# Firewall = Filter for incoming & outgoing network traffic

Through a Network Interface Card (NIC) a computer can send packets to whatever IP address or receive packets from whatever IP address. Firewall is a hardware/software piece that restricts this freedom.

A **firewall** is a software/hardware pieces that monitors and controls incoming and outgoing network traffic based on predetermined security **rules**.

# MAC address, IP address, port address

MAC address is at the Link level 🡪 determine a NIC

IP address is at the Transport level 🡪 determine a computer

Port address (port number) is at the Application level 🡪 determine which application on a given computer to process packets.

A socket = an ip address + port

Diagram

Description automatically generated

# Port: open/closed, listening, scanning

A port is a number, together with ip address, to determine which application to handle incoming/outgoing packets. Commonly used ports: HTTP – 80; HTTPS – 443; FTP – 21; FTPS / SSH – 22.

When an application is registered to its OS to process incoming packets with a specific port, the application is said to listening on that port. There can be only one application for a port and not every port has an application to listen on. If there is no application listening on a port, incoming packets to that port will simply be rejected by the computer's OS.

A firewall can be set to reject all packets for a port, i.e. all incoming/outgoing packets at this port will be dropped by the OS as if they don’t exist; in this case we say this port is closed. If a port is not restricted by any firewall, it is called an open port. An open port does not mean there is an application listening on it.

Malicious hackers commonly use [port scanning](https://en.wikipedia.org/wiki/Port_scanning) software to find which ports are "open" (unfiltered) in a given computer, and whether or not an actual service is listening on that port. They can then attempt to exploit potential [vulnerabilities](https://en.wikipedia.org/wiki/Vulnerability_(computing)) in any services they find. So it’s a good security practice to close all the unused ports on a computer to prevent public access to services on the computer.

# Terminal/console <-> Shell <-> stdin, stdout, stderr streams of process

Every process is associated with 3 streams: stdin, stdout, stderr.

Shell = program that launches processes. Shell will redirect those 3 stdin, stdout, stderr streams to files or terminal.

Terminal = console = keyboard + monitor. Terminal is associated with shell.

Shell is the middle between Process and Terminal

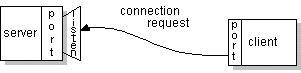
# Socket and Port

A socket is one endpoint of a two-way communication link between two programs running on the network. A socket is bound to a port number so that the TCP layer can identify the application that data is destined to be sent to.

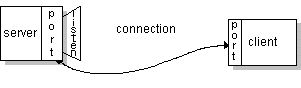
An endpoint is a combination of an IP address and a port number. Every TCP connection can be uniquely identified by its two endpoints. That way you can have multiple connections between your host and the server.

Normally, a server runs on a specific computer and has a socket that is bound to a specific port number. The server just waits, listening to the socket for a client to make a connection request.

On the client-side: The client knows the hostname of the machine on which the server is running and the port number on which the server is listening. To make a connection request, the client tries to rendezvous with the server on the server's machine and port. The client also needs to identify itself to the server so it binds to a local port number that it will use during this connection. This is usually assigned by the system.



If everything goes well, the server accepts the connection. Upon acceptance, the server gets a new socket bound to the same local port and also has its remote endpoint set to the address and port of the client. It needs a new socket so that it can continue to listen to the original socket for connection requests while tending to the needs of the connected client.



On the client side, if the connection is accepted, a socket is successfully created and the client can use the socket to communicate with the server.

The client and server can now communicate by writing to or reading from their sockets.

Note: Two programs cannot listen to a same port. Only one program can open a socket on one port on one IP address at one time.

# Service vs Process; Background vs Foreground; Register a service

A process is said to run in the foreground if it interacts with users; otherwise, the process is said to run in the background. Processes run in the foreground, services run in the background.

A process is an instance of an executable file. A service in Windows is often an instance of svhost.exe process. There are some exceptions for this.

For a process to run as a service, it needs to register with Windows Service Control Manager.

Every service has to have some typical feature so that Windows Service Control Manager can manage the service’s start/stop/status. So, for a program to become a service, it has to follow some pattern.

# Environment Variables

An environment variable is a variable about the environment in which programs run in.

Every program runs in an environment. The environment is obviously the operating system, but here we mean more: an environment for a program is always restricted to the permission of the user that runs the program and many other parameters. So an environment is a limited space of the operating system.

All kinds of programs use environment variables to answer questions like: What’s the [name of the computer](https://www.digitalcitizen.life/change-computer-name-windows/) where I’m installed? What is the name of the user account that is running me? What is my current working directory? Where is Windows installed? Where are the temporary files stored on this computer? Where is the OneDrive folder for this user account?

There are two types of variables: user variables, specific to each user account, and system variables that apply to all user accounts.

