EUROPEAN MIDDLEWARE INITIATIVE

LCMAPS

Document version:	**1.0.0**
EMI Component Version:	**1.4.29**
Date:	**April 28, 2011**

This work is co-funded by the EC EMI project under the FP7 Collaborative Projects Grant Agreement Nr. INFSO-RI-261611.

1 Introduction

The LCMAPS framework is designed to take various credentials as input, e.g. a certificate and/or VOMS[1] credentials, and map them to Unix credentials as output. Unix credentials are the basic POSIX credentials, i.e. User ID, Group ID and Secondary Group IDs. LCMAPS is a framework that can load and run one or more 'credential mapping' plugins. The framework will load and run plugins to perform the identity mapping. Site and organizations can create their own new functionality by creating new plugins. The LCMAPS framework exposes various APIs to push credentials into the framework and to get the account mapping results in return. The lcmaps.db configuration file configures the LCMAPS plugins and configures the order in which the plugins are launch. Some practical examples are shown below.

LCMAPS is used by gLExec, the lcas-lcmaps-gt4-interface to interface with a Globus GT4 and GT5 Gatekeeper, GridFTP daemon and GSI-OpenSSHd, in StoRM and somewhere in Xrootd.

2 History

The Gridification subtask of WP4 of the European Datagrid project[2] interfaces the local fabric to other middleware components by a number of services, among which the Local Centre Authorization Service (LCAS) handles authorization requests to the local computing fabric and the Local Credential Mapping Service (LCMAPS) provides all local credentials needed for jobs allowed into the fabric. This document describes a prototype version of LCMAPS, which is the second component released by the Gridification subtask, the first being LCAS.

3 The frameworks inner working and plugins

When an application intializes LCMAPS the plugins will be loaded based on the lcmaps.db configuration file. The application can use one of the APIs to provide credentials as input. The loaded plugins will be executed in the sequence described in the same lcmaps.db configuration file.

During a plugin's execution it has access to the credential data in the LCMAPS core memory. The plugin is also capable of writing credential mapping results in LCMAPS. The plugins can each resolve a part of the mapping and they can also perform actions based on these (intermediate) results, e.g. run setuid, setgid and setgroup calls or interact with an LDAP service.

The plugins are executed in a state machine. When a plugin finishes succesfully it can execute a di erent next plugin then when it failed. This allows LCMAPS to pass di erent plugins to resolve a credential mapping.

[1] Virtual Organization Management System by INFN CNAF
[2] European DataGrid: http://www.eu-datagrid.org

4 Installation

The easiest way of installing LCMAPS is to use the EPEL, Debian, Etics or Nikhef software repositories[3]. The installation experience is similar when manually build from a tarbal with the command: `./configure && make && sudo make install`. When you've downloaded the software from our Subversion repository the `./bootstrap` command is mandatory.

The software will be installed in the following system default locations. The locations can be altered at ./configure time or the system might have di erent default location specified then Debian and/or Fedora systems:

/usr/lib{64}/ The core library files will be located here. This includes the front-end API implementing libraries.

/usr/lib{64}/modules/ This is the default path for all the LCMAPS plugins. The same directory is used to hold the LCMAPS plugins.

/etc/lcmaps/ The configuration file lcmaps.db will be placed in this directory by default.

/usr/share/doc/lcmaps-${version} The documentation and example files will be located here.

4.1 Dependencies

The dependencies of LCMAPS are:

vomsapi 1.6 or higher

Globus 2.x through Globus 5.x. Older version of Globus are known to work.

OpenSSL 0.9.7 through 1.0

4.2 Custom configure options

The LCMAPS configure script is trying a few methods on finding the depending libraries on the system, including pkg-config. To be able to work with non-system distribution provided installation or personal compiles explicitly we provide several configure options to work with:

–enable-headers This switch will only build and install or distribute the header files to LCMAPS. The header files will contain the required type definitions for both the front-end/application interface as also the plugin interface.

–enable-gsi-mode This option is on by default and will build LCMAPS against the GSI interface libraries and OpenSSL. Switch this option to no to create the lcmaps-without-gsi flavor of LCMAPS.

[3]Nikhef software repository: http://software.nikhef.nl/

–enable-osg This option is `o` by default. This option is used in the Open Science Grid build-flavor and will explicitly switch o the VOMS Attribute Certificate verification in the VOMS-api. This is to overload a default failure condition when the VOMS api could not verify the VOMS credentials. Note: Only enable this option if your infrastructure's security does not depend on the VOMS AC verification at all service nodes.

–with-globus-prefix=PFX Allows you to select an alternative location to find the Globus headers and libraries needed to build LCMAPS.

–with-globus-libdir=DIR Allows you to select an alternative directory for the Globus libraries. The default behaviour is to use the $PFX setting of the previous option and add /lib, e.g. $PFX/lib or $PFX/lib64

–with-globus-thr-flavor=FLAVOR Sets the threaded flavor of Globus. This is not needed in Globus 5 (and up) installation.

–with-globus-nothr-flavor=FLAVOR Sets the non-threaded flavor of Globus. This is not needed in Globus 5 (and up) installation.

–with-voms-prefix=PFX Sets the directory where LCMAPS should look for the VOMS api libraries. The default is in /usr

–with-voms-includes=DIR Override to the default $VOMS_PFX/include directory to look for the required VOMS header files.

–with-voms-libdir=DIR Override to the default $VOMS_PFX/lib or $VOMS_PFX/lib64 directories to look for the required VOMS libraries.

5 Common used paths

Here is a small list of commonly used files and paths used in the context of LCMAPS and its plugins. It's important to know that these paths are not created by LCMAPS.

/etc/grid-security/grid-mapfile DN-based or VOMS FQAN-based auhtorization and mapping file used by various plugins. It maps DNs and/or FQANs to user accounts.

/etc/grid-security/vomapfile Meant to be exclusively used as VOMS FQAN-based auhtorization and mapping file used by VOMS-specific various plugins. It maps FQANs to user accounts.

/etc/grid-security/groupmapfile VOMS FQAN-based auhtorization and mapping file used by various VOMS plugins. It maps (secondary) Unix groups.

/etc/grid-security/vomsdir/ The VOMS directory will hold the VOMS .lsc files and/or PEM files to authenticate the VOMS Attributes Certificates. Subdirectories are named by the VO name and scope the .lsc and PEM files in their authentication to one particular VO.

/etc/grid-security/certificates/ The Certificate Authority (CA) directory filled with the supported CAs. It also contains the Certificate Revocation List (CRL) files and (Subject DN) signing namespace files of the CAs.

/etc/grid-security/gridmapdir/ The directory where all the user account mappings are held made by LCMAPS plugins that support the mapping of pool accounts. It will make a hardlink between a file entry representing the (Unix) account and the (encoded representation of the) certificate subject DN. The link is persistent and should be seen as a database that exposes the link between the Grid identity and the Unix identity.

/etc/grid-security/groupmapdir/ A similarly purposed directory, explicitly used by a VOMS-aware plugin that maps FQANs to a pool of (Unix) groups.

6 Control through environment variables

The library can be steered using environment variables used in the running process. It's important to understand that plugins can use an additional set of environment variables for plugins specific purposes. Feature overloading the following set of environment variables is not adviced.

GATEKEEPER_JM_ID Extra Gatekeeper log message to be able to more easily track a Job Manager ID.

GLOBUSID See $GATEKEEPER_JM_ID.

