Securing Lustre with Nodemap and Shared Key

Stephen Simms / Jeremy Filizetti
High Performance File Systems
Indiana University
hpfs-admin@iu.edu

Lustre Ecosystem Workshop
Baltimore, MD
March 8-9 2016

RESEARCH TECHNOLOGIES
INDIANA UNIVERSITY
University Information Technology Services

PERVASIVE TECHNOLOGY INSTITUTE
INDIANA UNIVERSITY

Current Build Instructions for this Tutorial (3/9/2016)

We'll follow the guide for building Lustre from source provided by Intel:
https://wiki.hpdd.intel.com/pages/viewpage.action?pageId=8126821

We also install: libgssglue libgssglue-devel openssl-devel krb5-libs krb5-devel

After step 3 in the "Preparing the Lustre source", we branch and patch:

```
[build@build lustre-release]$ git checkout a3e6b142c074df8ef20cbf3a9d9f7687c9ed9a5f -b shared-key
git fetch http://review.whamcloud.com/fs/lustre-release refs/changes/28/16728/10 && git cherry-pick FETCH_HEAD
git fetch http://review.whamcloud.com/fs/lustre-release refs/changes/97/17597/4 && git cherry-pick FETCH_HEAD
git fetch http://review.whamcloud.com/fs/lustre-release refs/changes/98/17598/4 && git cherry-pick FETCH_HEAD
git fetch http://review.whamcloud.com/fs/lustre-release refs/changes/99/17599/4 && git cherry-pick FETCH_HEAD
git fetch http://review.whamcloud.com/fs/lustre-release refs/changes/00/17600/4 && git cherry-pick FETCH_HEAD
git fetch http://review.whamcloud.com/fs/lustre-release refs/changes/01/17601/4 && git cherry-pick FETCH_HEAD
git fetch http://review.whamcloud.com/fs/lustre-release refs/changes/02/17602/4 && git cherry-pick FETCH_HEAD
git fetch http://review.whamcloud.com/fs/lustre-release refs/changes/50/16950/2 && git cherry-pick FETCH_HEAD
git fetch http://review.whamcloud.com/fs/lustre-release refs/changes/73/18773/1 && git cherry-pick FETCH_HEAD
```

We include the --enable-gss option to configure during step 1 in the "Configure and build Lustre" section:

```
[build@build lustre-release]$ ./configure --with-linux=path_to_linux_source --enable-gss [other-options]
```

Security and Lustre

- Lustre is widely deployed across disparate networks
- Security continues to be a major concern in operations
- There is pressure to mount Lustre system to new areas while maintaining compliance

Nodemap Use Cases

Nodemap was a technology preview in Lustre 2.7 which will be completed for the Lustre 2.9 release.

Using Nodemap, UIDs and GIDs provided by remote clients can be mapped onto a local set of UIDs and GIDs. You may find this useful if:

- You need to prevent UID and GID collisions between clients in different administrative domains
- Two or more partner organizations would like to share data
- You can limit access from a partner site

Nodemap terms

Nodemap is deployed on the MDS, MGS, and OSS nodes and is invisible to clients. Key elements include:

- NIDs, to which a unique *mapping* is defined
- *Policy groups*, which consist of one or more sets of NIDs
- Two *properties*, "trust" and "admin", which can optionally be applied to a policy group
- A collection of identity maps or *idmaps* which determine the translation table for a policy group

Basic setup for Biology lab case

- Bob's Biology lab runs a locally administered machine with multiple users.
- Bob wants to mount our Lustre file system locally.
- Only certain users on Bob's system require access to the Lustre file system.
- The UIDs and GIDs on Bob's lab system conflict with other existing clients that also mount our Lustre file system.

Our goal is to map the UID of one of Bob's users to that same user's UID on our system, and, at the same time, squash access from all others in Bob's lab.

Biology lab problem

Biology Server
IP: 172.18.0.14

jwilson
UID/GID 503

ghouse
UID/GID 626

We would like to allow jwilson access to the Lustre system, but not ghouse.

