NetApp®

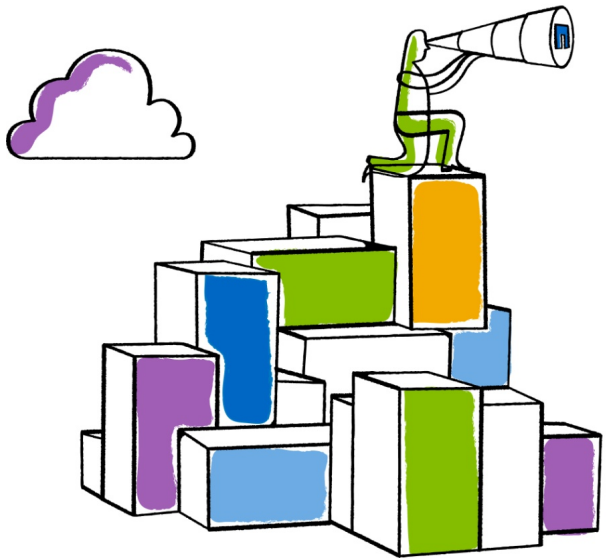
Go further, faster®

GSS Context Management for NFS: The NFS PAG

William A. (Andy) Adamson

andros@netapp.com

Connectathon 2014



The Problem

- GSS context is only destroyed on umount
- Context can be used even when Kerberos credentials are destroyed
  - GSS context lifetime equals the lifetime of the TGS and is renewed as long as the TGT has not expired
  - User destroys Kerberos credentials, but GSS context (TGS) lifetime is not expired.

The Problem

- We do not want to tie GSS NFS Kerberos access to the users Kerberos credentials as there can be other services that need Kerberos credentials
  - Keep the TGT and non-NFS TGS
- **Solution:** Design a way for a user to login and logout of NFS that is independent from Kerberos credentials.
  - NFS GSS context is created from the NFS TGS but is also independent from the NFS TGS

A Solution: Do What AFS Does

- kinit + aklog
- kinit obtains TGT
- aklog requires a TGT, obtains a TGS which becomes the AFS token for the AFS Cell and associates the toke with:
  - UID
  - Process Authentication Group or PAG
- unlog removes AFS access by removing the token, but leaves the Kerberos TGT (and all other non-AFS TGS) intact

Process Authentication Group

- PAG: Number guaranteed to uniquely identify the AFS user
- Used instead of the UID to identify an AFS user to the AFS cache manager
- Processes spawned by the user inherit the PAG and so share the AFS token; thus they gain access to AFS as the authenticated user.
- PAG is associated with a new shell via pagsh

Why Use PAGs?

- It closes a potential security loophole: root access to a user's AFS space.
- UNIX allows anyone already logged in as the local superuser root on a machine to assume any other identity by issuing the UNIX su command.
  - UID based token: root gets users AFS access
  - PAG based token: root gets no access

PAG: Advantages

- Printer and other daemons run under identities (such as the local superuser root) that the AFS server processes recognize only as anonymous.
- Unless PAGs are used, such daemons cannot access files in directories whose access control lists (ACLs) do not extend permissions to the system:anyuser group.

Linux Kernel Task Security Context

- The task credential contains:
  - real, effective, vfsop UIDs and GIDs
  - group_info
  - session, process, and thread keyrings
- Task credential keyrings:
  - session: inherited over fork
  - process: private to this process
  - thread: private to this thread

AFS PAG and Linux Keyring

- afslog with the pag option calls setpag.
- setpag is a "system call" that overloads the setgroups system call
- unlog uses an ioctl (pioctl)
- PAG is stored in the task credential keyring session key (inherited on fork) in an afs-pag key
  - Non-keyring: AFS requires 2 group slots in the task credential group_info for the PAG

NFS PAG Prototype

- nklog: creates a gss-pag key with a PAG payload
  - Uses the add-key system call
  - Added to the task credential session keyring
  - Prototype does not require TGT to run nklog ☺
- PAG fields added to the gss_cred and the auth_cred
  - Filled in when gss-pag key is created
- The crmatch routine called to lookup credentials on each NFS request will ONLY use the PAG
  - No PAG == no NFS GSS access

NFS PAG Prototype

- nulog: destroys the gss-pag key
  - Uses the keyctl system call
  - No TGT required to run
- gss-pag key destruction method marks the gss_cred as destroyed (RPCAUTH_CRED_KEY_DESTROYED)
  - gss_cred crmatch will not return a “destroyed” gss_cred.
- Leaves the Kerberos credentials intact

NFS PAG Prototype

- Buffered writes held in the VM after nulog:
  - rpc tasks servicing buffered writes are flagged (RPC_TASK_BUF_WRITE)
  - gss_cred crmatch routine will return a RPCAUTH_CRED_KEY_DESTROYED gss_cred/context if requesting task has RPC_TASK_BUF_WRITE set
  - GSS Expire patches still apply so GSS context expiring is (still) handled.

Issues:

- Will the Linux NFS community embrace the AFS kinit + aklog / ulog type of Kerberos access to NFS?
  - What about other client implementations?
  - AFS IT administrators are happy with kinit + aklog and make extensive use the AFS pagsh
- PAG only access?
  - Given the advantages of the PAG, why allow UID based access?
  - AFS had UID based Kerberos access for many years and could not enforce PAG only access

NFS PAG Prototype Issues

- NFS pagsh
  - Not yet implemented, but all of the pieces are there
- Design review WRT security
  - How PAG is created
  - Need TGT access for nlog
    - Get TGS for initial NFS server in nlog
  - Need TGT access for nulog
    - Else root + su to user could call nulog
  - Improve understanding and utilization of Linux keyrings

Thank you