FIELD NOTES FROM A PRESALES SOLUTIONS ARCHITECT

UFW Turns On After Every Reboot — It Doesn't. A Stale netfilter-persistent Snapshot Does

A field note about a firewall that "came back" after every reboot even though nobody turned it on — the old snapshot that was really loading, why ufw status said active, and the two checks I now run before I touch UFW.

Daanial Farrukh · 14 November 2026 · https://quabin.com/ufw-active-after-reboot-netfilter-persistent/

The ticket was short and tired.

"UFW turns itself on after every reboot. We disable it, and next boot it's back."

The server was an Ubuntu VM running cPanel. Web and mail traffic was already filtered upstream, by the platform's own firewall in front of the VM. The guest firewall was supposed to be off.

"What do you do when it comes back?"

"We run ufw disable. Everything works again."

"And then?"

"We also masked the UFW service so it can't start. It still comes back."

I asked one question.

"If the service is masked and it still comes back — what makes you sure it's UFW?"

Silence. Then: "ufw status says active."

That's the trap. The thing you see after boot looks like UFW, carries UFW's chain names, and ufw status even agrees with you. But UFW is not the program that put it there.

What Was Really Loading at Boot

netfilter-persistent is a small Ubuntu service that restores a saved iptables rule set at every boot. It comes with the iptables-persistent package. Its job is simple: read /etc/iptables/rules.v4 and rules.v6, and load them before the network is up.

It does not care where the rules came from. Whatever was in the kernel when the file was saved is what comes back — forever, until someone saves again or removes the file.

On this server, the story was in the timestamps:

- iptables-persistent was installed about eighteen months earlier.

- /etc/iptables/rules.v4 and rules.v6 were written **36 seconds** after the install.
- At that moment, UFW was running.

The installer asks one question: "Save current IPv4 rules?" Someone pressed Yes. That is my theory of how it happened — the timestamps fit it, but no one remembers the click. What is not a theory is what the file contained:

- Only UFW's own chains (ufw-user-input, ufw-before-input and the rest).
- A default policy of :INPUT DROP and :FORWARD DROP.

So the boot sequence was:

Kernel boots → netfilter-persistent loads the 18-month-old UFW snapshot → INPUT policy is DROP → traffic stops → someone runs ufw disable → chains flushed → traffic flows → next reboot → repeat

The same snapshot had also been written to /etc/nftables.conf a few minutes later. nftables.service tried to load it at every boot too, and failed on syntax errors. That failure was the only reason there were not two copies of the problem.

Where this shows up most:

- Ubuntu and Debian servers where someone installed iptables-persistent "just in case" while UFW was enabled.
- Servers handed over between teams, where the guest firewall was later replaced by an upstream or cloud firewall.
- Hosts with cPanel, CSF or Docker, where several tools write rules and nobody owns the whole table.

Why ufw disable "Worked" and Masking Didn't

Here's the part that trips people up.

UFW was never re-enabled. Its own switch lives in /etc/ufw/ufw.conf as ENABLED=no, and that line had not changed in almost a year.

When you run ufw disable on a UFW that is already disabled, it does not rewrite ufw.conf — there is nothing to change. But it still flushes the UFW chains from the kernel. That flush removed the snapshot's rules and reset the policies, so the team saw the server come back and concluded, reasonably, that UFW had been on.

Masking ufw.service could never help. UFW was not starting. A different service was loading UFW-shaped rules.

Why ufw status Said "Active"

This is the part I did not expect, and I checked the source to be sure.

ufw status does not read ENABLED from ufw.conf. In the iptables backend (backend_iptables.py), it decides "active" by checking whether the chain ufw-user-input exists in the kernel. If the chain is there, UFW reports itself as active.

So after every boot:

- netfilter-persistent created ufw-user-input from the snapshot.
- ufw status saw the chain and said **Status: active**.
- The team saw "active" and blamed UFW.

The status command was telling the truth about the kernel, not about UFW.

Why Nobody Sees It in Testing

Because nobody reboots a production server to test a firewall. The snapshot was saved once, during a package install, and the server ran for months without a restart. By the time the reboots started, nobody connected "the firewall is back" with a package installed a year and a half earlier.

And every symptom pointed at UFW: UFW chain names, UFW status output, and a UFW command that made it go away. Every clue was real. The conclusion was wrong.

How to Actually Fix This

Read-only checks first. Nothing in Steps 1 to 3 changes the server.

Step 1: Ask UFW's own config what it thinks.

```
grep ENABLED /etc/ufw/ufw.conf
```

If this says ENABLED=no while ufw status says active, UFW is not the one loading the rules. Stop blaming it.

Step 2: Check whether netfilter-persistent is enabled.

```
systemctl is-enabled netfilter-persistent
```

enabled means a saved rule set is restored at every boot. Also check its sibling:

```
systemctl is-enabled nftables
```

Step 3: Look inside the saved file.

```
grep -c ufw /etc/iptables/rules.v4
```

Any number above zero means the snapshot holds UFW chains. Then look at the default policies:

```
grep '^:' /etc/iptables/rules.v4
```

:INPUT DROP in a file that only has UFW chains is the snapshot of a running UFW. Check the file date against the package install date:

```
ls -l --time-style=full-iso /etc/iptables/
```

```
grep iptables-persistent /var/log/dpkg.log*
```

If the two times are seconds apart, the rules were saved during the install.

