STATUS OF THE RBAC INFRASTRUCTURE AND LESSONS LEARNT FROM ITS DEPLOYMENT IN LHC

I. Yastrebov, W. Sliwinski, P. Charrue, CERN, Geneva, Switzerland

Abstract

The distributed control system for the LHC accelerator poses many challenges due to its inherent heterogeneity and highly dynamic nature. One of the important aspects is to protect the machine against unauthorised access and unsafe operation of the control system, from the low-level front-end machines up to the high-level control applications running in the control room. In order to prevent an unauthorized access to the control system and accelerator equipment and to address the possible security issues, the Role Based Access Control (RBAC) project was designed and developed at CERN, with a major contribution from Fermilab laboratory. Furthermore, RBAC became an integral part of the CERN Controls Middleware (CMW) infrastructure and it was deployed and commissioned in the LHC operation in the summer 2008, well before the first beam in LHC. This paper presents the current status of the RBAC infrastructure, together with an outcome and gathered experience after a massive deployment in the LHC operation. Moreover, we outline how the project evolved over the last three years and give an overview of the major extensions introduced to improve integration, stability and its functionality. The paper also describes the plans of future project evolution and possible extensions, based on gathered users requirements and operational experience.

INTRODUCTION

In high energy and high intensity accelerators as the LHC, the energy stored in the beams is orders of magnitude above the damage level of accelerator components like magnets. Uncontrolled release of this energy can lead to serious damage of equipment and long machine downtimes. In order to cope with these potential risks CERN has developed a multi-pronged approach for machine safety that includes Role-Based Access Control (RBAC) [1] system. The main objectives of the project are: protection of accelerator equipment and control system against unauthorized access; definition of the operational access rules (who can do what and when) and providing facility to trace and audit the access requests.

The project was started in 2006 and was successfully commissioned and deployed for LHC in 2008, well before the first beam. Nowadays, RBAC protects all LHC equipment and selected experiments equipment against unauthorized access. Since 2008, the project has significantly evolved, following new operational requirements and collected users feedback.

RBAC ARCHITECTURE

Development of RBAC started in 2006, when the core part of the LHC control system was already in place. It was built on top of the existing software components and became an integral part of the Controls Middleware (CMW) [2] project. CMW provides communication infrastructure for all CERN accelerators, which is composed of centrally managed services (e.g. Directory Service) and middleware libraries (client/server) for all operational platforms. It implements two communication models: request-reply and publish-subscribe, organized in client-server relationships. By invoking the device access methods (i.e. *GET*, *SET* and *MONITOR*), clients can read, write and subscribe to device property values. Within this model, authentication (A1) is done on the client side and authorization (A2) is imposed on the server side by interception of the incoming client requests. The overall RBAC architecture including flow of an authentication token, accompanying every request, is illustrated in Figure 1.

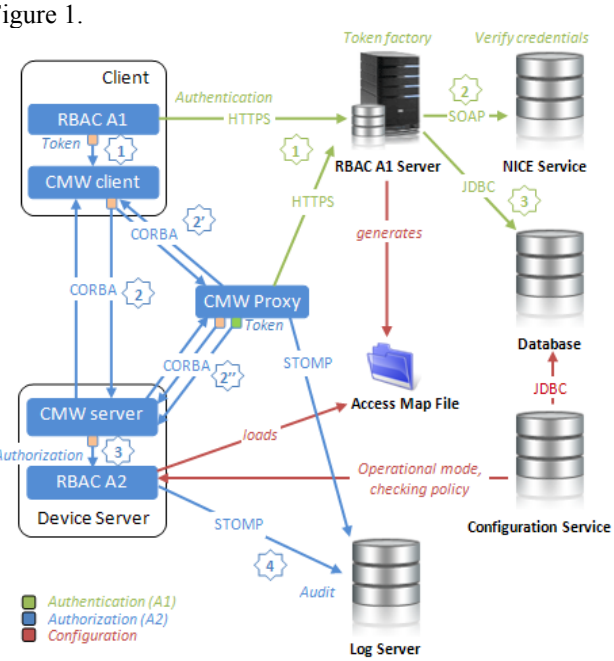

Figure 1: The overall RBAC architecture.

Configuration

Access rules are defined by equipment specialists for every device class. They are stored and managed centrally in the Controls Configuration Database [3]. Access map is a tab-separated text file, located on NFS, inside the CERN's technical network, which is in fact a snapshot of the database access rules for a given CMW device server. Every access map is digitally signed by the central RBAC A1 server to prevent against unauthorised modifications by third parties.

