows attackers to perform attacks such as brute-forcing (continually guessing potential passwords until they find the one that works) and then be able to log in with complete access to your system. Likewise, with your other user accounts, using SSH keys to log in instead of passwords eliminates an entire class of potential attacks (username and password guessing) against your systems.

Generating SSH Keys

SSH key pairs are generally accepted to be more secure than passwords or passphrases. SSH keys are cryptographically secure keys that can be used to authenticate a client computer (your local host) to an SSH server (your Ubuntu system). The first part of a key pair is your private key, which is held by and identifies your client, and must remain absolutely secret, just like a password. The other half is the public key, which can be freely shared. The public key is provided to your SSH server and is capable of decrypting your private key, thereby allowing authentication between the two endpoints. Each local user account that you want to use to log in to your Ubuntu system will need its own set of public and private keys.

To create an SSH key pair, open a terminal window on the computer you plan to use as your SSH client (the computer that will connect to your Ubuntu system via SSH). Enter `ssh-keygen` and press ENTER. Press ENTER again to accept the default file in which to save the keys. The default location of this file will be:

- Windows: *C:|Users|<user>|.ssh|id_rsa*
- macOS: */Users/<user>/.ssh/id_rsa*
- Linux: */home/<user>/.ssh/id_rsa*

Next, you'll be asked for a passphrase for your private key, which is optional but recommended. By using a passphrase in addition to your SSH keys, your private key is never exposed on the network, meaning that to get access to your private key, attackers need access to your

computer (if they have that, it's probably already game over). Once you've entered your passphrase (or not), press ENTER, and your key pair will be created.

To provide the public key file to your Ubuntu system (or any other server you want to use your key pair to connect with), enter the following command:

```
$ ssh-copy-id user@your_ubuntu_ip
```

```
The authenticity of host '192.168.1.10' can't be established.

ECDSA key fingerprint is
aa:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa:aa.

Are you sure you want to continue connecting (yes/no)? yes
```

The prompt about an ECDSA key fingerprint might be shown, which just means that the remote computer wasn't able to identify your local computer (because it hasn't connected to it in this way before). If you receive this prompt, type `yes` and press ENTER. Your Ubuntu system will ask you for the password of the user account you're trying to use to connect (that is, the password for the remote user account). Enter the password, and the process is complete. At this point, you can use `ssh user@your_ubuntu_ip` to log in to your Ubuntu system, and you'll be prompted for the SSH key passphrase (not the passphrase for the user on the Ubuntu system) if you set one.

Disallowing Password Authentication

Next, change the SSH configuration on your Ubuntu system to disallow password authentication, forcing the use of your SSH keys to log in. Log in to your Ubuntu system as a standard, non-root user, and open the SSH configuration file in the terminal using Nano, the text editor installed by default on many Linux distributions, with the following command:

```
$ sudo nano /etc/ssh/sshd_config
```

Find the line with the setting: `# PasswordAuthentication yes`. To search for text in Nano, press CTRL-W and then type your search term and press ENTER. The setting is currently commented out (the `#` at the start of the line tells SSH to ignore that line) because `yes` is the default configuration and doesn't need to be set explicitly. Remove the `#` from the beginning of the line, and change `yes` to `no`. For every system you create (and on which you enable SSH), you must change this setting.

Disabling Root Login

It's also prudent to disable the ability for the root user to log in remotely. As mentioned earlier, on Linux, the root user has the highest level of permissions or privileges on the system. By disabling its ability to log in, you remove the capability of any would-be attackers to gain privileged access to the system. Technically, the root account on the most recent versions of Ubuntu can't log in because it's locked by default, but it's always good to ensure it's unable to log in anyway. Find the line:

```
PermitRootLogin prohibit-password
```

and change `prohibit-password` to `no`. With that done, save the changes you've made to the file. Press CTRL-O and then press ENTER to overwrite the file you're editing. Press CTRL-X to exit the file and return to the terminal.

Restart the SSH service so that it's reloaded with the new configuration, using the following command:

```
$ sudo systemctl restart ssh
```

There's one more thing left to test. Earlier, you disabled the ability to log in via SSH using password authentication by modifying the configuration file in */etc/ssh/sshd_config*. From any computer in your network, try to SSH into your Ubuntu system using an account on that computer, with the password you configured (not the user account you provided with your SSH key):

```
$ ssh user@your_ubuntu_ip

user@your_ubuntu_ip: Permission denied (publickey).
```

Here, `user` is the username you use to log in to the system, and `your_ubuntu_ip` is the IP address of your Linux system. If you're able to log in successfully, go back to the "**Disallowing Password Authentication**" section and make sure your configuration is correct, or reboot Ubuntu. Leaving access open would create a vulnerability in your network, which is easy to fix but potentially a big problem if left unchecked.

