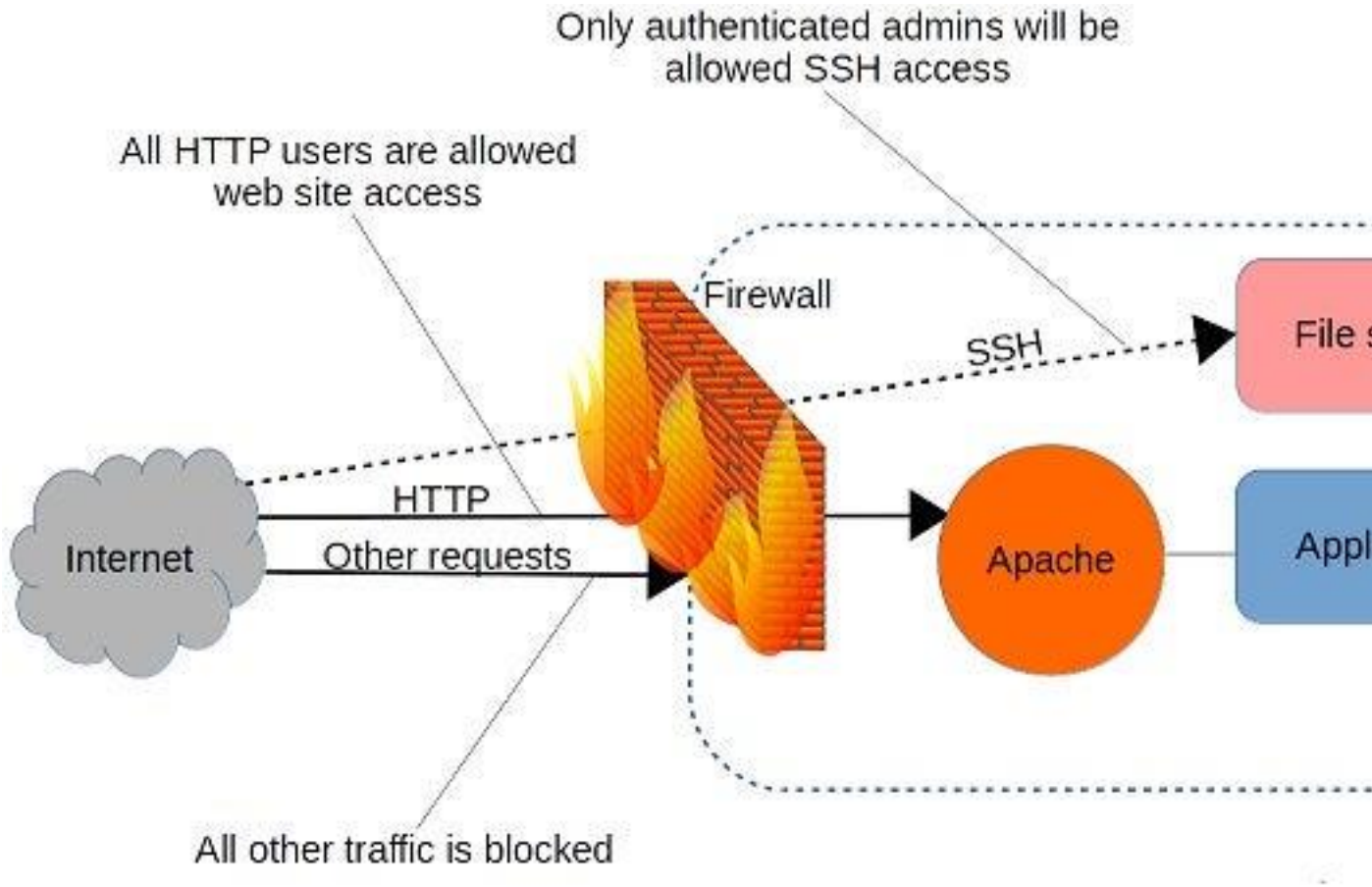
The firewall

A firewall is a set of rules. When a data packet moves into or out of a protected network space, its contents (in particular, information about its origin, target, and the protocol it plans to use) are tested against the firewall rules to see if it should be allowed through. Here's a simple example:



A firewall can filter requests based on protocol or target-based rules

What is IPTables?

The way the **Firewall** works is quite simple. It creates a barrier between trustworthy and untrustworthy networks so your system can be safe from malicious packets.

But how we are going to decide what is safe and what not? By default, you do get some privilege to set up rules for your Firewall but for more detailed surveillance of incoming and outgoing packages, **IPTables** are what you require the most.

IPTables can be used for personal computing or can also be applied to the entire network. Using **IPTables**, we will be defining a set of rules by which we can monitor, allow or block incoming or outgoing network packets.

Rather than just focusing on the entire theory part, we are only going to discuss what matters in the practical world. So let's start with understanding the core concepts of **IPTables**.

Understanding the Concept of IPTables

While discussing **IPTables**, we must understand 3 terms: **Tables**, **Chains**, and **Rules**. As these are the important parts, we are going to discuss each of them.

So let's start with **Tables**.

Tables in IPTables

There are 5 types of **tables** in IPTables and each has different rules applied. So let's start with the most common table "**Filer**".

- **Filter Table** – This is the default and main table while using **IPTables**. It means whenever you won't mention any specific table while applying rules, they will be applied to the filter table. As its name suggests, the role of the Filter table is to decide whether the packages should be allowed to reach their destination or deny their request.
- **NAT (Network Address Translation)**– As its name suggests, this table allows users to determine the translation of network addresses. The role of this table is to determine whether to modify and how to modify the source and destination of the packet address.
- **Mangle Table** – This table allows us to modify the IP headers of packets. For example, you can adjust **TTL** to either lengthening or shorting network hops that the packet can sustain. Similarly, other IP headers can also be modified according to your preference.
- **RAW Table** – The main use of this table is to track connections as it provides a mechanism for marking packets to view packets as a part of an ongoing session.
- **Security Table** – Using the Security table, users can apply internal **SELinux** security context marks on network packets.

