

cs-stream

Handbuch

Verschlüsselter TCP-Stream-Transport für ZFS-Replikation und Ordner-Sync - ersetzt ssh, mbuffer, netcat und pv

Version 2.1.0 · 13. Oktober 2026

Inhalt

1. Überblick	3
2. Benutzung	3
2.1 Optionen	3
2.2 Tunnel-Modus (rclone über SFTP)	3
3. Verschlüsselung und Sicherheit	4
3.1 Security-Audit v2.1.0	4
4. Installation	4
4.1 In napp-it CS	5
4.2 Eigenständig	5
5. Bauen	5
6. Gewährleistung und Lizenz	5

1. Überblick

cs-stream ist ein verschlüsselter TCP-Stream-Transport. In ZFS-send/receive-Pipelines ersetzt es ssh, mbuffer, netcat (nc) und pv durch ein einziges kleines Tool: eine verschlüsselte Pipe mit Read-ahead-Puffer, Ratenbegrenzung und Fortschrittsanzeige.

Es gehört zu napp-it CS und wird dort von den Jobs *job-replicate* (verschlüsselte ZFS-Replikation) und *job-filesync* (rclone-über-SFTP-Ordner-Sync zwischen Membern) genutzt. Es hat keine Abhängigkeit von einer napp-it-Installation und lässt sich eigenständig als allgemeiner verschlüsselter Stream-Transport verwenden.

Bis v1.3.3 hieß das Tool *zstream*. Ab v2.0.0 heißt es cs-stream (um eine Verwechslung mit dem gleichnamigen, unabhängigen OpenZFS-Tool zu vermeiden); Wire-Protokoll und Kommandozeile sind unverändert.

Modus	Zweck
<code>listen</code> / <code>send</code>	stdin nach stdout wie ein verschlüsseltes netcat - für <code>zfs send</code> / <code>zfs receive</code>
<code>tunnel-listen</code> / <code>tunnel-send</code>	einen lokalen TCP-Dienst (zum Beispiel den SFTP-Server von rclone) in einen verschlüsselten Tunnel packen

2. Benutzung

Der Empfänger wird immer zuerst gestartet: Er öffnet den Port und wartet (bis zum Accept-Timeout von 120 s). Ein Sender, bei dem nichts lauscht, kann schlicht keine Verbindung aufbauen.

```
# Empfänger (zuerst starten) - --allow-ip ist ab v2.1.0 PFLICHT
cs-stream listen 9000 SESSIONKEY --allow-ip=192.168.1.20 | zfs receive -F tank/backup

# Sender
zfs send tank@snap | cs-stream send 192.168.1.10 9000 SESSIONKEY

# Mit Optionen (Ratenbegrenzung, Fortschritt, Log)
zfs send tank@snap | cs-stream send 192.168.1.10 9000 SESSIONKEY --rate=100m --progress
--log=/tmp/cs-stream.log
```

2.1 Optionen

Option	Standard	Beschreibung
<code>--buf=SIZE</code>	128m	Read-ahead-Puffer (0 = aus), Einheiten k/m/g
<code>--rate=SPEED</code>	aus	Durchsatzbegrenzung pro Sekunde, Einheiten k/m/g (nicht im Tunnel-Modus)
<code>--progress</code>	aus	Live-Fortschritt auf stderr (nicht im Tunnel-Modus)
<code>--log=FILE</code>	aus	Zusammenfassung der Übertragung an eine Datei anhängen
<code>--bind=IP</code>	0.0.0.0	Listener an eine IP-Adresse binden
<code>--allow-ip=IP</code>	keine	Pflicht bei <code>listen</code> und <code>tunnel-listen</code> : nur diese Quell-IP darf verbinden

2.2 Tunnel-Modus (rclone über SFTP)

Der Tunnel umschließt einen lokalen Kindprozess. `--local=ADDR` ist die lokale Adresse, an die der umschlossene Prozess bindet (`tunnel-listen`) bzw. zu der er verbindet (`tunnel-send`); `--allow-ip=IP` ist die einzige Quell-IP, von der `tunnel-listen` Sitzungen annimmt. Beides ist bei `tunnel-listen` Pflicht.

```
# Empfänger (zuerst starten)
cs-stream tunnel-listen 9000 KEY --local=127.0.0.1:2222 --allow-ip=192.168.1.10 -- rclone serve sftp
/dst --addr 127.0.0.1:2222
```

```
# Sender
cs-stream tunnel-send HOST 9000 KEY --local=127.0.0.1:2222 -- rclone sync /src :sftp: --sftp-
host=127.0.0.1 --sftp-port=2222
```

3. Verschlüsselung und Sicherheit

- AES-256-GCM, authentifizierte Verschlüsselung (Go-Standardbibliothek, mit AES-NI beschleunigt).
- Ein Sitzungsschlüssel pro Übertragung; in napp-it CS wird er von job-replicate.pl und job-filesync.pl erzeugt.
- Jeder 64-KB-Block wird einzeln mit zufälliger Nonce verschlüsselt; das Authentifizierungs-Tag erkennt jede Manipulation und jeden Fehler.
- `--allow-ip` beschränkt, welche Quell-IP überhaupt einen Handshake versuchen darf.
- Accept-Timeout 120 s (der Listener wartet nie endlos); Lese-Timeout 5 Minuten (erkennt einen toten Sender oder einen Netzwerkfehler).
- `--bind=IP` beschränkt den Listener auf ein Interface.