We have an existing user on our Lustre system with UID/GID 503, so we will map jwilson to 20013. The UID and GID do not have to match -- this is a simplified example.

Pre-Nodemap setup

From a trusted client, the scratch directory of our Lustre file system looks like this:

```
# ls -l
total 107096
-rw-r----- 1 5173  2300  7884800 Feb 24 10:58 assemblyres.0001
-rw-r----- 1 5157  2300  7884800 Feb 24 10:58 assemblyres.0002
-rw-r----- 1 5157  2300  7884800 Feb 24 10:58 assemblyres.0003
-rw-r----- 1 5159  2300  7884800 Feb 24 10:58 assemblyres.0004
-rw-r----- 1 6781  2300  7884800 Feb 24 10:58 assemblyres.0005
-rw-r----- 1 8001 20013 10077184 Feb 24 10:57 sampleData1
-rw-r----- 1 8001 20013 15199232 Feb 24 10:57 sampleData2
-rw-r----- 1 8001 20013 17921024 Feb 24 10:58 sampleData3
-rw-r----- 1 8001 20013  3472384 Feb 24 10:58 sampleData4
-rw-r----- 1 9360  3350 23564288 Feb 24 10:59 tset0a
```

Making a quick map

These commands are issued on the Lustre MGS or MGS/MDS.

Create a policy group

```
mds01# lctl nodemap_add BiologyMap
```

Add at least one NID to the policy group

```
mds01# lctl nodemap_add_range --name BiologyMap --range 172.18.0.14@tcp
```

Add at least one idmap for the user and group

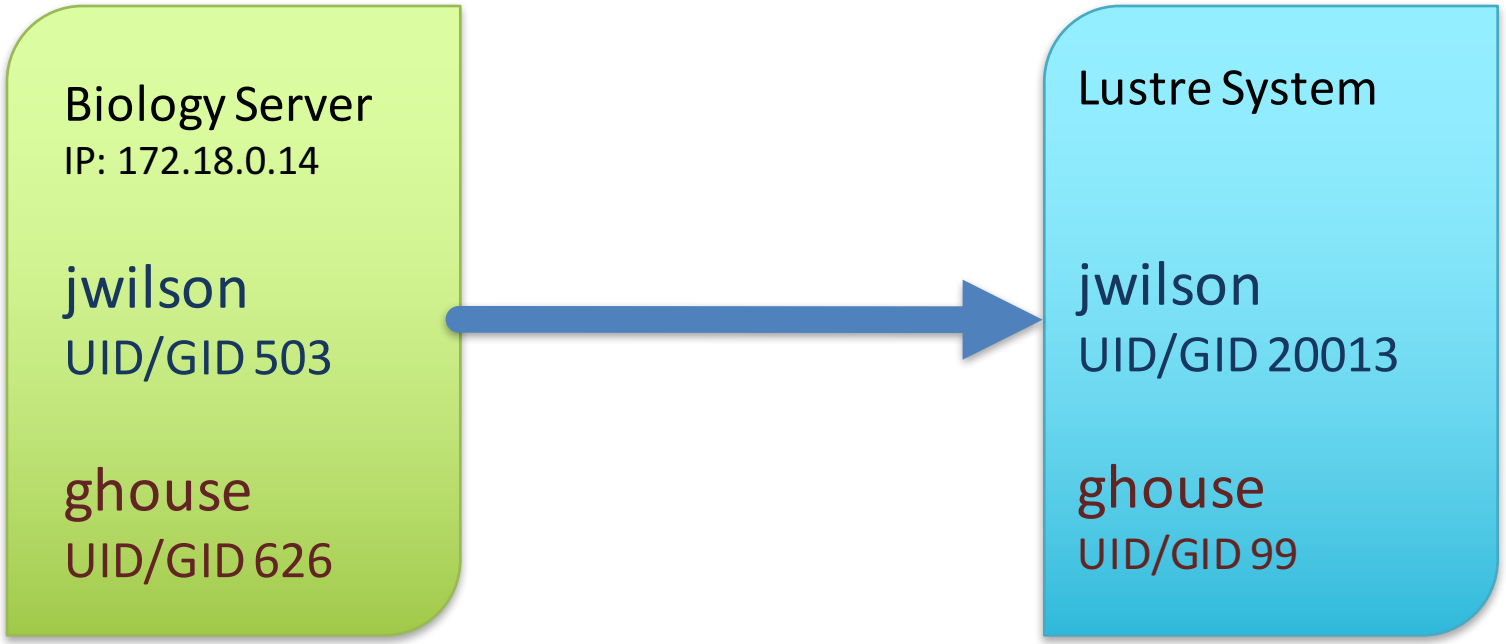
```
mds01# lctl nodemap_add_idmap --name BiologyMap --idtype uid --idmap 503:20013
mds01# lctl nodemap_add_idmap --name BiologyMap --idtype gid --idmap 503:20013
```