Remote Login with SSH

Both macOS and Windows have SSH built in. Using the computer for which you generated an SSH key and copied to your Ubuntu system, connect to your new Linux system by entering the following command:

```
$ ssh user@your_ubuntu_ip

Enter passphrase for key '/Users/user/.ssh/id_rsa':

❶ Welcome to Ubuntu (GNU/Linux 5.8.0-44-generic x86_6)

❷ * Documentation:  https://help.ubuntu.com

* Management:     https://landscape.canonical.com
```

```
* Support:        https://ubuntu.com/advantage

❸ 6 updates can be installed immediately.

5 of these updates are security updates.

To see these additional updates run: apt list --upgradable

Your Hardware Enablement Stack (HWE) is supported until ❹ April
2025.

❺ Last login: Mon Mar  8 17:02:46 from 192.168.1.12
```

When you log in via SSH to Ubuntu, the operating system outputs a lot of useful information. The first line indicates which version of the operating system is currently installed ❶. There are links to documentation and how to get help ❷, followed by a list of any available updates for the system or installed packages ❸. This useful list indicates when you need to run the update commands described earlier in "**Installing System Packages**." Next, Ubuntu shows when support for your operating system expires ❹, at which point you'll need to upgrade your distribution with the `sudo apt dist-upgrade` command or build a new system with the latest operating system. Finally, the last successful login to the system is shown ❺, which can be useful for identifying suspicious activity. If the last login was at 3 AM or from an unfamiliar IP address, you might want to investigate that activity (unless you're in the habit of administering your network and systems in the early hours).

#7: Capturing VM Configurations

At this point, if you're using a VM, your virtual machine is at a known-good state; you've finished configuring and hardening it, and it's ready to be used in your network. It's a good idea to save this state so that if something goes wrong, you can return to it without completely rebuilding the system. One of the benefits of using virtual machines is

the ability to take *snapshots*. Snapshots save the current state of a virtual machine, including its power state (on, off, suspended, and so on), so that you can quickly return to a saved state if necessary. You can't do that with a traditional, physical system, although we've all been in situations where we wish we could. You might choose to take a snapshot before installing a new program, for example, or before changing a VM's network settings or before adding or deleting a new user.

Taking Snapshots in VMware

Regardless of the version of VMware you're using, simply right-click the virtual machine for which you want to create a snapshot, click **Snapshots ▸ Snapshot**, name your snapshot, and wait for the process to complete. That's it. Now, if something happens to your VM, right-click the VM and then click **Snapshots ▸ Restore Snapshot** to revert to this known-good state. It's that simple.

Taking Snapshots in VirtualBox

In VirtualBox, click the menu button (three bullets and three lines) on the VM in the virtual machine panel on the left of the VirtualBox window and then click **Snapshots**. To create a snapshot, click **Take**. Name your snapshot, click **OK**, and wait for the process to complete. To revert to a snapshot, click the snapshot and then click **Restore**.

NOTE *Every snapshot you create will effectively make a duplicate of your virtual machine. Multiple snapshots can take up a large amount of space on your host computer. Keep this in mind when creating snapshots, and remove old snapshots when they're no longer needed. Some cloud providers charge for snapshot storage as well, so keep that in mind when creating snapshots in your cloud dashboard. Snapshots are also not a good long-term backup method. (We'll discuss backups at length in* **Chapter 9**.*)*

Network Topology

Understanding how your systems and devices connect to and communicate with each other is critical when it comes to cybersecurity. With that in mind, let's take a crash course on the *Internet Protocol (IP)* and IP addressing. IP is a standard protocol that defines the format of data sent over a network allowing computers and other network-connected devices to communicate with each other.

Each of your computers and other network-connected devices requires an *IP address*. An IP address is comparable to a street address or a post office box; computer A sends network traffic to computer B by embedding computer B's IP address in the data it sends. It's the same as writing an address on an envelope. Any intermediate devices between the two computers can interpret this address from the data and pass it along until it reaches its destination, just like the postal service.

Two commonly used versions of the internet protocol currently exist, version 4 and version 6, which means we have two types of IP addresses, IPv4 and IPv6. While IPv6 has been around since the 1990s, it still isn't used often today. We won't cover it in detail in later chapters as it's largely outside the scope of this book, but it's important to be aware of what it is and why it exists. IPv4 addresses are written in what is known as *dotted quad notation*, which is a fancy way of saying they're composed of four numbers separated by periods, such as 192.168.1.1. Each of the four numbers can range from 0 to 255, meaning that IPv4 addresses range from 0.0.0.0 to 255.255.255.255, or a total of 4,294,967,296 possible addresses.

