Whamcloud

# Lustre & Security

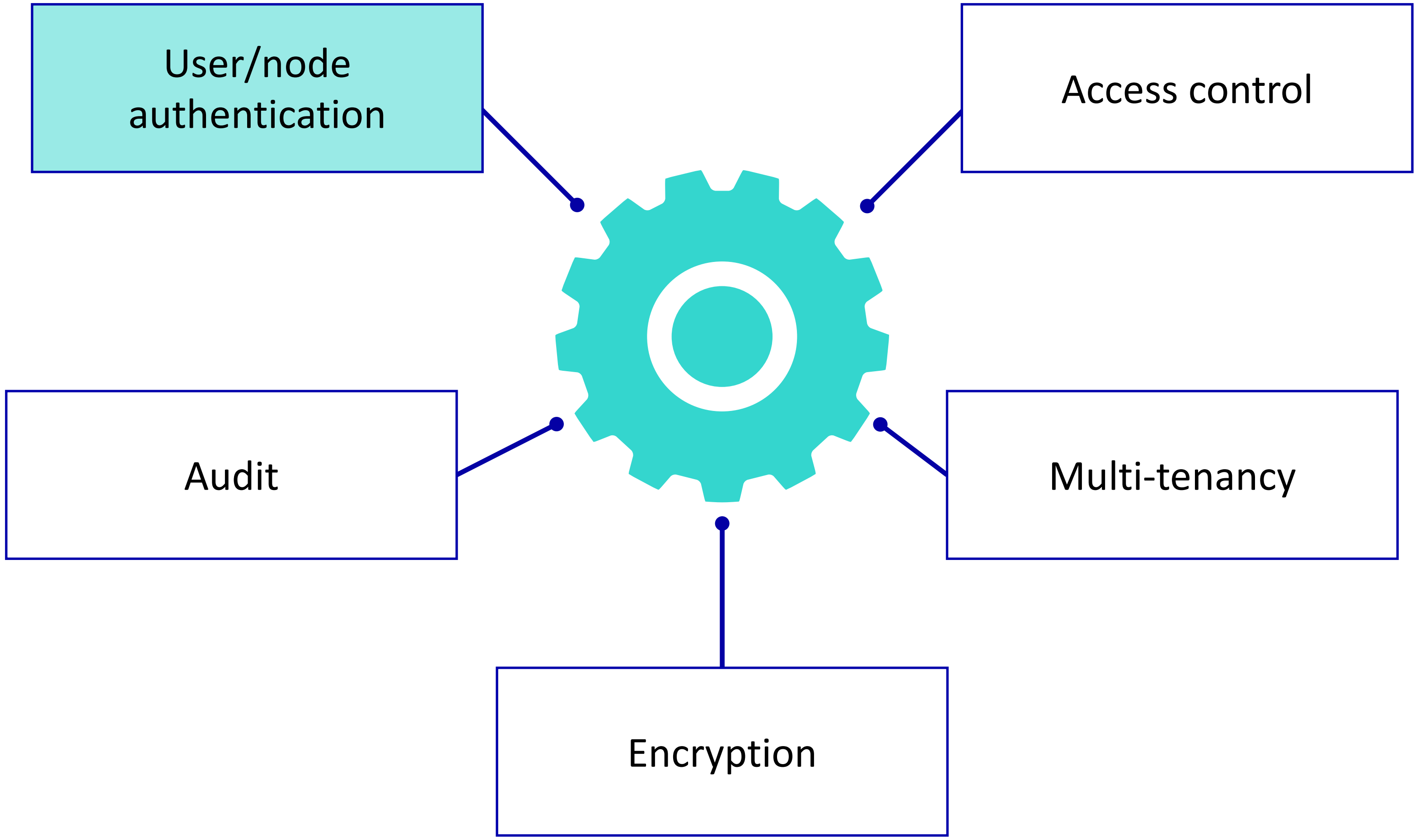
LUG 2019

sbuisson@whamcloud.com

DDN® STORAGE

## Different Security Requirements

- User/node authentication
  - Only authenticated users have access
  - Only authenticated nodes are part of Lustre
- Access control
  - DAC (Discretionary Access Control)
  - MAC (Mandatory Access Control)
- Multi-tenancy
  - Provides isolated namespaces from a single file system
  - Limited namespace exposed to clients
- Encryption
  - Wire Encryption (Network)
  - Data Encryption (Logical and Physical)
- Audit


User/node authentication
Access control
Audit
Multi-tenancy
Encryption

## Lustre User/Node Authentication

- Based on Kerberos Authentication Protocol
  - relies on a 3rd party Kerberos server
  - with Kerberized Lustre
    - users need their own Kerberos credentials to access Lustre file system
      - not just UID/GID perms
    - nodes need Kerberos credentials to be part of the file system
      - prevent from adding illegitimate client or target

  - Available with Lustre 2.8

*https://en.wikipedia.org/wiki/Kerberos_(protocol)*

# Kerberos on Lustre HOWTO: Credentials

- Every file system access needs to be authenticated with Kerberos credentials, named principals:
  - MGS

    `lustre_mgs/<mgt hostname on the interconnect network>.DOMAIN`
  - MDS

    `lustre_mds/<mds hostname on the interconnect network>.DOMAIN`
  - OSS

    `lustre_oss/<oss hostname on the interconnect network>.DOMAIN`
  - Client

    `lustre_root/<client hostname on the interconnect network>.DOMAIN`
- Note that users need their own principals

