The Security model of the ALICE next generation Grid framework

Miguel Martinez Pedreira[1], *Costin* Grigoras[1], *Volodymyr* Yurchenko[2,*], and *Maksim* Melnik Storetvedt[3]
[1]CERN, [2]National Academy of Sciences of Ukraine, Ukraine, [3]Western Norway University of Applied Sciences, Norway

Abstract. JAliEn (Java-AliEn) is the ALICE next generation Grid framework which will be used for the top-level distributed computing resources management during the LHC Run 3 and onward. While preserving an interface familiar to the ALICE users, its performance and scalability are an order of magnitude better than the currently used framework. To implement the JAliEn security model, we have developed the so-called Token Certificates – short lived full Grid certificates, generated by central services automatically or on the client's request. Token Certificates allow fine-grained control over user/client authorization, e.g. filtering out unauthorized requests based on the client's type: end user, job agent, job payload. These and other parameters (like job ID) are encrypted in the token by the issuing service and cannot be altered. The client-side security implementation is further described in aspects of the interaction between user jobs and job agents. User jobs will use JAliEn tokens for authentication and authorization by the central JAliEn services. These tokens are passed from the job agent through a pipe stream, not stored on disk and thus readily available only to the intended job process. The level of isolation of user payloads is further improved by running them in containers. While JAliEn doesn't rely on X.509 proxies, the backward compatibility is kept to assure interoperability with services that require them.

1 Introduction

During LHC Long Shutdown 2 the ALICE [1] experiment is going to acquire a significant amount of additional computing resources to cope with the higher volume of data received from the detector after its upgrade. These will be organized in the so-called O^2 facility [2]. At the same time ALICE is upgrading the software tools to gain better scalability and resource management. The JAliEn (Java ALICE Environment) framework [3, 4] is a key element of the new software and is a replacement of the legacy Grid middleware called AliEn [5].

The security of JAliEn is a key aspect of its development and this paper describes in detail the new security elements.

1.1 Current security considerations

The AliEn Grid framework is used in production since 2000 for Monte Carlo productions and since 2005 for all types of data processing, including end-user analysis [6]. For user authentication and authorization, it uses the X.509 proxy mechanism [7] with the current procedure requiring Grid users to periodically run a command to create a password-less proxy certificate. Here, a proxy is a temporary certificate with the same power as the identity certificate of the user. In the same system, Grid jobs are assigned a secret random string, which acts like a 'session cookie'. The secret is shared with the central task queue and associated with a job ID in the central database and is valid for the duration of the job.

In JAliEn, we introduce Token Certificates in order to unify authentication and authorization mechanisms for Grid users and Grid jobs. Section 2 describes how Token Certificates replace X.509 proxies and secret strings to achieve higher security consistency while working with user identities on the Grid.

By introducing Token Certificates, another issue is simultaneously addressed – payload isolation. In the current workflow the job has access to the pilot (Job Agent) started from the same sandbox. Splitting the pilot into two components, using Java pipes to pass the job credentials between them, and the use of containerization provide payload isolation in JAliEn.

The work on Token certificates is fully compatible with the directions and decisions taken by the WLCG Authorization Working Group at the time of writing of this manuscript. We will continue to adapt the code to the latest agreed upon standards.

[*] Corresponding author: volodymyr.yurchenko@cern.ch

2 Token Certificates

A Token Certificate is a time-limited certificate to represent a user, a job or a pilot submitter in JAliEn. Unlike X.509 proxies, generated in AliEn, it is a full X.509 certificate signed by our own internal certification authority (CA). This allows JAliEn to encode additional functions in the Distinguished Name (DN) of Token Certificate. This feature is used to divide the token identities into groups and treat them accordingly to define the privileges and limitations of each group.

Currently there are three types of identities:

- *Users*: tokens have the same rights as the user's identity certificate, restricted to 48 hours by default.
- *Jobs*: can only execute the payload, e.g. file operations with read-only access anywhere but write only in the output directory. Other actions like submitting new jobs are restricted. *Job* tokens can be requested only by *JobAgents*.
- *JobAgents*: can do the job matching, update the job status and upload job traces. *JobAgent* tokens can be requested only by VoBox services.

