Security against Brute attack

This volume of failed logins is **normal background noise** for any server with port 22 open to the internet — automated bots scan all IPv4 addresses and start hammering common usernames within minutes of your server going online. Your fail2ban and UFW are doing exactly what they're supposed to: stopping them from actually getting in.

That said, you can eliminate most of this traffic entirely. If you only have SSH keys enabled, the thousands of password guesses are harmless and simply never succeed.

Is this normal?

- **1564 failed logins / 192 bans / 7329 firewall blocks** is a typical-to-calm level for a public server. Many freshly-unprotected boxes see tens of thousands of attempts daily.
- The important test: do any of the failed logins *succeed*? If they don't (key-based auth only), this is nuisance traffic, not a breach.
- One thing worth checking: if most failures come from a **single IP**, fail2ban may be misconfigured or its filter may not match your auth log format — otherwise it should be banning that IP. Verify with `sudo fail2ban-client status sshd`.

What you can do to cut it down

The single most effective change is to make password login impossible, so the brute-force attempts can never succeed:

bash

- **Use SSH keys only** — this alone eliminates the overwhelming majority of brute-force risk.
- **Disable root login** (`PermitRootLogin no`) so attackers can't target root directly.
- **Rate-limit at the firewall**: `sudo ufw limit OpenSSH` (allows ~6 connections per 30s per IP, dropping brute-force spikes at the kernel level).
- **Use a non-standard port** (e.g., 2222) — reduces scanner traffic dramatically, though it's security-by-obscurity.

- **Restrict access** to known admin IPs or a VPN (Tailscale/WireGuard) if your IP is static — then port 22 isn't exposed at all.

Before you change anything

- Confirm key-based login **already works** from a fresh session before disabling passwords, or you'll lock yourself out.
- Keep one SSH session open, back up the config (`cp`, then `sshd -t` to validate syntax), then `sudo systemctl reload ssh` — not restart, so existing sessions survive.
- Check that your cloud provider has a web/serial console as a break-glass fallback.

Bottom line: there's nothing special or alarming in your numbers — it's the standard pre-scan traffic every internet-exposed server receives. If you want to reduce it (and the log noise), the fixes are key-only auth + disable root + firewall rate limiting. Once that's in place, the 1564 failures are wasted effort against a door that's already locked.

Want me to check whether any of the failed logins actually succeeded on a real account, or walk through a specific hardening step?

Used ssh keys only and disabled root login and I am using custom port 2010 for ssh. Is there anything if I changed ssh port for fail2ban ?

Yes — you should update fail2ban so its **ban action** targets your port 2010. The good news: fail2ban detects attacks by parsing your auth log, not by watching ports, so it still *sees* the failed logins on any port. The problem is only the ban itself — if fail2ban's action still blocks port 22 (or the `OpenSSH` app profile that points to 22), the banned IPs aren't actually being blocked from connecting to 2010.

What to change

On Ubuntu with UFW, the cleanest way is a dedicated UFW application profile for port 2010, then point the sshd jail at it.

1. Add an app profile for your port in `/etc/ufw/applications.d/openssh-server`:

```
text
```

2. Point the sshd jail at it in `/etc/fail2ban/jail.local`:

text

3. Reload and verify:

bash

The `port = 2010` line tells fail2ban the port to report/ban, and the `action = ufw[application="OpenSSH-2010"]` makes UFW insert a `deny from <ip> to any app OpenSSH-2010` rule that actually drops traffic to 2010.

Important caveat

There's a known failure mode: if you leave fail2ban's default `iptables`/`ufw` action pointing at 22 or forget to update it, `fail2ban-client status sshd` will show bans, but `ufw status` will have no matching deny rules and the IPs keep reaching 2010 — exactly the "fails to block ssh on custom port" problem.

Two checks to confirm it's working after changing the port:

- `sudo fail2ban-client status sshd` — shows the banned IP count.
- `sudo ufw status numbered` — verify a matching `DENY ... to 2010/tcp` (or `to ... app OpenSSH-2010`) rule exists for recently banned IPs.

If you'd rather not maintain a custom profile, the alternative is the simpler stock line `action = ufw` with `banaction = ufw` under `[DEFAULT]` — but that still binds to the `port = 2010` you set, so make sure that's updated.

One extra tip since you moved to 2010: add a second jail watching for activity on port 22 too — legacy scanners may still probe it, and banning them at 22 keeps noise out of your UFW logs.

[OpenSSH-2010] title=Secure shell server (custom) description=OpenSSH on custom port ports=2010/tcpJust append this?

Yes — just append that block to the end of `/etc/ufw/applications.d/openssh-server`. Adding a second `[OpenSSH-2010]` section alongside the existing `[OpenSSH]` (port 22) profile is fine; UFW supports multiple profiles in one file and each is referenced by its own bracketed name.

bash

Append at the bottom:

text

Two quick notes:

- Don't add a second `[OpenSSH]` with the same name, or UFW will complain about a duplicate app — a uniquely-named block is the correct approach.
- After saving, re-read the profile list and confirm it appears:

bash

Then you can use it in boththe firewall rule and the fail2ban action:

bash

If `ufw app list` doesn't show the new profile after saving, just re-run `sudo ufw app update OpenSSH-2010` (or reload UFW) to refresh its parsed application definitions.

Revision #1
Created 2026-09-01 06:26:17 UTC by meson
Updated 2026-09-01 06:27:24 UTC by meson