Draft-ietf-nfsv4-rpccsec-gssv3-05
William A. (Andy) Adamson
andros@netapp.com
IETF 88 Vancouver
Motivation

- Draft un-changed since Nov 07 2012
  - Refocus on NFSv4.2 use
- NFSv4.2 Secure Inter-server SSC
  - It’s use in draft-ietf-nfsv4-minorversion2-19 not clear
- NFSv4.2 LNFS Full mode or server-guest mode
  - Made the obvious changes
  - I did not do a detailed review of LNFS needs
Changes Between draft-04 and draft-05

- **Simplify**
  - Removed client identity assertions due to lack of interest
    - Can be added later, see extensibility section.
  - The `rpc_gss_credVers_3_t handle_version` field is not needed
  - Changed privilege assertions into structured privilege assertions
    - Draft-04 privilege assertions can be expressed as a structured PA

- **Clarify**
  - Restructured the draft including describing the encoding of each message type and its use.
Creating a GSSv3 Context

- Piggy back on existing GSS1 context, called the ‘parent’ context handle
  - Creates a ‘child’ GSS3 context handle bound to the payload
- Conceptually like v4.0 LOCK using confirmed OPEN sequence number stream for a new lockowner
- The RPC message header verifier field is always constructed using the parent context.
  - For all control and data messages
RPCSEC_GSS3_CREATE control msg

- Looks a lot like a GSS1 DESTROY message except:
  - It has as a NULLPROC payload
  - It uses version 3 instead of version 1

```
struct rpc_gss_cred_vers_3_t {
    rpc_gss3_proc_t      gss_proc;    RPCSEC_GSS3_CREATE
    unsigned int         seq_num;     handle field context seq_num
    rpc_gss_service_t    service;     NOT rpc_gss_svc_none.
    opaque               handle<>;    GSS1 (parent) handle
};
```
Creating a GSS3 Context

- The server verifies the payload, then returns a GSS3 (child) handle + per payload response.
  - The GSS3 handle is used in all future GSS3 destroy and data messages that require the create payload binding.

- Both the client and the server need to associate the GSS3 child handle with the parent GSS1(2) handle in their context caches.
  - Given GSS3 handle, lookup parent context for crypto..
Creating a GSSv3 Context

- The RPCSEC_GSS3_CREATE control message NULLPROC payload is any combination of:
  - Channel binding
  - Compound authentication
  - Structured privilege (RPC application defined)
  - Security label assertion
In addition to the parent context handle, the compound authentication payload contains a context handle called the ‘inner’ handle along with a
- version (inner handle version)
- Nonce
- MIC of the nonce using the ‘inner’ GSS-API security context

All uses of a child context handle that is bound to an inner context MUST be treated as speaking for the initiator principal of the inner context handle’s GSS-API security context
Structured Privilege Assertion

- An RPC application defined opaque structure. It’s encoding, server verification, and any server policies are described by the RPC application definition.

```c
struct rpc_gss3_privs {
    rpc_gss3_list_name listname;
    opaque privilege<>;
};
```

- All successful privilege assertions are returned in the `rpc_gss3_create_args assertions` field
Security Label Assertions

- The client performs a GSS3 LIST control message asking the server which security labels it supports
- A GSS3 CREATE control message is sent to bind a set of security labels to the resultant GSS3 handle

```c
struct rpc_gss3_lfs {
    unsigned int lfs_id;
    unsigned int pi_id;
};

struct rpc_gss3_label {
    rpc_gss3_lfs lfs;
    opaque label<>;
};
```
RPCSEC_GSS3_LIST control msg

- Used to list server supported security labels
- Client then chooses security label(s) from the list and creates a GSS3 handle bound to the chosen security labels.
- The LIST control message therefore uses the GSS1 parent handle and sequence number stream in the request, just like RPCSEC_GSS3_CREATE
  - Note: this is a change from the current -05 draft
RPCSEC_GSS3_LIST control msg

- Used to list server supported security labels

struct rpc_gss_cred_vers_3_t {
    rpc_gss3_proc_t gss_proc;            // RPCSEC_GSS3_LIST
    unsigned int seq_num;               // handle field context seq_num
    rpc_gss_service_t service;          // any security service
    opaque handle<>;                    // GSS1 (parent) handle
};

- NULLPROC payload rpc_gss3_list_res
RPCSEC_GSS3 DESTROY control msg

- Looks a lot like a GSS1 DESTROY message
  - With version 3 instead of 1

```c
struct rpc_gss_cred_vers_3_t {
    rpc_gss3_proc_t    gss_proc;    // RPCSEC_GSS3 DESTROY
    unsigned int      seq_num;     // handle field context seq_num
    rpc_gss_service_t  service;    // any security service
    opaque             handle<>;   // GSS3 (child) handle
};
```
The GSS3 data message auth header looks as expected:

```c
struct rpc_gss_cred_vers_3_t {
    rpc_gss3_proc_t    gss_proc;   // RPCSEC_GSS3_DATA
    unsigned int       seq_num;   // handle field context seq_num
    rpc_gss_service_t  service;   // any security service
    opaque             handle<>;  // GSS3 (child) handle
};
```
Using a GSSv3 Context

- All NFS operations using the GSS3 handle assert all successful privileges and features associated with its creation.

- Clients and servers therefore need to cache the specific privileges and features along with the GSS3 handles.
Issues

- Don’t destroy parent without first destroying child
  - GSS3 child handle depends on parent

- Review GSS3 LIST control message
  - Done prior to GSS3 create to find supported labels
  - Needs to use parent context handle

- Review NFSv4.2 and GSS3 security label handling
  - Ensure the two protocols handle the use cases

- Keep the void extensions arg and response?

- Error messages need to be completed
Thank you