An example of a token certificate for a payload is given below:

Subject: OU="queueid=1038905674/resubmission=3/user=mmmartin", OU=jobagent, CN=jobagent, CN=Jobs, O=AliEn, C=ch
Issuer: JAliEn-CS

With this token, the JobAgent can e.g. operate on job 1038905674 as user 'mmmartin' with the full rights of this user. The resubmission field shows how many times the job was executed, and that the previous tokens (for resubmissions 0–2) are invalidated by the central services (JCentral). Since all the required information can now be stored in the DN of the token, there is no need for the secret strings. The information cannot be altered or complemented by a third party in any way.

An *AlienPrincipal* Java object keeps the (token or identity) certificate in memory for every open connection. For each request, the authorization rights of the certificate are checked on both sides of the connection. This includes the access rights of the user and the limitations of the token.

Token Certificates are signed by the JAliEn CA and can be issued only by JCentral. The JAliEn CA is distributed via CVMFS together with other standard CAs as a part of *AliEn-CAs* package. This is already in place for existing ALICE software.

A user can get token certificate only by providing a valid Grid identity certificate. The expiration of a token is limited to two days by default, but it is possible to request tokens with custom duration from one hour up to the lifetime of the user identity certificate.

If a client requests the token duration to be extended, JAliEn will ask the certificate password to be typed once again. If the client is already connected using Token Certificate, JCentral will close the connection to force the client to reconnect with the identity certificate. It is done to limit the amount of time during which a token might be exploited if it were stolen.

The server closes the connection if it is idle for more than two hours. Periodical validity checks ensure connections close once Token certificate expires. In case of the user gets suspended, Token certificate becomes invalid as well, since it is bound to the same user's identity.

In case the user wants to change to a different role, a new token certificate is issued. Its subject DN now contains the new role, but keeps track of the initial user. This feature helps to keep log traces of all actions of every user.

3 WebSockets protocol

The WebSocket protocol [8] is an extension of HTTP and provides a persistent communication channel on top of a TCP connection. The difference from the usual HTTP protocol is that a WebSocket connection is not closed after the request has been executed, but is kept open until the client or the server closes it explicitly, as shown in Figure 1. The communication channel is bi-directional, the server is able to send data and update the connection status at any time. It is enabled by using the HTTP *Upgrade* header to change from HTTP to WebSocket protocol.

This feature reduces the number of handshakes and allows interaction with lower overheads, promoting real-time data transfer.

In JAliEn the WebSocket server endpoint is implemented by running an embedded Apache Tomcat webserver. It has native support for the secure WebSocket protocol and is started inside JCentral or JBox Java application instances.

The WebSocket protocol is standardized as RFC 6455. The WebSocket API is available in relevant programming languages and is provided by system libraries in most Linux distributions. Users can create their own custom clients to connect to the JAliEn endpoint, while we have implemented the reference client in the most prevalent case for ROOT [9] for the experiment software to interact with the Grid.

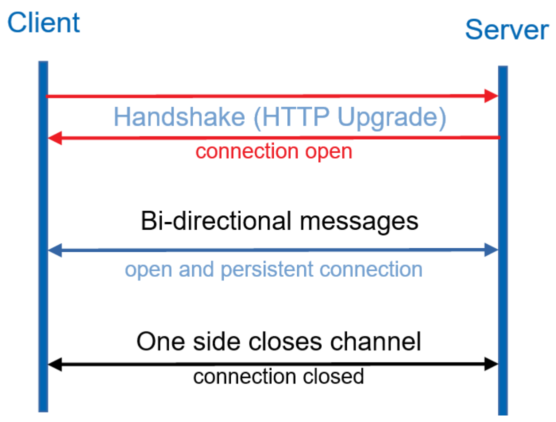

Fig 1. WebSockets visual representation

4 ROOT interface and JAliEn shell

4.1 TJAlien-ROOT plug-in

Two interfaces have been developed to access the ALICE Grid services. The first interface are the TJAlien classes [10] for AliRoot [11] and ROOT version 6. Unlike the TAlien classes, the new solution uses the ROOT plug-in mechanism to keep the ALICE-specific code out of the main ROOT repository. In general, TJAlien mimics the current TAlien implementation to provide a familiar user interface and functionality. At the same time, it takes advantages of the secure persistent connections with WebSockets and advanced privilege management of Token Certificates.

