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

Whamcloud

Lustre Kerberos Update

LAD 2023

sbuisson@whamcloud.com

ddn

Lustre Kerberos Update

Whamcloud

- Why strong authentication?
- What does Lustre offer for strong authentication?
- Focus on Kerberos with Lustre
  - How it works
  - How to implement
  - Recent improvements

Why strong authentication?

Whamcloud

- Customers are asking for it…
  - Legal requirements for security
  - ‘Shared’ file systems
    - User home directories
    - Multiple tenants
    - Cloud-based environment

Lustre features for strong authentication

Whamcloud

- Shared-Secret Key (SSK)
  - Offers strong authentication, by preventing clients from mounting without the shared key
  - Lightweight mechanism to allow rapid deployment
    - Directly implemented in Lustre
    - SSK does not rely on external server
    - Users do not need any key, only nodes are authenticated.
  - Tightly coupled with nodemap
  - Available from Lustre 2.9

Lustre features for strong authentication

Whamcloud

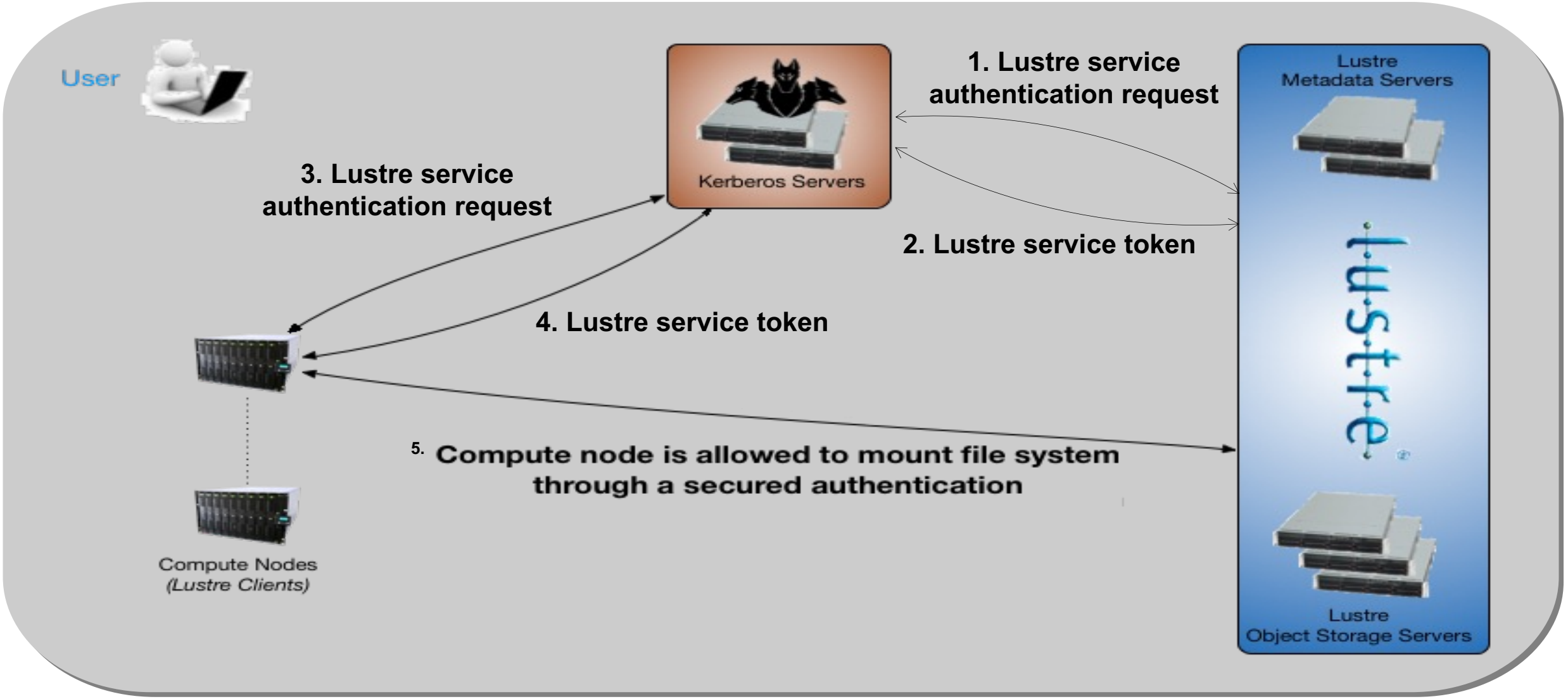
- Kerberos
  - Well-known authentication mechanism
  - Relies on a 3rd party Kerberos server
  - With Kerberized Lustre
    - Nodes need Kerberos credentials to be part of the file system
      - Prevent from adding illegitimate client or target
    - Users need their own Kerberos credentials to access Lustre file system
      - Not just UID/GID permissions
  - Available from:
    - Initial: 1.8/2.0
    - First revival: 2.8