JOB_REPOSITORY_ID See $GATEKEEPER_JM_ID, but explicitly for the purpose of the LCMAPS Job Repository plugin.

LCMAPS_DB_FILE Override the build-in default filename for the lcmaps.db configuration file with the value of this environment variable.

LCMAPS_DEBUG_LEVEL Tune the logging output verbosity. For near silence, set the value to '1'. For very verbose output, set the value to '5' (which is the maximum).

LCMAPS_DIR The base directory of the $LCMAPS_DB_FILE parameter. This variable is concatinated with the $LCMAPS_DB_FILE

LCMAPS_ETC_DIR See $LCMAPS_DIR

LCMAPS_LOG_FILE Overrides the build-in default file path to log the output to. When set, the logging will not go to Syslog.

LCMAPS_LOG_STRING Prepend all log output messages with value of this environment variable

LCMAPS_MODULES_DIR Directory to search for the LCMAPS plugins (or modules). Same as the path option in the lcmaps.db file..

LCMAPS_POLICY_NAME A colon separated list of LCMAPS plugin execution policies. When this environment variable is present, only the listed execution policies will be executed. They will be executed in the order as writen in the lcmaps.db file (from top to bottom).

LCMAPS_VERIFY_TYPE Depricated

LCMAPS_VOMS_EXTRACT Depricated

LCMAPS_X509_CERT_DIR Specific setting equal to the $X509_CERT_DIR environment variable

LCMAPS_X509_VOMS_DIR Specific setting equal to the $X509_VOMS_DIR environment variable

X509_CERT_DIR The directory where all the CA files, e.g. CA certificate and CRL files, are located. The default location is: /etc/grid-security/certificates/.

X509_VOMS_DIR This VOMS directory will hold the VOMS .lsc files and/or PEM files to authenticate the VOMS Attributes Certificates. Subdirectories are named by the VO name and scope the .lsc and PEM files in their authentication to one particular VO. The default location is: /etc/grid-security/vomsdir/.

7 Configuration

The default configuration file for LCMAPS is lcmaps.db. This file defines the plug-in configurations and policy definitions as used by tools like gLExec and a Globus GridFTPd[4]. As it is impossible to provide a default configuration that a) makes sense and b) doesn't require at least one plug-in that is not strictly required in every case. A few commonly used configration will be explained. The system administrator should pick and choose which parts are needed. The selected plug-ins are provided in separate packages.

For more information, see lcmaps.db(5) and the documentation for each individual plugin, or go to `http://www.nikhef.nl/pub/projects/grid/gridwiki/index.php/Site_Access_Control`

7.1 gLExec example

```
## Documentation example for gLExec
##
## The default path where modules are looked for. Do not change this.

path = /usr/lib64/modules

## ============================================================================
```

[4]The GridFTP daemon will need to be extended with the lcas-lcmaps-gt4-interface library

```
## Section: PLUG-INS
## ====================================================================

## Tracking group plug-in
##
## The Tracking Group ID plug-in can reattach secondary Group IDs from
## the calling user to the mapped user account. This is to ensure that
## special tracking group IDs, attached by the batch system to a running
## job, also remain when e.g. gLExec is switching the Unix/POSIX process
## ownership.
## The tracking group plug-in can either preserve a fixed range of Unix
## Group IDs or can auto-discover tracking GIDs. The auto-discover works
## by assuming tracking GIDs are 'nameless', but should be used with
## care.

# tracking_groupid = "lcmaps_tracking_groupid.mod"
#                        "--tracking-groupid-min MINGID"
#                        "--tracking-groupid-max MAXGID"

## verify-proxy
##
## The lcmaps_verify_proxy plugin verifies the validity of a proxy chain
## and (optionally) a valid delegation, including restrictions on the
## life time of any proxies in the chain.
## For more informaation about this plugin:
##    http://www.nikhef.nl/pub/projects/grid/gridwiki/index.php/Verify-proxy

## verify_proxy    = "lcmaps_verify_proxy.mod"
##                       " -certdir /etc/grid-security/certificates/"
##                       " --allow-limited-proxy"

## Posix enforcement plug-in
##
## The posix_enf plug-in takes the mapping as returned by earlier
## plug-ins and changes the user's identity according to the mapping,
## including any mapped secondary group ids. This is usually the last
## step in the chain.

# posix_enf        = "lcmaps_posix_enf.mod"
#                       "-maxuid 1"
#                       "-maxpgid 1"
#                       "-maxsgid 32"

##
## VOMS local group mapping
##
## The voms_localgroup plugin will map the VOMS FQANs (Fully Qualified
## Attribute Names) to one or more locally known groups by matching each
## FQAN to a line in the groupmapfile.
```

```
# vomslocalgroup   = "lcmaps_voms_localgroup.mod"
#                        "-groupmapfile /etc/grid-security/groupmapfile"
#                        "-mapmin 0"

## VOMS pool account mapping
##
## The voms_poolaccount plug-in maps FQANs to pool users, just like
## the plain poolaccount plug-in maps DNs.

# vomspoolaccount  = "lcmaps_voms_poolaccount.mod"
#                        "-gridmapfile /etc/grid-security/grid-mapfile"
#                        "-gridmapdir /etc/grid-security/gridmapdir"


## ============================================================================
## Section: POLICIES
## ============================================================================
##
## The policies are labeled, and gLExec will use the policy named in
## glexec.conf(5); if no policy is declared the first one will be used.

## One of the simplest policies is to verify the validity of the user
## proxy, and mapping the DN to a local account in
## /etc/grid-security/grid-mapfile.  This is the easiest setup to
## handle just a few users.  The verify_proxy and localaccount
## plug-ins in the plug-in section should be uncommented, and the
## lcmaps-plugins-basic package must be installed.

vomsawarelocalmapping:
verify_proxy -> vomslocalgroup
vomslocalgroup -> vomspoolaccount
vomspoolaccount -> tracking_groupid
tracking_groupid -> posix_enf | posix_enf
```

7.2 GridFTPd example

```
## Documentation example for gLExec
##
## The default path where modules are looked for. Do not change this.

path = /usr/lib64/modules

## ============================================================================
## Section: PLUG-INS
## ============================================================================

## Posix enforcement plug-in
##
## The posix_enf plug-in takes the mapping as returned by earlier
```

```
## plug-ins and changes the user's identity according to the mapping,
## including any mapped secondary group ids. This is usually the last
## step in the chain.

# posix_enf        = "lcmaps_posix_enf.mod"
#                       "-maxuid 1"
#                       "-maxpgid 1"
#                       "-maxsgid 32"

##
## VOMS local group mapping
##
## The voms_localgroup plugin will map the VOMS FQANs (Fully Qualified
## Attribute Names) to one or more locally known groups by matching each
## FQAN to a line in the groupmapfile.

# vomslocalgroup   = "lcmaps_voms_localgroup.mod"
#                       "-groupmapfile /etc/grid-security/groupmapfile"
#                       "-mapmin 0"

## VOMS pool account mapping
##
## The voms_poolaccount plug-in maps FQANs to pool users, just like
## the plain poolaccount plug-in maps DNs.

# vomspoolaccount  = "lcmaps_voms_poolaccount.mod"
#                       "-gridmapfile /etc/grid-security/grid-mapfile"
#                       "-gridmapdir /etc/grid-security/gridmapdir"


## ============================================================================
## Section: POLICIES
## ============================================================================
##
## The policies are labeled, and vomsawarelocalmapping is the first and
## only label

vomsawarelocalmapping:
vomslocalgroup -> vomspoolaccount
vomspoolaccount -> posix_enf
```

8 Debugging LCMAPS

LCMAPS will typically be logging in the same location as the service or tool that is initiating the LCMAPS library and its plugins. The plugin log records are embedded in the LCMAPS main framework logrecords.