Besides using the WebSocket protocol, the message text format was also changed. The JAliEn API uses JSON instead of XML. This way the TJAlien plug-in was able to skip the dependency on the legacy *gAPI* library [12]. For JSON it uses the lightweight *JSON-C* [13] library, while for WebSocket support the *libwebsockets* [14] library is used.

The TJAlien plug-in does not require running the session creation command periodically to create X.509 proxies. For the first call of the *TGrid::Connect("alien://")* command from ROOT, TJAlien is loaded because of the *"alien"* protocol specification. It prompts the user to type the password of the Grid identity certificate stored in the default location (*~/.globus*) or specified through *$X509_USER_CERT*. Token Certificate is then automatically issued by JCentral and is passed to the client once the starting credentials are verified. TJAlien keeps the communication channel open with the identity certificate, but from then on will use the token to make subsequent connections. In case of short network glitches, TJAlien will automatically reconnect to the server making several retries, thus excluding the need for human intervention.

For Grid jobs the token is provided by the JobAgent. TJAlien picks it up from the *$JALIEN_TOKEN_CERT* environment variable or the */tmp* directory.

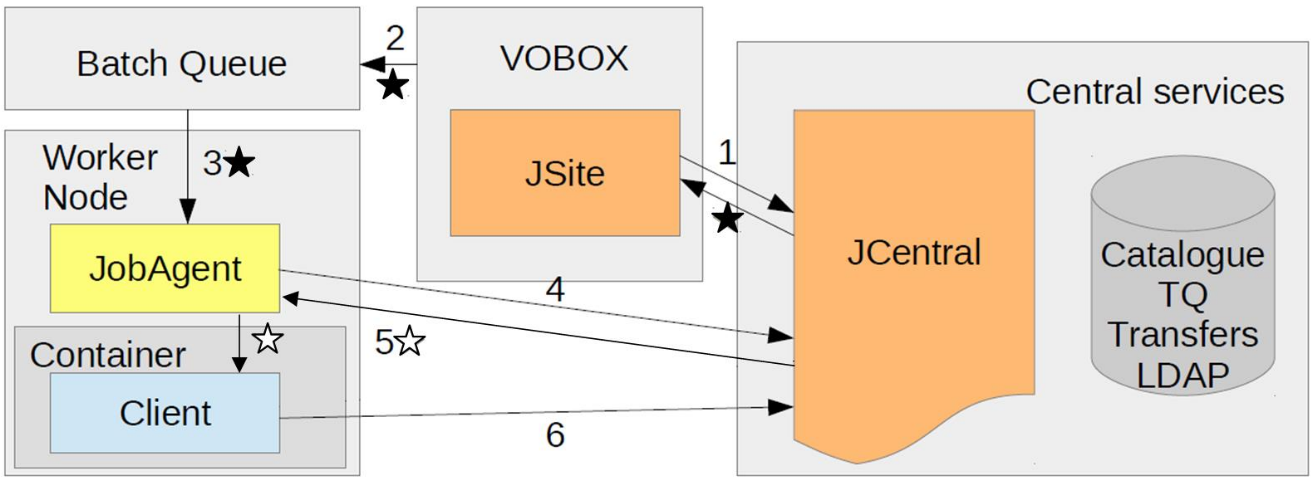

Fig 2. Token certificates for JobAgents in JAliEn

The typical sequence of operations during job submission and execution is described in Figure 2:
1. Request new job agent Token Certificate (★) every 5 minutes;
2. Put job script with JobAgent token into the local site batch queue;
3. Start the JobAgent with generic identity;
4. Use JobAgent token to connect to JCentral and request a job;
5. Get job-specific certificate (user+job ID, ☆) and pass it to the JobWrapper + payload;
6. Connect with job token, execute the payload.

The *TJAlien::Token()* command was introduced to allow the user to request new token certificates, extend their duration and change roles. To renew the token, TJAlien will close the existing connection and ask the user to type the password of the Grid identity certificate.

4.2 JAliEn shell

The JAliEn shell (JSh) is the second interface for accessing JAliEn services. It is implemented in Java and provides a command-line interface (CLI) with history and tab-completion similarly to the AliEn shell utility. JSh follows the same security logic as the TJAlien ROOT plug-in.