# Kerberos on Lustre HOWTO: Activation

- Start server-side daemon
  - on all server nodes (MDS, OSS), userspace daemon responsible for checking authentication credentials

```
# lsvcgssd -vv -k
```

- Enable Kerberos authentication by setting flavor

```
mgs# lctl conf_param <fs>.srpc.flavor.default = krb5n
mgs# lctl conf_param <fs>.srpc.flavor.o2ib0 = krb5n
mgs# lctl conf_param <fs>.srpc.flavor.default.client2ost = krb5n
```

  - MGS particular case

```
mgs# lctl conf_param _mgs.srpc.flavor.default=krb5n
```

⇒'`-o mgssec=flavor`' mount option required when mounting Lustre targets and clients

## Shared-Secret Key (SSK)

- If not possible to implement Kerberos for policy or resource reasons
  - Lightweight authentication mechanism is possible in Lustre to allow rapid deployment
- SSK offers strong authentication, by preventing clients from mounting without the shared key
  - directly implemented in Lustre
  - SSK does not rely on external server
  - users do not need any key, only nodes are authenticated
- Available with Lustre 2.9

## SSK HOWTO: Shared Secret Key

- Secret Keys are generated ahead of time with lgss_sk…

```
# lgss_sk -t server -f testfs -w testfs.server.key
# lgss_sk -t client -m testfs.client.key
```

- …then distributed to all Lustre servers and clients that share these keys
  - usually via SSH

## SSK HOWTO: Activation

- Start server-side daemon
  - on all server nodes (MDS, OSS), userspace daemon responsible for checking authentication credentials

```
# lsvcgssd -vv -s
```

- Enable SSK authentication by setting flavor

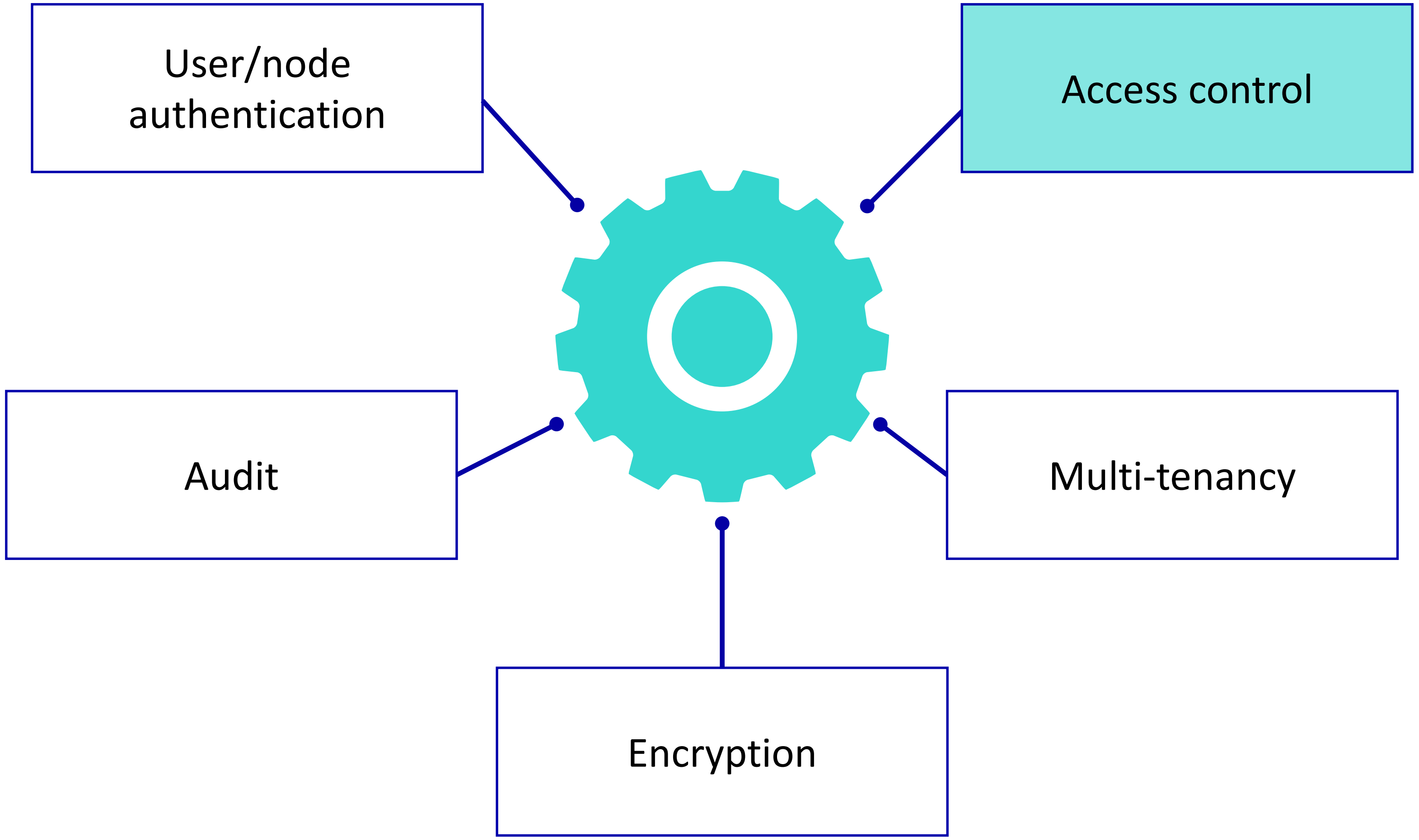
```
mgs# lctl conf_param <fs>.srpc.flavor.default = skn
mgs# lctl conf_param <fs>.srpc.flavor.o2ib0 = skn
mgs# lctl conf_param <fs>.srpc.flavor.default.client2ost = skn
```