To start debugging a problem in LCMAPS, we recommend to look at the

log file and increase the logging verbosity ultimately to level "5" by using the LCMAPS_LOG_LEVEL. You can now see the plugins loading succesfully:

```
lcmaps_log_open(): setting debugging level to 5
lcmaps.mod-lcmaps_startPluginManager(): doing lcmaps_startEvaluationManager(/etc/lcmaps/lcmaps-gle
Checking policy 'localonly' for recursions.
No recursions were found.
[...]
lcmaps.mod-lcmaps_startPluginManager(): initializing plugin /usr/lib64/modules/lcmaps_verify_proxy
lcmaps.mod-PluginInit(): found "plugin_initialize()"
lcmaps.mod-PluginInit(): found "plugin_run()"
lcmaps.mod-PluginInit(): found "plugin_terminate()"
lcmaps.mod-PluginInit(): found "plugin_introspect()"
lcmaps.mod-PluginInit(): found "plugin_verify()"
lcmaps.mod-PluginInit(): creating first pluginlist entry
[...]
```

The start of an mapping request initiated using the LCMAPS PEM and return account API. When available the LCMAPS framework will also pre-load its memory with VOMS credentials, of which a summary is given here. Note: This feature depricates the LCMAPS VOMS extract plugin:

```
Using "lcmaps_run_with_pem_and_return_account" interface of LCMAPS
Got individual certificate with subject: /O=dutchgrid/O=users/O=nikhef/CN=Oscar Koeroo/CN=proxy
 lcmaps_x509_to_voms_fqans(): voms data structure initialized
 lcmaps_x509_to_voms_fqans(): setting voms data for VO == dteam
 lcmaps_x509_to_voms_fqans(): setting voms data for VO server == /C=GR/O=HellasGrid/OU=hellasgrid.
 lcmaps_x509_to_voms_fqans(): TYP_STD
lcmaps_get_attributes: found 0 generic attributes.
 lcmaps_x509_to_voms_fqans(): extracted '0' generic voms attributes
 lcmaps_x509_to_voms_fqans(): Success, VOMS destroy
 print_lcmaps_vomsdata(): lcmaps_vomsdata->voms[ 1 / 1 ]
 print_lcmaps_vomsdata(): lcmaps_vomsdata->voms[1].user_dn           : /O=dutchgrid/O=users/O=nikh
 print_lcmaps_vomsdata(): lcmaps_vomsdata->voms[1].user_ca           : /C=NL/O=NIKHEF/CN=NIKHEF me
 print_lcmaps_vomsdata(): lcmaps_vomsdata->voms[1].voms_issuer_dn    : /C=GR/O=HellasGrid/OU=hella
 print_lcmaps_vomsdata(): lcmaps_vomsdata->voms[1].voms_issuer_ca    : /C=GR/O=HellasGrid/OU=Certi
 print_lcmaps_vomsdata(): lcmaps_vomsdata->voms[1].uri               : voms.hellasgrid.gr:15004
 print_lcmaps_vomsdata(): lcmaps_vomsdata->voms[1].date1             : 20110421100431Z
 print_lcmaps_vomsdata(): lcmaps_vomsdata->voms[1].date2             : 20110422100431Z
 print_lcmaps_vomsdata(): lcmaps_vomsdata->voms[1].voname            : dteam
 print_lcmaps_vomsdata(): lcmaps_vomsdata->voms[1].fqan_unix[ 1 / 1 ]
 print_lcmaps_vomsdata(): lcmaps_vomsdata->voms[1].fqan_unix[1].fqan : /dteam/Role=NULL/Capability=
 print_lcmaps_vomsdata(): lcmaps_vomsdata->voms[1].fqan_unix[1].uid  : -1
 print_lcmaps_vomsdata(): lcmaps_vomsdata->voms[1].fqan_unix[1].gid  : -1
lcmaps.mod-lcmaps_credential_store_x509(): Found LCMAPS vomsdata structure, adding it to the lcmap
[...]
```

Each of the plugins will be loaded in the sequence described in the configuration section. Here is an example of the LCMAPS Verify Proxy plugin.

```
lcmaps.mod-lcmaps_runPluginManager(): Do lcmaps_runEvaluationManager()
```

```
evaluationmanager: found plugin: /usr/lib64/modules/lcmaps_verify_proxy.mod
evaluationmanager: Resetting credential data.
lcmaps.mod-lcmaps_resetCredentialData(): Called
lcmaps.mod-lcmaps_runPlugin(): looking for plugin /usr/lib64/modules/lcmaps_verify_proxy.mod
lcmaps.mod-lcmaps_runPlugin(): found plugin /usr/lib64/modules/lcmaps_verify_proxy.mod
lcmaps.mod-lcmaps_runPlugin(): running plugin /usr/lib64/modules/lcmaps_verify_proxy.mod
 lcmaps_plugin_verify_proxy-plugin_run():
 lcmaps_plugin_verify_proxy-plugin_run(): found X.509 chain.
 lcmaps_plugin_verify_proxy-plugin_run(): found PEM string
 lcmaps_plugin_verify_proxy-plugin_run(): found lcmaps voms_data_list placeholder.
 lcmaps_plugin_verify_proxy-plugin_run(): vomsdata->nvoms = 1
 lcmaps_plugin_verify_proxy-plugin_run(): vomsdata->voms = 19f9f500
 lcmaps_plugin_verify_proxy-plugin_run(): vomsdata->voms[0] = 0
 lcmaps_plugin_verify_proxy-plugin_run(): vomsdata->voms[0].userdn = /O=dutchgrid/O=users/O=nikhef
Debug:   --- Reading the Private Key From PEM ---
Info:    Reading PEM string
Debug:   Reading Private key
Debug:   --- Welcome to the grid_verifyCert function ---
Info:    Using CA Directory: /etc/grid-security/certificates/
[...]
```

At the end of each plugin execution the LCMAPS framework will log if the module ended succesfully or not. This is important to understand the execution/control flow of the plugins by LCMAPS.

```
[...]
lcmaps_runEvaluationManager: running plugin: /usr/lib64/modules/lcmaps_verify_proxy.mod.
                    : result true.
```

At the end of the exeuction of the LCMAPS framework a summary will be printed. The following example didn't map the user based on the VOMS FQANs but only on the DN to a `uid` (Unix User ID) and `pgid` (Unix primary Group ID):

```
LCMAPS CRED FINAL: DN:"/O=dutchgrid/O=users/O=nikhef/CN=Oscar Koeroo"->mapped uid:'539',pgid:'100'
lcmaps.mod-lcmaps_run_with_pem_and_return_account(): LCMAPS found no secondary groups
```

NAME

lcmaps-plugins-c-pep - LCMAPS plug-in PEP-C Policy Enforcement Point Daemon client

SYNOPSIS

lcmaps_c_pep.mod *--pep-daemon-endpoint-url <url>* [*--resourcetype rb|ce|se|wn*] [*--actiontype queue|execute-now|access*] [*--pep-c-debug*] [*--[no]-check-certificates*] [*--capath <dir>*] [*--cafile <path>*] [*--cert <path>*] [*--key <path>*] [*--passfile <path>*] [*--pass <plaintextpassword>*] [*--resourceid <uri>*] [*--actionid <uri>*] [*--profile <profile name>*]

DESCRIPTION

The LCMAPS plug-in **lcmaps_c_pep** utilizes the PEP-C library to contact the PEP daemon. It will send the user credentials and, if applicable, the pilot job credentials and extra information to the PEP daemon.