The JBox utility was developed to act as a long-living authentication agent for TJAlien and JSh. It uses the Grid identity certificate to make a serialized Java object connection [15] to JCentral and requests a token to create a local WebSocket endpoint using a Tomcat webserver. From then on JBox renews token certificate every two hours automatically. TJAlien plug-in and JSh clients become aware of the JBox endpoint by reading the *jclient* configuration file in */tmp* and take advantage of password-less access for as long as JBox is running.

This is very important for long-lived services on the site frontend machines (VoBoxes). Similarly, users can avoid typing passwords by running the ssh-agent-like service on their machines.

5 Payload isolation

The job pilot in JAliEn consists of two parts. The first is the JobAgent, that is initialized with a Pilot Token Certificate (created and attached to the JobAgent script submitted to the batch system of the site). Its only purpose is job matching – reporting the worker node conditions to JCentral and asking for a user payload. The reply contains a Job Token Certificate together with the job ID and description. The JobAgent starts the second component of the job pilot – the JobWrapper. It is responsible for setting the environment for the payload, downloading input files and creating a sandbox. The JobWrapper maintains a communication channel with the JobAgent through a Java piped I/O stream used for transferring credentials and any sensitive data without leaving any trace in the environment or the file system. The JobWrapper uploads the output files once the payload has finished.

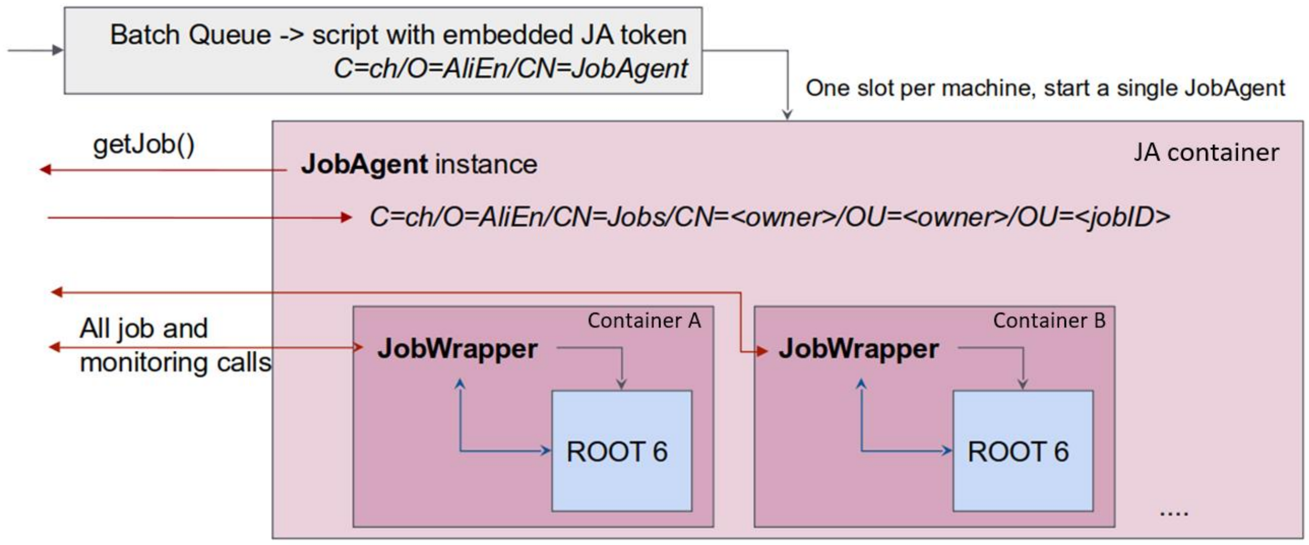
Batch Queue -> script with embedded JA token
C=ch/O=AliEn/CN=JobAgent
One slot per machine, start a single JobAgent
getJob()
JobAgent instance
JA container
C=ch/O=AliEn/CN=Jobs/CN=<owner>/OU=<owner>/OU=<jobID>
All job and monitoring calls
Container A
JobWrapper
ROOT 6
Container B
JobWrapper
ROOT 6
....

