DIRAC distributed secure framework

Journal of Physics: Conference Series **219** (2010) 042033 doi:10.1088/1742-6596/219/4/042033

DIRAC Distributed Secure Framework

A Casajus
Universitat de Barcelona
E-mail: adria@ecm.ub.es

R Graciani
Universitat de Barcelona
E-mail: graciani@ecm.ub.es

on behalf of the LHCb DIRAC Team

Abstract. *DIRAC*, the *LHCb* community Grid solution, provides access to a vast amount of computing and storage resources to a large number of users. In *DIRAC* users are organized in groups with different needs and permissions. In order to ensure that only allowed users can access the resources and to enforce that there are no abuses, security is mandatory. All *DIRAC* services and clients use secure connections that are authenticated using certificates and grid proxies. Once a client has been authenticated, authorization rules are applied to the requested action based on the presented credentials. These authorization rules and the list of users and groups are centrally managed in the *DIRAC* Configuration Service. Users submit jobs to *DIRAC* using their local credentials. From then on, *DIRAC* has to interact with different Grid services on behalf of this user. *DIRAC* has a proxy management service where users upload short-lived proxies to be used when *DIRAC* needs to act on behalf of them. Long duration proxies are uploaded by users to a MyProxy service, and *DIRAC* retrieves new short delegated proxies when necessary. This contribution discusses the details of the implementation of this security infrastructure in *DIRAC*.

1. Introduction

The *LHCb* [1] experiment is the Large Hadron Collider Beauty experiment at *CERN*, primarily intended for precise measurements of CP violation and rare decays in b-physics. *LHCb* expects to start taking data by the end of 2009. The data rate is expected to be about 5 Peta Bytes per full year of running. This data will need large computing resources at a scale in which the only reasonable solution is the worldwide computing grid.

LHCb physicists will have to process a large amount of data. *DIRAC* [2] is the community grid solution *LHCb* will use to manage all the available resources to process the data. All the resources are distributed across different countries and they have to be accessed by the collaboration in a secure way. *DIRAC* has a security framework called *DISET* built on top of *OpenSSL*, an industry standard widely used. *DISET* provides all the communication, authorization and authentication framework for *DIRAC* to build its services on top.

2. Secure framework

One of the *DIRAC* design goals was to manage a large amount of distributed resources. That required a secure communications layer that allowed *DIRAC* to set up access control across the resources. *DIRAC* was designed using a service-agent achitecture [3]. Due to *DIRAC* complexity, it also required a powerful framework that allowed to build services and agents easily. To meet all these needs *DISET* was made. *DISET* is a framework that provides a secure connection layer, an authorization mechanism and a powerful and flexible framework to build agents and services. To provide secure connectivity, *DISET* uses a python wrapper around *OpenSSL* to manage the authentication and encryption of the data.

2.1. Authentication

DISET follows the same authentication mechanism the grid uses. Clients and services are identified by *X509* certificates [4]. To be able to forward the user credentials to the final destination, a proxy certificate [5] has to be provided. Therefore, the authentication mechanism has to support *X509* certificates and certificate proxies.

All connections require mutual authentication: clients authenticate services and vice versa. Authentication is done by checking the credentials received against a list of valid *Certification Authorities (CAs)*. If the credentials presented by the client or the service fail to be signed by one of the valid *CAs*, the connection is closed. Clients can use certificates or proxies to connect to services but services can only use host certificates.

2.2. Authorization

Before any action is executed, *DIRAC* checks if the client is allowed to request the execution of the action. *DIRAC* has a set of authorization rules to verify if the action can be executed. Each action has a set of required properties. Every time a client connects to a *DIRAC* service, *DISET* authenticates it and extracts the client credentials from the *SSL* handshake. The client credentials identify the requester in *DIRAC* and associate a set of properties to the credentials. There are two ways in how a set of properties can be associated to a credential depending on the requester:

- If the requester is a host, *DISET* checks if the Distinguished Name (*DN*) is registered in the *DIRAC Configuration Service (CS)*. If it's registered, there will be a set of properties associated to the host *DN*.
- If the requester is a user, the properties are not directly associated to the user's *DN*. Because there can be lots of users, users are organized in groups. When a user wants to connect to any *DIRAC* service, an active group has to be selected. *DISET* extracts this group with the rest of the user's credentials when the connection is established, and uses this group and the user's *DN* to discover which set of properties the user has in the *CS*.

