# RPC-with-TLS

## Progress and challenges

**Chuck Lever** <chuck.lever@oracle.com>

# Status of draft-ietf-nfsv4-rpc-tls

- The document is in the RFC Editor queue awaiting a missing normative reference (MISSREF)
  - draft-ietf-kitten-tls-channel-bindings-for-tls13 is waiting for AD write-up
- The following normative references have been received by the RFC Editor
  - draft-ietf-tls-dtls-connection-id
  - draft-ietf-tls-dtls13

# Status of draft-ietf-nfsv4-rpc-tls

- Matters to be handled during final author approval (AUTH48):
  - The included ASN.1 module does not compile, a suggested replacement is available
  - “RPC-over-TLS” could be renamed “RPC-with-TLS”
- Proposed changes can be mocked up and reviewed in the document’s github repo

# Implementations of RPC-with-TLS

- FreeBSD client and server
- Java-based client and server (DESY)
- Hammerspace server
- Linux client prototype (and eventually server too)
- Cloud data center NFS server implementations

# Community Testing

- First 100% virtual bake-a-thon held in February 2021
  - Two implementations of RPC-with-TLS showed up
  - Testing continues after the event amongst several prototypes
- No “cthon04”-like suite of tests yet
- Discussions beginning on how to assure product quality

# Linux Kernel Implementation Challenges

## Code duplication concerns

- Two possible handshake architectures
  - Traditional upcall mechanism would utilize existing user space TLS implementation
  - In-kernel handshake would duplicate user space but could be independent of user space components, easier container support, possibly more scalable
- QUIC
  - How much TLS handshake logic can be shared with in-kernel QUIC?

# Linux Kernel Implementation Challenges

## Maintainability concerns

- Implementing new TLS versions
- Curating the set of available crypto algorithms
- Responding to CVEs
- Managing the usual churn of maturing kernel infrastructure
- Executing CI and quality assurance plans

# Linux Kernel Implementation Challenges

## Performance concerns

- TLS handshake scalability
  - Handshake crypto is CPU intensive
  - Clients should have a low connection rate, but servers can experience many handshakes at once (*e.g.* after an unplanned server reboot)
- TLS record protocol
  - Storage protocols benefit from encryption offload
  - Hardware devices lag behind latest crypto and other features

# Additional Proposed Standards Actions

- Clarify NFS operation when using RPC-with-TLS
  - Using TLS peer authentication for EXCHANGE_ID and friends
  - Advertising TLS security level requirements via SECINFO and MNT
  - Best security policies for NFS clients and servers when using Transport Layer Security
    - Semantics of mount options and share security settings
    - Address NFS user authentication as best we can

# Supplemental Material

# Bibliography


- https://datatracker.ietf.org/doc/draft-ietf-nfsv4-rpc-tls/
- https://datatracker.ietf.org/doc/draft-ietf-kitten-tls-channel-bindings-for-tls13
- https://datatracker.ietf.org/doc/draft-ietf-tls-dtls-connection-id
- https://datatracker.ietf.org/doc/draft-ietf-tls-dtls13