Kerberos Node Authentication

Whamcloud

- Objective
  - Control which nodes can be part of a Lustre file system
- Without authentication
  - Whichever node that
    - Is connected to the Interconnect network
    - Knows the MGS and file system names
  - Can mount Lustre as a client!
  - Can format a target and mount Lustre as a server!
- Kerberos is a solution
  - Nodes need Kerberos credentials to be part of the file system

How Kerberos Works with Lustre Mount

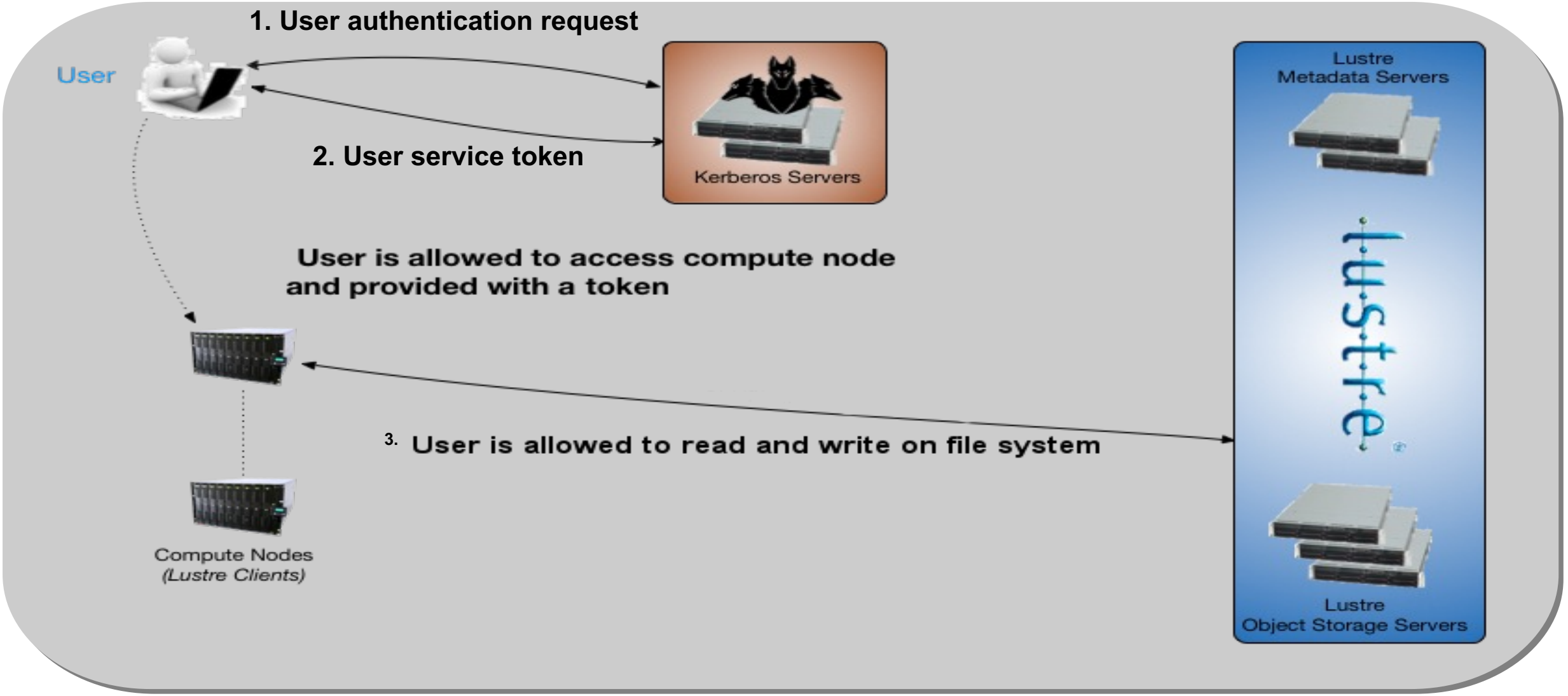


Kerberos User Authentication

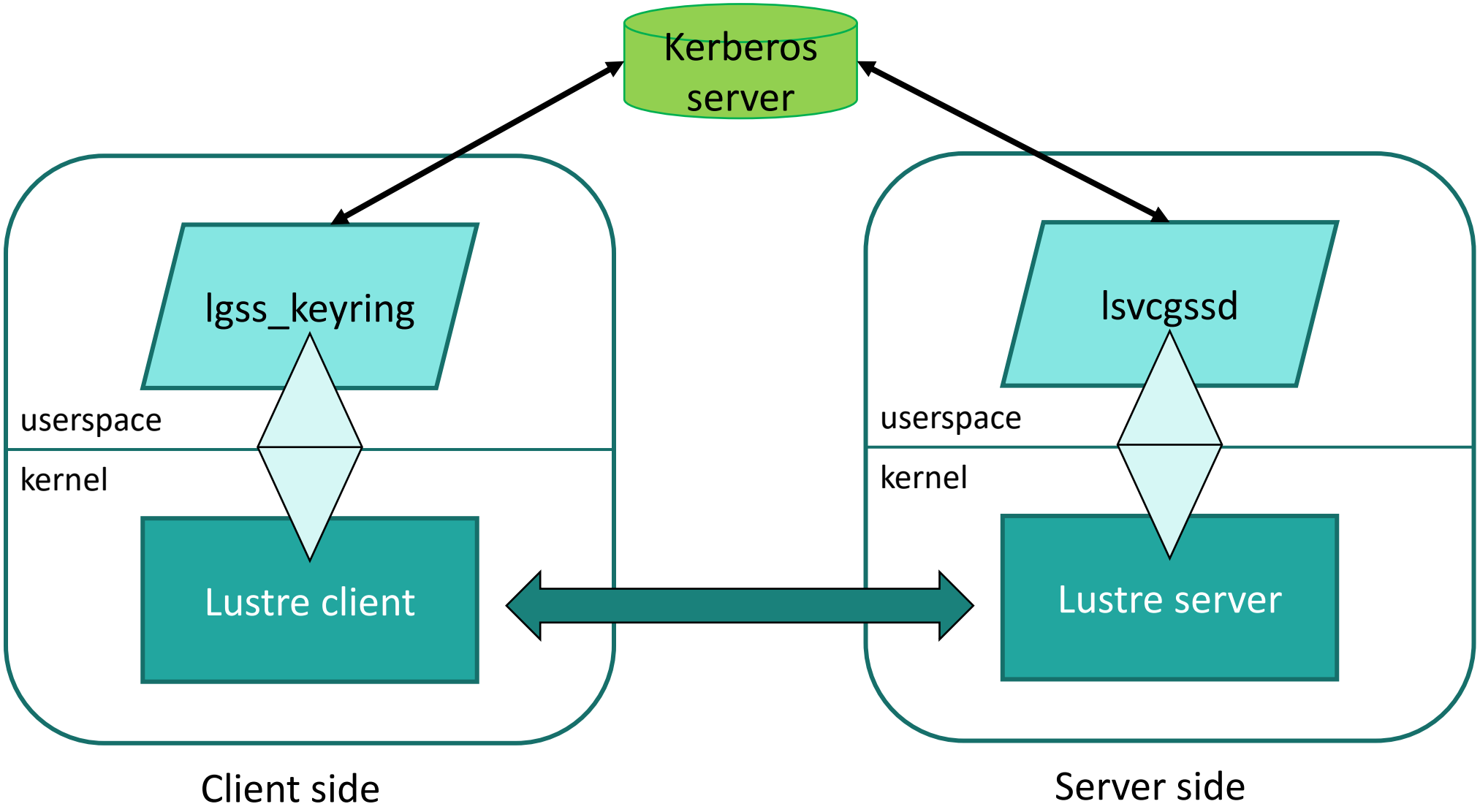
Whamcloud

- Objective
  - Control which users can access a Lustre file system
- Without authentication
  - Whichever user that
    - Has access to a client node where Lustre is mounted
    - Is declared on MDS side
  - Can access Lustre files based on UID/GID permissions
- Kerberos is a solution
  - Users need their own Kerberos credentials to access the Lustre file system

How Kerberos Works with Lustre File Access



Some implementation details



Kerberos on Lustre HOWTO: Credentials

Whamcloud

- Every file system access needs Kerberos credentials, named principals:
  - MGS

    `lustre_mgs/<mgs hostname on the network>.DOMAIN@REALM`