For the most use cases, the last 2 types (**RAW** and **Security**) of the table don't have much to do and only the first 3 options are counted as main tables.

Now, let's talk about **Chains**.

Chains in IPTables

They behave at points in the route of the network where we can apply rules. In IPTables, we 5 types of **chains** and we will discuss each of them. Keep in mind that not each type of chain is available for each type of table.

- **Pre-routing** – This chain is applied to any incoming packet once it is entered the network stack and this chain is processed even before any routing decision has been made regarding the final destination of the packet.
- **Input Chain** – It is the point where a packet enters the network stack.
- **Forward Chain** – It is the point where the packet has been forwarded through your system.
- **Output Chain** – The output chain is applied to the packet when it originated through your system and goes out.
- **Post-routing** – This is the complete opposite of the pre-routing chain and is applied to forwarded or outgoing packets once the routing decision has been made.

Now, the only thing left to discuss is **rules**, and it's the easiest one out of the 3 that we have discussed here. So let's complete what's left on the theoretical part.

Rules in IPTables

Rules are nothing but the set or individual commands by which users manipulate network traffic. Once each chain will come into action, the packet will be checked against defined rules.

If one rule does not satisfy the condition, it will be skipped to the next one and if it satisfies the condition, the next rule will be specified by the value of the target.

Each rule has two components: the **matching component** and the **target component**.

- **Matching Component** – They are different conditions to define rules which can be matched by protocol, IP address, port address, interfaces, and headers.
- **Target Component** – This is an action that will be triggered once the conditions are satisfied.

This was the explanation part and now we will be covering basic commands related to **IPTables** in Linux.

What is SELinux?

The "SE" in SELinux stands for **Security-Enhanced**. Linux is basically an operating system like Windows, Android, and iOS. However, rather than being developed by a single company, Linux has always been an open-source project. The source code of the Linux kernel – the "core" of Linux – is freely available to developers both for non-profit and commercial projects. Based on the Linux kernel, several Linux-based operating systems have been created. These are referred to as "distributions" and some of the most well-known are Ubuntu, Debian, and Fedora.

What does "Security-Enhanced" actually mean?

The code of the Linux kernel is constantly being developed by companies, volunteers, and non-profit organizations. Security-Enhanced Linux is an **extension of the Linux kernel** and is available as a standalone security module. It was officially integrated in the Linux kernel in 2003. Some Linux distributions offer SELinux as standard, but you can easily disable the module if you don't need it. SELinux gives administrators **greater control over the processes**running on their system. Any processes that are not considered essential are blocked. This greatly reduces the risks associated with security vulnerabilities in user programs.

Even if you trust a program, it can still be a good idea to restrict access rights, because if the program were to be hijacked by a third party this could have very serious consequences indeed. If **programs infected by malware** have access to all of the data and processes on a system, they can do a lot of damage. By restricting access, SELinux limits the potential for damage.

Strict control of operating system access

The special SELinux security architecture is based on the principle of Mandatory Access Control (MAC). Unlike the standard Linux kernel, SELinux only allows access to operating system processes and files if this is absolutely essential. The aim is to ensure **data confidentiality and integrity** by implementing a strict access control strategy and corresponding security policies. With SELinux, the operating system and the user programs are clearly separated from one another.

How does SELinux limit access rights?

The standard Linux setup uses Discretionary Access Control (DAC). With this type of mechanism, if users and applications have the necessary privileges, they generally have unlimited access to operating system data and processes. When Mandatory Access Control is implemented, as in SELinux, an administrator uses precisely defined security policies to define additional attributes that determine the **conditions and contexts in which a user may access certain operating system processes or files**. If the conditions or contexts (i.e. attributes) have not been approved, access is denied.

For the purposes of control in SELinux, the administrator assigns the **following labels**:

- User
- Role
- Type
- Level

These labels can be assigned for every process and file and then integrated in the defined security policies. For example, an application might only be granted access to folders that have a specific label. The process of checking the security policies is referred to as **SELinux enforcement**.

The advantages and disadvantages of SELinux

SELinux hinders or prevents the abuse of user rights that can occur when user programs have security flaws. The operating system is, therefore, well-protected. Linux distributors offer the SELinux module with various different policy packages and corresponding security policies, which simplifies configuration of the security layer. Authorized administrators can also **define the security policies themselves**.

Although SELinux gives administrators far more control over processes and systems, it does not really help them to resolve problems. Whenever SELinux blocks access, it issues an error message, but these **messages are often very vague**, which makes troubleshooting rather difficult. SELinux is also a relatively complex module. Many administrators feel that dealing with the security policies and defining attributes is **too complicated or requires too much effort**. Moreover, implementing SELinux can have a slightly negative effect on the performance of the operating system.

ACL

Access Control Lists (ACLs) provide access control to directories and files. ACLs can set read, write, and execute permissions for the owner, group, and all other system users.

An ACL consists of a set of rules that specify how a specific user or group can access ACL enabled files and directories. A regular ACL entry specifies access information for a single file or directory. A default ACL entry is set on directories only, and specifies the default access information for any file within the directory that does not have an access ACL.

When setting a default ACL on a directory, its subdirectories inherit the same rights automatically. ACLs can be used with the `btrfs`, `ext3`, `ext4`, `OCFS2`, and `XFS` file systems, as well as mounted NFS file systems.