To execute the requested action, the requester has to have at least one property in the set of properties required by the action. Actions can allow any authenticated user to execute them if no property is required. A default set of properties can also be defined. If an action does not have a set of required properties, the default set will be the one required by the action. Figure 1 shows the authorization state machine.

2.3. Setting an active group

Users can belong to more than one group. Before any action can be requested, users need to select which group will be the active one. To request any action from a service, users have to present a valid grid proxy, which can have a *DIRAC* group embedded in an *X509* extension inside it. *DISET* only accepts this extension if it is embedded in the fist level after the user

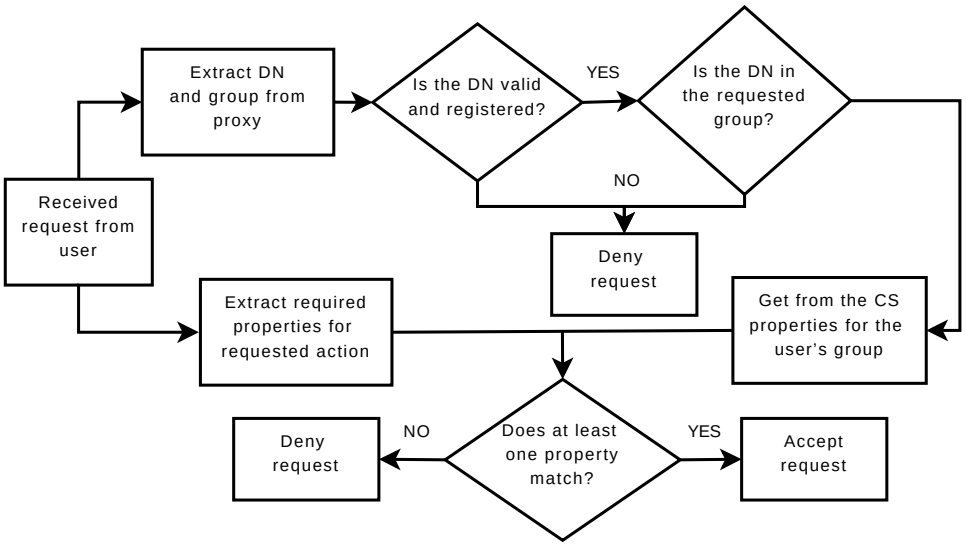

Figure 1. Authorization algorithm applied by *DISET* to each incoming request.

certificate in the proxy chain. By embedding the group in an extension in a predefined level in the proxy chain, *DISET* ensures that:

- Only the user can set the active group. No other entity can define the active group because it is embedded in the level signed directly by the user certificate.
- Services know that the user has selected the group in person. The user cannot say that the group hasn't been selected because it has been signed directly by the user certificate.
- The selected group travels with the proxy. Future delegations only add levels to the proxy. These extra levels can add anything in their extensions but only the fist level is checked to ensure no modifications can be made.

3. Proxy Management service

DIRAC manages users' payloads. There are multiple types of payload, e.g. a job that has to run in a cluster or a data transfer request. Typically users' payloads are submitted from one host and get executed in another host, but the payload has to be executed under the user's credentials. This requires the user's credentials to travel with the payload to the final destination. There is also a delay between the payload upload to *DIRAC* and its execution. To manage all the user's proxies, *DIRAC* has the *Proxy Management* service.

The *Proxy Management* service allows users to upload long-lived proxies with a *DIRAC* group embedded. There is a strict set of rules to allow download of any stored proxy. All proxy movements through the network are done by delegating proxies. Delegation is a method that allows movement of public credentials through any channel avoiding movement of private keys. Instead of simply sending the user's proxy through the network, there are three steps to create a new proxy on the other side of the connection:

(i) The proxy receiving end generates a public key and a private key. With the public key, a certificate request is generated and sent to the other end of the connection.

(ii) The proxy sending end uses the certificate request received to create a new proxy chain. This proxy chain will contain the originating proxy chain plus a new level. The new level will be generated using the certificate request received and signing it with the previous level.

(iii) The new chain is sent to the receiving end where the private key will be added.

Users upload their proxies to the *Proxy Management* service by delegating them. And authorized agents download them by doing the same.

