rpcsec_gss support for kernel RPC servers

This document gives references to the standards and protocols used to implement RPCGSS authentication in kernel RPC servers such as the NFS server and the NFS client's NFSv4.0 callback server. (But note that NFSv4.1 and higher don't require the client to act as a server for the purposes of authentication.)

RPCGSS is specified in a few IETF documents:

- RFC2203 v1: https://tools.ietf.org/rfc/rfc2203.txt
- RFC5403 v2: https://tools.ietf.org/rfc/rfc5403.txt

There is a third version that we don't currently implement:

- RFC7861 v3: https://tools.ietf.org/rfc/rfc7861.txt

Background

The RPCGSS Authentication method describes a way to perform GSSAPI Authentication for NFS. Although GSSAPI is itself completely mechanism agnostic, in many cases only the KRB5 mechanism is supported by NFS implementations.

The Linux kernel, at the moment, supports only the KRB5 mechanism, and depends on GSSAPI extensions that are KRB5 specific.

GSSAPI is a complex library, and implementing it completely in kernel is unwarranted. However GSSAPI operations are fundementally separable in 2 parts:

- initial context establishment
- integrity/privacy protection (signing and encrypting of individual packets)

The former is more complex and policy-independent, but less performance-sensitive. The latter is simpler and needs to be very fast.

Therefore, we perform per-packet integrity and privacy protection in the kernel, but leave the initial context establishment to userspace. We need upcalls to request userspace to perform context establishment.

NFS Server Legacy Upcall Mechanism

The classic upcall mechanism uses a custom text based upcall mechanism to talk to a custom daemon called rpc.svcgssd that is provide by the nfs-utils package.

This upcall mechanism has 2 limitations:

A. It can handle tokens that are no bigger than 2KiB

In some Kerberos deployment GSSAPI tokens can be quite big, up and beyond 64KiB in size due to various authorization extensions attacked to the Kerberos tickets, that needs to be sent through the GSS layer in order to perform context establishment.

B) It does not properly handle creds where the user is member of more than a few thousand groups (the current hard limit in the kernel is 65K groups) due to limitation on the size of the buffer that can be send back to the kernel (4KiB).

NFS Server New RPC Upcall Mechanism

The newer upcall mechanism uses RPC over a unix socket to a daemon called gss-proxy, implemented by a userspace program called Gssproxy.

The gss_proxy RPC protocol is currently documented here.

This upcall mechanism uses the kernel rpc client and connects to the gssproxy userspace program over a regular unix socket. The gssproxy protocol does not suffer from the size limitations of the legacy protocol.

Negotiating Upcall Mechanisms

To provide backward compatibility, the kernel defaults to using the legacy mechanism. To switch to the new mechanism, gss-proxy must bind to /var/run/gssproxy.sock and then write "1" to /proc/net/rpc/use-gss-proxy. If gss-proxy dies, it must repeat both steps.

Once the upcall mechanism is chosen, it cannot be changed. To prevent locking into the legacy mechanisms, the above steps must be performed before starting nfsd. Whoever starts nfsd can guarantee this by reading from /proc/net/rpc/use-gss-proxy and checking that it contains a "1"--the read will block until gss-proxy has done its write to the file.