  - MGS particular case

```
mgs# lctl conf_param _mgs.srpc.flavor.default=skn
```

- Use ‘skpath’ option to mount targets and clients

```
-o skpath=/path/to/ssk.key
```


User/node authentication
Access control
Audit
Multi-tenancy
Encryption

## Lustre Access Control

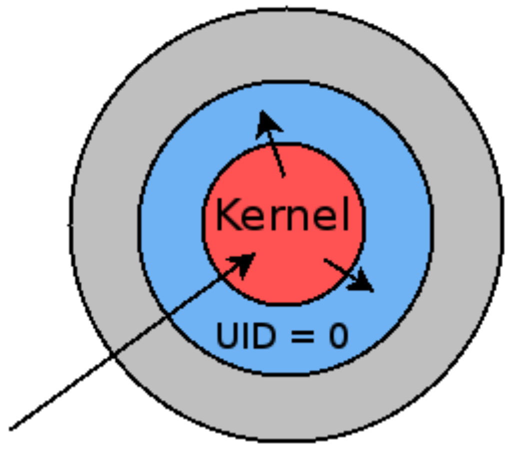
Whamcloud

- DAC (*Discretionary Access Control*): *always been there*
  - o traditional Unix system of users, groups, and read-write-execute rights is a DAC implementation
  - o enforced on MDS side
  - ⇒ MDS servers must have access to users and groups database, similarly to client nodes.

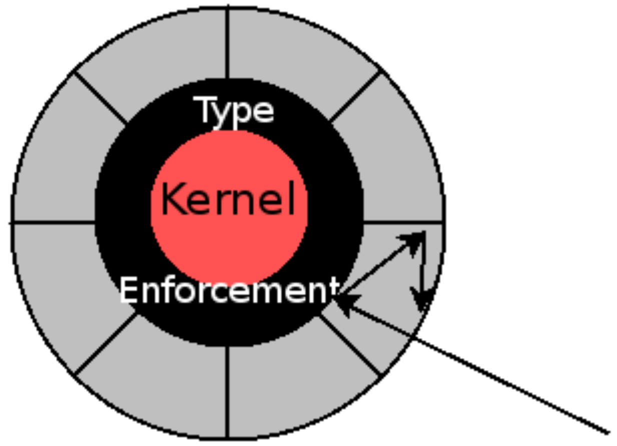
- MAC (*Mandatory Access Control*): available with Lustre 2.8
  - SELinux support in Lustre
    - o Targeted policy
    - o MLS policy
  - enforced on client side

# Mandatory Access Control

- Objective
  - protect from privilege escalation in OS
- Support of SELinux Targeted Policy in Lustre: Lustre 2.8
  - defines confined and unconfined domains for processes and users
  - enforced on client side
  - need to store security information permanently in file xattr
    - use of `security.selinux` xattr to store security context



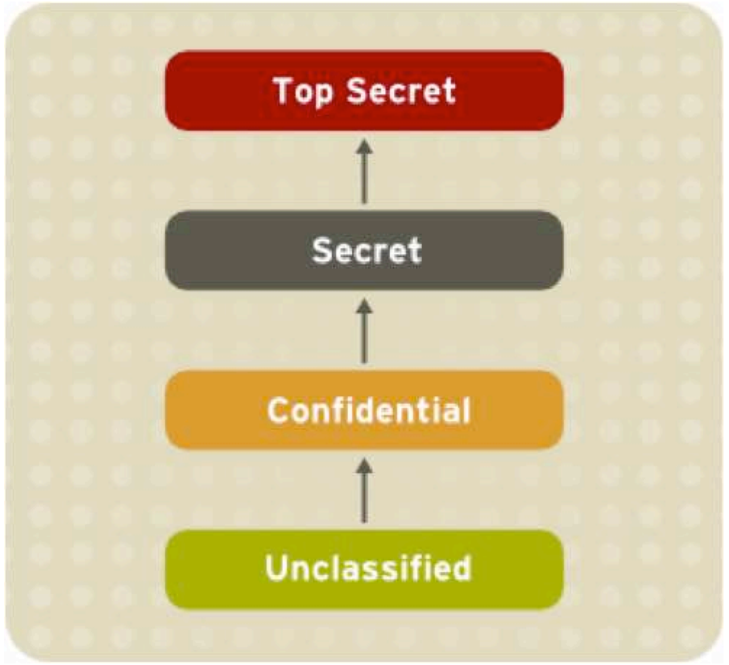
Traditional access control.
UID 0 have full access.



Domain/Type enforcement.
Programs confined in sandboxes.