Authentication (A1)

The purpose of authentication is to verify the digital identity of a principal. If the authentication process succeeds, its result is a digitally signed authentication token that is returned to the application. The token is a short-term uniform substitute of the real credentials. It is issued by the central A1 authentication server that can reliably verify the users identity [3]. Authentication is a 3-step process: application sends request over encrypted HTTPS channel (1); authentication A1 server verifies credentials using remote, encrypted SOAP access to the NICE service (2) and then in case of success returns a token containing users roles and other details extracted from the Controls Configuration Database (3) [4].

Authorization (A2)

Authorization is the process of verifying that a known identity has an authority to perform a certain operation. Prior to authorization, a client application has to be authenticated (1). Following, it can use the obtained token to interact with various parts of the control system: access the equipment devices directly (2) or through a proxy (2') or another middle-tier server. Front-ends and the middleware library that are receiving such calls verify the token, thus confirming the identity of the remote party and can use it as a base for authorization (3) [5].

OPERATIONAL DEPLOYMENT IN 2008

During the deployment campaign in 2008, RBAC was successfully integrated in all layers of the LHC control system (see Figure 2).

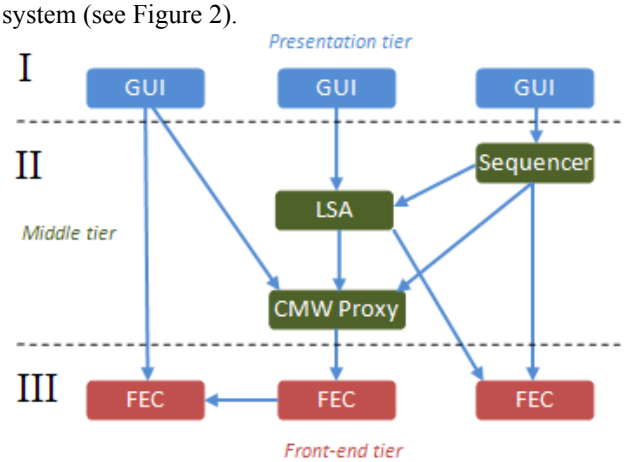

Figure 2: Layers of the Control System using RBAC.

In the presentation tier (I), RBAC authentication library was introduced for client applications running in the CERN Control Centre (CCC), where all CERN accelerators are controlled by the Operation team. For Java clients, RBAC provides login services, Spring (open-source, lightweight container for Java) beans and GUI components targeted for quick and easy integration of RBAC into existing applications.

In the middle tier (II), security components provided by RBAC were integrated in the high-level control systems (e.g. LSA, Sequencer) that help operators and physicists to commission, monitor and control the LHC machine. RBAC became an integral and core component of all major control subsystems including: LHC Software Architecture (LSA) [6], Software Interlock System (SIS) [7], Sequencer [8] and CMW Proxies.

In the front-end layer (III), RBAC integrates with CMW and Front-End Software Architecture (FESA) [9]. At this level the software is written in C++ and it uses RBAC A2 authorization library. Selected front-end machines might also use the A1 authentication service in order to communicate with protected equipment accessible through other front-ends.

To facilitate the introduction of RBAC into such a large and distributed system the algorithm of dynamic authorization was designed. It takes into account not only the access rules, but also an internal state of the subject under question. For RBAC, three different checking policies were introduced: NO-CHECK (no protection, no authentication required), LENIENT (access restriction for protected properties, authentication is optional) and STRICT (authentication and authorization is obligatory). They treat differently the access rules and have different requirements for the authentication token. For each front-end server the associated checking policy is stored in database, from where it is propagated to the running servers by the CMW Configuration Server and it can be changed at runtime from the administrative GUI application. The major advantage of this approach is the possibility to make a phased introduction of the access control in a large, distributed system as LHC.

During the deployment campaign in 2008 and 2009, the STRICT policy was enforced for all equipment systems in LHC and LENIENT policy for all other machines, i.e. injector chain (e.g. PS, SPS).

EVOLUTION OVER LAST 3 YEARS

Over the last 3 years the RBAC infrastructure has been significantly improved in order to comply better with the operational requirements and to provide seamless integration with all layers of the control system. In this section we list the most important changes in the project, which were introduced over the last 3 years.

Temporary Roles

In 2009, the concept of temporary roles was introduced. It brings the possibility to assign a certain role to some user (e.g. piquet role to an equipment expert) and grant him specific access rights but only during the limited intervention time. After that time the role is automatically revoked for that user.

