6 WS-PGRADE/gUSE Security

Zoltán Farkas

Abstract. If any data is stored on publicly available services, then it is obligatory for the service to apply different security policies in order to prevent information leakage to unauthorized users. This also applies to e-Science gateways, where sensitive (for example, medical) data might be stored. Moreover, e-Science gateways usually enable the usage of distributed computing infrastructures that apply some sort of authentication, which should either be provided by the users, or be completely hidden and implemented inside the gateway in the form of robot credentials. In this chapter we discuss some aspects of security applied within WS-PGRADE/gUSE.

6.1 Introduction

Security of e-Science gateways is an important topic. The gateway offers easy access to different distributed computing infrastructures (DCIs) that can be used to process the data of the e-scientist, mostly of sensitive nature (for example, patient data). It follows from this, that securing access to the gateways, its services, the infrastructure, and the data accessed by them is of major importance.

In this chapter we discuss different aspects of security related to e-Science gateways: how the gateways can be accessed, how access to their services can be limited, and how the included components can be made secure. Finally we present two possible approaches for using credentials to access the computing infrastructure and data behind the gateways. The chapter tries to discuss this topic in general, but in order to ease understanding, we present a real use-case of WS-PGRADE/gUSE.

6.2 Access to the Gateway

Access to e-Science gateways is usually possible by accessing a login page, where users have to identify themselves by providing some sort of credential. The credential can be a login name or e-mail address with a relevant password, some sort of single sign-on credential, or a one-time password.

As WS-PGRADE/gUSE builds on the Liferay portlet container, login possibilities offered by WS-PGRADE/gUSE are those offered by Liferay. This includes the following:

- Username or e-mail with a password: in this case gateway users can register using either a username or e-mail address, and a password.

After successful registration, users can log in using the credentials provided during the registration process.

- Facebook: in this case the gateway uses Facebook's user authentication mechanism [FBLogin] (based on OAuth [OAuth]) to identify users. This method assumes that a Facebook application is created for the e-Science Gateway, and this application can access some basic properties of the Facebook user's profile. Through this login method, users have to provide their Facebook account details in order to log in.
- Single sign-on: this method is similar to the Facebook one. Some sort of external identity provider (for example, Shibboleth [SHIB]) is used to identify users, and the information provided by the identity providers is used to store users' information in Liferay.

The above-described methods are the most common ones applied by e-Science gateways. Thanks to the fact that Liferay offers the possibility to use additionally deployed authentication methods, these can be extended toward additional needs.

6.3 Visibility of User Interface Components

When a user registers to an e-Science gateway, the user account associated to the user receives some default roles [LiferayRoles]. These roles can be preconfigured in the portlet framework or can be inherited from an external identity provider. In either case, the set of roles associated to the user account may determine the set of tools available for use by the user account. Some example roles used by Liferay and WS-PGRADE/gUSE-based e-Science gateways are as follows:

- Guest: this role represents any non-authenticated user accessing the gateway (the role is defined by Liferay).
- User: this role represents any authenticated user accessing the gateway (this role is defined by Liferay).
- Owner: this role represents any authenticated gateway administrator user accessing the gateway (this role is defined by Liferay).
- End user: this role represents any authenticated user who is going to access the end user interface of WS-PGRADE/gUSE (this role is defined by WS-PGRADE/gUSE and is discussed in Chap. 8).
- Power user: this role represents any authenticated user, who is going to access all the portlets provided by WS-PGRADE/gUSE (this role is defined by WS-PGRADE/gUSE).

The above example roles are the basic set of roles available in a WS-PGRADE/gUSE-based e-Science gateway. The gateway administrator has the freedom to add any new roles as necessary.

In the Liferay portlet container used by WS-PGRADE/gUSE, the different portlets placed on the user interface have a visibility property, which describes the set of user roles that can actually view, and thus use the portlet. This method of-

fers a flexible way to set up a single e-Science gateway instance for users with different roles and even for different scientific domains.

