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

Linux und die Realtek RTL9210 USB-zu-NVMe-Brücke

> **Zusammenfassung:**
> • **Symptome:** Wiederholte USB-Resets, I/O-Fehler oder verschwindende Laufwerke unter Linux.
> • **Betroffen:** Realtek RTL9210 (bestätigt) und RTL9220 (möglicherweise).
> • **Ursache:** Rückfall auf interne ROM (**f0.01**) nach Prüfsummenfehler.
> • **Auswirkungen:** Dauerhafte Instabilität, keine Linux-Reflash-Tools verfügbar.
> • **Lösung:** Nur OEM-Windows-Dienstprogramme können die Firmware wiederherstellen – Realtek blockiert Open-Source-Alternativen.

Vorwort

Im Jahr 2025 sollte es völlig normal sein, einen Raspberry Pi von einer über USB angeschlossenen SSD zu starten. Doch dank der Eigenheiten der Realtek-Firmware ist dieses vernünftige Ziel zu einem Abenteuer geworden. Nach Monaten ungeklärter Instabilität – zufällige Resets, verschwindende Laufwerke, beschädigte Dateisysteme – hat der Autor alle üblichen Lösungen ausgeschöpft: neue Kabel, stromversorgte Hubs, Kernel-Updates, USB-Anpassungen und Firmware-Optimierungen. Der Durchbruch kam erst, als ChatGPT auf eine seltsame nächtliche Frage antwortete: „Ist es möglich, dass die USB-zu-NVMe-Brücke auf eine alte Firmware zurückgefallen ist?“

Einführung

Wenn Ihr Realtek-basierter NVMe-Gehäuse plötzlich nach Wochen einwandfreien Betriebs instabil wird – wiederholte USB-Resets, I/O-Fehler oder verschwindende Laufwerke – sind Sie nicht allein. Das Muster ist bei mehreren Marken aufgetreten, von No-Name-Geräten bis hin zu bekannten OEMs wie Sabrent und Orico. Der gemeinsame Nenner: **Realteks RTL9210 und möglicherweise RTL9220** USB-zu-NVMe-Brückenchips.

Anfangs funktioniert alles. Dann, scheinbar ohne Grund, beginnt das Gerät unter Last oder bei längerem Gebrauch, insbesondere auf Linux- oder Raspberry Pi-Systemen, die Verbindung zu verlieren. Die wahre Ursache ist weder die SSD noch die Stromversorgung – es ist der Firmware-Controller selbst, der leise auf seinen **in der ROM eingebetteten Rückfallcode** zurückgreift, eine Version, die Realtek intern immer noch als **f0.01** ausliefert.

Der verborgene Mechanismus – Firmware-Rückfall durch Design

Die Brückenchips von Realtek speichern ihre Betriebsfirmware und Konfigurationsdaten in einem externen SPI-Flash. Beim Einschalten überprüft der Controller eine einfache Prüfsumme. Wenn diese Prüfsumme nicht übereinstimmt, verweigert er das Laden der externen Firmware und startet stattdessen aus seiner internen ROM.

Diese Rückfall-Firmware ist veraltet und fehlerhaft. Sie enthält mehrere fehlende USB-Stabilitätskorrekturen und Verbesserungen der Link-Zustandsverwaltung, die in späteren Revisionen vorhanden sind, was zur klassischen Sequenz führt, die jeder Linux-Benutzer erkennt:

> usb 3-2: Reset des Hochgeschwindigkeits-USB-Geräts Nummer 2 mit xhci-hcd
> usb 3-2: Gerätebeschreibung lesen/64, Fehler -71
> EXT4-fs Warnung (Gerät sda2): I/O-Fehler beim Schreiben in Inode ...

Die Prüfsumme kann ungültig werden, wenn Konfigurationsdaten überschrieben werden – zum Beispiel, wenn die Brücke ihre Energieverwaltungs- oder UAS-Einstellungen aktualisiert – und das Gerät während des Schreibens die Stromversorgung verliert. Beim nächsten Start erkennt es eine beschädigte Prüfsumme und fällt dauerhaft auf die ROM-Firmware zurück.

An diesem Punkt verhält sich Ihr „hochleistungsfähiges NVMe-Gehäuse“ genau wie das billigste No-Name-Gehäuse, da es intern nun denselben fehlerhaften Basiskode ausführt, der in das Silizium eingebrannt ist.

Überprüfung des Problems

Sie können diesen Zustand unter Linux einfach bestätigen:

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

Eine gesunde Realtek-Brücke meldet eine Firmware-Revision (**bcdDevice**) über 1.00. Eine zurückgefallene zeigt:

```
bcdDevice f0.01
```

Diese **f0.01**-Signatur bedeutet, dass der Controller aus der ROM startet – und kein Entstecken, Neuformatieren oder Kernel-Tuning wird dies beheben.

Dieser Rückfallmechanismus wurde **für den RTL9210 bestätigt**. Der **RTL9220** scheint die gleiche Designarchitektur und Firmware-Layout zu teilen, sodass er wahrscheinlich identisches Verhalten zeigt, aber dies bleibt **wahrscheinlich, nicht bewiesen**.