## Mandatory Access Control

- **Objective**
  - protect data sensitivity
- **Support of SELinux MLS Policy in Lustre: Lustre 2.8**
  - comes on top of Targeted Policy
  - defines the concept of security levels in addition to domains
  - enforced on client side
  - need to store security information permanently in file xattr
    - use of `security.selinux` xattr to store security context



## Mandatory Access Control

- Distributed file systems specificity:
  - really need to make sure data is always accessed by nodes with SELinux policy **enforced**
    - otherwise data is not protected

- SELinux status checking: safeguard for security admins
  - retrieve SELinux status on client nodes:
    - SELinux is enforced
    - which policy module loaded
    - policy is not altered
    - decide on status retrieval frequency: only at mount, for every request, once in a while
  - send clients' SELinux status to servers along with requests
  - on servers, compare info received from clients with reference status stored in nodemap
    - deny access if no match

- Available with Lustre 2.13 / 2.12.1

# SELinux for Lustre HOWTO

- Just enforce desired SELinux policy on **all** Lustre clients
- Nothing required on servers

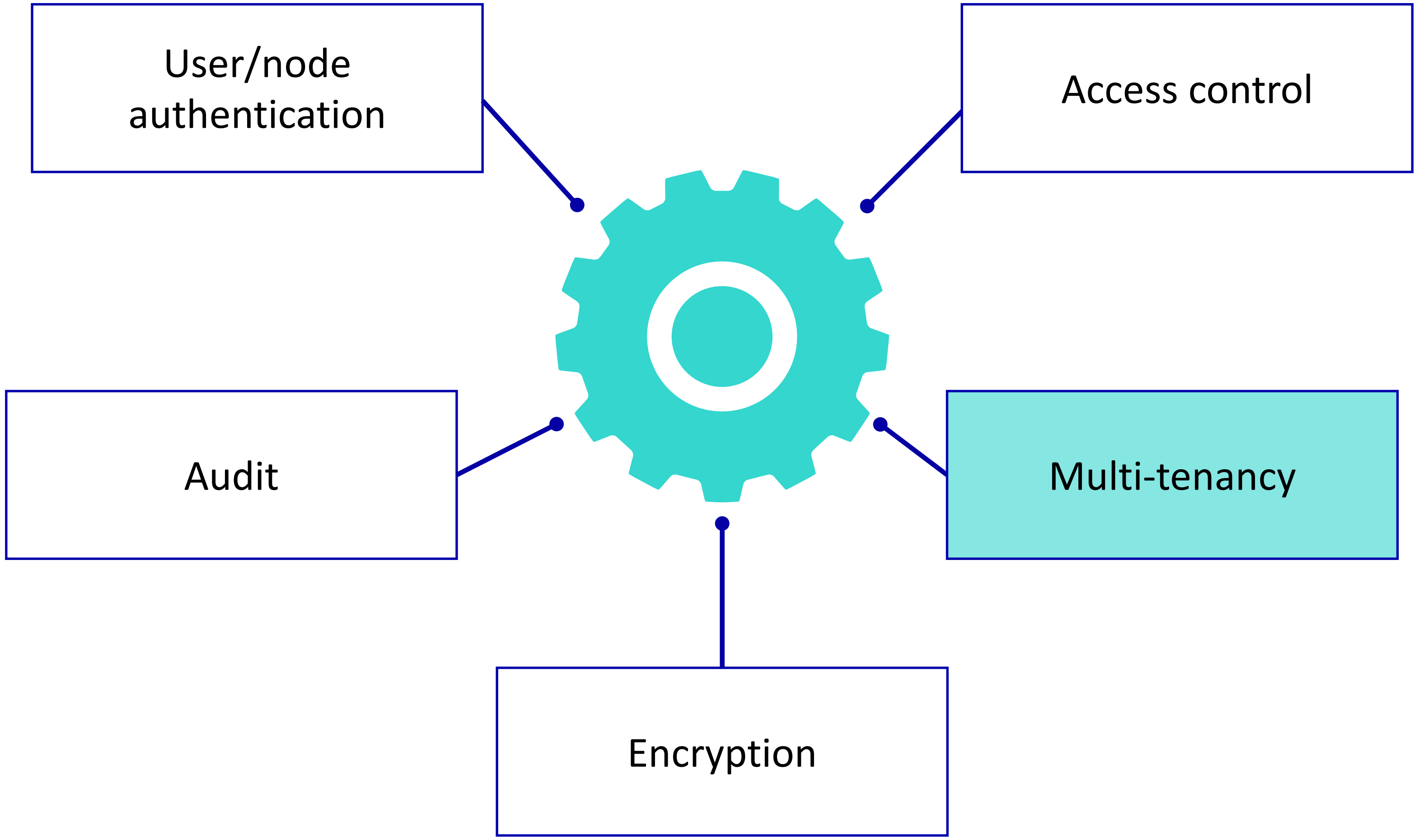
- If you want more: SELinux status checking
  - determine SELinux Policy Info

```
client# l_getsepol
SELinux status info: 1:mls:31:40afb76…
```