Fig 3. Containerization of job execution

An additional isolation may be achieved by usage of containers. Due to the separation of functions between the JobAgent and the JobWrapper, they can be instantiated in different containers, thus making sure none of them has access to resources of the other. When several payloads are started concurrently on the same worker node, containers ensure strict isolation of the sandboxes of each of them. All of the above is illustrated in Figure 3.

6 File access

The data access model in JAliEn follows the existing design, implemented in the AliEn framework. For read and write access a client requests a so-called *Access Envelope*. It is a set of metadata information signed with the central services private key. Additionally, the *Envelope* is encrypted using the Storage Element (SE) public key. The client presents the *Access Envelope* to the destination SE, which decrypts the envelope and checks the signature. Each *Envelope* is unique for an operation on a file, and has a time-limited validity.

7 Conclusions

JAliEn is a new Grid framework, which is partially in production, gradually increasing its presence and will be fully deployed before LHC Run 3. It is built using modern technologies and has better performance and scalability compared to the existing Grid middleware. Several security aspects are addressed in the development of JAliEn, including the separation of authorities between JobAgents and payloads, isolation of sandboxes on worker nodes and better management of user credentials.

The communication channels between the two Grid interfaces (TJAlien-ROOT plug-in and JSh) and JCentral services are based on secure WebSocket endpoints. Token Certificates are used to distinguish and set the necessary security boundaries for JobAgents, Jobs and Users. Each of these groups has its own set of privileges and authorized actions, which are filtered and validated by the central JAliEn services.

By running the JBox service on their machines, users can enjoy the benefits of largely password-less access to the JAliEn ecosystem.

Jobs can achieve enhanced security through containers, which can be organically used together with the new job pilot, consisting of two separate components – the JobAgent and the JobWrapper.

The data access model implements the Access Envelope mechanism, ensuring that only valid user requests (authorized by the central services) are processed by the storages on the sites.

References

1. ALICE Collaboration, K. Aamodt et al., *The ALICE experiment at the CERN LHC*. JINST **3** (2008) S08002
2. ALICE O Project, *O* [software]. Available from http://alice-o2.web.cern.ch/ [accessed 2018-11-15]
3. ALICE Off-line Project, *Java ALICE Environment – JAliEn* [software]. Available from https://gitlab.cern.ch/jalien/jalien [accessed 2018-11-15]
4. A.G. Grigoras, C. Grigoras, M.M. Pedreira, P. Saiz, S. Schreiner. *JAliEn – A new interface between the AliEn jobs and the central services*. Journal of Physics Conference Series **523** (2014)
5. ALICE Off-line Project, *AliEn - ALICE Environment Grid Framework* [software]. Available from http://alien2.cern.ch/ [accessed 2018-11-15]
6. S. Bagnasco, L. Betev, P. Buncic, F. Carminati, C. Cirstoiu, C. Grigoras, A. Hayrapetyan, A. Harutyunyan, A.J. Peters, P. Saiz. *Alien: Alice environment on the grid*. Journal of Physics Conference Series **119**. (2008)
7. V. Welch et al., *X.509 proxy certificates for dynamic delegation*. Proceedings of the 3rd Annual PKI R&D Workshop **14** (2004)
8. *WebSocket communication protocol*. https://en.wikipedia.org/wiki/WebSocket [accessed 2018-11-15]
9. ROOT project, *ROOT* [software]. Available from http://root.cern.ch/ [accessed 2018-11-15]
10. ALICE Off-line Project, *JAliEn implementation of the TGrid interface* [software]. Available from https://gitlab.cern.ch/jalien/jalien-root [accessed 2018-11-15]
11. ALICE Off-line Project, *AliRoot* [software]. Available from http://alice-offline.web.cern.ch/ [accessed 2018-11-15]
12. ARDA project, *ARDA/ALICE packages for API Client+Server* [software]. Available from http://project-arda-dev.web.cern.ch/project-arda-dev/alice/apiservice/ [accessed 2018-11-15]
13. Eric Haszlakiewicz, *JSON-C* [software]. Available from https://github.com/json-c/json-c [accessed 2018-11-15]
14. Andy Green, *Libwebsockets* [software]. Available from https://libwebsockets.org [accessed 2018-11-15]
15. S. Schreiner, C. Grigoras, A. Grigoras, L. Betev, J. Buchmann. *A security architecture for the ALICE grid services*. PoS ISGC **2012:027** (2012)