The PEP daemon will process the request and query the PDP, PAP, EES chain for a policy decision. The PEP daemon will return a Permit statement with a Unix account. The Unix account must be composed of a Unix User ID and Unix Group ID. Optionally Unix Secondary GIDs may be returned. All of these IDs must be returned in numerical form. The results will then be published in the LCMAPS framework.

The plug-in will use the credentials loaded in the LCMAPS framework for the primary authorization decision. The returned Unix account will reflect this identity. Additionally to this identity, in a multi-user pilot job scenario, the X509_USER_PROXY environment variable is read to add information about the identity that executes the pilot job framework and triggered the execution of this plug-in. This probably with the use of gLExec.

OPTIONS

--actiontype queue|execute-now|access

The action type option will declare the type of action that is intended to be performed. The **queue** option signifies an execution to a queue. That's mostly due to a submission of a computer job to a queue. The **execute-now** option signifies the direct execution of a command or a job. Use cases for this action are the LCG-CE's fork-queue or gLExec where there is no (significant) delay for the operation's execution.. The **access** option signifies the access of a file at a storage facility of any kind. The true type of (file) access, like reading, writing, execution or listing, is not declared because this would be too detailed and poses practical limitation in the interaction with different storage system.

--pep-daemon-endpoint-url <url>

This will configure the plugin to contact the PEP daemon at <url>. Multiple endpoints can be set in the order specified by the **--endpoint-strategy** option.

--resourcetype rb|ce|se|wn

The resource type option will identify this PEP by its type of resource. The possible types that can be signified are rb, ce, se and wn. The **rb** option is to signify a Resource Broker or Workload Managment System or an differently named high level scheduler. The **ce** is to signify a Computing Element as a front-end node to a compute cluster like a LCG-CE or CREAM-CE. The **se** option is to signify a Storage Element, like DPM, dCache, Castor, StoRM or something else. The **wn** is to signify a Worker Node, like an LCG-WN, a compute node with gLExec on it for example.

--pep-c-debug

This option will trigger the PEP-C library to log at the maximum verbosity level and will let output detailed message, like HTTP protocol interactions with the PEP daemon, to be written to stderr. Without this option the detailed information is directed to /dev/null. Most error messages can be retrieved in sufficient detail from the PEP-C library in a different way. If you enable this option, the option_client_keypassword will be printed in clear-text to the invoker on stderr!

--[no]-check-certificates
This option will trigger the PEP-C library to disable or enable SSL validaiton on the connection to the PEPd server. By default, SSL validaiton is enabled. Using --no-check-certificates will ignore any SSL validation errors on the PEPd interconnect, but does not influence any checks on the invoking user or the target proxy.

--capath <directory>
Path to a directory containing the trust anchors. Each trust anchor must be in PEM format, and should be named according to the OpenSSL c_hash convention. The default will be one of $X509_CERT_DIR, $HOME/.globus/certificates, or /etc/grid-security/certificates (in that order).

--cafile <file>
Path to a file containing the trust anchors.

--cert <file>
Specify the file that contains the client certificate used to connect to the PEPd. When this option is set, client authentication is implicitly enabled, and a corresponding key file (--key option) must be provided.

--key <file>
Specify the file that contains the client private key used to connect to the PEPd. When this option is set, client authentication is impli citly enabled, and a corresponding certificate chain (--cert option) must be provided.

--passfile <file>
Specify the file that contains the password used to decrypt the private key for the PEPd handshake. This option is mutually exclusive with the --pass option.

--pass <password>
Specify the password in clear-text that will be used to decrypt the private key for the PEPd handshake. This option is mutually exclusive with the --passfile option. Using a password file (with appropriate permissions) is preferred.

--resourceid <urn/url>
This option will set the resourceID in the request protocol message.

--actionid <urn/url>
This option will set the actionID value in the request protocol message.

--profile <profile name>
The name of the profile that the plug-in must use. Currently supported Profiles names are "http://glite.org/xacml/profile/grid-wn/1.0" (default) and "http://authz-interop.org/profile/1.1". For more information, see the SUPPORTED PROFILES section.
The profile setting of "http://glite.org/xacml/profile/grid-wn/1.0" (default) requires the installation and availability of the libpepc.so library version 1.3.0-1 or newer.

ENVIRONMENT

X509_USER_PROXY
The value of the X509_USER_PROXY environment holds the path to the proxy certificate. This is not the primairy identity on which the authorization decision is based on. This proxy certificate identifies the Pilot Job executor. This identity is responsable for pulling a pilot job payload associated with a proxy onto a Worker Node during a job execution.

NOTES

When an https end point is used for the PEP daemon, client-side authentication can be enabled by specifying a file with a certificate chain and a file with the associated private key. If an https end point is specified but no certificate or key is provided, an anonymous secure connection is established. The server identity is always verified using the trust anchor repository specified by the --capath or --cafile options. If neither of

these is specified, the plugin will use the directory referred to by the X509_CERT_DIR environment variable, or fall back to /etc/grid-security/certificates. If neither directory can be found, the system default trust anchor store is used.

SUPPORTED PROFILES

There are different profiles to which the plug-in can adhere. Each profile describes the way in which the XACML attributes and obligations are to be used and what each of their intentions are. In the previous versions (all versions before 1.0.1-1) of this plug-in only the attributes and obligations mentioned in the "An XACML Attribute and Obligation Profile for Authorization Interoperability in Grids" document, version 1.1 (October 09, 2008) were used. Since January 13th the first draft of the "XACML Grid Worker Node Authorization Profile, Version 1.0" has been submitted for review. As there is a strong push to support the attributes and obligations stated in the Grid WN profile this profile is the new default.

NAMESPACE USAGE

The namespace usage complies with "An XACML Attribute and Obligation Profile for Authorization Interoperability in Grids" document, version 1.1 (October 09, 2008).

EDMS : https://edms.cern.ch/document/929867
FNAL CD : http://cd-docdb.fnal.gov/cgi-bin/ShowDocument?docid=2952

The Subject section of the request message will use the namespace "http://authz-interop.org/xacml/subject/cert-chain" to set the proxy certificate of the (real) user (job) credentials. The value is a base64 encoded string of the proxy certificate.

The action-id namespace is by default: "http://authz-interop.org/xacml/action/action-id".

It's value is set by the parameter value provided through the **--actionid** option, which can be anything as we do not check if the provided string value is a proper URL/URN. The option **--actiontype** also sets the action-id namespace, but the provided values **access-file**, **queue** and **execute-now** are expanded internally to the following namespace values:
http://authz-interop.org/xacml/action/action-type/access-file
http://authz-interop.org/xacml/action/action-type/queue
http://authz-interop.org/xacml/action/action-type/execute-now
When both the **--actionid** and **--actiontype** options are set, the value of the **--actionid** will take presidence.

The resource-id namespace is by default: "http://authz-interop.org/xacml/resource/resource-id".