Critical Roles

The Management of Critical Settings (MCS) [10] system is aimed to protect the most critical parameters in either potentially dangerous equipment or protection devices (e.g. Beam Loss Monitors, Collimators). It is complementary to the RBAC infrastructure. Both systems, RBAC and MCS, are fully integrated in the

control system for LHC and SPS and were successfully commissioned already before the first beam in LHC. In addition to RBAC's authentication and authorization MCS provides a mechanism to guarantee data integrity at all times using the digital signatures.

RBAC was extended to provide management of the public/private key pairs (stored in the private, local key store) as a part of the role management module. RBAC database keeps association of the critical MCS roles with the public/private key pairs. The private keys are always kept secret and never leave the RBAC A1 server. The digital signatures are generated by the RBAC A1 server using the private key from the key store. The public keys are made available to front-ends, namely to FESA servers and other applications, which verify the MCS signature.

For MCS roles several additional restrictions were implemented in RBAC: role lifetime constraints and limit of active critical MCS roles in a token.

Operational Mode

During accelerator shutdown or technical stop it is often necessary to allow access to machine for a wider range of users than during normal operation. In such cases an expansion of the access rules is not always desirable and appropriate. Firstly, it may weaken the system security and secondly it requires significant administrative costs. To address this problem notion of the *Operational Mode* was introduced to the access rules as an additional condition, which facilitated extensions of the access rules for non-operational interventions, while keeping operational access rules unchanged.

Support for CMW Proxy

CMW Proxy servers act as an intermediary layer between clients and actual front-end servers. A typical use-case for a proxy is to reduce the number of open connections to the front-end and to reduce the load that is associated with processing of subscriptions. When several clients subscribe on the same property, proxy establishes just one connection to the front-end and then broadcasts updates to all interested clients. The major security issue in this model is how to enforce the access control for subscriptions.

In order to address this problem authentication for CMW Proxies was introduced. At start-up, each proxy performs authentication by location, without using stored credentials and obtains a token that contains the 'CMW-PROXY' role. Access rules for devices working behind a proxy must allow the client subscriptions on desired properties for that role. This approach has several advantages. First, being very simple, efficient and non-intrusive it enforces access control in a single place. Second, it helps equipment specialists to impose usage of a proxy for certain devices thus preventing direct access and performance problems.

Virtual Devices

Virtual device is a special type of device that usually represents a unit of high-level business logic, which is not deployed on a physical front-end machine. A typical use-case for a virtual device is implementation of the high-level control parameters on top of existing physical devices. Since there is no front-end device server, which hosts a virtual device it is problematic to enable authorization. Therefore, RBAC framework was extended to provide a lightweight authorization scheme for any type of device, including both hardware and virtual. In this case authorization is performed by requesting RBAC A1 server that checks privileges for the supplied token.

QUALITY ASSURANCE

After introduction and commissioning of the new extensions, RBAC project team focused on test-driven development for C++ and Java components, in order to improve the overall quality and reliability. In the following paragraphs the detailed description of this process is presented.

Quality Assurance and Testing

A major effort was invested in preparing the tests for all core components. All tests can be split into 3 groups: *unit tests* that verify functionality of individual classes; *integration tests* for libraries that guarantee compatibility between different versions and *system tests* for the whole set of products. System testing verifies that the completely integrated system meets the requirements. For this, the setup called Testbed [11] was used, to test core control components before their operational deployment. For integration testing Bamboo server was used that provides the Continuous Integration environment. Each source code update in the core components triggers recompilation and execution of all unit tests. Next, code is analysed with the Clover tool that computes the code coverage and detects most risky and most complex classes and generates a detailed report. An example report is demonstrated in Figure 3.

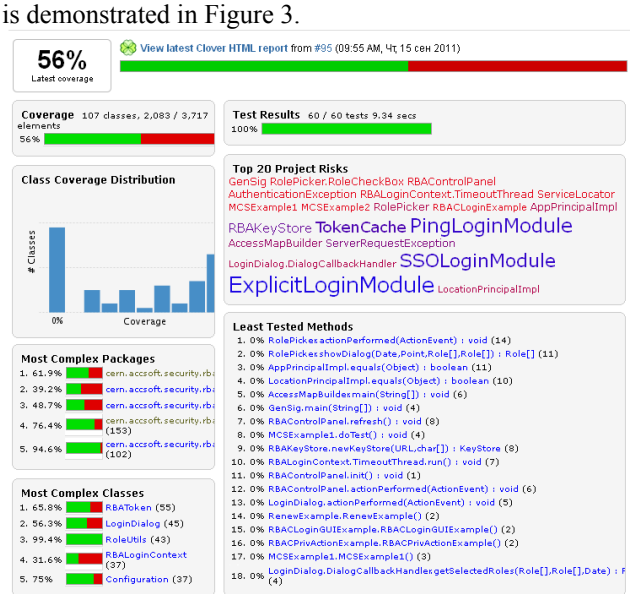

Figure 3: Example of the Clover report for RBAC.

