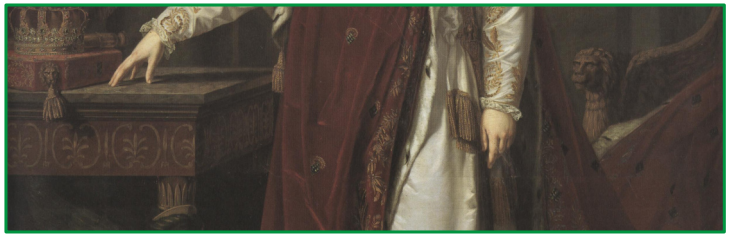
LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

GRIDKA SCHOOL 2014
KARLSRUHER INSTITUT FÜR TECHNOLOGIE
KARLSRUHE

September 4, 2014

# dCache Introduction Course

Christoph Anton Mitterer
christoph.anton.mitterer@lmu.de



# V. Access Control

# Access Control Systems In dCache

dCache contains several access control systems, used with different types of protocols and providing a varying set of features:

- gPlazma + door (NFS 4.1, WebDAV, FTP, DCAP, rootd and SRM)
  The NFS 4.1-, WebDAV-, FTP-, DCAP-, rootd- and SRM-doors implement their own (very similar) access control systems by using the services provided by gPlazma.
- xrootd-ALICE (rootd)
  This system implements the so called “ALICE security model” and is only used for authorisation by rootd-doors.
- NFS server export table `/etc/exports` (NFS 3 and 4.1)
  Controls which and how clients can access NFS exports.
- SpaceManager-LinkGroup-authorization-file (SRM)
  Controls which entities are allowed to perform space reservations.

*This course covers the access control systems based on gPlazma.*

## gPlazma

gPlazma (**G**rid-Aware **Pl**uggable **A**uthori**z**ation **Ma**nagement) is a service of dCache, providing functionalities for access control, which are used by doors in order to implement their respective access control system.

It should be noted, that not every type of door makes necessarily use of all the features provided by gPlazma.

It is used by NFS 4.1-, WebDAV-, FTP-, DCAP-, rootd- and SRM-doors.
To some extent it is further used by the `admin`- and `httpd`- services.

In order to serve different needs, gPlazma utilises plug-ins as back-end for its tasks and services.

*This course gives only an overview of the plug-in system itself and the available plug-ins.*

## Requirements And Constraints

gPlazma has a number of requirements and constraints:

- Of course, the specific door must make use of gPlazma.
- For some of the plug-ins, gPlazma requires the CA- and VOMS-root-certificates, that it should trust, to be present, (typically) in `/etc/grid-security/certificates/` and `/etc/grid-security/vomsdir` respectively.
- For some of the plug-ins, gPlazma requires X.509-host-certificates to be present, (typically) in `/etc/grid-security/`.
- The configuration for gPlazma and its used plug-ins must be present on each host that runs a `gplazma`-service.
- Multiple DNs per client-certificate are currently generally not supported (but multiple FQANs are).
- `gplazma` must be restarted in order to notice changes to its configuration (not however to the certificates).
- In advanced setups, there might be more than one instance of `gplazma` per cluster. It is then usually desired to have gPlazma's configuration synchronised between all of its instances.

# Enabling The gPlazma-Cell And General Configuration

In order to enable gPlazma for a cluster, the `gplazma`-service must be added to a domain in the "layout-configuration-file" on a node.

General configuration-parameters are:

- `gplazma.cell.limits.threads`
  Specifies the number of requests gPlazma may process concurrently.
- `gplazma.cell.name`
  Specifies the name of the gPlazma-cell.
  Only important when running several instances of `gplazma` per cluster.
- *service-name*`.service.gplazma`
  Specifies the name of the gPlazma-cell that the service *service-name* will use.
  Only important when running multiple (non-local) instances of `gplazma` per cluster.
- `gplazma.cell.export`
  Can be used to have only "local" gPlazma-cells, run by the respective doors.
- `gplazma.configuration.file`
  The location of gPlazma's plug-in configuration file, per default `/etc/dcache/gplazma.conf`.

# The gPlazma Plug-In System

gPlazma's plug-in system is similar to that of PAM:
There are different classes of plug-ins, with a number of such plug-ins of the same class being a phase.

The following classes/phases exist:

- `auth`
  Authenticating an accessing entity, for example via an X.509 certificate or a Kerberos ticket. This step leads to authentication information like a DN, FQANs or a Kerberos principal.
- `map`
  Mapping of authentication or mapping information (this is basically "recursive" mapping), for example a DN and FQANs to UNIX User IDs and Group IDs.
- `account`
  Determining the status of an account, for example whether it was banned.
- `session`
  Determining the session environment of the access, for example home- and root-directories.

# The gPlazma Plug-In System

- `identity`
  Mapping of (UNIX) User IDs and Group IDs to user names and group names – and vice-versa.

# The gPlazma Plug-In System

Each plug-in of a phase either succeeds or fails.
The control-type of the respective plug-in determines what happens then.

The following control-types exist:

- `optional`
  Continue with the “next” plug-in, regardless whether the “current” one succeeds or fails.
- `sufficient`
  If the “current” plug-in succeeds, finish the phase with success and do not process its “next” plug-ins.
- `requisite`
  If the “current” plug-in fails, finish the phase with failure and do not process its “next” plug-ins.
- `required`
  If the “current” plug-in fails, finish the phase with failure but process its “next” plug-ins nevertheless.

# Currently Available gPlazma Plug-Ins

Currently the following plug-ins are available:

- for the `auth`-phase
  `x509`, `voms`, `xacml`, `jaas` and `kpwd`
- for the `map`-phase
  `vorolemap`, `authzdb`, `krb5`, `nsswitch`, `ldap`, `nis`, `gridmap`, `kpwd` and `mutator`
- for the `account`-phase
  `argus` and `kpwd`
- for the `session`-phase
  `authzdb`, `nsswitch`, `ldap`, `nis`, and `kpwd`
- for the `identity`-phase
  `nsswitch`, `ldap`, and `nis`

# Authentication- and Authorisation-Process

The following is a very brief description of the authentication- and authorisation-process:

1. A client makes an access request on some resource via a door using its protocol.
   Depending on the protocol, the access is either secured or not:
   - unsecured protocols
     Depending on configuration, the access is either denied or mapped to some special user account, typically with only read-rights.
   - secured protocols
     In this case, the client presents credentials, for example via a certificate with a DN and optionally one or more FQANs or via a Kerberos ticket.
2. The access control system determines possible mappings (these include the actual UNIX user-ID(s) and group-IDs as well as some other information).
   - Depending on the door, gPlazma is used for this.
   - With unsecured protocols, the client receives usually a special limited mapping.
3. The access control system evaluates the found mappings against the policy data for the requested resource in order to determine the access decision.
4. The access decision is enforced and the access granted or denied.

## Policies

Policies contain the rules that describe how resources might be accessed.

dCache can use two types of policies:

- Traditional POSIX file permission modes
- Access Control Lists
  dCache's "native" ACLs (not the ones from the "ALICE security model") are a subset of the NFS version 4 ACLs, providing nearly all of their features.

  They are evaluated in addition to the traditional POSIX file permission modes, which they generally outvote.

The data for both is stored within dCache's file hierarchy and thus the `chimera`-database.