  - enforce SELinux Policy Check

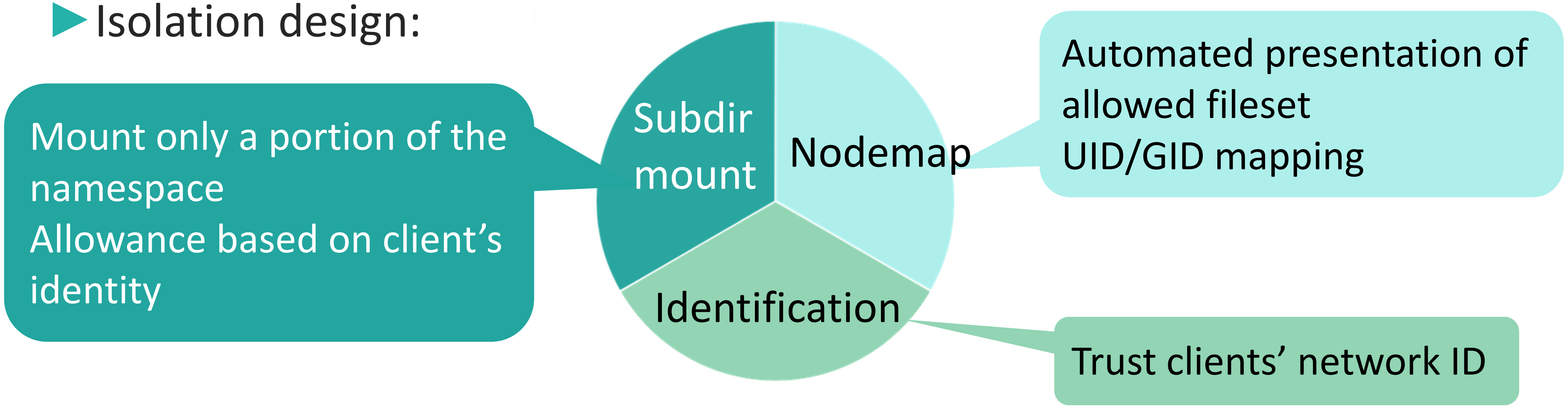
```
mgs# lctl nodemap_set_sepol --name restricted --sepol '1:mls:31:40afb76…'
```

  - send SELinux Status Info from clients
    - *send_sepol* ptlrpc kernel module's parameter


User/node authentication
Access control
Audit
Multi-tenancy
Encryption

## Multi-Tenancy: Concept

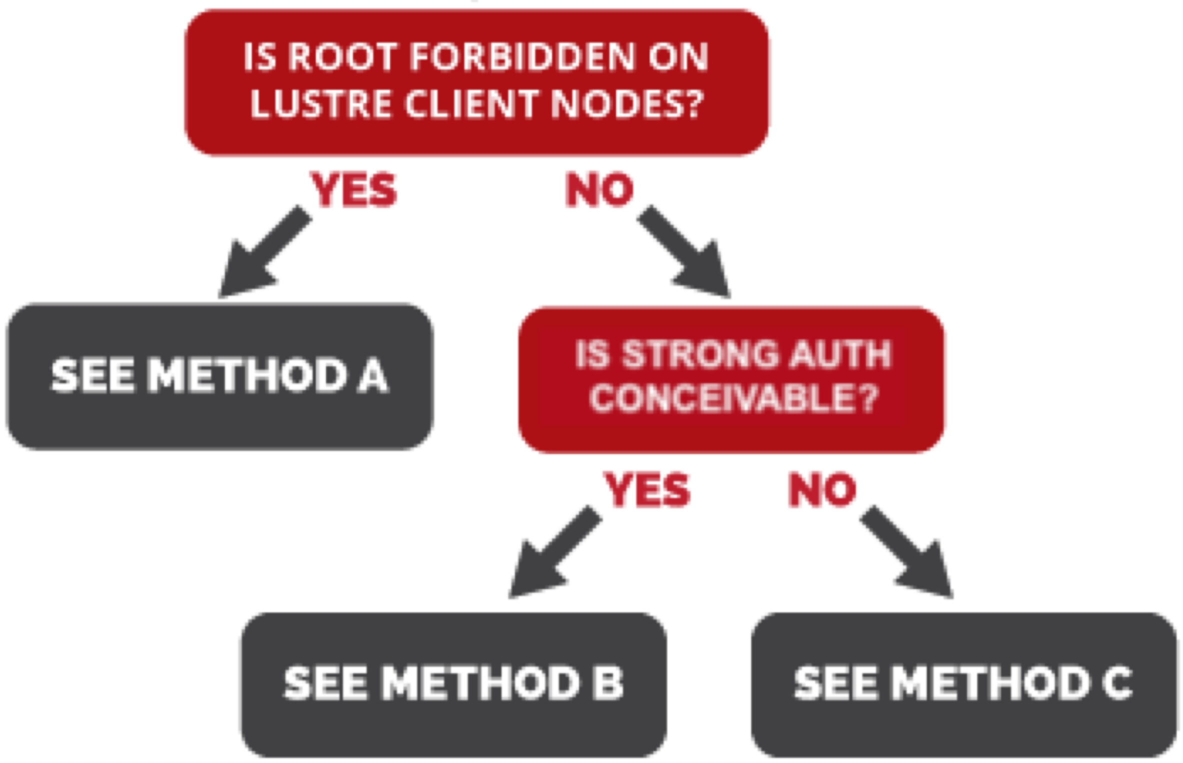
- Isolation design:


Mount only a portion of the namespace
Allowance based on client's identity
Subdir mount
Nodemap
Automated presentation of allowed fileset
UID/GID mapping
Identification
Trust clients' network ID

- Isolation enables Multi-tenancy:
  - different populations of users on the same file systems
  - isolation of these different populations of users
- Available from Lustre 2.10

# Multi-tenancy: How to Implement

- Narrows down to
  - ability to properly identify the client nodes used by a tenant
  - trust those identities


IS ROOT FORBIDDEN ON LUSTRE CLIENT NODES?
YES
NO
SEE METHOD A
IS STRONG AUTH CONCEIVABLE?
YES
NO
SEE METHOD B
SEE METHOD C

## Multi-tenancy: Method A

- Users cannot be root
  - clients's NIDs can be trusted
  - multi-tenancy guaranteed by subdirectory mount and nodemap

