Integrating Lustre with User Security Administration

LAD’15 // Chris Gouge // 2015 Sep
Topics

- User Security in Linux
- POSIX Permissions
- The Requirement for Upcall in Lustre
- Upcall Utilities Overview
- Upcall with Samba Gateways
User Security in Linux
## User Security: Authentication vs. Authorization

<table>
<thead>
<tr>
<th>Establishes:</th>
<th>By means of:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Authentication</strong></td>
<td><strong>Identity</strong></td>
</tr>
<tr>
<td><strong>Authorization</strong></td>
<td><strong>Access</strong></td>
</tr>
</tbody>
</table>
Common Linux Frameworks for User Security

- **PAM (Pluggable Authentication Module)**
  - framework for logging into a Linux system, i.e. proving identity
  - result of login is a user account (username/uid/default gid)
  - multiple methods of login are supported
    - as hinted by the name

- **NSS (Name Service Switch)**
  - framework for queries to various “name services”, including user/group records
  - can be used to resolve group membership while checking permissions
  - multiple kinds of name services are supported
    - as hinted by the name
Common Choices for User Security in Lustre Environments

Many nodes requires synchronization of user accounts across all nodes

- LDAP (Lightweight Directory Access Protocol)
- AD (Active Directory)
  - Mostly LDAP compatible; older versions have quirks
- NIS (Network Information System)
- Local files (/etc/passwd and others)
  - Replicated manually, or via puppet, etc. to all nodes
User Security for Lustre

- **Users login** on the Lustre client nodes (compute nodes)
  - User accounts configured using PAM and NSS on these nodes
  - Upon login, users get their `uid` and default `gid` on these client nodes
    - This is all irrelevant to the Lustre filesystem.

- Upon file I/O requests, the Lustre client passes the `uid/gid` to the Lustre MDS for **permissions check**
  - Users do not login to the Lustre MDS directly, or otherwise directly interact with any Lustre server.
POSIX Permissions
POSIX Identity

- User account
  - username – systemwide unique name
  - uid – systemwide unique number
  - default gid

- Group account
  - groupname – systemwide unique name
  - gid – systemwide unique number
  - member list (a list of usernames)

- “Supplemental groups” – groups other than the default group, that a user belongs to
POSIX I/O Request

- Each *process* running in the system maintains:
  - effective *uid*
  - effective *gid*

- In the simplest case, these are the *uid* and *default gid* of the logged-in user.

- The process sends these 2 numbers in each I/O request to the filesystem.
POSIX File Permissions

● The finest granularity of POSIX filesystem permissions is a file.

● Each file has the following permission info recorded in its directory record:
  ○ owner uid
  ○ owner gid
  ○ read/write/execute bits for the owner uid
  ○ read/write/execute bits for the owner gid
  ○ read/write/execute bits for everyone else

● Access Control Lists (ACLs) are optional, and stored in extended attributes.
  ○ Lustre supports ACLs; stored on MDS
POSIX Permission
Checking: Examples
Permission Check Scenario #1

- Suppose we have a file:

  ```
  filename   uid   gid   user/group/other
  foo.txt   1003:3001   rwxr-x--x
  ```

- And the filesystem gets a “write” request for this file with
  uid=1003 and gid=3001
  - The uids match, so the “owner uid” (or “user”) permission bit for
    “write” is checked. Writes are allowed to this file. In fact rwx
    means reads and executes are also allowed for user 1003.
Permission Check Scenario #2

- Suppose we have a file:

<table>
<thead>
<tr>
<th>filename</th>
<th>uid</th>
<th>gid</th>
<th>user/group/other</th>
</tr>
</thead>
<tbody>
<tr>
<td>foo.txt</td>
<td>1003:3001</td>
<td>rwxr-x--x</td>
<td></td>
</tr>
</tbody>
</table>

- And the filesystem gets a “read” request for this file with uid=1071 and gid=3001
  - The uids don’t match; user bits are ignored.
  - The “owner gid” (or “group”) permission bit for “read” is checked. Reads are allowed, due to r-x, for users in group 3001.
  - (Or really, processes running with that effective gid.)
Permission Check Scenario #3

- Suppose we have a file:

<table>
<thead>
<tr>
<th>filename</th>
<th>uid</th>
<th>gid</th>
<th>user/group/other</th>
</tr>
</thead>
<tbody>
<tr>
<td>foo.txt</td>
<td>1003:3001</td>
<td></td>
<td>rwrx---x</td>
</tr>
</tbody>
</table>

- And the filesystem gets a “read” request for this file with uid=1071 and gid=3449
  - The uids don’t match; user bits are ignored.
  - The gids don’t match… but now we have an interesting question:
  - Is user 1071 in group 3001?
  - The filesystem cannot answer this on its own, it must call out (or call up) to user security services.
Scenarios with Lustre
User Security for Lustre (redux)

- **Users login on the Lustre client nodes (compute nodes)**
  - User accounts configured using PAM and NSS on these nodes
  - Upon login, users get their `uid` and default `gid` on these client nodes
    - This is all irrelevant to the Lustre filesystem.