For example, let us assume that an e-Science gateway is set up for astrophysicists and biologists. In this case the e-Science gateway administrator can follow these steps to properly configure the gateway for the different communities:

1. Two new roles have to be defined: for example “astro” and “bio”.
2. All the portlets for the astrophysicist and biologist communities can be deployed onto the gateway.
3. The portlets targeting the different science domains should be set up such that they are only visible for the targeted science domain role (for example, portlets targeting the astrophysicist community should be visible only for users possessing the “astro” role).
4. Any new users registering to the gateway should have the proper role assigned.

6.4 Securing the Services

In a publicly available e-Science gateway, securing the services building up the gateway is an important task of the gateway administrator. If the gateway is based on a single service (that is, it is not built based on the SOA concept), then securing the gateway is really simple. However, if the gateway relies on the cooperation of multiple services (for example, a job submission service or a workflow interpreter service), then the communications between the different services must also be secured ensuring that no sensitive data is leaked from the gateway. In this section we present three best practices that can be followed to make a service-oriented e-Science gateway secure.

The first rule of thumb of securing a service-oriented e-Science gateway is to make the service communicate through a secure channel. For this, the different interfaces exposed by the services should be accessible only through a secure protocol, preferably relying on client authentication as well. Making use of a secure channel ensures that the communication between the client and the server cannot be decoded. On the other hand, relying on client-side authentication ensures that only allowed clients can use the service. For example, in case of WS-PGRADE/gUSE, the workflow interpreter service (WFI) sends jobs for execution to the DCI Bridge service. In this case, if the administrator wants to enable job submission through a secure channel, then the DCI Bridge’s services must be made accessible only through the HTTPS protocol. Additionally, if the gateway administrator wants to make sure that only the WFI component of the given gateway is able to submit jobs to the DCI Bridge, then the administrator has to create a client certificate for the WFI service, which will be used to authenticate to the DCI Bridge service. These two steps make sure that the communication channel cannot be intercepted, and that a given service (the job submission component, DCI Bridge in our example) can only be used by dedicated clients.

The second rule of thumb of securing service-oriented e-Science gateways is to make service components publicly available if and only if it is really necessary. For example, in case of WS-PGRADE/gUSE, the web interface (front-end) is a component that has to be made publicly available, but it is not necessary to make the back-end components (WFI, WFS, DCI Bridge service) publicly available. Of course, the front-end and back-end components should be able to communicate with each other. Such a requirement can be fulfilled, for example, with the following setup:

- All of the gateway services should reside on an internal network, where they can freely communicate with each other, but they are not necessarily accessible from a public network.
- All of the front-end components should additionally placed onto a network accessible from a public network as well.

Following this setup, only the components really important for public operation are publicly available, but they can still communicate with the back-end components as well.

Finally, if the e-Science gateway technology is not built on widespread web servers (for example, Apache), then it is desirable to put it behind one. For example, WS-PGRADE/gUSE makes use of the Apache Tomcat servlet container, which is connected to an Apache web server using AJP [AJP]. This setup enables use of the web server's features while configuring the WS-PGRADE/gUSE access. For example, host certificates can simply be added to the front-end service.

An important example for the need of securing the services is the storage component of WS-PGRADE/gUSE. This service is used by other components to up- and download files belonging to workflows. However, the storage service doesn't make use of any authentication, and enables the client to specify the file path to upload or download. This means that, if the storage service is publicly available, then everyone in the world can fetch or modify data stored on the gateway, which the Storage service has access to. Thus, it is recommended to close access to this service. However, in case of accessing cloud services directly (Chap. 4) or BOINC-based desktop grids, access to the storage service from the computing infrastructure is necessary. In such cases, properly configured firewalls should be set up to grant access to the Storage service only from the given computing infrastructures.

6.5 User Credentials

Once logged in, the users are able to run experiments on different computing infrastructures, some of which require users to provide credentials. These credentials can, for example, be a username and a password, or an X.509 [X509] certificate. Thus, in order to be able to run the experiments, the user somehow has to present the necessary credentials.

For this, two basic possibilities are available: either the user has to define these credentials on the user interface of the e-Science gateway, or the e-Science gate-

way should include some predefined credentials (robot credentials) for running different applications on the different computing infrastructures.

In the first case (when the users have to provide their credentials) the gateway assumes that its users are familiar with the security concepts of the different computing infrastructures attached to the experiments, and that the users already posses the necessary credentials. This means that the entry level to gateways offering such experiments is higher; the users not only need to be familiar with the targeted science domain, but the used infrastructure's security concept as well.