Performance Improvements

Authorization check is performed for each access to front-end, therefore it should execute fast and should not hinder the performance of CMW. Having this in mind, the authorization algorithm was completely redesigned, which resulted in 10 times performance gain.

Codebase Refactoring

During last 3 years a major effort was invested into refactoring of the RBAC codebase. Most components were significantly redesigned and improved. The main aspects of refactoring: removal of code duplication, separation of interfaces from implementation, usage of dependency injection to reduce coupling and make the code more testable, split of heavy classes to make them more cohesive and reusable, code documentation.

For RBAC components written in C++ the new release system based on Maven [12] is used since 2011. It helped to standardise and to unify the development process by introducing versioning and dependency management.

STATUS OF THE PROJECT

The RBAC infrastructure was integrated with all layers of the CERN Control System and successfully deployed and commissioned in LHC.

Collected Experience

Lessons learnt from RBAC deployment in LHC:

1. Close collaboration with equipment experts and the Operation team in CCC was essential for successful deployment and operation of RBAC.
2. The crucial success factor was the staged deployment strategy: new functionality was tested during several 1-3 days LHC dry-runs, then verified during machine checkout and commissioning phases before the final deployment for beam operations.
3. Dynamic authorization based on several checking policies allowed for step-by-step introduction of RBAC without interrupting the running systems.
4. Several administrative tools had to be upgraded to support introduction of RBAC; to provide additional diagnostics and to allow changing checking policies for device servers in the runtime.
5. Better logging and diagnostics at all layers was required. This was a critical requirement for debugging and helped to solve many issues.
6. Database driven approach provided means for enforcing data consistency and simplified development of web-based administrative forms.
7. Configuration of access rules for sub-systems (e.g. BLM, BPM, Collimation) was delegated to equipment experts (reduced administration costs).

Future plans

This paragraph lists the major new features that are planned for implementation in the coming year:

1. Introduce Spring framework in RBAC server.
2. Make authentication server resistant to the database connection failures.
3. Revise the authentication by location mechanism and make it more secure.
4. Automate propagation of the access rules from database to the running device servers.

CONCLUSIONS

The RBAC project was developed at CERN, with a major contribution from Fermilab laboratory. It was successfully deployed and commissioned in LHC operations in the summer 2008.

The system successfully passed many centrally organized tests. The feasibility, performance and overhead of RBAC were experimentally evaluated. The results show that the overhead is acceptable and the chosen approach can be effectively used to enforce access control in the CERN control system.

Currently RBAC is used to protect all LHC equipment and selected equipment of other machines. Nevertheless, there are still few areas where current implementation can be improved and extended in order to expand the area of its applicability.

REFERENCES

[1] S.R. Gysin *et al.*, "Role-Based Access Control for the Accelerator Control System at CERN", ICALEPCS'07, Knoxville, Tennessee, USA.

[2] N. Trofimov *et al.*, "Remote Device Access in the New Accelerator Controls", ICALEPCS'01, San Jose, USA.

[3] A. Petrov *et al.*, "User Authentication for Role-Based Access Control", ICALEPCS'07, Knoxville, Tennessee, USA.

[4] Z. Zaharieva *et al.*, "Database Foundation for the Configuration Management of the CERN Accelerator Controls System", ICALEPCS'11, Grenoble, France.

[5] K. Kostro *et al.*, "Role-Based Authorization in Equipment Access at CERN", ICALEPCS'07, Knoxville, Tennessee, USA.

[6] G. Kruk *et al.*, "LHC Software Architecture (LSA) – Evolution toward LHC beam commissioning", ICALEPCS'07, Knoxville, Tennessee, USA.

[7] J. Wozniak *et al.*, "Software Interlocks System", ICALEPCS'07, Knoxville, Tennessee, USA.

[8] V. Baggiolini *et al.*, "A Sequencer For the LHC Era", ICALEPCS'09, Kobe, Japan.

[9] M. Arruat *et al.*, "Front-End Software Architecture", ICALEPCS'07, Knoxville, Tennessee, USA.

[10] W. Sliwinski *et al.*, "Management of Critical Machine Settings for Accelerators at CERN", ICALEPCS'09, Kobe, Japan.

[11] J. Nguyen Xuan *et al.*, "Testbed for Validating the LHC Controls System Core Before Deployment", ICALEPCS'11, Grenoble, France.

[12] J. Nguyen Xuan *et al.*, "A C/C++ Build System Based on Maven for the LHC Controls System", ICALEPCS'11, Grenoble, France.