It's value is set by the parameter value provided through the **--resourceid** option, which can be anything as we do not check if the provided string value is a proper URL/URN. The option **--resourcetype** also sets the resource-id namespace, but the provided values **ce**, **se**, **wn** and **rb** are expanded internally to the following namespace values:
http://authz-interop.org/xacml/resource/resource-type/se
http://authz-interop.org/xacml/resource/resource-type/ce
http://authz-interop.org/xacml/resource/resource-type/wn
http://authz-interop.org/xacml/resource/resource-type/rb
When both the **--resourceid** and **--resourcetype** options are set, the value of the **--resourceid** will take presidence.

The dns-name of the host is set in the Resource section of a request and set in the namespace "http://authz-interop.org/xacml/resource/dns-host-name". The value is the FQDN of the resource which sends out the request.

The Environment section of the request message will use the namespace "http://authz-interop.org/xacml/subject/cert-chain" to set the proxy certificate of the pilot job executor's credentials. The value is a base64 encoded string of the proxy certificate.

The registered Obligation handlers are trigger by the namespaced IDs:

http://authz-interop.org/xacml/obligation/uidgid
http://authz-interop.org/xacml/obligation/secondary-gids
http://glite.org/xacml/obligation/local-environment-map/posix

The obligation identified by "http://authz-interop.org/xacml/obligation/uidgid" can only be set once in a Response message. The obligation handler will trigger an error state when this obligation is stated multiple times in a response message. The obligation requires that two attributes are set. These attributes must provide the Unix UserID and Unix Primary Group ID in numerical form. The attribute with the namespace "http://authz-interop.org/xacml/attribute/posix-uid" is expected to be filled with an integer value of the Unix User ID which must exist on the system on which the Response message is going to be processed. The attribute with the namespace "http://authz-interop.org/xacml/attribute/posix-gid" is expected to be filled with an integer value of the Unix Group ID which must exist on the system on which the Response message is going to be processed.

The obligation identified by "http://authz-interop.org/xacml/obligation/secondary-gids" can be set zero or multiple times in a Response message. The obligation requires that at least one or more attribute are set. These attributes must provide the Unix Group ID in numerical form. This may be the primary Unix Group ID, but it is intended to only be filled with the secondary Unix Group IDs. The attribute with the namespace "http://authz-interop.org/xacml/attribute/posix-gid" is expected to be filled with an integer value of the Unix Group ID which must exist on the system on which the Response message is going to be processed. The obligation requires that at least one or more attribute are set. These attributes must provide the Unix Group ID in numerical form. This may be the primary Unix Group ID, but it is intended to only be filled with the secondary Unix Group IDs. The attribute with the namespace "http://authz-interop.org/xacml/attribute/posix-gid" is expected to be filled with an integer value of the Unix Group ID which must exist on the system on which the Response message is going to be processed.

The obligation identified by "http://authz-interop.org/xacml/obligation/username" can only be set once in a Response message. The obligation handler will trigger an error state when this obligation is stated multiple times in a response message. The obligation requires that one attribute is set. These attributes must provide the Unix Username in string form. The attribute with the namespace "http://authz-interop.org/xacml/attribute/username" is expected to be filled with an string value of the Unix Username which must exist on the system on which the Response message is going to be processed. The attribute declaring the username will result in system lookup to acquire the Unix User-ID, Unix primary Group-ID and the list of secondary Group-IDs of which the username is a member. This information will then be stored for further processing in LCMAPS.

The obligation identified by "http://glite.org/xacml/obligation/local-environment-map/posix" can be set once Response message. The obligation handler will trigger an error state when this obligation is stated multiple times in a response message. The obligation requires that two attributes are set, optionally more. The two required attributes are identified with "http://glite.org/xacml/attribute/user-id" and "http://glite.org/xacml/attribute/group-id/primary". They state the name of the User account and the name of the primary Group on the system. Ootionally a list of "http://glite.org/xacml/attribute/group-id" attributes can be set which state the secondary Group affifiliation on the system. These names are resolved in the obligation handlers to the actual Unix User ID and Unix Group IDs and registered in the LCMAPS system for

enforcement.

A source code rebuild with the define 'USE_STANDARDIZED_NAMESPACE' will force the default namespace usage to for the resource-id to "urn:oasis:names:tc:xacml:1.0:resource:resource-id" and for the action-id to "urn:oasis:names:tc:xacml:1.0:action:action-id". As this is not used by the interoperability group as the default, and would thus break interoperability, we've choosen not to promote these settings.

EXAMPLE

The following example config file can be used for LCMAPS:

```
### gLExec on the WN for the AuthZ WG PEP-C to PEP-D interaction only
# default path for the modules
path = /usr/lib64/modules

# Plugin definitions:
posix_enf = "lcmaps_posix_enf.mod"
          "-maxuid 1"
          "-maxpgid 1"
          "-maxsgid 32"

verifyproxy = "lcmaps_verify_proxy.mod"
            "-certdir /etc/grid-security/certificates"

pepc = "lcmaps_c_pep.mod"
     "--pep-daemon-endpoint-url http://localhost:8080/PEPd/authz"
     "--resourceid http://cnaf.infn.it/wn"
     "--actionid http://glite.org/xacml/action/execute"
     "--capath /etc/grid-security/certificates"
     "--pep-certificate-mode implicit"


# Policies:
# AuthZ WG PEP-C PEP-Daemon interaction
glexec_get_account:
verifyproxy ->  pepc
pepc -> posix_enf
```

BUGS

There are no bugs found (yet).

FILES

/etc/lcmaps/lcmaps.db
/usr/lib64/modules/lcmaps_c_pep.mod
/usr/lib64/modules/liblcmaps_c_pep.a
/usr/lib64/modules/liblcmaps_c_pep.so
/usr/lib64/modules/liblcmaps_c_pep.so.0
/usr/lib64/modules/liblcmaps_c_pep.so.0.0.0

SEE ALSO

lcmaps(3), **glexec**(1)

AUTHOR

Writen by Oscar Koeroo

COPYRIGHT

lcmaps_tracking_groupid.mod – LCMAPS plugin to add tracking groupids

The Tracking Group ID plugin will add one or more secondary Group IDs to the final mapping result. This is too ensure that the batch system's specially issues secondary GIDs are kept, even if a process like gLExec is switching the Unix/POSIX process ownership. This feature is required to ensure that stray processes from users are cleaned up from a cluster's Worker Node when the job is done.

The plugin can preserve a fixed range of Unix Group IDs or the plugin is able to auto-discover the tracking GIDs. The automatic discovery should be carefully used. It could for instance preserve AFS specific secondary Group ID. The auto discovery mechanism will lookup all secondary Group IDs of the running process and will preserve the Group IDs which are left nameless. This might come in handy, but this doesn't work if system administrators (for good reasons) assign group names to the tracking Group IDs.

Declare the lowest numerical representation of a secondary Group ID used as a Tracking Group ID. Note: the number specified is included in the Group ID match with an equal or higher statement. Declare the highest numerical representation of a secondary Group ID used as a Tracking Group ID. Note: the number specified is included in the Group ID match with an equal or lower statement.

Success. Failure.

Please report any errors to the Nikhef Grid Middleware Security Team <grid-mw-security-support@nikhef.nl>. LCMAPS and the LCMAPS plug-ins were written by the Grid Middleware Security Team <grid-mw-security@nikhef.nl>.

NAME

lcmaps_dummy_bad.mod – dummy LCMAPS plugin that always returns failure

SYNOPSIS

lcmaps_dummy_bad.mod

DESCRIPTION

LCMAPS plug-in that always fails. This plugin should be configured in the **lcmaps.db**(5) file, following the specific syntax as described in the documentation.