On the other hand, in the case of using robot credentials, the whole back-end infrastructure can be hidden from the e-Scientists. This means, that the users only have to focus on their experiments, and no knowledge beyond their science domain is needed in order to use the e-Science gateway's services. However, applying the robot credential concept on the gateway assumes that the gateway records all the necessary information that is needed to connect any interaction with the computing infrastructure to a gateway user. The policy for e-Science gateways that would like to expose robot certificates in the EGI infrastructure is described in the EGI VO Portal Policy document [EGIVO].

Of course, gateways have the freedom to offer these two possibilities simultaneously if needed. For example, some experiments can be run with robot credentials, while some others may ask the users to enter their own credentials. Additionally, if the gateway experiments are built on workflows, mixing these two usage scenarios within an experiment is also possible.

6.5.1 User-Defined Credentials

The first possibility of using credentials as presented in the introduction part of this section is when users provide their own credentials for running the experiments. We are going to discuss the following topics related to this possibility: a user interface for entering the credentials, storing the provided credentials, and usage of the provided credentials.

User Interface for Providing Credentials

After the user is logged in to the e-Science gateway, and before he has started any experiment relying on credentials, there should be user interfaces for providing any necessary credentials to run the experiments. These interfaces can be included in the experiment's interface, or can be placed into a dedicated, security-related place on the gateway's user interface.

WS-PGRADE/gUSE follows the latter approach: all the portlets related to setting necessary credentials for the different distributed computing infrastructures are grouped in the "Security" tab. This tab contains subpages with portlets for setting the credentials to be used. The organization of this "Security" tab is shown in Fig. 6.1.

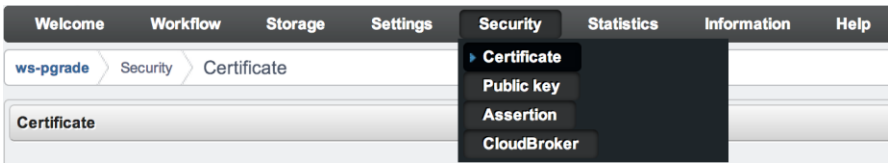

Fig. 6.1 Security tab in WS-PGRADE/gUSE

As can be seen, WS-PGRADE/gUSE currently offers four different credential management portlets:

- Certificate: for managing X.509 proxy certificates,
- Public key: for displaying public keys for public-key-based SSH authentication,
- Assertion: for managing SAML [SAML] assertions, and
- CloudBroker: for defining e-mail and password for cloud-directed jobs.

All of these portlets implement a user interface for defining the necessary credentials depending on the credential's properties.

The "Certificate" portlet offers a complete X.509 proxy certificate management tool based on the MyProxy Credential Management Service [MyProxy]. Through the interface provided by the portlet, the users can: upload their X.509 certificates to a MyProxy server, download X.509 proxy certificates from a MyProxy server and assign them to resources relying on X.509 proxy authentication, and can manage X.509 proxy certificates stored on MyProxy servers. This latter interface is shown in Fig 6.2. Although the portlet can be used to upload proxies to MyProxy servers, this is achieved by transferring the user's certificate and key to the portal server from the user's machine. In order to overcome this security risk, one may use other tools that run on the user's machine and are capable of uploading proxies to MyProxy servers, like the NGS Certificate Wizard [CertWizard] or the GSISSH-Term [GSISSH-Term].