3.1 Security-Audit v2.1.0

Ein vollständiges Source-Audit fand und behob drei Punkte:

1. `listen` und `tunnel-listen` nahmen Verbindungen von jeder Quelle an. `listen` akzeptiert pro Prozesslebenszeit genau eine Verbindung, ein fremder Host konnte also diesen einen Platz belegen und den echten Sender blockieren (nur ein Verfügbarkeitsproblem - AES-256-GCM verhinderte ohne Schlüssel schon bisher jedes Lesen und Schreiben). Behebung: `--allow-ip` ist Pflicht; falsche IPs werden abgewiesen und geschlossen, der Listener wartet weiter, und die Accept-Frist läuft weiter, sodass eine Flut falscher Verbindungen das Fenster nicht verlängern kann. Das ist eine inkompatible Änderung gegenüber v2.0.0.
2. `tunnel-listen` hatte keine Obergrenze für gleichzeitige Sitzungen: jetzt höchstens 64 (`maxTunnelSessions`), atomar gezählt, als zusätzliche Absicherung selbst für die erlaubte IP.
3. `--rate` und `--progress` wurden im Tunnel-Modus stillschweigend ignoriert; jetzt wird einmalig eine Warnung geloggt.

Drei Live-Tests liefen auf Windows, Linux, FreeBSD, macOS und OmniOS/illumos und wurden alle bestanden: (A) `listen/tunnel-listen` starten ohne `--allow-ip` nicht, (B) eine Verbindung von einer falschen IP wird abgelehnt und geloggt, während der Listener weiterläuft, (C) eine Verbindung von der erlaubten IP funktioniert durchgehend und wird korrekt entschlüsselt.

Geprüft und nicht als Handlungsbedarf gewertet: die zufälligen 96-Bit-Nonces unter einem festen Schlüssel (ein Kollisionsrisiko wird erst bei etwa 2^{32} Blöcken relevant, also rund 256 TB unter einem Schlüssel - ein Schlüssel gilt nur für eine Übertragung) und die Schlüsselableitung per einfachem SHA-256 (das Schlüsselmaterial ist immer ein frisch erzeugter String mit hoher Entropie aus dem Perl-Aufrufer, nie ein vom Benutzer gewähltes Passwort).

4. Installation

Plattform	Datei
macOS x86_64 / ARM64	<code>cs-stream-darwin-amd64</code> , <code>cs-stream-darwin-arm64</code>
Linux x86_64 / ARM64	<code>cs-stream-linux-amd64</code> , <code>cs-stream-linux-arm64</code>
Windows x86_64	<code>cs-stream-windows-amd64.exe</code>
FreeBSD x86_64	<code>cs-stream-freebsd-amd64</code>
illumos x86_64	<code>cs-stream-illumos-amd64</code>
OmniOS / Solaris x86_64	<code>cs-stream-solaris-amd64</code>

Die fertigen Binaries hängen an den GitHub-Releases.

4.1 In napp-it CS

Das Binary liegt in `data/cs_server/tools/cs-stream/<os>.amd64/cs-stream` (Linux, BSD, OmniOS, Solaris) bzw. `data\cs_server\tools\cs-stream\mswin.amd64\cs-stream.exe` (Windows). napp-it CS verteilt und pflegt es auf den Mitgliedern über das Menü **About > Download cs-tools**; eine manuelle Ablage ist nur nötig, um den Admin-Host selbst mit einer neuen Version zu versorgen.

4.2 Eigenständig

cs-stream ist ein in sich geschlossenes Go-Binary aus einer einzigen Datei, ganz ohne Bezug zu napp-it. Es braucht keine Konfigurationsdatei, keinen Registry-Eintrag und keine Umgebung - nur den gleichen Sitzungsschlüssel auf beiden Seiten und, ab v2.1.0, `--allow-ip` auf der Empfängerseite.

```
git clone https://github.com/guenther-alka/cs-stream.git
cd cs-stream
go build -o cs-stream .

# Empfänger
cs-stream listen 9000 SESSIONKEY --allow-ip=192.168.1.20 > out.bin
# Sender
cs-stream send 192.168.1.10 9000 SESSIONKEY < in.bin
```

5. Bauen

Go 1.21 oder neuer, keine externen Abhängigkeiten.

```
G00S=linux GOARCH=amd64 go build -o cs-stream-linux-amd64 .
G00S=windows GOARCH=amd64 go build -o cs-stream-windows-amd64.exe .
G00S=freebsd GOARCH=amd64 go build -o cs-stream-freebsd-amd64 .
G00S=solaris GOARCH=amd64 go build -o cs-stream-solaris-amd64 .
```

6. Gewährleistung und Lizenz

cs-stream ist Open Source. Es darf kostenlos genutzt, analysiert und verändert werden, "wie es ist", und die Verantwortung für die Nutzung liegt allein beim Anwender. Dieser Hinweis fasst die Lizenzbedingungen in einfachen Worten zusammen und ersetzt sie nicht.

BSD 2-Clause License - Copyright (c) 2026 Guenther Alka / napp-it.org. Die Weitergabe in Quell- oder Binärform erfordert die Beibehaltung des Copyright-Hinweises; siehe Datei LICENSE.