So many network-connected devices now exist in the world that there aren't enough IPv4 addresses to go around, which is one of the reasons IPv6 was created. IPv6 has a larger address space, with a total of more than 340 trillion, trillion, trillion addresses. To put that in perspective, that's 100 times more addresses than there are atoms on the

surface of Earth, which is convenient as more and more internet-connected devices come online. Eventually, IPv6 will be in common use, and every device will be able to have its own public IPv6 address, until we run out of those (probably not in my lifetime).

As there aren't enough IPv4 addresses for everyone, we've had to come up with clever workarounds to connect all of our devices to the internet. One of those solutions is *network address translation (NAT)*. Using NAT, several devices can be contacted via one IP address.

When you connect your home or office to the internet through your router, your internet service provider assigns you (and your network) a public IP address. You can find your IP address using services like ***https://www.whatismyip.com/***. IP addresses are usually *dynamic*, meaning that when you disconnect from and reconnect to the internet, you will often receive a different IP address.

Your internet router is responsible for routing traffic from your private, internal network, to the public internet, and vice versa, which is how you're able to access services and browse the internet generally. At a high level, NAT takes the public IP address assigned to your router and translates the traffic it receives so that traffic bound from the internet to one of your internal computers or devices receives the traffic destined for that specific device. It's similar to the way letters and packages are delivered to office buildings at their street address, and then a clerk or mail department determines where that package needs to go internally, forwarding it to the right recipient. It also works in the reverse; traffic outbound from your computer to the internet must be translated from the internal IP address of your computer to the public IP address of your router before being forwarded on to reach its intended destination and return a service, like a web page.

Different IP addresses are reserved and available for use on the public internet and your private network. The private address ranges that can be used for private networks are:

- 10.0.0.0 to 10.255.255.255
- 172.16.0.0 to 172.31.255.255
- 192.168.0.0 to 192.168.255.255

All other IP addresses form part of the publicly available IP ranges or are as yet unassigned.

#8: Checking Your IP Address

Addressing is usually handled by a router or server. If you have a wireless router, you can log in to that device, take a look at the client list or DHCP settings, and find out which address range is being used. Alternatively, you can just check the address of your computer(s). Understanding the addressing in your network, in addition to maintaining an asset inventory and network map, also means you can keep account of the addresses assigned to specific devices, the users who are responsible for or assigned to a specific piece of hardware, and where a particular device is physically located, among other metadata. (We discuss asset management further in **Chapter 8**.)

On Windows

On Windows, click the Start menu, enter cmd, and press ENTER to open the command prompt. Next, enter `ipconfig` and press ENTER:

C:\Users\user>`ipconfig`

Windows IP Configuration

Ethernet adapter Ethernet:

`IPv4 Address` . . . . . : 192.168.1.126

`Subnet Mask` . . . . . : 255.255.255.0

`Default Gateway` . . . . . : 192.168.1.1

```
C:\Users\user>
```

The output will show your current IP address, the relevant *subnet mask*, and the *default gateway* your computer is using. The subnet mask indicates to which segment of a network your computer belongs, and the default gateway is the address of the device your computer uses to access other networks, such as the internet. The gateway is probably your router.

On a Mac

Finding the IP address of a Mac is similar. Open a terminal window, type `ifconfig`, and press ENTER:

```
$ ifconfig

--snip--

en0:
flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST>
mtu 1500

    options=400<CHANNEL_IO>

    ether 78:8d:43:a4:ce:29

    inet 192.168.1.120 netmask 0xffffff00 broadcast 192.168.1.255

    media: autoselect

    status: active
```

On macOS, the `ifconfig` output is a bit different: `inet` is the internet or IP address, `netmask` is the subnet mask, and `broadcast` is the *broadcast address* of the network. The subnet mask here is displayed in *hexadecimal (hex)* instead of decimal. Hex is another notation used

by computers, different from the dotted quad notation. A broadcast address is a reserved address in a network used for sending traffic to all devices in that network segment (we'll cover this more in **Chapter 10**, which discusses network security monitoring).