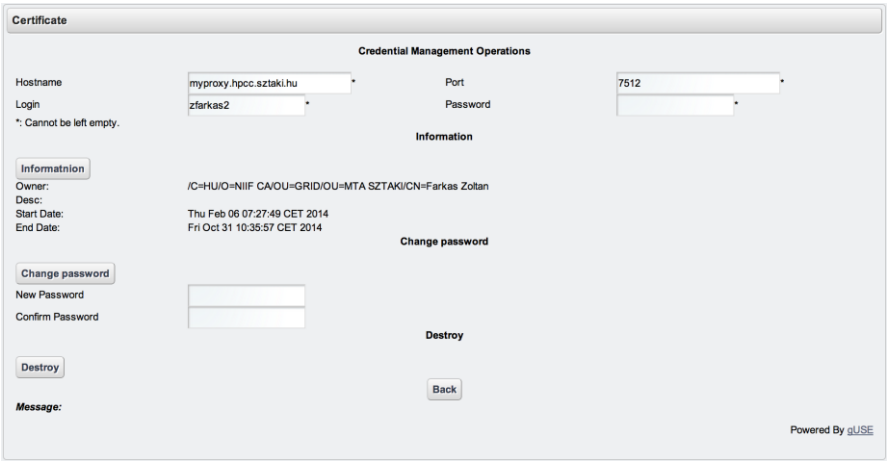

Fig. 6.2 Managing X.509 proxy certificates stored on MyProxy server

The "Public key" portlet can be used in the case of infrastructures where public key-based SSH authentication is used to connect to the computing infrastructure. Typical examples are PBS [PBS] or LSF [LSF] local resource management systems, where WS-PGRADE/gUSE uses a generated keypair to connect to the submission node of the target infrastructure.

The "Assertion" portlet enables the users to generate or upload already existing SAML assertion data that is typically used to access UNICORE resources.

Finally, the "CloudBroker" portlet offers an interface for entering e-mail and passwords used to authenticate with different CloudBroker services.

Interested users can find the user interface description details of these security portlets in Chap. 10 of the WS-PGRADE User's Manual [Manual/2014].

Every computing infrastructure supported by WS-PGRADE/gUSE has an authentication method attached. This can be "basic authentication" for e-mail or username and password-based, "SSH key" for public key-based, "X509" for X.509 proxy-based, and "SAML" for assertion-based authentication. It follows from this, that if the user invokes a given portlet from the "Security" tab on the WS-PGRADE/gUSE user interface, then the given portlet can be used to set credentials for computing infrastructures having the authentication method type implemented by the given portlet. For example, the "Public key" portlet can be used to set SSH keys for LSF-based and PBS-based resources (as these computing infrastructures use the "SSH key" authentication).

Storing User-Provided Credentials

Once the user has set the necessary credentials through a given credential portlet, the credential data is stored by WS-PGRADE/gUSE. Each WS-PGRADE/gUSE user has a dedicated credential storage space allocated on the front-end component of the gateway, which is the user's directory, and is only ac-

cessible, by the given user (and by WS-PGRADE/gUSE services acting on behalf of the user). The credentials provided by the user are stored in this location, and follow the naming scheme of "x509up.<name>", where <name> represents the name of a computing infrastructure resource. Let us assume for example, that the user has set credentials for a gLite resource (using authentication type "x509") called "seegrid", and a CloudBroker resource (using authentication type "basic authentication") called "platform". In this case two credential-related files exist in the user's directory, one called "x509up.seegrid", and another called "x509up.platform". It follows from this naming scheme that it is not possible to store credentials of different computing resources using the same name, even if they belong to a different computing infrastructure.

The content of the credential files depends on the computing infrastructure's authentication method they store the credential for:

- Basic authentication: in this case the credential file contains two strings, a username (or e-mail address) and a password, placed into separate lines,
- X.509: in this case the credential file contains the X.509 proxy certificate as it has been downloaded from the MyProxy server,
- SAML: in this case the credential file contains the SAML assertion data,
- SSH key: in this case actually two credential files exist; one containing the SSH key pair's private part (in a file called x509up.<name>), and one containing the SSH key pair's public part (in a file called x509up.<name>.pub).

Using User-Provided Credentials for Job Execution

For each job submitted as part of an experiment, the job's description contains the name of the user who submitted the job, the type of the computing infrastructure the job should be handled by, and the name of the resource the job should be run on. All this information is necessary to identify unambiguously the credential set by the user for the given job. The steps to get the credential for the given job are as follows:

1. When the DCI Bridge (the job submission component of WS-PGRADE/gUSE) receives the job, it checks the type of the computing infrastructure, and asks the relevant plugin to handle the job.
2. The plugin in charge gets the name of the user, and the name of the resource from the job description.
3. The plugin asks the WS-PGRADE component's CredentialProvider service for the credential belonging to the given user and given resource.
4. The CredentialProvider service serves the credential for the DCI Bridge plugin.
5. The plugin can submit the job to the selected computing infrastructure's resource, given that the credential is a valid one.