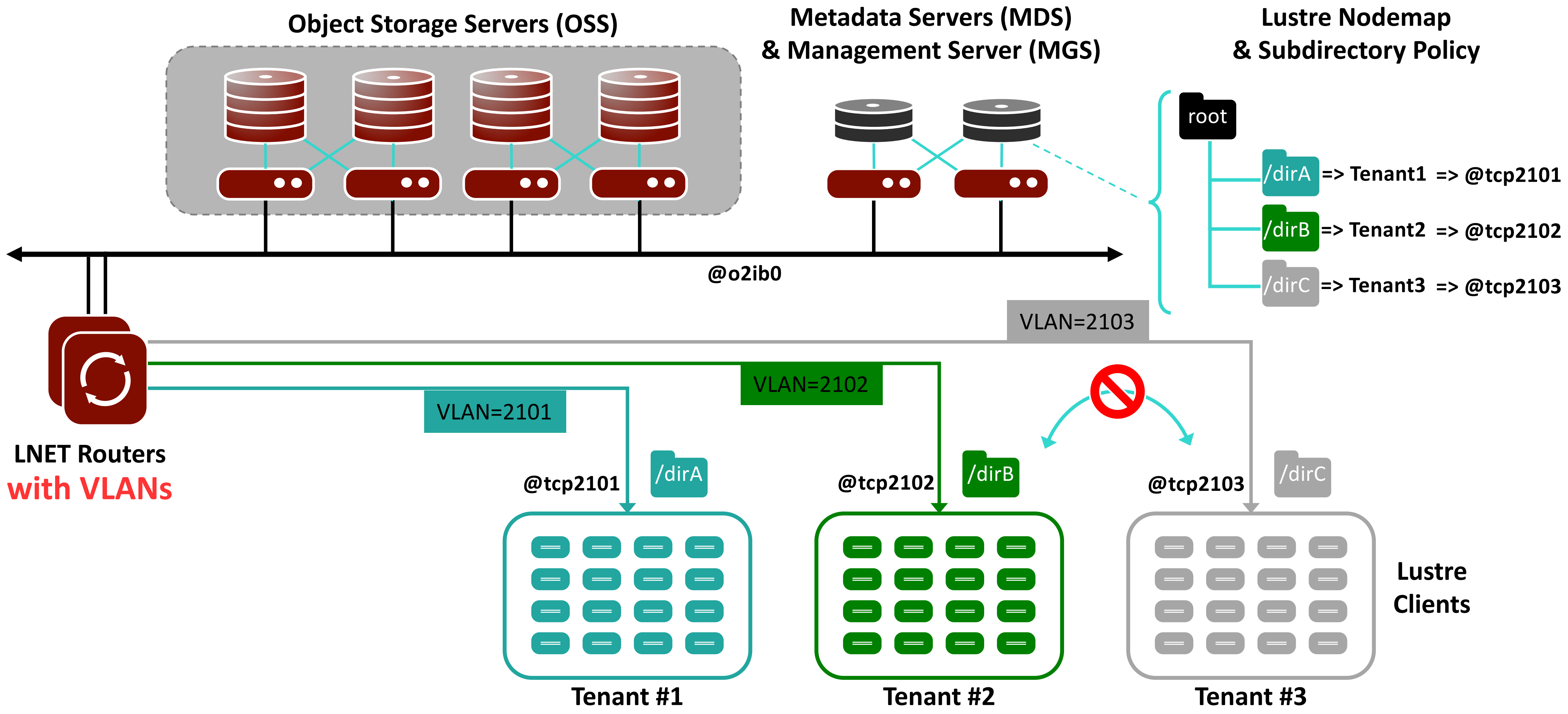
```
lctl set_param nodemap.<nodemap_name>.fileset='/<directory>'
```

  - groups of clients assigned to each tenant can change over time
    - needs to update tenants definitions in nodemaps

## Multi-tenancy: Method B

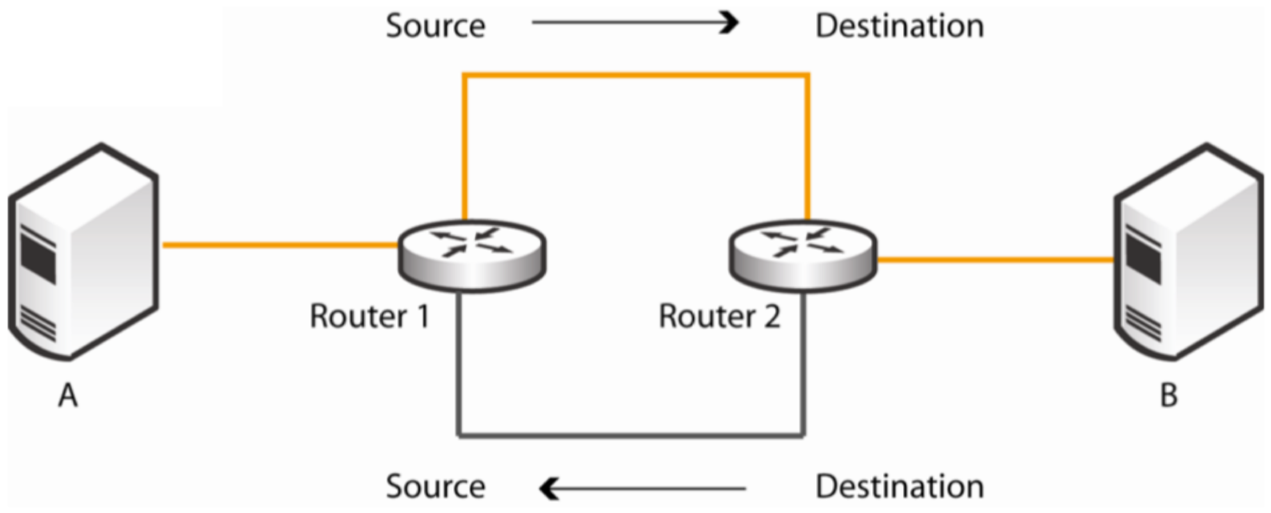
- If Root is Possible on Clients
  - are Lustre clients running inside VMs or containers?
    - advantage: dynamically assign NIDs to clients used by tenants
    - drawback: malicious user may use root privileges to change Lustre client NIDs
  - make use of strong authentication
    - Kerberos - if already in place at customer site
    - Shared-Secret Key is Lustre-specific alternative, much easier to implement
  - how does it work?
    - maliciously modified client NID will not match client’s key
      - installed in VM or container by sec admin
    - Lustre servers will refuse connection