On Linux

On a Linux system, open a terminal window, and enter the following:

```
$ ip addr

--snip--

2: ens32: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
fq_codel state UP group default qlen 1000

    link/ether 00:0c:29:db:ee:7c brd ff:ff:ff:ff:ff:ff

    altname enp2s0

    inet 192.168.1.30/24 brd 192.168.1.255 scope global dynamic no-
prefixroute ens32

       valid_lft 4106sec preferred_lft 4106sec

    inet6 fe80::66e:1ae7:861f:9224/64 scope link noprefixroute

       valid_lft forever preferred_lft forever
```

The IP address is listed under `inet` again; the broadcast address is `brd`, and the subnet mask is shown as `/24`, in *CIDR notation*. CIDR is another way of representing the same subnet mask information in a shorter format.

#9: Creating a Network Map

To get a better understanding of your network and see more granular details, like ingress and egress points (that is, the places where traffic enters and leaves your network), creating a *network map* or *network diagram* is beneficial. A network diagram is a graphical representation of your network that allows you to see the overall architecture at a glance and makes it easier to identify potential issues when it comes to securing your network.

draw.io (***https://www.draw.io/***) is a free and easy-to-use cloud editor that allows users to create various types of diagrams, one being the network diagram. Alternatively, Microsoft Visio is a commercial solution that achieves the same objective. If you choose to use draw.io, load the site and open the Citrix drop-down from the menu on the left. You can then drag and drop the relevant representations, as shown in **Figure 1-2**, from the menu on the left to the canvas on the right.

***Figure 1-2**: draw.io diagramming tool*

At its most basic, a small network typically comprises a modem/router, usually provided by your internet service provider, which connects your network and all your devices to the internet, as well as a few devices: computers, laptops, mobile devices, peripherals such as printers, and so on. Keeping track of all the devices on your network better enables you to secure your network, because you know what should be connected and allowed to communicate, both within the network and between your network and the internet.

Always keep your network diagram up-to-date. Whenever you add a new computer, laptop, mobile device, switch, virtual machine, or

other system or device to your network (and remove them as well), you should update your network diagram. When it comes to a transient device whose IP addresses aren't static, it might be worth assigning the device a static IP address in your router (see **Chapter 4**). Otherwise, you can track the IP range that might be assigned to those devices. Even if a device isn't always connected to your network, maintain a record of devices that will be consistently expected to connect (**Chapter 4** discusses this in greater detail), and note those IP addresses in your network diagram.

A network diagram also allows you to see where you can implement additional security controls to improve your security posture. For example, **Figure 1-3** shows a small, basic network.