Activate the Nodemap feature

```
mds01# lctl nodemap_activate 1
```

Biology lab with Nodemap

Our selected user gets mapped, but all other user access is squashed.



jwilson's view from mapped Biology server

```
[jwilson@cli01 scratch]$ id
uid=503(jwilson) gid=503(jwilson) groups=503(jwilson)
[jwilson@cli01 scratch]$ ls -l
total 107096
-rw-r----- 1 nobody nobody   7884800 Feb 24 10:58 assemblyres.0001
-rw-r----- 1 nobody nobody   7884800 Feb 24 10:58 assemblyres.0002
-rw-r----- 1 nobody nobody   7884800 Feb 24 10:58 assemblyres.0003
-rw-r----- 1 nobody nobody   7884800 Feb 24 10:58 assemblyres.0004
-rw-r----- 1 nobody nobody   7884800 Feb 24 10:58 assemblyres.0005
-rw-r----- 1 nobody jwilson 10077184 Feb 24 10:57 sampleData1
-rw-r----- 1 nobody jwilson 15199232 Feb 24 10:57 sampleData2
-rw-r----- 1 nobody jwilson 17921024 Feb 24 10:58 sampleData3
-rw-r----- 1 nobody jwilson  3472384 Feb 24 10:58 sampleData4
-rw-r----- 1 nobody nobody  23564288 Feb 24 10:59 tset0a
[jwilson@cli01 scratch]$ file sampleData1
sampleData1: data
```

ghouse's view from mapped Biology server

```
[ghouse@cli01 scratch]$ id
uid=626(ghouse) gid=626(ghouse) groups=626(ghouse)
[ghouse@cli01 scratch]$ ls -l
total 107096
-rw-r----- 1 nobody nobody   7884800 Feb 24 10:58 assemblyres.0001
-rw-r----- 1 nobody nobody   7884800 Feb 24 10:58 assemblyres.0002
-rw-r----- 1 nobody nobody   7884800 Feb 24 10:58 assemblyres.0003
-rw-r----- 1 nobody nobody   7884800 Feb 24 10:58 assemblyres.0004
-rw-r----- 1 nobody nobody   7884800 Feb 24 10:58 assemblyres.0005
-rw-r----- 1 nobody jwilson 10077184 Feb 24 10:57 sampleData1
-rw-r----- 1 nobody jwilson 15199232 Feb 24 10:57 sampleData2
-rw-r----- 1 nobody jwilson 17921024 Feb 24 10:58 sampleData3
-rw-r----- 1 nobody jwilson  3472384 Feb 24 10:58 sampleData4
-rw-r----- 1 nobody nobody  23564288 Feb 24 10:59 tset0a
[ghouse@cli01 scratch]$ file sampleData1
sampleData1: regular file, no read permission
```

Creating new files as squashed or mapped user

New files created by a squashed user will become owned by the squash user and group, typically nobody/nobody (99/99).