# Multi-tenancy: Method C - make use of LNet routers



# Multi-tenancy: asymmetrical route detection

- Asymmetrical route



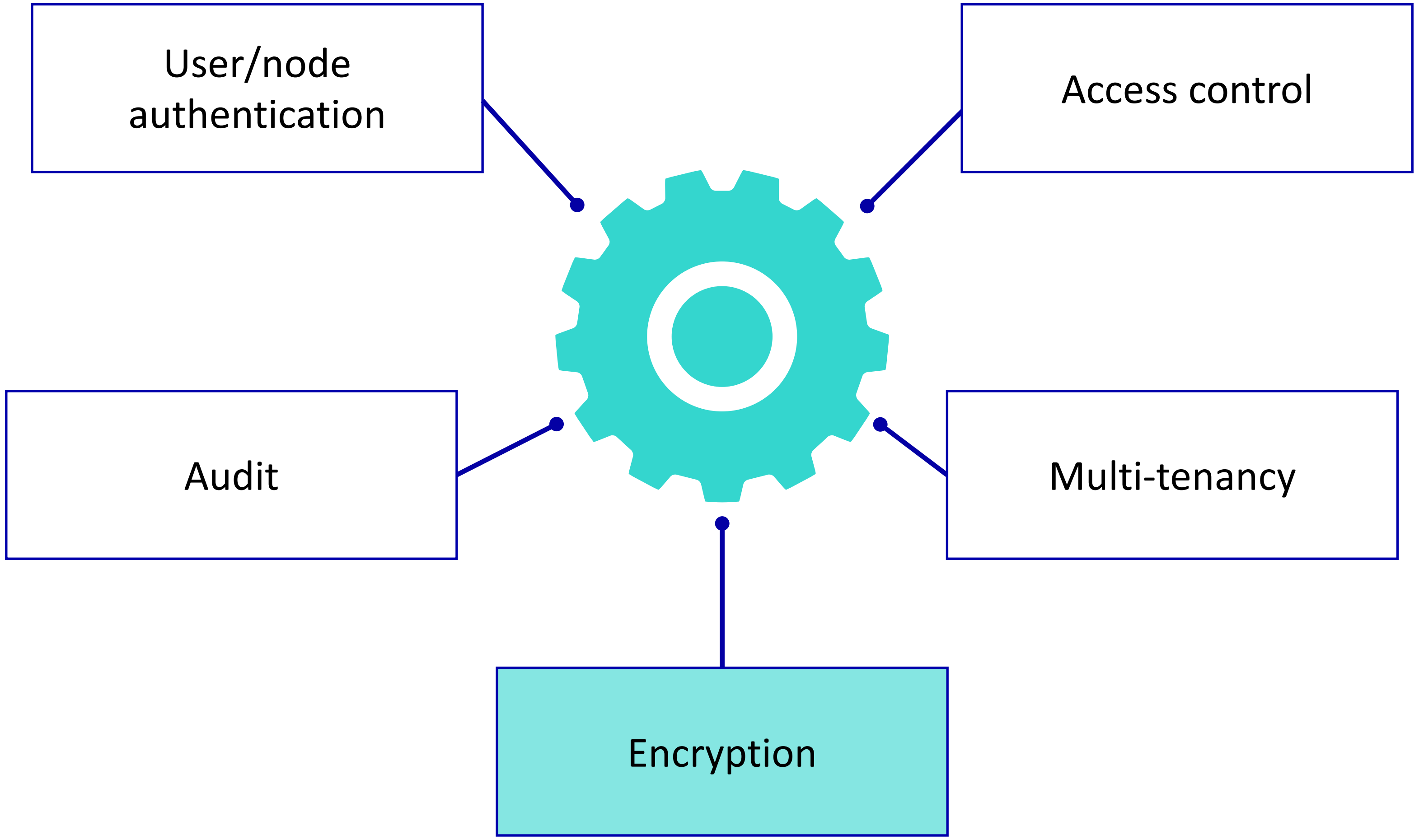
https://support.riverbed.com/bin/support/download?did=84

  - could be the clue of hostile clients injecting data to the servers

- Purpose is to drop asymmetrical route messages

```
lnetctl set drop_asym_route 1
```