OPTIONS

There are no options.

RETURN VALUE

This plug-in always returns **LCMAPS_MOD_FAIL**.

SEE ALSO

lcmaps(3), **lcmaps.db**(5). More information can be found on-line at the Nikhef Wiki on Site Access Control

BUGS

Failure is not an option.

Please report any errors to the Nikhef Grid Middleware Security Team <grid-mw-security-support@nikhef.nl>.

AUTHOR

LCMAPS and the LCMAPS plug-ins were written by the Grid Middleware Security Team <grid-mw-security@nikhef.nl>.

NAME

lcmaps_dummy_good.mod – dummy LCMAPS plugin that always returns success

SYNOPSIS

lcmaps_dummy_good.mod [{**--dummy-uid** *uid*|**--dummy-username** *user*}] [**--dummy-gid** *gid*] [**--dummy-sec-gid** *gid*]

DESCRIPTION

LCMAPS plug-in that always returns successfully, optionally with a user and group mapping.

OPTIONS

This plugin is configured in the **lcmaps.db**(5) file, following the specific syntax as described in the documentation.

--dummy-uid *uid*

--dummy-username *user* **--dummy-gid** *gid* **--dummy-sec-gid** *gid* The user id, user name, group id, secondary group id and to return in a successful mapping. The given user and group(s) must exist in the system. The default is to return no user or group information.

RETURN VALUE

This plugin always returns LCMAPS_MOD_SUCCESS, but if the given user or group was not found on the system, the plugin returns LCMAPS_MOD_FAIL.

BUGS

Please report any errors to the Nikhef Grid Middleware Security Team <grid-mw-security-support@nikhef.nl>.

SEE ALSO

lcmaps(3), **lcmaps.db**(5).
More information can be found on-line at the Nikhef Wiki on Site Access Control

AUTHOR

LCMAPS and the LCMAPS plug-ins were written by the Grid Middleware Security Team <grid-mw-security@nikhef.nl>.

NAME

lcmaps_ldap_enf.mod – LCMAPS plugin to update ldap according to credentials

SYNOPSIS

lcmaps_ldap_enf.mod **-maxuid** *maxuid* **-maxpgid** *maxpgid* **-maxsgid** *maxsgid* **-hostname** *hostname* **-port** *port* [**-require_all_groups** {**yes**|**no**}] **-dn_manager** *DN* **-ldap_pw** *filename* **-sb_groups** *searchbase* **-sb_user** *searchbase* **-timeout** *seconds*

DESCRIPTION

Ldap enforcement plugin will alter the user and group settings in the ldap database, using the user and groups settings provided by the credential acquisition plugins. Note that LDAP has to be used as the source of account information for PAM or NSS and has to be RFC 2307 compliant.

OPTIONS

-maxuid *maxuid*
Maximum number of uids to be used. Strongly advised is to set this to 1.

-maxpgid *maxpgid*
Maximum number of primary gids to be used.

-maxsgid *maxsgid*
Maximum number of (secondary) gids to be used (not including primary group). Advised is to set this to 1.

-hostname *hostname*
The hostname on which the LDAP server is running, e.g. asen.nikhef.nl

-port *port*
The port number to which to connect, e.g. 389

-require_all_groups {**yes**|**no**}
Specify if all groups set by the PluginManager shall be used. Default is 'yes'.

-dn_manager *DN*
DN of the LDAP manager, e.g. "cn=Manager,dc=root"

-ldap_pw *filename*
Path to the file containing the password of the LDAP manager. Note: the mode of the file containing the password must be read-only for root (400), otherwise the plugin will not run.

-sb_groups *searchbase*
Search base for the (secondary) groups, e.g. "ou=LocalGroups, dc=example, dc=com"

-sb_user *searchbase*
Search base for the user, e.g. "ou=LocalUsers, dc=example, dc=com"

-timeout *timeout value*
timeout (in seconds) that will be applied to the ldap binding

RETURN VALUE

LCMAPS_MOD_SUCCESS
Success.

LCMAPS_MOD_FAIL
Failure.

BUGS

Please report any errors to the Nikhef Grid Middleware Security Team <grid-mw-security-support@nikhef.nl>.

SEE ALSO

lcmaps.db(5), **lcmaps**(3), **ldap**(3).

AUTHORS

LCMAPS and the LCMAPS plug-ins were written by the Grid Middleware Security Team <grid-mw-security@nikhef.nl>.

NAME

lcmaps_localaccount.mod – LCMAPS plugin to switch user identity

SYNOPSIS

lcmaps_localaccount.mod [**-gridmapfile** *gridmapfile*]

DESCRIPTION

This plugin is an Acquisition Plugin and will provide the LCMAPS system with Local Account credential information. To do this it needs to look up the Distinghuished Name (DN) from a user's certificate in the gridmapfile. If this DN is found in the gridmapfile the plugin knows the mapped local (system) account username. By knowing the username of the local account the plugin can gather additional information about this account. The plugin will resolve the UID, GID and all the secondary GIDs. When this has been done and there weren't any problems detected, the plugin will add this information to a datastructure in the Plugin Manager. The plugin will finish its run with a **LCMAPS_MOD_SUCCESS**. This result will be reported to the Plugin Manager which started this plugin and it will forward this result to the Evaluation Manager, which will take appropriate actions for the next plugin to run. Normally this plugin would be followed by an Enforcement plugin that can apply these gathered credentials in a way that is appropriate to a system administration's needs.

OPTIONS

-gridmapfile *gridmapfile*

When this option is set it will override the default path of the gridmapfile. It is advised to use an absolute path to the gridmapfile to avoid usage of the wrong file(path).

RETURN VALUES

LCMAPS_MOD_SUCCESS

Success.

LCMAPS_MOD_FAIL

Failure.

BUGS

Please report any errors to the Nikhef Grid Middleware Security Team <grid-mw-security-support@nikhef.nl>.

SEE ALSO

lcmaps.db(5), **lcmaps**(3).

AUTHORS

LCMAPS and the LCMAPS plug-ins were written by the Grid Middleware Security Team <grid-mw-security@nikhef.nl>.

NAME

lcmaps-plugins-scas-client - LCMAPS plug-in Site Central Authorization Service

The LCMAPS plug-in SCAS Client utilizes the SAML2-XACML2 protocol to contact the SCAS daemon. It will send the user credentials, (if applicable) the pilot job credentials and extra information to the SCAS service.

The SCAS service will process the request and provide a Unix account in return. The Unix account must be composed of a Unix User ID and Unix Group ID. Optionally Unix Secondary GIDs may be returned. All of these IDs must be returned in numerical form. The results will then be published in the LCMAPS framework.

OPTIONS

-actiontype queue|execute-now|access

The action type option will declare the type of action that is intended to be performed. The **queue** option signifies an execution to a queue. That's mostly due to a submission of a computer job to a queue. The **execute-now** option signifies the direct execution of a command or a job. Use cases for this action are the LCG-CE's fork-queue or gLExec where there is no (significant) delay for the operation's execution.. The **access** option signifies the access of a file at a storage facility of any kind. The true type of (file) access, like reading, writing, execution or listing, is not declared because this would be too detailed and poses practical limitation in the interaction with different storage system.

--authorization-only

This will make the returned Obligations optional. The plug-in will normally expect Obligations in the server's Response message, this setting will disable the requirement in a positive response from the service.

