https://farid.ps/articles/rtl9210_usb_to_nvme_bridge/nl.html

Linux en de Realtek RTL9210 USB-naar-NVMe-brug

> **Samenvatting:**
> • **Symptomen:** Herhaalde USB-resets, I/O-fouten of verdwijnende schijven onder Linux.
> • **Betroffen:** Realtek RTL9210 (bevestigd) en RTL9220 (mogelijk).
> • **Oorzaak:** Terugval naar interne ROM (**f0.01**) na een checksumfout.
> • **Impact:** Permanente instabiliteit, geen Linux-reflash-tools beschikbaar.
> • **Oplossing:** Alleen OEM Windows-hulpprogramma's kunnen de firmware herstellen – Realtek blokkeert open-source alternatieven.

Voorwoord

In 2025 zou het volkomen redelijk moeten zijn om een Raspberry Pi te starten vanaf een SSD die via USB is verbonden. Door de eigenaardigheden van Realtek's firmware is dit redelijke doel echter een avontuur geworden. Na maanden van onverklaarbare instabiliteit – willekeurige resets, verdwijnende schijven, beschadigde bestandssystemen – heeft de auteur alle gebruikelijke oplossingen uitgeput: nieuwe kabels, gevoede hubs, kernelupdates, USB-aanpassingen en firmware-finetuning. De doorbraak kwam pas toen ChatGPT een vreemde vraag laat op de avond beantwoordde: "Is het mogelijk dat de USB-naar-NVMe-brug is teruggevallen op een oude firmware?"

Inleiding

Als jouw Realtek-gebaseerde NVMe-behuizing plotseling instabiel wordt na weken van vlekkeloze werking – herhaalde USB-resets, I/O-fouten of verdwijnende schijven – ben je niet alleen. Dit patroon is opgedoken bij verschillende merken, van naamloze eenheden tot bekende OEM's zoals Sabrent en Orico. De gemeenschappelijke deler: **Realtek's RTL9210 en mogelijk RTL9220** USB-naar-NVMe-brugchips.

In het begin werkt alles. Dan, schijnbaar zonder reden, begint het apparaat te ontkoppelen onder belasting of bij langdurig gebruik, vooral op Linux- of Raspberry Pi-systemen. De echte oorzaak is niet de SSD of de voeding – het is de firmware-controller zelf die stilletjes terugvalt op zijn **in ROM ingebedde back-upcode**, een versie die Realtek intern nog steeds levert als **f0.01**.

Het verborgen mechanisme – Firmware-terugval door ontwerp

Realtek's brugchips slaan hun operationele firmware en configuratiegegevens op in een externe SPI-flash. Bij het opstarten controleert de controller een eenvoudige checksum. Als die checksum niet overeenkomt, weigert het de externe firmware te laden en start het in plaats daarvan vanuit zijn interne ROM.

Deze back-upfirmware is verouderd en defect. Het mist verschillende USB-stabiliteitscorrecties en verbeteringen in linkstatusbeheer die aanwezig zijn in latere revisies, wat leidt tot de klassieke reeks die elke Linux-gebruiker herkent:

> usb 3-2: reset van high-speed USB-apparaat nummer 2 met xhci-hcd
> usb 3-2: apparaatdescriptor lezen/64, fout -71
> EXT4-fs waarschuwing (apparaat sda2): I/O-fout bij schrijven naar inode ...

De checksum kan ongeldig worden wanneer configuratiegegevens worden herschreven – bijvoorbeeld wanneer de brug zijn energiebeheer- of UAS-instellingen bijwerkt – en het apparaat tijdens het schrijven stroom verliest. De volgende opstart detecteert een beschadigde checksum en valt permanent terug naar de ROM-firmware.

Op dat moment gedraagt jouw "high-performance NVMe-behuizing" zich precies als de goedkoopste naamloze behuizing, omdat het intern nu dezelfde defecte bascode uitvoert die in het silicium is gebrand.

Verificatie van het probleem

Je kunt deze toestand eenvoudig bevestigen onder Linux:

```
lsusb -v | grep -A2 Realtek
```

Een gezonde Realtek-brug rapporteert een firmware-revisie (**bcdDevice**) boven 1.00. Een teruggevallen brug toont:

```
bcdDevice f0.01
```

Deze **f0.01**-handtekening betekent dat de controller opstart vanuit ROM – en geen enkele hoeveelheid ontkoppelen, herformatteren of kernelafstemming zal het repareren.

Dit terugvalmechanisme is **bevestigd op de RTL9210**. De **RTL9220** lijkt dezelfde ontwerparchitectuur en firmware-indeling te delen, dus het kan identiek gedrag vertonen, maar dit blijft **waarschijnlijk in plaats van bewezen**.

