Engineer

Firewall

the rule set that decides which traffic passes

What it is

A firewall is a filter on a network path that compares each packet or connection with a list of rules and then lets it through, drops it or answers with a refusal. It can be software on the machine it protects, a separate appliance between two networks, or a service a cloud provider runs in front of a virtual machine. Linux administrators often say packet filter for the same thing, and cloud providers sell firewalls under names such as security group. In plural, firewalls usually come in layers: one at the provider, one on the host.

NIST's guidance on the subject, SP 800-41 Rev. 1 from September 2009, sorts them by how much of the traffic they read. A packet filter looks at addresses, protocol and port numbers, one packet at a time. A stateful firewall also remembers connections, so the reply to a request the server made itself comes back in without a rule of its own. An application-proxy gateway terminates the connection and reads the protocol inside it. A host-based firewall runs on the server or laptop it guards and knows which program owns a socket.

On a Linux server the filtering happens in the kernel's netfilter framework. nftables, available since kernel 3.13, replaces the older iptables tools, and front ends such as ufw write the rules for people who would rather state intent than chains. Whatever writes them, the model is the same: an ordered list, read from the top, where the first rule that matches decides.

A firewall is often confused with neighbouring tools. A web application firewall reads HTTP requests and blocks attack patterns in them; a network firewall never looks that deep. An intrusion-banning tool reads logs and adds temporary rules against addresses that misbehave, which means it depends on a firewall underneath. A VPN decides who is on the network at all. Each of them answers a different question, and none replaces the plain rule list.

Use the concept in a decision

Put “What is a firewall?” to work in a real decision. We can help relate the concept to the system you are building or buying. Accounts, passwords and permissions arranged so people hold exactly the access their job needs, and leavers lose it that day.

Explore our services Get in touch →

Test the assumptions behind the term

Use the explanation and linked work to frame questions for your project. We can examine the assumptions with you before a tool is chosen. Wrote 1,000 RBAC rules to secure Linux servers — 1,000 access rules written to close a security gap

Wrote 1,000 RBAC rules to secure Linux servers Get in touch →

Why it matters

Every service that listens on a public address is a door, and the firewall is the inventory of which doors exist. The decision a team actually faces has three parts: where the firewall lives, what it allows by default, and who is allowed to change it.

A provider firewall stops traffic before it reaches the machine, keeps working if someone gains root on the host, and costs the server nothing in CPU. Its rules live in a console or an API, outside the server’s own configuration, so they drift unless they are written as code. A host firewall travels with the machine, can rate-limit a single port and can be tested from the machine itself, but a compromised root account can rewrite it, and so can software that manages its own rules. Running both is common and sensible, at the price of keeping two lists in step: every port opened or closed is a change in two places.

The default is the second part. NIST recommends denying all traffic that is not expressly permitted, and for inbound traffic almost everyone agrees. Outbound filtering is where the trade-off bites: it limits what a compromised service can reach, and it breaks package updates, certificate renewal and monitoring until each destination has its rule.

The third part is ownership. A firewall in default-deny mode is only as correct as the list of ports someone declared, so the useful artefact is that declared list, kept in version control and checked against the live rules. The software is free on every mainstream Linux distribution; what costs money is the attention needed to keep the declaration true. And a firewall is a second line. A service bound to the loopback address cannot be reached from outside whatever the rules say, which is a stronger guarantee than a deny rule in front of a service listening everywhere.

What it looks like in practice

On one host a rate limit for the SSH port sat in the rule list for a year without a single packet reaching it, because a broader allow above it matched first. The status tools listed the limit all year, and they were right to: it existed. The checks were rewritten to read rule position rather than presence: they now compare the order of the numbered rules, and of the chain the kernel is actually filtering with, against what was intended. Every port gets one rule and every rule names a protocol; SSH is rate-limited and has no separate allow. The five public ports are declared together and checked as a set, so an open port the declaration leaves out turns the check red, and the security report lists the rules in the order packets meet them, with a note on how to read that order.

The first server was built by hand and only later got its firewall, which left a window in which a fresh machine was reachable from the internet. For a second server at a different provider, the firewall is created by its own play ahead of the machine, so the new machine meets the internet with its rules already in place. That provider attaches firewalls by tag, so the tags are created first: a rule aimed at a tag needs the tag to exist. Each provider's firewall is written directly against that provider's REST API with a plain HTTP task. The task does nothing in a dry run, so changes are rehearsed on the firewall play itself, since a provider firewall costs nothing to create and can be deleted again.

Containers can get past a firewall without touching it. Docker writes its own entries into the kernel's packet filter, and a port it publishes for a container is matched before anything the host's hardening role put there, so a container can be reachable from outside whatever the firewall was told. This platform's containers run under systemd through Podman and Quadlet, share the host's network and publish no ports, so there is no container port for anything to route around the rules.

Where it goes wrong

A rule appears in the status output and never fires. Rules are evaluated in order and the first match wins, so a broad allow placed above a narrower limit or deny shadows it completely. The listing still shows both. Read the rules as the kernel does, numbered and in order, put specific rules above general ones, and test from a machine outside:

```
ufw status numbered
nft list ruleset
```

A container answers on a port the firewall denies. Docker publishes ports by redirecting traffic in the kernel's NAT table, before it reaches the chains ufw filters, and Docker's own documentation says this effectively ignores the firewall configuration. Publish to the loopback address only, put a provider firewall in front, or run containers without published ports as described above.

IPv4 is locked down and IPv6 is wide open. The two address families have separate rule sets in iptables, and ufw writes IPv6 rules only when IPv6 is enabled in its defaults file. A host with a public IPv6 address and an IPv4-only rule set is exposed on every port it listens on. Use an nftables table of the `inet` family, which sees both, or audit both listings, and scan the host's IPv6 address from outside.

The SSH session freezes the moment the firewall is enabled. Default deny went live before the rule for SSH, or the SSH daemon was moved to another port while the rule still names 22. Add the SSH rule first, keep the provider's console open in another window, and schedule an automatic rollback a few minutes out that is cancelled once a new session gets in.

A rule opens more than its author meant. In ufw, a rule that names a port without a protocol opens it for both TCP and UDP. Name the protocol on every rule, and declare the expected surface as a whole so that anything extra fails a check rather than waiting for an audit.

Further reading

- Firewall — the definitions NIST publications use, each linked to its source document.
- SP 800-41 Rev. 1: Guidelines on Firewalls and Firewall Policy — firewall types, default-deny policy and rule management, from NIST.
- ufw(8) — Ubuntu's firewall front end, including rule order and the `limit` rule.
- nftables wiki — the netfilter project's documentation for the framework that replaces iptables.
- Packet filtering and firewalls — Docker's own account of how published ports pass around a host's rules.

Glossary

Choose for your actual constraints

Bring the deployment path, the operational risks and the people on call. We can scope a change with checks, a recovery route and clear ownership.

See what it costs Get in touch →

Make the terminology practical

Ask how “What is a firewall?” applies to your organisation. A concrete example from your current system is enough to start the discussion.

Explore our services

welcome@engineer.company +45 50 10 61 84 Copenhagen, Denmark

Take this page with you

Contact card

Carefully Crafted With Love by D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – present · Engineer ApS · All rights reserved.

Printed from https://engineer.company/glossary/firewall/