RPC-with-TLS

Progress report

Chuck Lever <chuck.lever@oracle.com>
Publication Status

• The document is in the RFC Editor queue awaiting a publication of a normative reference (REF)
  • draft-ietf-kitten-tls-channel-bindings-for-tls13 is now in AUTH48
• The following normative references have now been published:
  • draft-ietf-tls-dtls-connection-id as RFC 9146
  • draft-ietf-tls-dtls13 as RFC 9147
Publication Status

• 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” has been renamed “RPC-with-TLS”
• Proposed changes have been mocked up in the document’s github repo:
  • https://github.com/chucklever/i-d-rpc-tls
Implementation Status

- FreeBSD client and server
- Java-based client and server (DESY)
- Hammerspace server
- Linux client prototype
- nginx module
Community Testing

• Virtual bake-a-thons held in October 2021 and April 2022
  • FreeBSD, DESY, Linux, and nginx implementations all present and interoperating at most recent event
• Discussions continue on how to assure product quality and how to make administrative interfaces similar among the implementations
Linux NFS Client Implementation

• Prototype upcall-style TLS handshake mechanism
  • Kernel passes connected socket descriptor to a user space agent, which uses a standard TLS library to perform the handshake

• `xprtsec= none | tls | mtls` mount option

• Currently supports server TLS authentication
  • Implementation of client TLS authentication is in progress

• Available via [kernel.org](http://kernel.org) and GitHub
Additional Proposed Standards Actions
NFS operation when using RPC-with-TLS

• Use TLS peer authentication for EXCHANGE_ID and friends

• Best security policies for NFS clients and servers when using Transport Layer Security

• Still no mechanism for servers to indicate that TLS is required when clients want to use AUTH_SYS, but not clear one is necessary

• Proposal: allow rpc-tls-pseudoflavors to expire; explore conventions for servers to use for this case
Supplemental Material
Linux Kernel Implementation

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, would work for NFSROOT

- QUIC and other transport security protocols
  - How much TLS handshake logic can be shared with in-kernel QUIC?
Bibliography

- https://datatracker.ietf.org/doc/draft-ietf-nfsv4-rpc-tls/
Linux Prototype Source Code

- Kernel component:

- User TLS handshake agent:
  - [https://github.com/oracle/ktls-utils](https://github.com/oracle/ktls-utils)

- nfs-utils with client TLS authentication mount options:
  - Coming soon