--capath <CA and CRL path>

Directory where the CA and CRL files are to be found. This is needed to verify the SCAS service credentials when contacting the service.

--cert-owner <certifcate owner>

This is an optional parameter, mostly interesting for sites that use gLExec that use a host certificates to authenticate the SSL connection to GUMS, SCAS or something else. The SCAS Client will switch the process Unix Identity to the <certifcate owner> Unix account (only its UID) with the purpose to read in the certificate with the <certifcate owner> ID, and switch back to root. Bewarned that you take care of possible root-squash issues that might arise here. We recommend to create a special account (besides the special glexec account) to read in the host credentials. The root account is a valid option, but might be root-squashed if the host certificate is to be read from an NFS mount with that option enabled.

--cert <path to certificate file>

Public certificate file belonging to the service or host to identify itself with in the SSL handshake when connecting to the SCAS service.

--enable-poolindex-fix

Add the faked poolindex to the LCMAPS framework. This is to cope with LCMAPS 1.4.6 and older which expects a poolindex string to be set. A poolindex is typically set when the poolaccount or VOMS poolaccount plug-in is successfully executed.

--endpoint <url>

This will configure the plugin to contact the SCAS service at <url>.

--endpoint-strategy round-robin|round-robin-random-start|random

The endpoint strategy tells the client in which order the configured endpoints should be tried to be contacted. With **round-robin** the list of endpoints will be tried from top to bottom as written in the lcmaps.db file. The option **round-robin-random-start** will follow the list of endpoints as written

in the lcmaps.db file, but it will randomly start somewhere in the list of endpoints. The **random** option will randomly choose an endpoint to try. When unlucky the same endpoint could be tried twice. This is true pseudo-random. The **round-robin-random-start** is made default. This will automagically provide a load balancing effect by randomly selecting a configured endpoints.

--key <path to private key file>
Private key file belonging to the certificate file for the SSL connection to the SCAS service.

--override-expected-hostname <hostname>
This option will override the expected hostname from the service it connects to. The service must present a valid host certificate, but during the validation of the hostname and the certificate the set <hostname> string will be used to check the expected host to be OK to communicate with.

--resourcetype rb|ce|se|wn
The resource type option will identify this plugin by its type of resource. The possible types that can be signified are rb, ce, se and wn. The **rb** option is to signify a Resource Broker or Workload Managment System or an differently named high level scheduler. The **ce** is to signify a Computing Element as a front-end node to a compute cluster like a LCG-CE or CREAM-CE. The **se** option is to signify a Storage Element, like DPM, dCache, Castor, StoRM or something else. The **wn** is to signify a Worker Node, like an LCG-WN, a compute node with gLExec on it for example.

--retry <0-9+>
This will alter the retry count when interacting with an SCAS endpoint. By default each endpoint is tried twice by default before any other endpoint is tried (this excludes the various TCP/IP layer retries that are always performed at a lower level). This option can alter this default behavior. It can be set to any number as long as its more then '1'. Between two tries there is a small amount of time of delay build-in.

ENVIRONMENT

X509_USER_PROXY
This variable will be used for the SSL handshake to the SCAS service. In this case the user proxy certificate file (must include the private key) will be used to establish the SSL connection. In a gLExec-on-WNs scenario, this is the identity of the pilot job framework executor, not the real user job. This variable will be read in the plug-in run phase to trigger a proper authz failure in gLExec and when not used,the X509_USER_CERT, X509_USER_PROXY, --cert or --key must be set. If those are not properly setup a failure will occure in the initialization phase of the plug-in.

X509_USER_CERT
Same value as the option **--cert <path to certificate file>**

X509_USER_KEY
Same value as the option **--key <path to private key file>**

EXAMPLE

The following example config file can be used for LCMAPS:

```
### gLExec on the WN for the SCAS client to SCAS service interaction only
# default path for the modules
path = /usr/lib64/modules

# Plugin definitions:
posix_enf = "lcmaps_posix_enf.mod"
        "-maxuid 1"
        "-maxpgid 1"
        "-maxsgid 32"
verifyproxy = "lcmaps_verify_proxy.mod"
        "-certdir /etc/grid-security/certificates"
```

```
scasclient = "lcmaps_scas_client.mod"
        "-resourcetype wn"
        "-actiontype execute-now"
        "-capath /etc/grid-security/certificates"
#         " -cert   /etc/grid-security/hostcert.pem"
#         " -key    /etc/grid-security/hostkey.pem"
        "--endpoint https://eir.nikhef.nl:8443"
        "--endpoint https://grasmaaier.nikhef.nl:8443"
#         "--endpoint-strategy round-robin"
        "--endpoint-strategy round-robin-random-start"
#         "--endpoint-strategy random"
# Policies:
# SCAS
glexec_get_account:
verifyproxy -> scasclient
scasclient -> posix_enf
```

BUGS

None so far.

FILES

/etc/lcmaps/lcmaps.db
/usr/lib64/modules/lcmaps_scas_client.mod
/usr/lib64/modules/liblcmaps_scas_client.a
/usr/lib64/modules/liblcmaps_scas_client.so
/usr/lib64/modules/liblcmaps_scas_client.so.0
/usr/lib64/modules/liblcmaps_scas_client.so.0.0.0

SEE ALSO

lcmaps(3), **glexec**(1), **scas**(8), **scas.conf**(5)

AUTHOR

Writen by Oscar Koeroo

COPYRIGHT

NAME

lcmaps_poolaccount.mod – LCMAPS plugin to switch user identity by pool accounts

SYNOPSIS

lcmaps_poolaccount.mod [**-gridmapfile** *gridmapfile*] [**-gridmapdir** *gridmapdir*]
[**-override_inconsistency**] [**-max_mappings_per_credential** *max nr of mappings*]

DESCRIPTION

This plugin is a Acquisition Plugin and will provide the LCMAPS system with Pool Account information. To do this it needs to look up the Distinguished Name (DN) from a user's certificate in the gridmapfile. If this DN is found in the gridmapfile the plugin now knows to which pool of local system accounts the user wil be mapped. The poolname (starting with a dot ('.') instead of an alphanumeric character) will be converted into the an account from a list of local accounts. This list is located in the \ gridmapdir and is made out of filenames. These filenames correspond to the system poolaccount names. (E.g. if a DN corresponds to **.test** in the gridmapfile, it will be mapped to **test001**, **test002**, etc., which names can be found in the gridmapdir.

If there is no pool account assigned to the user yet, the plugin will get a directory listing of the gridmapdir. This list will contain usernames corresponding to system accounts specially designated for pool accounting. If the plugin resolved the mapping of a certain pool name, let's say '.test', the plugin will look into the directory list and will find the first available file in the list corresponding with 'test' (e.g. 'test001') by checking the number of links to its i-node. If this number is 1, this account is still available. To lease this account a second hard link is created, named after the URL-encoded, decapitalized DN.

When a user returns to this site the plugin will look for the DN of the user (URL encoded) in this directory. If found, the corresponding poolaccount will be assigned to the user.

The plugin will resolve the UID, GID and all the secondary GIDs belonging to the poolaccount. When this has been done and there weren't any problems detected, the plugin will add this information to a datastructure in the Plugin Manager. The plugin will finish its run with a **LCMAPS_MOD_SUCCESS**. This result will be reported to the Plugin Manager which started this plugin and it will forward this result to the Evaluation Manager, which will take appropriate actions for the next plugin to run. Normally this plugin would be followed by an Enforcement plugin that can apply these gathered credentials in a way that is appropriate to a system administration's needs.