6.5.2 Robot Credentials

If a gateway administrator is about to offer the gateway's services for end-user scientists, then it is recommended to hide the computing infrastructure details from the users. (The complicated procedure to get X.509 certificates often distracts end-users from using DCIs even in the case of science gateways.) This also includes the necessity to hide any credential-related interface; otherwise the users will have to take care of acquiring and providing credentials to run their experiments on the gateway. Only a user interface presenting solely experiment-related interfaces is really handy for the e-Scientists.

The need to provide user credentials for using the computing infrastructures behind the experiments can be solved by applying the robot credential concept. In this concept, the developer who sets up the experiments for the end users is responsible for attaching any necessary credential for running the experiments (workflows) on the targeted computing infrastructures. The attached credentials later can be used transparently by the end-users to actually run the experiments, meaning they do not have to take care of providing their own credentials.

The EGI VO Portal Policy document [EGIVO] describes policies that should be followed by e-Science gateways, assuming they would like to use robot certificates (credentials) in the EGI infrastructure. In this section we give a brief overview of this document, and present how WS-PGRADE/gUSE has applied this policy when implementing robot credentials.

The Concept of Robot Credentials

As described previously, the certificate provided by the users is used by the computing infrastructure to identify any interaction with the infrastructure selected by the user. The robot credentials are also used to identify some entity, but are more related to a community, an application, or a science gateway, than a user. It follows from this, that a robot credential can be used by a number of users to run applications on computing infrastructures. As a consequence, the entity offering the usage of the robot credentials must keep track of the credentials' usage, so that any interaction with the computing infrastructure performed with the robot credentials can be tracked back to a user.

An additional requirement is that if robot credentials are used, then the entity applying the credential should negotiate with the target computing infrastructure's operator about the maximum job submission rate performed with the robot credentials.

The EGI VO Portal Policy document defines the following user groups: web user (any human accessing a gateway), anonymous web user (a web user not providing unique credentials), identified web user (web user providing personal identification information), strongly identified web user (identified web user providing computing infrastructure credentials as well).

The EGI VO Portal Policy document describes the following portal classes (Table 6.1). Science gateways belonging to the job management portal class mustn't offer robot certificates.

Table 6.1 Portal classes as defined by the EGI VO Portal Policy document

Portal Class	Executable	Parameters	Input
Simple one-click	Provided by portal	Provided by portal	Provided by portal
Parameter	Provided by portal	Chosen from enumerable and limited set	Chosen from repository vetted by the portal
Data processing	Provided by portal	Chosen from enumerable and limited set	Provided by user
Job management	Provided by user	Provided by user	Provided by user

In the case of the other portal classes, the portal may offer the usage of robot credentials, depending on the type of user making use of the gateway's services. The EGI VO Portal policy document describes in detail the policies that should be followed in each of the different portal classes.

Of course, a single gateway deployment may implement multiple portal classes. For example, a WS-PGRADE/gUSE deployment can operate both as a job management and as a simple one-click portal, depending on the type of users accessing it.

Robot Credentials in WS-PGRADE/gUSE

After the short introduction to the EGI VO Portal Policy document, we present how this policy has been implemented in WS-PGRADE/gUSE. We describe how the workflow developers can define robot credentials, how the robot credentials (or their references) are stored, and how they are used during the workflow submission process.

In WS-PGRADE/gUSE, workflow developers willing to attach robot credentials to workflows should possess a special user role called "RobotPermissionOwner". This means that traditional users of the gateway do not have the right to assign robot credentials to workflow nodes.

Defining Robot Credentials

The definition of robot credentials is available through the workflow nodes' configuration window. Once the target computing infrastructure has been selected, the "Create association" button becomes visible if the gateway user has the "RobotPermissionOwner" role. Once this button is pressed, a robot credential definition dialog, depending on the type of authentication applied by the computing infrastructure, is presented for the workflow developer. Figure 6.3 shows the

workflow node configuration and a robot credential association for a CloudBroker-based job.

