TLS handshake for Linux kernel consumers

A High-level Overview

Chuck Lever – October 2023

Acknowledgements

- Jamal Hadi Salim and the NetDev 0x17 program committee
- Jakub Kicinski and the netdev maintainers and community
- My friends and colleagues on kernel-tls-handshake@

Presenter’s Biography

- Nearly a quarter century working on the Linux NFS implementation
- Author or co-author of numerous NFS-related IETF RFCs
- Co-maintainer of NFSD (the Linux kernel NFS server)
    - Before that, extensive focus on NFS/RDMA
- But only peripheral computer security experience

In Scope

- The facility described in this presentation is already in mainline Linux
  - See commit 3b3009ea8abb (“net/handshake: Create a NETLINK service for handling handshake requests”) [4/23]
- Which kernel consumers want TLS and why (our use cases)
- Alternative approaches to providing TLS handshakes in-kernel
- Thoughts on the use of TPM, NIC offload, keyrings, and other technologies

Out Of Scope

- User space applications cannot directly see or use this new facility, since they already have access to TLS handshake mechanisms via libraries
- I’m not going to perform a demo today
- Still no user authentication with x.509 certificates
- Our handshake mechanism will never officially support TLS versions older than TLS v1.3

Our Initial In-Kernel Use Cases

- SunRPC with TLS
  - RFC 9289 Towards Remote Procedure Call Encryption by Default [9/22]
- NVMe on TCP with TLS
  - NVM Express TCP Transport Specification 1.0c [10/22]
- QUICv1
  - RFC 9000 QUIC: A UDP-based Multiplexed and Secure Transport [5/21]

Sidebar: RPC-with-TLS

- RPC already has GSSAPI, why does it need TLS too?
  - GSS Kerberos has heavyweight infrastructure requirements
  - TLS is now a commodity technology (web, email, etc)
    - GSS Kerberos encryption cannot easily be offloaded (key-per-user versus key-per-host)
    - TLS gets new encryption algorithms more quickly than Kerberos does
  - TLS encryption can be enabled with a single server-side certificate, which can enable better security for deployments that wish to continue using AUTH_SYS

The Benefits of kTLS

- Existing kTLS implements the TLS Record protocol in the kernel. Each endpoint looks like a regular network socket.
- Without much modification, kernel kTLS consumers can utilize either:
  - A software TLS implementation based on the kernel's crypto
  - A hardware TLS implementation provided in the NIC
- To initialize the session, first a handshake must optionally authenticate, negotiate a session key, and select encryption and MAC algorithms

Alternative Approaches

- Grow an in-kernel TLS handshake implementation
- Run a full user space library in a protected middle layer
- Pass open sockets to a user space library
  - accept(2)
  - call_usermodehelper
  - netlink

The Selected Approach

- A new netlink protocol was constructed for passing an open file descriptor to user space
- A new daemon was created that waits for these fds, passes them to a library (GnuTLS), then sets kTLS socket options with the negotiated results
- A kernel consumer can open a socket and probe for TLS support. Then:
  - The new handshake mechanism dups that socket and passes the dup'd fd up to the daemon
  - The kernel consumer sleeps while waiting for the handshake result

Netlink Protocol

- READY (kernel -> multicast group)
  - Indicates an in-kernel consumer wants a handshake
- ACCEPT (user space -> kernel)
  - Takes an MC group, and returns a socket descriptor and handshake parameters. Agent can then perform a TLS handshake on the socket.
- DONE (user space -> kernel)
  - The agent has either primed a socket for use with kTLS, or the handshake failed

Managing Authentication Material

- Certificates, PSKs, CA bundles, and private keys are typically stored in files
  - The ULP has to select and provide the material,
  - The handshake agent can have suitable default material,
  - The kernel or handshake agent can retrieve the material from a TPM, or
  - The ULP or kernel can copy the material into a long-lived keyring

Keyrings

- Although tlshd reads default authentication material from files, upper layer protocols can provide material in keys
    - tlshd checks its process group keyring, and possibly other keyrings
    - ULPs can pass key serial numbers for PSKs, x.509 certificates, and private keys
    - Some of these can be long-lived

Future Work

- Support for DTLS is planned but not started
- Support for QUIC is under way (see slide 6)
- Support for session re-key has been proposed for kTLS; planned for the netlink protocol and tlshd, but not started
- Support for storing certs in TPM is planned but not started
- Tackling TLS protection for root filesystem resources is still being discussed

Component Availability

- A TLS handshake user agent (tlshd) is part of ktls-utils
  - Upstream is https://github.com/oracle/ktls-utils
  - ktls-utils has been packaged for Fedora, SuSe, and Debian
- The kernel handshake API was merged in v6.4, along with server-side SunRPC and NFSD support for RPC-with-TLS
- Client-side SunRPC and NFS client support is in v6.5
- NVMe with TLS is coming soon (patches are under review)
- In-kernel QUIC prototype: https://github.com/lxin/quic

AMA & Discussion