OPTIONS

-gridmapfile *gridmapfile*

If this option is set, it will override the default path of the gridmapfile. It is advised to use an absolute path to the gridmapfile to avoid usage of the wrong file(path).

-gridmapdir *gridmapdir*

If this option is set, it will override the default path to the gridmapdir. It is advised to use an absolute path to the gridmapdir to avoid usage of the wrong path.

-override_inconsistency

Moving a user from one pool to another (because of a VO change) should only be done by changing the gridmapfile indicating the new pool for this user. If a user has already been mapped previously to a poolaccount, there is a link present between this poolaccount and his DN. In the good old days prior to LCMAPS, a 'pool change' would still result in a mapping to the old pool account, neglecting the administrative changes in the gridmapfile. LCMAPS corrects this behaviour: By default the poolaccount plugin will fail if the pool designated by the gridmapfile doesn't match the previously mapped poolaccount leasename. If the site doesn't want a failure on this inconsistency it can turn on this parameter. When the inconsistency is detected the plugin will automatically unlink the previous mapping and will proceed by making a new lease from the new pool.

-max_mappings_per_credential *max nr of mappings*

This value indicates the maximum number of accounts a user, or more specifically a set of credentials (=DN + FQANS), can be mapped to. Normally this number is 1. But if each job should run under its own account the number should be increased. The leasename (or poolindex) in this case looks like:

url_encoded(<DN>):mapcount=<mapnumber>)

-no_wildcard
If this option is set, wildcards cannot be used in the grid-mapfile (on by default)

-strict_poolprefix_match {**yes**|**no**}
Default is 'yes'. If this is set to 'yes', a line in the gridmapfile like <DN>.pool will result in accounts matching the regexp 'pool[0-9]+'. Otherwise it will be allowed to match 'pool.*' (legacy behaviour).

RETURN VALUES

LCMAPS_MOD_SUCCESS
Success.

LCMAPS_MOD_FAIL
Failure.

BUGS

Please report any errors to the Nikhef Grid Middleware Security Team <grid-mw-security-support@nikhef.nl>.

SEE ALSO

lcmaps.db(5), **lcmaps**(3).

AUTHORS

LCMAPS and the LCMAPS plug-ins were written by the Grid Middleware Security Team <grid-mw-security@nikhef.nl>.

NAME

lcmaps_posix_enf.mod – LCMAPS plugin to switch user identity

SYNOPSIS

lcmaps_posix_enf.mod [**-maxuid** *number of uids*] [**-maxpgid** *number of primary gids*] [**-maxsgid** *number of secondary gids*]

DESCRIPTION

The Posix Enforcement plugin will enforce (apply) the gathered credentials that are stacked in the datastructure of the Plugin Manager. The plugin will get the credential information that is gathered by one or more Acquisition plugins. This implies that at least one Acquisition should have been run prior to this Enforcement. All of the gathered information will be checked by looking into the 'passwd' file of the system (FIXME: shouldn't that be **getpwent**(2)?). These files have information about all registered system account and its user groups.

The Posix Enforcement plugin does not check whether the secondary groups have the primary UID as a member, so it is possible to end up with more group memberships than what is defined in the group database.

The (BSD/POSIX) functions **setreuid**(2), **setregid**(2) and **setgroups**(2) are used to change the privileges of the process from root to that of a local user.

OPTIONS

-maxuid *number of uids*

In principle, this will set the maximum number of allowed UIDs that this plugin will handle, but at the moment only the first UID found will be enforced; the others will discarded. By setting the value to a maximum there will be a failure raised when the amount of UIDs exceed the set maximum. Without this value the plugin will continue and will enforce only the first found value in the credential data structure.

-maxpgid *number of primary gids*

This will set the maximum number of allowed Primary GIDs that this plugin will handle, similar to **-maxuid**. Also here only the first primary GID found will be taken into account.

-maxsgid *number of secondary gids*

This will set the maximum allowed Secondary GIDs that this plugin will handle. This number is limited by the system (NGROUPS) and is usually 32. If the plugin cannot determine the system value, it limits itself to 32.

The remaining options are considered dangerous, as they have the potential to allow a client process to gain root privileges. **The use of these options is strongly discouraged.**

-set_only_euid {**yes**|**no**}

The result of setting this option to 'yes' is that only the effective uid is set. In other words, it is still possible to regain root (uid) privileges for the process. This is definitely undesirable if this module is used from a process like the gatekeeper, since it would be possible for user jobs to get root privileges.

-set_only_egid {**yes**|**no**}

Analogue to the previous option the result of setting this option to 'yes' is that only the effective (primary) gid is set. In other words, it is still possible to regain root (gid) privileges for the process. This is definitely undesirable if this module is used from a process like the gatekeeper, since it would be possible for user jobs to get root privileges. Possibly this option should be set if the module is used by gridFTP, since this service does not spawn user jobs and has to regain root pivileges at the end.

RETURN VALUES

LCMAPS_MOD_SUCCESS
Success.

LCMAPS_MOD_FAIL
Failure.

BUGS

Please report any errors to the Nikhef Grid Middleware Security Team <grid-mw-security-support@nikhef.nl>.

SEE ALSO

lcmaps.db(5), **lcmaps**(3), **getpwent**(3), **getgrent**(3), **setreuid**(2), **setregid**(2), **setgroups**(2).

AUTHORS

LCMAPS and the LCMAPS plug-ins were written by the Grid Middleware Security Team <grid-mw-security@nikhef.nl>.

NAME

lcmaps_tracking_groupid.mod – LCMAPS plugin to add tracking groupids

SYNOPSIS

lcmaps_tracking_groupid.mod [**--tracking-groupid-min** *number of secondary gid* --tracking-groupid-max *number of secondary gid*] [**--auto-discover**]

DESCRIPTION

The Tracking Group ID plugin will add one or more secondary Group IDs to the final mapping result. This is too ensure that the batch system's specially issues secondary GIDs are kept, even if a process like gLExec is switching the Unix/POSIX process ownership. This feature is required to ensure that stray processes from users are cleaned up from a cluster's Worker Node when the job is done.

The plugin can preserve a fixed range of Unix Group IDs or the plugin is able to auto-discover the tracking GIDs. The automatic discovery should be carefully used. It could for instance preserve AFS specific secondary Group ID. The auto discovery mechanism will lookup all secondary Group IDs of the running process and will preserve the Group IDs which are left nameless. This might come in handy, but this doesn't work if system administrators (for good reasons) assign group names to the tracking Group IDs.

OPTIONS

--tracking-groupid-min*number of gid*

Declare the lowest numerical representation of a secondary Group ID used as a Tracking Group ID. Note: the number specified is included in the Group ID match with an equal or higher statement.

--tracking-groupid-max*number of gid*

Declare the highest numerical representation of a secondary Group ID used as a Tracking Group ID. Note: the number specified is included in the Group ID match with an equal or lower statement.

RETURN VALUES

LCMAPS_MOD_SUCCESS

Success.

LCMAPS_MOD_FAIL

Failure.

BUGS

Please report any errors to the Nikhef Grid Middleware Security Team <grid-mw-security-support@nikhef.nl>.

SEE ALSO

lcmaps.db(5), **lcmaps**(3), **setgroups**(2).

AUTHORS

LCMAPS and the LCMAPS plug-ins were written by the Grid Middleware Security Team <grid-mw-security@nikhef.nl>.