- Available with Lustre 2.13 / 2.12.1

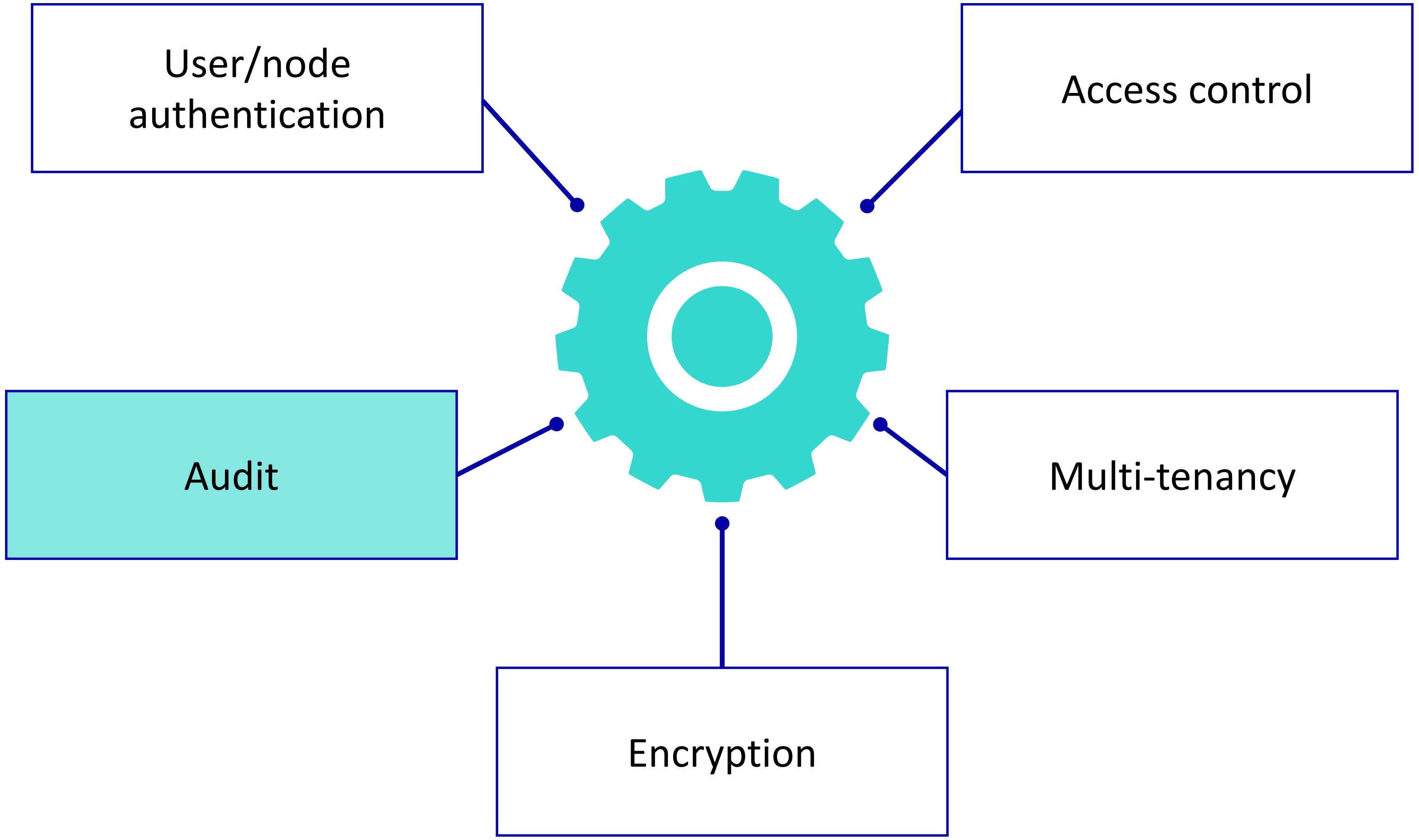

User/node authentication
Access control
Audit
Multi-tenancy
Encryption

## Encryption – On the Wire

- Objective
  - protect data transfers between nodes
    - ‘Man-in-the-middle’ attacks
- Encryption over the network with Kerberos krb5p or SSK skpi flavors
  - for communications between Lustre clients and servers
  - data encrypted on emitter’s side before sending
  - data decrypted on recipient’s side upon receipt
  - large performance impact
- Available from Lustre 2.8 (Krb) / 2.9 (SSK)

# Encryption – Data at REST

- Objective
  - protect against storage theft
  - protect against network snooping
- Encryption at Lustre client level
  - applications see clear text
  - data is encrypted before being sent to servers
  - data is decrypted upon receipt from servers
  - servers only see encrypted data
  - only client nodes have access to encryption keys
- *Available in 2.14+*


User/node authentication
Access control
Audit
Multi-tenancy
Encryption

## Lustre Audit Facility

- **Objective**
  - provide records of all Lustre access

- **Use Lustre changelogs**
  - log activity on MDTs
  - record file system namespace & metadata events
    - with UID:GID and NID info
  - record *even failed access attempts*
  - limit duplicate `open()` and `close()` events
  - restrict nodes from which activity is recorded

- **Available from Lustre 2.11**

# Lustre Audit HOWTO

- All Changelog record types must be enabled, to be able to record events such as OPEN, ATIME, GETXATTR and DENIED OPEN

- Enable all changelog entry types:

```
# lctl set_param mdd.<fsname>-*.changelog_mask=ALL
```

- Then, just register a Changelogs user:

```
# lctl --device <fsname>-<MDT number> changelog_register
```

- Control which Lustre client nodes can trigger the recording of file system access events to the Changelogs

```
# lctl nodemap_modify --name <nodmap_name> \
  --property audit_mode --value=<0,1>
```

Whamcloud

# Thank you!

sbuisson@whamcloud.com

DDN® STORAGE