Step 4: Stop the boot-time restore. (This changes the server.)

```
systemctl disable netfilter-persistent
```

```
systemctl mask netfilter-persistent
```

Do the same for nftables if /etc/nftables.conf holds the same snapshot. Then move the old files aside instead of deleting them, so you can always look back:

```
mkdir /root/fw-snapshot-old
```

```
mv /etc/iptables/rules.v4 /etc/iptables/rules.v6 /etc/nftables.conf /root/fw-snapshot-old/
```

Before you do this, be sure the server is protected somewhere else. On this server, the platform firewall in front of the VM already allowed web traffic only from Cloudflare. If the guest firewall is your only firewall, do not remove it — enable UFW properly instead and make it the single owner.

Also leave other tools' tables alone. cPanel's cPHulk had its own inet filter table in nftables; it was not part of the snapshot and stayed untouched.

Step 5: Verify with a real reboot.

This step is where most tutorials stop and where mine starts. A firewall fix that has not survived a reboot is not a fix. Agree a window with the owner, reboot, and then run:

```
ufw status
```

Expected: Status: inactive.

```
nft list ruleset | grep -c ufw
```

Expected: 0.

```
iptables -S | grep -- '-P'
```

Expected: every policy is ACCEPT.

```
journalctl -b | grep -Ei 'netfilter-persistent|nftables|ufw'
```

Expected: nothing.

On this server all four came back clean after the reboot, and the sites, mail on port 993 and cPanel services returned without anyone touching UFW. If ufw status still says active after this, something else is creating the ufw-user-input chain — find it before you change anything else. This result is not a bug. It is the verification.

The Checklist I Run Before Touching UFW

1. **Is UFW really enabled?** grep ENABLED /etc/ufw/ufw.conf
2. **Is a saved rule set restored at boot?** systemctl is-enabled netfilter-persistent
3. **Does the saved set contain UFW chains?** grep -c ufw /etc/iptables/rules.v4
4. **Is nftables loading its own copy?** systemctl is-enabled nftables
5. **Who else writes rules here?** nft list tables
6. **What does the kernel hold right now?** iptables -S | grep -- '-P'
7. **Will it survive a reboot?** Only a real reboot answers this one.

And one rule I now tell every team: **never run netfilter-persistent save while UFW is up.** You are freezing UFW's state into a second system that does not know UFW exists.

The Two Mail Problems Found on the Same Day

Once the firewall was clean, I checked the security alerts on the same server, because a server you have just rebooted should be able to tell you when something is wrong. It could not. Two small things were broken, and both are worth a line here.

- The malware monitor was failing to start because inotifywait was missing. Installing inotify-tools brought it back.
- Its alert mail was never delivered. Its fallback called sendmail -t with the address as an argument and no To: header. Exim with -t reads recipients from the headers, found none, and dropped the message — while the tool logged "sent". Installing a real mail binary (bsd-mailx) fixed the sending. Then the remote gateway rejected it, because the server's hostname had no SPF record and the parent domain's DMARC policy was strict. Adding one SPF record for the hostname was the only change between fail and pass.

Absence of alerts is not health. Send yourself a test alert after every fix like this.

What I Tell Every Customer Now

When someone says "UFW keeps turning itself on", I ask one question:

"Is netfilter-persistent enabled, and when was /etc/iptables/rules.v4 last written?"

If the answer is "not installed", UFW really is the one to look at. If the answer is "enabled, and the file is older than anyone remembers", we're not done yet.

I write about the questions I ask in real customer rooms — the ones that change architectures and surface the problems no one else noticed. If you found this useful, drop your email below and I'll send the next one when it's ready.

FAQs

Why does UFW show active after reboot when it is disabled?

ufw status checks whether the ufw-user-input chain exists in the kernel; it does not read ENABLED in /etc/ufw/ufw.conf. If netfilter-persistent restores an old snapshot that contains UFW chains, ufw status reports active even though UFW itself is disabled.

Why doesn't masking ufw.service stop the firewall from coming back?

Because UFW is not the service loading the rules. netfilter-persistent reads /etc/iptables/rules.v4 at boot and restores whatever was saved there, including UFW's chains and a DROP policy. Masking UFW does nothing to that service.

How do I check if netfilter-persistent is loading old rules?

Run systemctl is-enabled netfilter-persistent, then grep -c ufw /etc/iptables/rules.v4. If the service is enabled and the file contains UFW chains, an old UFW snapshot is being restored at every boot. Compare the file date with the install date in /var/log/dpkg.log.

Is it safe to disable netfilter-persistent?

Only if the server is protected by another firewall, or you then enable UFW properly as the single owner of the rules. Move the saved files aside instead of deleting them, and leave other tools' tables, such as cPHulk's, untouched.

How do I prove the fix worked?

Reboot in an agreed window. Afterwards ufw status should say inactive, nft list ruleset | grep -c ufw should return 0, all iptables policies should be ACCEPT, and the boot journal should show no entries for netfilter-persistent, nftables or ufw.

Read online: https://quabin.com/ufw-active-after-reboot-netfilter-persistent/