New files created by a mapped user will be translated to their UID and GID on the Lustre file system.

```
[jwilson@cli01 ~]$ dd if=/dev/zero of=poultryData bs=1024 count=1300

[jwilson@cli01 scratch]$ ls -l poultryData
-rw-r----- 1 jwilson jwilson 1331200 Feb 24 13:41 poultryData

[root@trusted scratch]# ls -l poultryData
-rw-r----- 1 20013 20013 1331200 Feb 24 13:41 poultryData
```

Integration with existing systems

- The trusted property will need to be enabled for maps including existing clients. These maps will need to be created before enabling Nodemap.
- Unmapped hosts match the default policy which squashes by default. This protects against both root access and by access from unknown hosts, should firewalls not be blocking those hosts.
- Hosts and maps can be added while the file system is online. In Lustre 2.9, it will take around ten seconds to sync configuration.

Security improvements from Nodemap

Nodemap can allow us to:

- Cherry-pick which users on a remote system can access a Lustre system.
- Ensure that unknown systems are squashed when mounting the file system.
- Re-map either UID or GIDs, or both, to prevent conflicts or restrict access.

What does Shared Key offer?

- Isolation
  - Prevents clients from mounting without the shared key
  - Group and isolate NID ranges to a specific key (w/ Nodemap)
- Message Integrity
  - Prevents man-in-the-middle attacks
  - Ensure RPCs cannot be altered without detection
- Privacy
  - Prevents eavesdropping
  - Encryption of RPCs
- Ability to choose security flavors between OSS, MDS, MGS, and client nodes, as site policy dictates

Shared Key Mechanism Security Flavors

	skn	**ska**	**ski**	**skpi**
Name	Shared Key Null	Shared Key Auth	Shared Key Integrity	Shared Key Privacy and Integrity
Key Required				
RPC Integrity				
RPC Privacy				
Bulk Integrity				
Bulk Privacy				

Command Parameters for lgss_sk

In the Bob's Biology Lab example, we will call lgss_sk with the following parameters:

-f lustre : The file system using our key is named lustre

-e 86400 : Contexts for this key expire in 86400 seconds (one day)

- default is over 68 years so good idea to set this

-n Biology : Nodemap name for this key is Biology

-w /root/Biology.key : Generate key file in this location

-d /dev/urandom : Source of entropy

- /dev/random is the default (and higher quality) entropy source
- /dev/urandom generated more quickly for testing

Shared Key Integrity Setup

We create a Shared Key key file for the Biology lab machines and distribute the key to all Lustre servers and Biology lab clients:

```
[root@mds ~]$ /usr/sbin/lgss_sk -f lustre -e 86400 -n Biology -w
/root/Biology.key -d /dev/urandom
[root@mds ~]$ scp /root/Biology.key oss01:/root/Biology.key
[root@mds ~]$ scp /root/Biology.key cli01:/root/Biology.key
```

Then, unmount the Biology clients:

```
[root@cli01 ~]$ umount /mnt/lustre
[root@cli02 ~]$ umount /mnt/lustre
```

On the MGS, we set SPTLRPC security flavor to Shared Key Integrity (ski) for connections between clients and the MDT, and between clients and OSTs on the Lustre TCP network:

```
[root@mds ~]$ lctl conf_param lustre.srpc.flavor.tcp.cli2ost=ski
[root@mds ~]$ lctl conf_param lustre.srpc.flavor.tcp.cli2mdt=ski
```

Shared Key Integrity Setup, continued

- Install keyutils package on all Lustre servers and clients.
- Unmount all Lustre resources.
- On all Lustre servers and Biology lab clients, we create /etc/request-key.d/lgssc.conf, adding the following:

```
create lgssc * * /usr/sbin/lgss_keyring  %o %k %t %d %c %u %g %T %P %S
```

- Enable lsvcgssd Shared Key and MDS service support:
  - Add -s –m to LSVCGSSDARGS variable in /etc/sysconfig/lsvcgss on MDS
- Enable Shared Key and OSS service support:
  - Add -s –o to LSVCGSSDARGS variable in /etc/sysconfig/lsvcgss on OSS
- Start lsvcgssd on the MDS and OSS servers with service lsvcgss start
- Load the ptlrpc_gss module on all servers with modprobe ptlrpc_gss

- Mount the MGT/MDT and OSTs with the -o skpath=/root/Biology.key option.