  - MDS

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

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

    `lustre_root/<client hostname on the network>.DOMAIN@REALM`

    or (new!)

    `host/<client hostname on the network>.DOMAIN@REALM`
- Note that users need their own principals

Kerberos on Lustre HOWTO: Activation

Whamcloud

- Start server-side daemon
  - On all server nodes (MGS, 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

The question of the Kerberos server

Whamcloud

- Lustre is Kerberos server agnostic
  - MIT Kerberos
  - Heimdal
  - Microsoft Active Directory
  - …

- What if you do not have a Kerberos server at hand?

`https://github.com/DDNStorage/lustre-docker-kdc`

🚨 **NOT FOR PRODUCTION USE** 🚨

A necessary Kerberos Update

Whamcloud

- Previous revival dates back from Lustre 2.8
  - Code was barely compiling…
  - … and certainly crashing when the feature was enabled.

- Since 2016, same concerns
  - Code that is not widely used
    - Contains bugs
    - Is not exposed to more modern conditions
      - Newer kernels
      - Newer userspace libraries

Kerberos Update – tests

- Kerberos now goes through non-regression tests
  - Thanks to Kerberos env setup in Maloo
  - And fixes in tests

⇒ `Test-Parameters: kerberos=true testlist=sanity-krb5`

LU-16799 tests: fix sanity-krb5
https://review.whamcloud.com/50864

LU-17050 tests: test Kerberos env in sanity-krb5
https://review.whamcloud.com/52068

LU-16804 tests: load CONFIG at beginning of init_test_env
https://review.whamcloud.com/50914

Kerberos Update – cleanup / bugfix

Whamcloud

- "Some" cleanup / bugfixing

LU-9243 gss: fix GSS struct definition badness
https://review.whamcloud.com/46543

LU-16911 sec: quiet messages from identity upcall retry mech
https://review.whamcloud.com/51355

LU-16532 sec: session key bad keyring
https://review.whamcloud.com/49909

LU-16888 gss: fix ptlrpc_gss automatic loading
https://review.whamcloud.com/51264

LU-15896 gss: support OpenSSLv3
https://review.whamcloud.com/47717

Kerberos Update – local client

Whamcloud

- Address the ‘local client mount’ use case
  - Useful for data movement or protocol re-export
  - Strong authentication + integrity + privacy can hurt performance
    - and local client requires dedicated credentials

⇒ Disable security flavor for local client

LU-13343 gss: no sec flavor on loopback connection
https://review.whamcloud.com/46704