- **Upon file I/O requests, the Lustre client passes the `uid/gid` to the Lustre MDS for permissions check**
  - Users do not login to the Lustre MDS directly, or otherwise directly interact with any Lustre server.
Lustre User Login - 1 of 4

Security Service

Lustre client

MDS

OSS

Jane

OSS

OSS
Lustre User Login - 2 of 4

Security Service

Lustre client

Lustre client

Lustre client

jane/h3LLo

MDS

OSS

OSS

OSS
Lustre User Login - 3 of 4

Security Service

Lustre client

jane/h3LLo ?

MDS

OSS

OSS

OSS

OSS
Lustre User Login - 4 of 4

Security Service

uid:3013
gid:4069

...done!
Lustre File Access
Lustre File Access (Simple) - 1 of 2

May jane
[uid:3013/gid:4069]
access foo.txt
[owner: 3013/4443]
[modes:-rwxr-x---]

Notes:
- jane is uid 3013
- foo.txt owner is 3013
- jane owns this file and owner has rwx permission
Lustre File Access (Simple) - 2 of 2

Security Service

MDS

OSS

OSS

file handle for foo.txt

...done!
Lustre File Access (Group Upcall) - 1 of 2

May jane
[uid:3013/gid:4069]
access secret.log
[owner: 3456/4443]
[modes:-rwxr-x---]?

Notes:
jane does not own the file
jane's default gid does not match
What is jane's permission?
* MDS does not know.
MDS must have an *upcall utility* in place to be able to contact the security service.
Upcall Utilities for Lustre

● **External / Separate from Lustre / Not part of Lustre**
  ○ Although, a default is provided
  ○ And it works well enough for most sites

● **The Magic Incantation:**

```bash
[root@mgs]
# lctl conf_param myfs-MDT0000.mdt.identity_upcall=/usr/sbin/l_getidentity
```

● No reason there can’t be other ways of doing it
Common Lustre Upcall Utilities

- **l_getidentity** *(used to be l_getgroups)*
  - Default
  - Looks for user/group membership on the MDS using NSS
    - NSS, in turn, relies on:
      - Separate configuration of external security service
      - Locally-defined user/group accounts on the node
  - This is both versatile & vulnerable
    - Any kind of user/group account configuration can be used.
    - But users should never need to login to these nodes!

- **l_adsidentity**
  - Looks for user/group membership on an Active Directory server
  - Mostly relevant to older AD server implementations
  - Supports only 1 AD server, no SSL connection
**l_getidentity** versus **l_adsidentity**

- **l_getidentity** *can do anything that l_adsidentity can!*  
  *with correct configuration of relevant Linux services*
  - Protocol for Active Directory is LDAP
  - Schema configurable via `ns1cd.conf`
    - RFC2307bis
    - Active Directory mappings (current schema)
    - Microsoft Services For Unix
- **l_getidentity** *can do even more!*
  *indirectly, by leveraging Linux services*
  - Backup servers
  - SSL connections
  - Alternate Bind methods
Limitations of \texttt{l_getidentity}

- Requires extensive configuration of the base system
  - Its greatest strength is its weakness

- If you do the minimal configuration to make Lustre work, then users will be able to login to Lustre server nodes.
  - This is a security issue at some sites
  - Multiple ways to prevent this:
    - Set user shell to \texttt{/sbin/nologin}
    - Disable PAM modules
    - etc.
  - Requires even more configuration!
Introducing \texttt{l\_getidentity\_nss}

- Leverages NSS without requiring PAM configuration
  - LDAP (including Active Directory)
  - NIS
  - others

- Supports local user/group definition \textit{without} creating any accounts
  - \texttt{passwd and group files in /etc/lustre}

- Supports multiple types of security service (user/group definitions) at once
  - Search order is set separately from \texttt{nsswitch.conf}
l_getidentity versus l_getidentity_nss

- **l_getidentity_nss** can do anything that **l_getidentity** can!
  * relevant to Lustre

- **l_getidentity_nss** doesn’t open up user logins on Lustre servers in its simplest usage

- Both still require configuration of the base system.
  - Maybe **l_getidentity_nss** requires a little bit less in some cases.
Is `l_getidentity_nss` even needed?

- We don’t hear a lot of users clamoring for new upcall utilities

- We do hear general concerns about securing systems
  - Every site has different policies and different focus areas
  - Preventing server logins is a concern for some sites

- Some Lustre developer opinions (my paraphrasing)
  - “It’s a small thing.”
  - “Not worth bothering anyone to commit it upstream.”

- Is there any interest?
Upcall with a Samba Gateway
Samba Gateway as a Lustre Client

Security Service(s)

Lustre client

access file?

MDS

Samba gateway

access file?

jane/h3LLo

uid:3013
gid:4069

jane/h3LLo

uid:3013
gid:4069

jane/h3LLo

uid:3013
gid:4069
Upcall due to CIFS Client of Samba Gateway
Security Service: NIS + AD + ID Mapping Service

Presumably AD has no POSIX uid/gid support in this scenario. Otherwise the 3rd-party ID Mapper (eg. Centrify) would be redundant.
Questions?

chris.gouge+lad@seagate.com