Mounting the Shared Key Integrity file system

```
[root@cli01 ~]$ modprobe ptlrpc_gss
[root@cli01 ~]$ mount -t lustre -o skpath=/root/Biology.key
172.16.0.51@tcp:/lustre  /mnt/lustre
```

Verify OSC and MDC client connections are using ski:

```
[root@cli01 ~]$ lctl get_param *.*.srpc_*
mdc.lustre-MDT0000-mdc-ffff88007ca7b000.srpc_contexts=
ffff88007b85fbc0:  uid 0, ref 2, expire 1457035621(+86374),  fl uptodate,cached,,  seq 4, win 2048,
key 01dc5f9f(ref 1), hdl 0xfd4cae388b699bb6:0xf0d58784672e4c93,  mech: sk
mdc.lustre-MDT0000-mdc-ffff88007ca7b000.srpc_info=
rpc flavor:     ski
bulk flavor:    ski
[…]
osc.lustre-OST0000-osc-ffff88007ca7b000.srpc_contexts=
ffff88007b85f5c0:  uid 0, ref 2, expire 1457035621(+86374),  fl uptodate,cached,,  seq 1, win 2048,
key 2cec1f5a(ref 1), hdl 0xfd23f559c34ef35:0xf0d58784672e4c94,  mech: sk
osc.lustre-OST0000-osc-ffff88007ca7b000.srpc_info=
rpc flavor:     ski
bulk flavor:    ski
[…]
osc.lustre-OST0001-osc-ffff88007ca7b000.srpc_contexts=
ffff88007b85f380:  uid 0, ref 2, expire 1457035621(+86374),  fl uptodate,cached,,  seq 1, win 2048,
key 0ed7fca8(ref 1), hdl 0xfd23f559c34ef36:0xf0d58784672e4c95,  mech: sk
osc.lustre-OST0001-osc-ffff88007ca7b000.srpc_info=
rpc flavor:     ski
bulk flavor:    ski
```

Without a valid key, mount fails!

If we try to mount Lustre somewhere besides Bob's Biology Lab, the mount is refused:

```
[root@cli02 ~]$ mount -t lustre 172.16.0.51@tcp:/lustre  /mnt/lustre

mount.lustre:  mount 172.16.0.51@tcp:/lustre  at /mnt/lustre  failed:
Connection  refused
```

Enabling Privacy and Integrity using Shared Key

On MGS node, we set SPTLRPC security flavor to Shared Key Privacy and Integrity (skpi) for all connections between clients and the Lustre MDT:

```
[root@mds ~]$ lctl conf_param lustre.srpc.flavor.tcp.cli2mdt=skpi
```

Next, we set flavor between clients and OSTs:

```
[root@mds ~]$ lctl conf_param lustre.srpc.flavor.tcp.cli2ost=skpi
```