Waarom je het niet zelf kunt repareren

In principe is de oplossing triviaal: flash de juiste firmware opnieuw naar SPI. In de praktijk maakt Realtek dit onmogelijk.

Het bedrijf biedt een gesloten-bron Windows-updater aan OEM's en integrators. Linux-gebruikers krijgen niets aangeboden. Gemeenschapsontwikkelaars hebben reverse-engineered compatibele flash-hulpprogramma's (**rtsupdater**, **rtl9210fw**, **rtsupdater-cli**) die

volledige firmwareherstel vanuit Linux-systemen mogelijk maakten – totdat Realtek **DMCA-verwijderingsverzoeken** uitgaf om ze te onderdrukken.

Er is geen plausibele reden voor intellectueel eigendom om dergelijke hulpprogramma's te blokkeren: ze onthullen geen microcode, maar organiseren alleen de updatesequentie via USB. De verwijderingen van Realtek gingen niet over bescherming. Ze waren ideologisch.

De prijs van een ideologie

Dit gaat niet over open-source idealisme. Het gaat over de **ideologische vijandigheid van een hardwareleverancier tegenover open systemen** die apparaten kapotmaakt die worden verkocht als *Linux-compatibel*.

Realtek's weerstand tegen documentatie en open tools bestaat al twee decennia, van Wi-Fi, Ethernet, audio tot nu opslagcontrollers. Deze insulariteit kan onopgemerkt blijven in een wereld met alleen Windows, maar wordt giftig wanneer dezelfde chips worden geïntegreerd in multi-platformproducten zoals de **Sabrent EC-SNVE**, die openlijk het Linux-logo op zijn verpakking toont.

Door Linux-flash-hulpprogramma's te verbieden en gemeenschapsonderhoud te blokkeren, heeft Realtek effectief **zelfreparatie gecriminaliseerd**. De gevolgen verspreiden zich naar buiten:

- Linux-gebruikers zien "ondersteunde" hardware degraderen tot instabiliteit.
- OEM's zoals Sabrent en Orico worden geconfronteerd met onnodige RMA- en garantiekosten.
- Realtek's langdurige reputatie voor slechte Linux-compatibiliteit wordt opnieuw versterkt.

Uiteindelijk is het niet open-source dat Realtek-apparaten kapotmaakt – het is **Realtek's vijandigheid tegenover open-source** dat ze kapotmaakt.

Een rationele weg vooruit

De oplossing vereist geen ideologische verschuiving, alleen pragmatisme. Realtek zou kunnen:

1. Een door de leverancier ondertekend opdrachtregel-updateprogramma voor Linux uitbrengen (geen openbaarmaking van broncode nodig).
2. Het checksum-algoritme publiceren zodat integrators flash-images veilig kunnen valideren.
3. Een DFU-achtige modus aannemen die updates accepteert via USB-mass storage, onafhankelijk van het besturingssysteem.

Elk van deze zou garantiekosten voorkomen, OEM-relaties beschermen en het vertrouwen in Realtek's brugchips herstellen onder professionele Linux-gebruikers – van werkstationbouwers tot Raspberry Pi-ontwikkelaars.

Wat je kunt doen

Als je vermoedt dat jouw behuizing is teruggevallen naar ROM-firmware:

- Controleer met **lsusb -v | grep bcdDevice**.
- Als het **f0.01** toont, meld het probleem aan je OEM.
- Voeg het **dmesg**-fragment toe en wijs op dit gedocumenteerde terugvalmechanisme.
- Vraag je leverancier om het probleem te escaleren naar Realtek, met vermelding van de behoefte aan een Linux-compatibel updateprogramma.

Realtek's firmwarebeleid hindert niet alleen enthousiastelingen; het creëert tastbare financiële verliezen voor hun eigen ecosysteem. Hoe eerder deze realiteit binnen het bedrijf wordt erkend, hoe eerder Linux-gebruikers en OEM-partners kunnen stoppen met tijdverspilling aan vermijdbare RMA-cycli.

Reacties van fabrikanten

Zowel Realtek als Sabrent zijn uitgenodigd om verklaringen af te geven over het bovenbeschreven firmware-terugvalprobleem. Hun antwoorden – indien ontvangen – worden hier toegevoegd.

Bijlage – Identificatie van getroffen apparaten

Controller	Leverancier-ID	Product-ID	Opmerkingen	Status
RTL9210	0x0bda	0x9210	USB 3.1 Gen 2 10 Gb/s brug	**Bevestigd** terugvalgedrag
RTL9220	0x0bda	0x9220	USB 3.2 Gen 2×2 20 Gb/s brug	**Mogelijk**, vergelijkbare architectuur

Firmware-terugvalhandtekening: **bcdDevice f0.01**
Bekende stabiele revisies: **1.23** – **1.31**