3.1. Authorization rules for downloading proxies
Proxies allow to act on behalf of users so they are very sensitive data. Only a restricted set of users and agents are allowed to download proxies from the *Proxy Management* service. There are only two cases where downloading full proxies is required:

- Administrators (a handful of people) can download any proxy.
- Agents that run in predefined hosts and need to interact with resources on behalf of users, can download the required proxy. An example of this case is the *DIRAC* component that submits jobs to the grids *DIRAC* has access to.

There is another case where downloading a proxy is required. *DIRAC* uses *pilot* jobs to access the resources. A *pilot* job is a job sent to the grid to reach the final resource. Once it is running on the final resource, it installs a *DIRAC* client if it is not there, and then contacts the DIRAC service to obtain the details for a real user job. Before a user's job can start on a resource, a limited proxy has to be downloaded to run the user's job using the user's credentials. Limited proxies are proxies that have *limited proxy* as the last Common Name in the proxy's Distinguished Name. These proxies are not allowed to submit new jobs, and jobs are only allowed to run if they use limited proxies as user credentials. There are two types of *pilot* jobs: generic and private. Private pilot jobs are submitted to the grid using the user's credentials. This type of pilot job can only run with the submitter's credentials. These jobs are directly submitted with a user proxy.

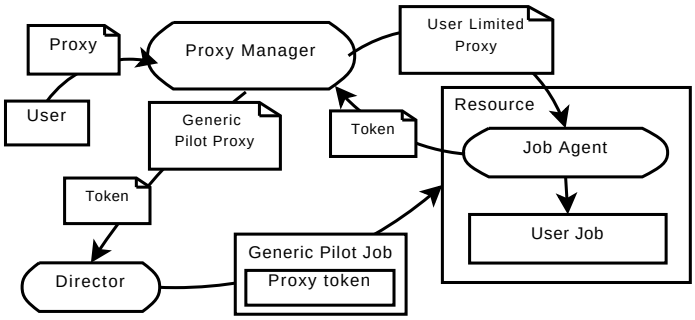

Figure 2. How *DIRAC* handles the proxies to execute the user's payload under a generic pilot.

Generic pilot jobs are submitted with a generic credential. Generic credentials are privileged users with a special group that are allowed to change the effective credential in the final resource before executing the real user payload. Before running the real user job, a limited user proxy is needed. The pilot credentials must be able to download any limited proxy. To avoid someone stealing a generic pilot proxy from a resource and downloading a newer proxy for the same credentials, generic pilot credentials cannot download generic pilot credentials. To avoid being able to download all proxies from the *Proxy Management* service, each generic pilot job has a proxy token. This token is randomly generated for every job. A token has to be presented to the *Proxy Management* service for any proxy to be downloaded by a pilot job. Tokens have an expiration time and a limited number of uses. If a generic pilot proxy is stolen, downloading non-pilot proxies will be allowed only while the proxy and the token haven't expired. Figure 2 shows how the proxies are requested and delegated.

3.2. Extending proxies

The *Proxy Management* service can keep long-lived proxies but it can also use external mechanisms like the *MyProxy* service to extend the proxies it holds. If an authorized source requests a proxy longer than the one the *Proxy Management* service has, it can request a new proxy from the *MyProxy* service, store the new and longer lived proxy, and use it to delegate a proxy that fits the request.

3.3. VOMS interaction

DIRAC does not require *VOMS* [6], but can use it if the resources *DIRAC* accesses require it. Any *DIRAC* group that can interact with these resources has a *VOMS* mapping. When a *DIRAC* component wants to interact with the resources, a *VOMS* proxy is requested from the *Proxy Management* service. Before delegating the proxy to the requester, the *Proxy Management* service adds the required *VOMS* extensions based on the requested group.

References

[1] Antunes-Nobrega R *et al.* (LHCb) CERN-LHCC-2003-030
[2] Tsaregorodtsev A *et al.* 2008 *Computing in High-Energy Physics and Nuclear Physics 2007*
[3] Graciani R and Casajus A 2008 *Computing in High-Energy Physics and Nuclear Physics 2007*
[4] Housley R, Polk W, Ford W and Solo D 2002 [RFC3280] Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile
[5] Tuecke S, Welch V, Engert D, Pearlman L and Thompson M 2004 [RFC3820] Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile
[6] Virtual organization membership service URL `https://twiki.cnaf.infn.it/cgi-bin/twiki/view/VOMS`