Verifying privacy using Shared Key

Wait about a minute, then verify security flavor for the Biology lab is now skpi:

```
[root@cli01 ~]$ lctl get_param *.*.srpc_*
mdc.lustre-MDT0000-mdc-ffff88007ca7b000.srpc_contexts=
ffff88007c0aa140: uid 0, ref 2, expire 1457037129(+83108), fl uptodate,cached,, seq 131, win
2048, key 2e6a9ffc(ref 1), hdl 0xfd4cae388b699bb7:0xf0d58784672e4c96, mech: sk
mdc.lustre-MDT0000-mdc-ffff88007ca7b000.srpc_info=
rpc flavor:     skpi
bulk flavor:    skpi
[…]
osc.lustre-OST0000-osc-ffff88007ca7b000.srpc_contexts=
ffff88007c0aa740: uid 0, ref 2, expire 1457037125(+83104), fl uptodate,cached,, seq 131, win
2048, key 09b4c5bd(ref 1), hdl 0xfd23f559c34ef38:0xf0d58784672e4c97, mech: sk
osc.lustre-OST0000-osc-ffff88007ca7b000.srpc_info=
rpc flavor:     skpi
bulk flavor:    skpi
[…]
osc.lustre-OST0001-osc-ffff88007ca7b000.srpc_contexts=
ffff88007c0aa8c0: uid 0, ref 2, expire 1457037123(+83102), fl uptodate,cached,, seq 131, win
2048, key 05ef0d0c(ref 1), hdl 0xfd23f559c34ef37:0xf0d58784672e4c98, mech: sk
osc.lustre-OST0001-osc-ffff88007ca7b000.srpc_info=
rpc flavor:     skpi
bulk flavor:    skpi
```

Thank You!

OpenSFS for helping fund this project

Generous Volunteers: Nathan Rutman, Andreas Dilger, John Hammond, and Ken Hornstein.

Andrew Korty, Indiana University Security Officer

Kit Westneat, Jeremy Filizetti, Nathan Lavender, Chris Hanna and the rest of IU's HPFS team past and present for their extraordinary efforts.

Nodemap has been added to the 2.9 Lustre release manual:
http://review.whamcloud.com/#/c/17634/

Watch for Shared Key to fully land in 2.9!

Shared Key Security Process

- Upon server connect, security context lookup triggers upcall to request-key
- Resulting action is determined by key type, which is lgssc for Lustre
- Scan /etc/request.key.d/lgssc.conf for execution program and arguments
- Upcall information parsed by lgss_keyring and token sent back to kernel space for SEC_CTX_INIT RPC
- RPC received by server, sent to pipe between kernel and user space
- lsvcgssd responds back over the same file descriptor, and server sends RPC reply
- Client parses reply and updates the key which imports the context into the kernel
- Now the OBD connect can continue!

Shared Key File

- Key is generated by lgss_sk utility
- Must be loaded into kernel keyring
  - The -t option specifies the context (mgs, server, client)
  - Mount via mount.lustre with –o skpath=path
  - Options and devices from command determines context
    - The skpath can be a directory or a file
    - For directory, all key files will be loaded
- Any key load failures cause mount to fail!
- Big endian on disk; native byte-ordering in keyring

Shared Key File Contents

- Encryption algorithm - Used for Privacy mode (skpi)
  - AES256 (default)
- HMAC algorithm (Used for Auth, Integrity, and Privacy modes)
  - SHA256 (default)
  - SHA512
- File system name and/or MGS NIDs (limited to 16 NIDs)
  - One or both must be specified
- Context expiration in seconds (default: INT_MAX)
- Session key length
- DH Shared Secret Key length

lsvcgss daemon

- Server side user-space component
- Handles token from SEC_CTX_INIT RPC and reply token
- Allows specific mechanisms to be enabled:
  - Kerberos: -k
  - Shared-Key: -s
  - GSS Null: -n
- Allows individual services to be enabled:
  - MGS: -g
  - MDS: -m
  - OSS: -o

- In foreground, logs to STDOUT; otherwise, to syslog
- Up to four levels of verbosity (-v, -vv, -vvv, -vvvv)
- Modify /etc/sysconfig/lsvcgss to change flags

Shared Key and Kernel Keyring

- Keys are loaded into user session keyring
- Specific key description, which can be found with keyctl_search()
  - MGS
    - lustre:MGS:nodemap (e.g. lustre:MGS:Biology)
  - MDS/OSS
    - lustre:fsname:nodemap (e.g. lustre:scratch:Biology)
  - Client
    - lustre:fsname (e.g. lustre:scratch)

      OR
    - lustre:MGC<NID> (e.g. lustre:MGC192.168.1.1@tcp)