# High Level Design of User Revoke

Peter Braam, Eric Mei

Jan 31, 2005

## 1 Requirement

- Be able to revoke a user, prevent it from accessing lustre immediately.
- Be able to pass sub-test of HP acceptance 4.1.51.
- user & mapping databases manipulation API.

## 2 Functional Specification

A sub-command "revoke" will be added into existing tool 'lctl'. When system administrator want to kick somebody off from lustre filesystem (e.g. a certain user has known be malicious or an account be compromised), he could use this functionality on MDS's to prevent the victim user from access lustre filesystem right away. The command format could be:

```
lctl revoke user|all
```

- Here the 'user' format is: uid[@nid[.netid]]
- option @nid.netid is only for remote users. The uid is in term of local uid, thus 'uid@remote_nid.netid' means remote users on node 'remote_nid.netid' who are mapped to local 'uid', it's not intend to remove a certain user on specific node.
- Specified uid without nid or netid means match all nid or netid.
- 'all' means revoke all users.

Actually lctl only remove those in-kernel cache for the victim user, usually there's many other configuration work need to be done by using other admin tools:

- Kerberos Database: For removing a user from kerberos principal database, sysadmin must use kerberos admin tools. And this change will not take effect right away if the victim user has authenticated with MDS's before the removal (because of client side credential cache).

- User Database: For removing a user from user database, sysadmin also must resort to other tools, usually standard unix tools. This change will not take effect right away if this user had ever accessed lustre before the removal (because of in-kernel LSD cache).
- User Mapping Database: For removing a user from remote user mapping database, sysadmin need edit the configure file manually. This only affect certain user on certain remote client. This change will not take effect right away if this user had ever acessed lustre before the removal (because of in-kernel uid mapping cache).

So when sysadmin actually revoke a user, he usually at first did one or more steps of above according to requirement, then invoke lctl to finally revoke the user. In cases that user database or user mapping database are not centrally managed by e.g. LDAP, sysadmin must remove the user from all configure files on each MDS's, this could be done by using 'pdsh', etc.

What above described is the basic requirement. There's an additional one: for user and mapping database, write a C API library (probably later add python support), which can query, add, remove, and enumerate users in each database. 'edit' could be implemented as remove + add.

By using this API, we could provide much complete functionality. Sysadmin could do everything about user account within single lctl tools; Kernel upcall helper also could use this API to obtain information from mapping database, etc.

# 3 Use Cases

## 3.1 Revoke Alice's access right on all clients, permanently

1. Sysadmin remove Alice from user database on all MDS's.
2. Sysadmin invoke 'lctl revoke alice_uid' on all MDS's.
3. Alice from local clients will not be able to access lustre.
4. Any remote users who are mapped to Alice will not be able to access lustre.

## 3.2 Revoke Alice's access right on remote client remote1

1. Suppose alice@remote1 is mapped to local user Bob.
2. Sysadmin remove mapping entry of 'alice_uid@remote1 -> bob' from user mapping database.
3. Sysadmin invoke 'lctl revoke bob_uid@remote1' on all MDS's.
4. Alice will not be able to access lustre from remote1.
5. Bob from an local client could still work fine.

# 4 Logic Specification

There's several kinds of in-kernel cache for certain user: LSD, gss context, and uid-mapping. In the future we might add consideration of removing OSS access capability.

1. LSD: On each MDS, each user (uid) correspond to at most one LSD entry. There's already an existing interface to flush LSD for a certain user: simply write an uid into '/proc/fs/lustre/mds/lsd_flush' (Note this is subject to change). Write in '-1' will flush all LSD entries.

2. GSS Context: On each MDS, each user (principal) might correspond to several(even many) gss contexts. The gss module should export a proc entry. When provided uid and remote nid/netid, it should be able to find out the initiating/established gss contexts and destroy them. Providing a special tag will flush all gss contexts.

3. UID Mapping: Firstly found out per-client structure for specified nid/netid, then destroy the mapping entries for specified uid. Since this is strongly related to GSS context, we can use the export proc entry for gss context to initiate this flush. Thus when sysadmin trying to flush gss contexts for certain user, we also flush associated uid-mapping.

This work should be done after the completion of GSS and remote uid/gid handling implementation.

The user and mapping databases manipulation API could be simple not much restriction, and the details is very much related to the actual database structure. we leave the details to the following DLD document.

# 5 State Management

Since we'll flush several cache separately, we might have situation that not strictly consistency. For example, after we flushed alice from cache1, someone re-populate it in cache1 while do it on cache2. In fact, the inconsistency between LSD and gss context is perfectly allowed. Only one thing need be sure is: since uid mapping is established after that of gss context, thus we need flush uid mapping at first, and then flush gss context. This could prevent unnecessary error when doing 'revoke' while we don't actually remote it from mapping database.

No serious locking issues, no special recovery consideration.

# 6 Alternatives

None.

# 7 Focus of Inspection

- Is the lctl interface reasonably reflect the facts?
- Could it pass acceptance test?