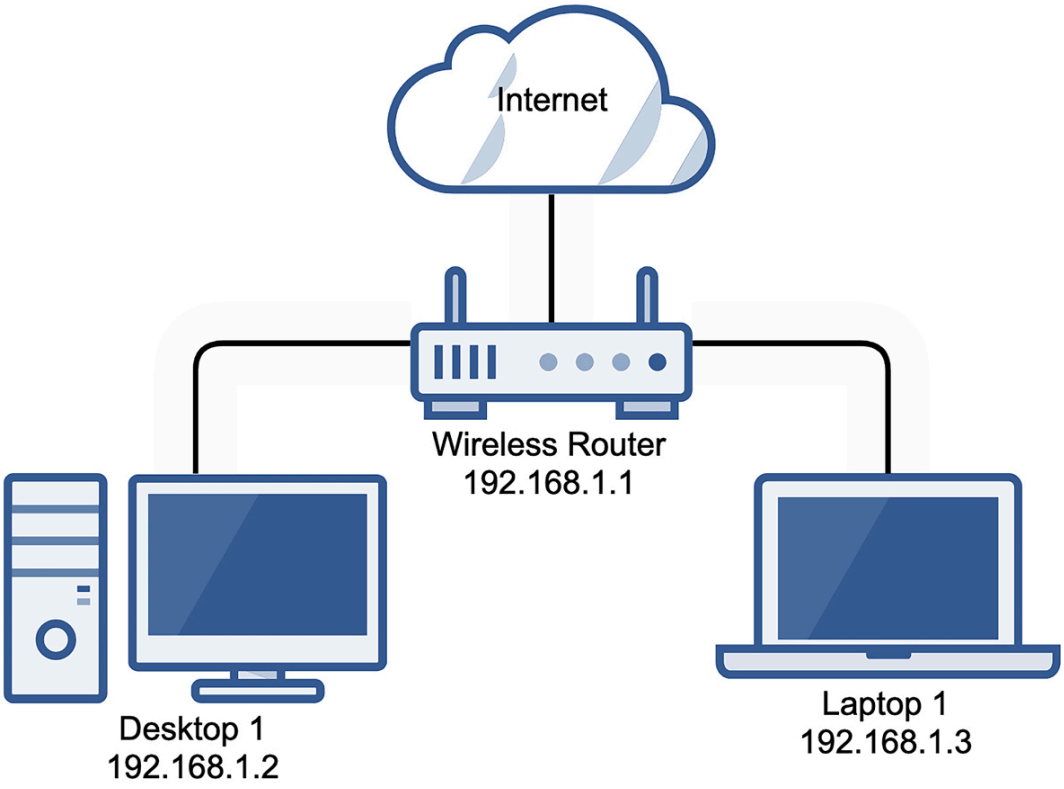

***Figure 1-3**: A small network*

This network layout is typical of most home networks, where a modem/router connects the network to the internet, and all the endpoints use that device as a gateway to the public network. The prob-

lem with this network architecture is that it also allows adversaries to use the same network infrastructure to access the private network, without many obstacles in their path. Later in this book, you'll learn how to improve this network's security by adding a firewall between the wireless router and the internet to better manage the ingress and egress traffic and block suspicious traffic entering or leaving the network (see **Chapter 3** for more details on firewalls).

When mapping your network, collect as much information as possible about all the devices it includes, such as their IP addresses, MAC addresses, hostnames, purpose, primary user or owner, location, serial numbers, and so on. Start with your computers and move on to mobile devices, such as phones and tablets, and then any IoT devices you might have—if your TV or refrigerator connects to the internet, be sure to capture those as well.

#10: Transferring Files

You may want to transfer files to your Linux machine from another system or from your Linux machine to your local computer. The intuitive rsync tool can synchronize files and folders, either between two locations on one system or between two systems across a network. To transfer specific files from one computer to another, use the following:

```
$ rsync -ruhP --remove-source-files --protect-args
" /path/to/source/ " \

  " user@computer_ip:/path/to/destination/ "
```

Immediately following the `rsync` command are four flags. The `r` flag stands for *recursive*, meaning that everything inside the source folder will be copied to the destination. The `u` flag stands for *update*, which indicates to rsync that if it finds a copy of a file in the destination location that is newer than the copy in the source directory, skip it. Next, `h` is the typical flag used for *human-readable output*: any numbers

(dates, file sizes, and so on) will be shown in an easier-to-read format. The `P` flag is for *progress*, which tells rsync to output the progress of the copy to the screen so you can see how much data has been transferred, how much remains, and how long until the process is expected to complete.

Following this first set of flags, the `--remove-source-files` argument tells rsync to delete the source files once they've been successfully copied, and `--protect-args` tells rsync to interpret the following arguments (the source and destination directories) as one continuous string each, even if they're separated by a space character, which would normally indicate to the terminal that the directories were separate and independent. Without this argument, if your source path has a space in it, the command will interpret each section of the path on either side of the space character as a separate path. The same is true of the destination path. You can exclude one or both of those arguments if you don't want to delete the source files after copying them or if your source and/or destination directories have no space characters in them.

In practice, the following shows how the command might look between two Linux servers (which we'll cover in greater detail in **Chapter 5**):

```
$ rsync -ruhP --remove-source-files --protect-args test.txt \
user@192.168.1.30:/tmp

Enter passphrase for key '/Users/user/.ssh/id_rsa':

sending incremental file list

test.txt

          0 100%    0.00kB/s    0:00:00 (xfr#1, to-chk=0/1)
```

As we mentioned in the "**Securing Remote Access**" section, remember to enter the passphrase for the SSH key pair you created earlier, not the passphrase for the user account you're using for the file transfer. The bottom of this listing contains the progress percentage, the current transfer speed, expected time remaining, and the number of files remaining in the transfer.

NOTE *The best feature of rsync is its ability to resume transfers if they get interrupted. The Secure File Transfer Protocol (SFTP) and Secure Copy Protocol (SCP) are alternatives you could use to transfer files between systems, but neither can stop a transfer and pick it up at the same point so that you don't lose your transfer progress while transferring potentially large files or directories. For these reasons, rsync is superior to SFTP and SCP, so we'll primarily use rsync throughout the remainder of this book.*

Summary

In this chapter, you built and secured your first Linux machine, which you'll need to follow the examples in several chapters of this book. You also learned how to harden your Linux system to increase its security and your overall security posture, including creating a secure SSH configuration by utilizing SSH key pairs, and the fundamentals of how to manage users in your network. You learned about mapping your network topology, how computers and other devices are interrelated, and how they communicate with each other. In **Chapter 2**, you'll learn how to layout your network to passively increase your overall security.