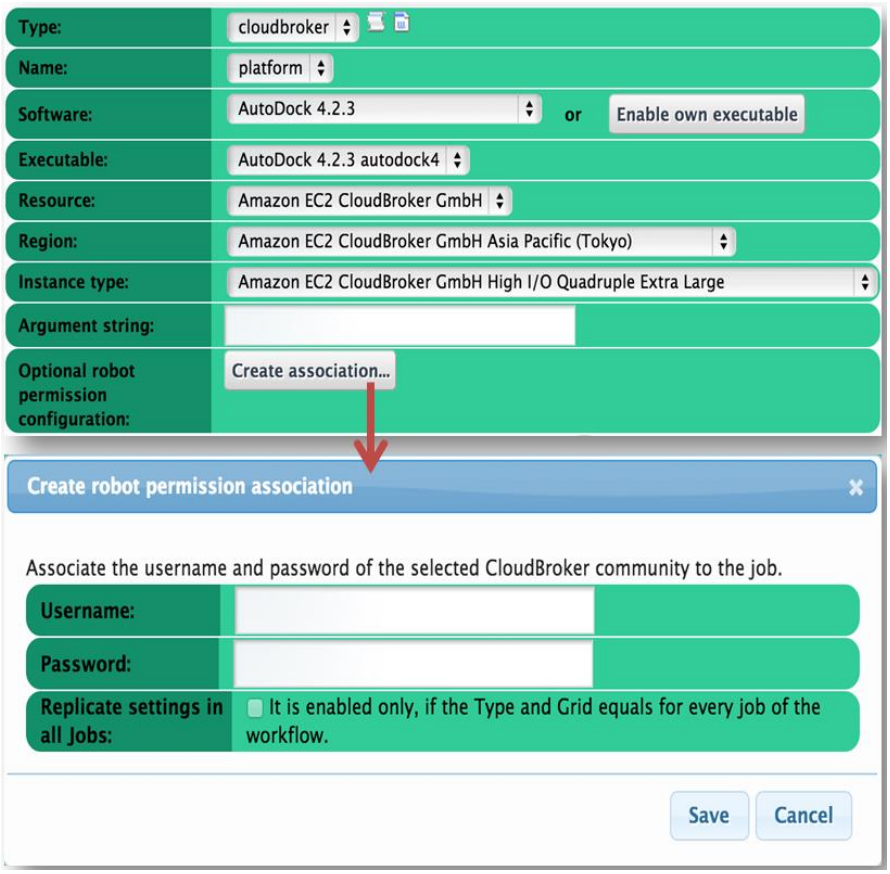

Fig. 6.3 Robot credential association to a workflow node

The robot credential association dialog depends on the type of authentication applied by the target computing infrastructure. Notice that if all the nodes of a workflow are executed on the same target computing infrastructure then the associated robot credential can be applied for all the nodes of the workflow by simply clicking on the tick-box of the "Replicate settings in all jobs" (Fig. 6.3).

Once the robot credentials have been set and the workflow has been saved, any robot credential association set in the workflow is stored. From this point on, the executable and the computing resource defined in the workflow nodes set up with robot credentials cannot be changed, unless the robot credential association is removed. This ensures that end users cannot run their own application with the robot credentials.

If the workflow developer opens a workflow node that already has robot credentials defined, then it is possible to remove the robot credential association. It is important to note that the "RobotPermissionOwner" role is not needed to remove the robot credential association. Once the robot credential association has been removed, the target computing infrastructure and the executable of the workflow node can be overridden.

The above-described mechanism applied in WS-PGRADE/gUSE's user interface ensures that if a given workflow node has some sort of robot credential assigned, then the users do not have the possibility to modify the target computing infrastructure and the executable defined for the node unless the robot credential association is removed. Once the robot credential association is removed, the users have to provide their own credentials to run their own applications.

Storing Robot Credentials

Once a robot credential association has been set for a workflow node and the workflow has been saved, WS-PGRADE/gUSE stores the robot credentials. The storing mechanism conforms to the following policy:

- The credentials are stored where they are really needed. This is the job submission component called DCI Bridge in WS-PGRADE/gUSE.
- The stored workflow description contains only a reference to the robot credential stored on the DCI Bridge.

This policy ensures that users cannot acquire robot credentials.