Kerberos Update – uid remapping

Whamcloud

- Support Kerberos standards: **cross-realm trust remapping**
  - Lustre's idmap.conf is nice…

```
<remote principal> <NID> <local uid>
```

  - but the GSSAPI can handle it - via krb5.conf

```
auth_to_local_names = {
    <remote name> = <local user>
}
```

LU-16630 sec: improve Kerberos cross-realm trust remapping
https://review.whamcloud.com/50259

Kerberos Update – credentials cache

Whamcloud

- Support Kerberos standards: **credentials cache**
  - Lustre supports FILE ccache, and hardcoded…

    `FILE:/tmp/krb5cc_<xxx>`
  - but the GSSAPI can handle it - via krb5.conf

    `default_ccache_name = KEYRING:persistent:%{euid}`
  - and we can fallback to /tmp/*krb5cc* and /run/user/<uid>/*krb5cc*

LU-16646 krb: use system ccache for Lustre services
https://review.whamcloud.com/50342
LU-16646 krb: improve lookup of user's credentials
https://review.whamcloud.com/50377

Kerberos Update – client principal

Whamcloud

- Support Kerberos standards: **principal name**
  - Lustre **client**'s principal expected to be `lustre_root`
    - allows handling Lustre authentication independently of node authentication
  - but for simpler credentials management of **client** nodes
    - could **also** use standard Kerberos machine principal `host/<hostname>@REALM`

LU-16758 krb: use Kerberos machine principal in client
https://review.whamcloud.com/50709

Kerberos Update – realm

Whamcloud

- Choose Kerberos realm
  - Lustre uses the `default_realm`
  - but nodes can be part of multiple Kerberos realms

⇒ Admins need to be able to specify realm to use

- Client side: '-R' option to lgss_keyring

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

- Server side: '-R' option to lsvcgssd

```
lsvcgssd -vvv -k -R REALM
```

LU-17023 krb: use a Kerberos realm different from default

https://review.whamcloud.com/51914

Kerberos Update – large tokens

Whamcloud

- Handle large authentication tickets and tokens
  - Can be 64KiB due to:
    - authorization extensions attached to the Kerberos tickets
    - large number of supplementary groups
  - Limit in Lustre client code
    - Client can pack token in request up to 1KiB only

    ⇒ Just increase buffer size ☺

LU-17015 gss: support large kerberos token on client
https://review.whamcloud.com/51946

Kerberos Update – large tokens

Whamcloud

- Handle large authentication tickets and tokens (cont.)
  - Limit in code used by Lustre server side
    - Server is relying on sunrpc cache implementation
      - Token exchanged with userspace limited to PAGE_SIZE
    - Need to rework implementation of cache and exchange pattern with userspace

      ⇒ Leverage Lustre's upcall cache mechanism

      » Already used for identity cache (supplementary groups)

      » Do not touch existing GSS context negotiation routines

LU-17015 gss: support large kerberos token for rpc sec init
https://review.whamcloud.com/52224
LU-17015 gss: support large kerberos token for rpc sec ctxt
https://review.whamcloud.com/52305

Lustre Kerberos Update – wrap-up

Whamcloud

- Easy to implement
  - If you already have a Kerberos infrastructure
  - Otherwise SSK is a valid alternative

- New Kerberos revival: 2.16
  - Now code is tested regularly
  - And much nicer than before ☺ (usability, standard practices)
  - So please use Kerberos authentication (and SSK)!

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

Whamcloud

Thank you!

sbuisson@whamcloud.com

ddn

Kerberos flavors

Whamcloud

flavor	auth	RPC message protection	Bulk data protection
krb5n	yes	no	checksum
krb5a	yes	headers integrity	checksum
krb5i	yes	integrity	integrity
krb5p	yes	privacy	privacy

Kerberos on Lustre

Whamcloud

- Performance impact
  - with Kerberos authentication: very modest
    - no impact on bandwidth
    - 5-10% on metadata operations