Warum Sie es nicht selbst reparieren können

Im Prinzip ist die Lösung einfach: Die richtige Firmware auf den SPI flashen. In der Praxis macht Realtek dies unmöglich.

Das Unternehmen stellt OEMs und Integratoren ein Closed-Source-Windows-Update-Tool zur Verfügung. Linux-Benutzern wird nichts angeboten. Community-Entwickler haben kompatible Flash-Tools (**rtsupdater**, **rtl9210fw**, **rtsupdater-cli**) reverse-engineered, die eine vollständige Firmware-Wiederherstellung von Linux-Systemen aus ermöglichten – bis Realtek **DMCA-Abbauanforderungen** ausstellte, um sie zu unterdrücken.

Es gibt keinen plausiblen Grund für den Schutz des geistigen Eigentums, um solche Tools zu blockieren: Sie legen keinen Mikrocode offen, sondern orchestrieren lediglich die Update-Sequenz über USB. Die Abbauanforderungen von Realtek waren nicht zum Schutz gedacht. Sie waren ideologisch.

Der Preis einer Ideologie

Hier geht es nicht um Open-Source-Idealismus. Es geht um die **ideologische Feindseligkeit eines Hardware-Herstellers gegenüber offenen Systemen**, die Geräte zerstört, die als *Linux-kompatibel* vermarktet werden.

Realteks Widerstand gegen Dokumentation und offene Tools besteht seit zwei Jahrzehnten und erstreckt sich über WLAN, Ethernet, Audio und jetzt Speichercontroller. Diese Abschottung mag in einer reinen Windows-Welt unbemerkt bleiben, wird jedoch toxisch, wenn dieselben Chips in Multi-Plattform-Produkte wie das **Sabrent EC-SNVE** integriert werden, das offen das Linux-Logo auf seiner Verpackung zeigt.

Durch das Verbot von Linux-Flash-Tools und die Blockierung der Community-Wartung hat Realtek **Selbstreparatur effektiv kriminalisiert**. Die Folgen breiten sich aus:

- Linux-Benutzer sehen, wie „unterstützte“ Hardware in Instabilität abgleitet.
- OEMs wie Sabrent und Orico stehen vor unnötigen RMA- und Garantiekosten.
- Realteks langjähriger Ruf für schlechte Linux-Kompatibilität wird erneut verstärkt.

Am Ende ist es nicht Open Source, das Realteks Geräte zerstört – es ist **Realteks Feindseligkeit gegenüber Open Source**, die sie zerstört.

Ein rationaler Weg nach vorn

Die Lösung erfordert keinen ideologischen Wandel, sondern nur Pragmatismus. Realtek könnte:

1. Ein vom Hersteller signiertes Kommandozeilen-Update-Tool für Linux veröffentlichen (keine Offenlegung des Quellcodes erforderlich).
2. Den Prüfsummenalgorithmus veröffentlichen, damit Integratoren Flash-Images sicher validieren können.
3. Einen DFU-ähnlichen Modus einführen, der Updates über USB-Massenspeicher akzeptiert, unabhängig vom Betriebssystem.

Jeder dieser Schritte würde Garantiekosten verhindern, OEM-Beziehungen schützen und das Vertrauen in Realteks Brückenchips unter professionellen Linux-Benutzern – von

Workstation-Bauern bis hin zu Raspberry Pi-Entwicklern – wiederherstellen.

Was Sie tun können

Wenn Sie vermuten, dass Ihr Gehäuse auf die ROM-Firmware zurückgefallen ist:

- Überprüfen Sie mit **lsusb -v | grep bcdDevice**.
- Wenn es **f0.01** anzeigt, melden Sie das Problem an Ihren OEM.
- Fügen Sie den **dmesg**-Auszug hinzu und verweisen Sie auf diesen dokumentierten Rückfallmechanismus.
- Bitten Sie Ihren Anbieter, das Problem an Realtek zu eskalieren, und verweisen Sie auf die Notwendigkeit eines Linux-kompatiblen Update-Tools.

Realteks Firmware-Politik stört nicht nur Enthusiasten; sie verursacht greifbare finanzielle Verluste für ihr eigenes Ökosystem. Je früher diese Realität im Unternehmen anerkannt wird, desto schneller können sowohl Linux-Benutzer als auch OEM-Partner aufhören, Zeit mit vermeidbaren RMA-Zyklen zu verschwenden.

Antworten der Hersteller

Sowohl Realtek als auch Sabrent wurden eingeladen, Stellungnahmen zu dem oben beschriebenen Firmware-Rückfallproblem abzugeben. Ihre Antworten – falls erhalten – werden hier angehängt.

Anhang – Identifizierung betroffener Geräte

Controller	Hersteller-ID	Produkt-ID	Anmerkungen	Status
RTL9210	0x0bda	0x9210	USB 3.1 Gen 2 10 Gb/s Brücke	**Bestätigtes** Rückfallverhalten
RTL9220	0x0bda	0x9220	USB 3.2 Gen 2×2 20 Gb/s Brücke	**Möglicherweise**, ähnliche Architektur

Firmware-Rückfallsignatur: **bcdDevice f0.01**
Bekannte stabile Revisionen: **1.23** – **1.31**