GSS/Kerberos setup guide for Lustre for RHEL 6 distros

Author: Andrew Perepechko
2014-10-29

KDC setup

Assuming we chose LUSTRE as the Kerberos realm name.

1. *yum install krb5-server krb5-libs*
2. edit **/var/kerberos/krb5kdc/kdc.conf**, replace EXAMPLE.COM with the uppercase domain name, e.g. LUSTRE (this is called the realm)
3. edit **/etc/krb5.conf**, replace EXAMPLE.COM with the uppercase domain name, e.g. LUSTRE (the realm), set “kdc” and “admin_server” to the domain name of KDC, e.g. localhost.localdomain, add domain->realm translation to [domain_realm] section, e.g. “.localdomain = LUSTRE”, “localdomain = LUSTRE”.
4. run *kdb5_util create -s* to create the key database
5. edit **/var/kerberos/krb5kdc/kadm5.acl**, add “*/admin@LUSTRE *” to allow admin actions
6. *kadmin.local -q "addprinc root/admin"*
7. make sure krb5kdc will be started automatically, e.g. run *ntsysv* or *chkconfig krb5kdc on /sbin/service krb5kdc start && /sbin/service kadmin start*

It is important that every client and server should be able to translate each other’s addresses to host names. This is usually achieved by using a single domain name system, which can be replaced by filling out /etc/hosts and keeping them synchronous, but the latter solution is error-prone.

Firewall needs to be configured for communication through a few ports:

TCP/UDP Port 88 should be accessible in order to be able to reach KDC from network.
TCP/UDP Ports 464, 749 should be accessible in order to be able to reach kadmin from network.

Generic Kerberos client setup

1. *yum install krb5-libs krb5-workstation*
2. edit /etc/krb5.conf as described for the KDC setup (be sure to use the real domain name for kdc/admin_server, localhost will not work for a host which is not itself KDC!)

Assuming client host name is dac-1.

1. [on KDC] *kadmin.local -q “addprinc -randkey host/dac-1”*
2. [on client] kadmin> *ktadd -k /etc/krb5.keytab host/dac-1*

Testing generic Kerberos setup with ssh

This section is only relevant if you want to test basic (non-Lustre) Kerberos installation and setup.

Choose a client and a server. Assume dac-1 is the client and dac-2 is the server, LUSTRE is the realm.

1. add “*GSSAPIKeyExchange yes*” to /etc/ssh/ssh_config on the client
2. add “*GSSAPIAuthentication yes*” to /etc/ssh/sshd_config on the server
3. add “*GSSAPIKeyExchange yes*” to /etc/ssh/sshd_config on the server
4. add “*root/admin@LUSTRE*” to /root/.k5login on the server
5. execute “kinit root/admin@LUSTRE” from the client, enter password
6. *ssh root@dac-2* from the client

You should be able to log in without using a key or entering your password.

Building Lustre

In order to build Lustre, one needs to install libgssglue, libgssglue-devel, krb5-libs, krb5-devel and run *configure --enable-gss* for Lustre itself.

Lustre client Kerberos setup

Assuming client host name is dac-2. Run the following commands from the client:

1. kadmin> *addprinc -randkey lustre_root/dac-2@LUSTRE*
2. kadmin> *ktadd -k /etc/krb5.keytab lustre_root/dac-2@LUSTRE*
3. *echo "create lgssc * * /usr/sbin/lgss_keyring %o %k %t %d %c %u %g %T %P %S" >> /etc/request-key.conf*

One more step can be useful in order to run sanity-krb5.sh in the future:

1. [on KDC] kadmin.local: *addprinc sanityusr@LUSTRE*

Lustre server Kerberos setup

Assuming the Lustre server host name is dac-1. “lustre_mds” should be replaced by lustre_oss or lustre_mgs for OSS and MGS servers. Run the following commands from the respective server:

1. kadmin> *addprinc -randkey lustre_mds/dac-1@LUSTRE*
2. kadmin> *ktadd -k /etc/krb5.keytab lustre_mds/dac-1@LUSTRE*
3. *echo "create lgssc * * /usr/sbin/lgss_keyring %o %k %t %d %c %u %g %T %P %S" >> /etc/request-key.conf*
4. start */usr/sbin/lsvcgssd*, run *chkconfig lsvcgss on* to start lsvcgssd automatically on boot

Starting Lustre servers with Kerberos

In order to activate Kerberos, the following command should be ran on the MGS:

lctl conf_param $fsname.srpc.flavor.default=krb5p
lctl conf_param _mgs.srpc.flavor.default=krb5p

where fsname is the fsname for the Lustre cluster.

You can use the flavors from the following table (copied from the official Lustre manual):

Basic flavor	**Authentication**	**RPC protection**	**Bulk protection**	**Remarks**
null	N/A	N/A	N/A	No performance overhead.
plain	N/A	null	checksum (adler32)	Carries checksum (which only protects data mutating during transfer, cannot guarantee the genuine author because there is no actual authentication).
krb5n	krb5	null	checksum (adler32)	No protection of the RPC

				message, adler32 checksum protection of bulk data; light performance overhead.
krb5a	krb5	partial integrity	checksum (adler32)	Only the header of the RPC message is integrity protected, adler32 checksum protection of bulk data, more performance overhead compared to **krb5n**.
krb5i	krb5	integrity	integrity [sha1]	RPC message integrity protection algorithm is determined by actual Kerberos algorithms in use; heavy performance overhead.
krb5p	krb5	privacy	privacy [sha1/aes128]	RPC message privacy protection algorithm is determined by actual Kerberos algorithms in use; heaviest performance overhead.

Note only krb5p has been tested thorougly.

The changes will take place in a few seconds after the command. After the first kerberized communication, e.g. ping, you will see in the logs something like

Lustre: 3206:0:(sec_gss.c:2085:gss_svc_handle_init()) create svc ctx ffff880113da8e40: user from 0@lo authenticated as mds
Lustre: 3226:0:(sec_gss.c:399:gss_cli_ctx_uptodate()) client refreshed ctx ffff88010f3cd980 idx 0x6bf680d47128b0ea (0->lustre-MDT0000_UUID), expiry 1448199208(+69245s)
Lustre: 3225:0:(gss_svc_upcall.c:818:gss_svc_upcall_install_rvs_ctx()) create reverse svc ctx ffff8800b991ac40 to lustre-OST0000_UUID: idx 0x6bf680d47128b0e7

Starting Lustre clients with Kerberos

mount -t lustre 172.18.1.20@o2ib:/black2fs /mnt/testfs2 -o mgssec=krb5p

Testing Kerberized Lustre

Lustre source repository contains *tests/sanity-krb5.sh*. It can be started using usual acc-sm syntax. Additional GSS_USER and GSS_PASS parameters containing the kerberos user and password can be used for the tests:

GSS_USER=sanityusr GSS_PASS=sanityuserpass REFORMAT=--reformat bash sanity-krb5.sh