Robot credentials consist of the following components: the executable used and the actual authentication data. The latter is similar to the authentication data of user-provided credentials, and depends on the type of authentication applied by the target computing infrastructure:

- Basic authentication: in this case the username and password are stored.
- SSH key: in this case the SSH key pair's private part is stored, along with a username needed to log in to the resource.
- X.509: in this case storing the X.509 proxy is not feasible (being a short-term credential) nor allowable (as robot X.509 certificates must not leave a secure token). Instead, the MyProxy availability of the robot certificate is defined.
- SAML: in this case the SAML assertion is stored.

Once the workflow developer saves the workflow, the following process is performed for each workflow node with a robot credential association:

1. The WS-PGRADE portlet connects the DCI Bridge service, sending the node's executable and the robot credential information entered by the workflow developer.
2. The DCI Bridge generates a robot credential identifier (a universally unique identifier, UUID), stores the information provided by the port-

let in a file using the generated identifier, and returns the identifier to the caller.
3. The WS-PGRADE portlet stores the returned identifier in the workflow node's description.

Using Robot Credentials for Job Execution

The concluding step of using robot credentials in WS-PGRADE/gUSE happens during workflow node execution. In this case the workflow interpreter (WFI) is interacting with the job submission component (DCI Bridge) based on the workflow's description, where the WFI inserts the identifier of the robot credential to be used for the job's submission if the job is set to be run using robot credentials. Once the DCI Bridge receives the job description, it fetches any input file and executable defined for the job, and checks if a robot credential identifier is defined in the job description. If yes, the DCI Bridge checks if there is a robot credential stored with the given identifier. If the requested credential exists, then it checks if the executable defined for the robot credential is the same as the one provided within the job description. If everything matches, then the DCI Bridge prepares the credentials based on the information stored in the relevant robot credential, and arranges the job's execution on the selected target resource. In case of any problem (for example, a different executable is provided or the robot credentials with the given identifier don't exist), the DCI Bridge checks if the user has provided his or her own credential, and makes use of that one to execute the job. If the user hasn't provided any credentials, the job fails.

6.6 Conclusions

Security is very important in case of publicly available services, like an e-Science gateway. The gateway not only has to make sure that users accessing the tools exposed are authenticated properly, but also is responsible for not allowing leakage of data stored and produced by the users' experiments. In this chapter we discussed some aspects of security in e-Science gateways. We have covered the following main topics: how the gateways can be accessed, how access to their services can be limited, how the included components can be made secure, and finally how credentials needed to access computing infrastructures can be defined and used at job submission time.

In the first part, we described some methods that can be followed to grant access to an e-Science gateway. Some examples are e-mail (or username) and password-based login methods, or inheriting authentication from Facebook. Currently, work is in progress to enable federated identity management-based login to WS-PGRADE/gUSE services, based on the Higher Education External Attribute Authorities (HEXAA). In this approach, the science gateway not only offers federated login, but also has the possibility to receive additional attributes of the user

logging in; thus visibility of user interface components can be set in an external attribute store.

Next, we presented the user role concept that can be applied successfully to fine-tune the visibility of user interfaces for different groups of users. For example, there can be dedicated user accounts having a power user role, and general user accounts having the end user role. Users with the power user role are free to access all the user interface components of the e-Science gateway (including, for example, workflow editing as well, like in case of WS-PGRADE/gUSE), whereas users having only the end user role can have limited access only to customized interfaces running different experiments on the gateway.

Next, we discussed how to secure the services a gateway is built up from. Basically, access to the services must be restricted as much as possible, either by running the services on a private network or by putting them behind a firewall. Additionally, making the services accessible through a secure communication channel helps to make sure data sent between the user and the gateway services is not leaked.

In the biggest part of this chapter we discussed the usage possibilities of credentials necessary to access distributed computing infrastructure services. We have shown the flexibility WS-PGRADE/gUSE provides through its different credential handling mechanisms: on one hand, the individual users can provide their own credentials to run experiments in the different computing infrastructures. On the other hand, the gateway framework offers the possibility to use robot credentials. This latter approach provides workflow developers with a tool for creating really user-friendly e-Science gateways which completely hide the details of using some sort of credentials to interact with the computing infrastructure from the end-user's point of view.