Graphical user interface, text, application, email

Description automatically generated

For CM:

View all the environment variables: set

View environment variable myVar: echo %myVar%

Assign value X for environment variable myVar: set myVar=X

# Shell

A shell is a [user interface](https://en.wikipedia.org/wiki/User_interface) for access to an [operating system](https://en.wikipedia.org/wiki/Operating_system)'s services. In general, operating system shells use either a [command-line interface](https://en.wikipedia.org/wiki/Command-line_interface) (CLI) or [graphical user interface](https://en.wikipedia.org/wiki/Graphical_user_interface) (GUI). It is named a shell because it is the outermost layer around the operating system.

Sometimes, a shell means just a command interpreter, and there comes terminals that provide user interface. So users type commands on terminal, and then terminal call shell to interpret user commands to system calls to the OS.

Bash shell is a popular Unix shell. It is used also on MAC.

**Wordplay: kernel vs shell**

Kernel: core services; Shell: wrap around kernel. Shell is an user interface for users to interact with core service.

# SSH = secured shell = shell that uses public key

SSH, also known as Secure Shell, is a network protocol that gives users, particularly system administrators, a secure way to access a computer over an unsecured network. SSH also refers to the suite of utilities that implement the SSH protocol.

SSH utilities are widely used by network administrators for managing systems and applications remotely, enabling them to log in to another computer over a network, execute commands and move files from one computer to another.   
The name “secured shell” has 2 parts:

* SSH is a shell (i.e. a user interface) to operate network services. Bash shell is a user interface for controlling OS’s service, and now Secure Shell is a user interface for controlling network services.
* SSH is secured. It uses public key cryptography

SSH refers both to the cryptographic network protocol and to the suite of utilities that implement that protocol. SSH uses the client-server model, connecting a Secure Shell client application, which is the end where the session is displayed, with an SSH server, which is the end where the session runs. SSH implementations often include support for application protocols used for terminal emulation or file transfers. SSH can also be used to create secure tunnels for other application protocols, for example, to securely run X Window System graphical sessions remotely. An SSH server, by default, listens on the standard Transmission Control Protocol (TCP) port 22.

# Fingerprint key

In public-key cryptography, a public key fingerprint is a short sequence of bytes used to identify a longer public key. Fingerprints are created by applying a cryptographic hash function to a public key.

# Installing software in Linux

There are 2 kinds of software packages: .deb (Debian) and .rpm (RedHat)

## .deb and apt-get

APT (Advanced Packaging Tool) is a tool, commonly used to install packages, remotely from the software repository.

Install a new package:

apt-get install firefox

Remove an installed package:

apt-get remove firefox

To update the local APT database:

apt-get update

To upgrade **all** the installed packages to the newest version:

apt-get upgrade

One usually combines update and upgrade:

apt-get update; apt-get upgrade

To upgrade a single package

apt-get --only-upgrade install firefox

## .rpm and yum

Install a new package:

yum install firefox

Remove an installed package:

yum remove firefox

yum does not have a local database of packages like apt. But to install the newest version of a package:

yum update firefox

To install the newest versions of all current packages:

yum update

## apt vs apt-get

The APT tool is more complicated than just apt-get, it has apt-cache, etc, and many options for low-level management.

The [apt commands](https://manpages.debian.org/jessie/apt/apt.8.en.html) have been introduced to solve this problem. apt consists some of the most widely used features from apt-get and apt-cache leaving aside obscure and seldom used features.

While apt does have some similar command options as apt-get, it’s not backward compatible with apt-get.

|  |  |  |
| --- | --- | --- |
| **apt command** | **the command it replaces** | **function of the command** |
| apt install | apt-get install | Installs a package |
| apt remove | apt-get remove | Removes a package |
| apt purge | apt-get purge | Removes package with configuration |
| apt update | apt-get update | Refreshes repository index |
| apt upgrade | apt-get upgrade | Upgrades all upgradable packages |
| apt autoremove | apt-get autoremove | Removes unwanted packages |
| apt full-upgrade | apt-get dist-upgrade | Upgrades packages with auto-handling of dependencies |
| apt search | apt-cache search | Searches for the program |
| apt show | apt-cache show | Shows package details |

## **Ubuntu Linux and root account password**

By default root account is locked under Ubuntu Linux. Therefore, you cannot log in as root or use ‘su -‘ command to become a superuser. To run all administrative command use the sudo command on Ubuntu. sudo allows a permitted user to execute a command as the superuser or another user. Ubuntu setup your default account (the one created during installation) to run all administrative commands. For example create a new user called bar, you need to type sudo command as follows:  
$ sudo adduser bar  
Password:  
When sudo asks for a password, you need to supply **YOUR OWN** password. In other words a root password is not needed. Here are few more examples.

Start / stop / restart services stored in /etc/init.d/ directory

$ sudo /etc/init.d/ssh stop  
$ sudo /etc/init.d/networking restart  
$ sudo systemctl restart ssh