Computing and Informatics, Vol. 31, 2012, 1151–1172

SECURING CONTROLS MIDDLEWARE OF THE LARGE HADRON COLLIDER

Ilia Yastrebov, Natalia Yastrebova

European Organization for Nuclear Research (CERN)
CH-1211 Switzerland, Geneva 23
e-mail: {ilia.yastrebov, natalia.yastrebova}@cern.ch

Communicated by Isabel Campos Plasencia

Abstract. The distributed control system of the Large Hadron Collider (LHC) presents many challenges due to its inherent heterogeneity and highly dynamic nature. One critical challenge is providing access control guarantees within the middleware. Role-based access control (RBAC) is a good candidate to provide access control. However, in an equipment control system transactions are often dependent on user context and device context. Unfortunately, classic RBAC cannot be used to handle the above requirements. In this paper we present an extended role-based access control model called CMW-RBAC. This new model incorporates the advantages of role-based permission administration together with a fine-grained control of dynamic context attributes. We also propose a new technique called dynamic authorization that allows phased introduction of access control in large distributed systems. This paper also describes motivation of the project, requirements, and overview of its main components: authentication and authorization.

Keywords: Role-based access control, information security, equipment protection, middleware, distributed systems

Mathematics Subject Classification 2010: 68N19, 68N30, 68U35, 94A62

1 INTRODUCTION

The Large Hadron Collider was built by the European Organization for Nuclear Research (CERN) with the intention of testing various predictions of high-energy physics, including the existence of the hypothesized Higgs boson and of the large

family of new particles predicted by supersymmetry. The unprecedented energy it achieves may even reveal some unexpected results that no one has ever thought of [1].

The LHC is the world's largest and highest-energy particle accelerator. The collider is contained in a circular tunnel, with a circumference of 27 kilometers, at a depth ranging from 50 to 175 meters underground [2]. The LHC uses some of the most powerful dipoles and radio-frequency cavities in existence. The size of the tunnel, magnets, cavities and other essential elements of the machine, represent the main constraints that determine the design energy of 7 TeV per proton beam [3].

The energy stored in the LHC magnets and beam is enormous, and their potential for crippling the machine is a serious concern. That is why European Organization for Nuclear Research has developed a multi-pronged approach for machine safety [4]:

1. Hardware Protection
2. LHC Beam Interlock System
3. Powering Interlock System
4. Software Interlock System
5. Access Control.

As part of the preparation to operate the LHC it has been requested that an access control is implemented for the existing Controls Middleware (CMW). The CMW design reflects the Accelerator Device Model in which devices, named entities in the control system, can be controlled via properties. Each device belongs to a Device Class and it is the Device Class, which defines the properties which can be used to access the device. CMW implements this model in a distributed environment with devices residing in servers that can run anywhere in the controls network. It provides a location-independent and reliable access to the devices from control programs. By invoking the device access methods, clients can read, write and subscribe to device property values [5].

From the point of view of the CMW, each accelerator device is a device server. Control applications maintaining communication with accelerator devices are viewed as CMW clients. Figure 1 shows the architecture of CMW.

Given the significant dangers of the LHC operations, protection against unauthorized or accidental access to the accelerator controls system is required at the level of CMW. There is a need to define among the machine operation and equipment groups, who can do what and when. Access control is a preventative and therefore inexpensive way to protect the accelerator equipment. It keeps users from making wrong settings. Other machine protection systems such as interlocks are reactive and once triggered it is expensive to recover operations.

It is important to mention that access control is not a security system against hackers; it is designed only to prevent well meaning people from making wrong setting, and unauthorized users who have no credentials from running control applications.

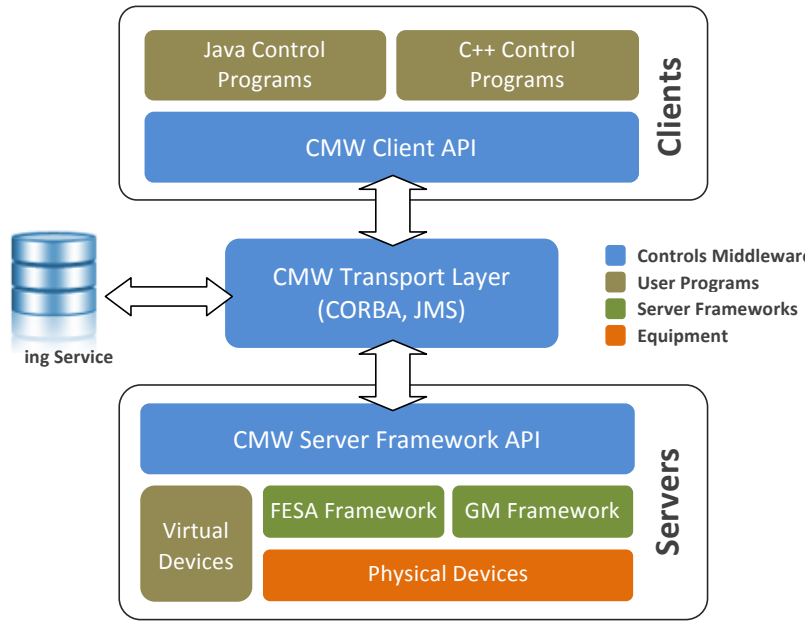

Fig. 1. Controls middleware architecture

2 ACCESS CONTROL REQUIREMENTS

The access control requirements of the distributed middleware system are characterized by four concepts. Firstly, the granting of access rights should be determined by the operation to be performed on the equipment. The CMW architecture imposes a specific device model structure that must be followed also for access control. Secondly, the access control mechanism must be aware of the user and device context. Thirdly, the access control mechanism should support separation of duty policies that may be required for management of critical settings. Finally, a dynamic authorization algorithm is needed to allow phased introduction and flexibility in operation. Each of these requirements is now discussed in more detail.

2.1 Device Model for Access Control

The device model structure of the controls middleware must be taken into account for access control. An access control implementation should allow defining access privileges for the following operations on each device property: read, monitor, and write. In practice, it is the write operation and exceptionally the monitor operation which will be restricted. Access to equipment must be restricted according to access rules defined jointly by the equipment and operation groups.

2.2 Context Information

Access decisions must include other factors that characterize both the user and the device to be accessed. The specific values of those factors form a context that must be taken into account while performing authorization checks. The term "dynamic security attributes" is used in contrast to traditional "static" attributes. Dynamic attribute values must be obtained at the time when an access decision is required while static attribute values are usually obtained when a session is established.

For example, consider a user accessing a device server from outside the CERN. In this case the user's access privileges depend on his location. His privileges will change as his context changes, for example, if he moves from an unauthorized location to an authorized one.

2.3 Separation of Duty

Separation of duty is a security principle used to formulate multi-person control policies. In essence it requires that two or more different people are responsible for the completion of a business process. It would thus, in principle, discourage fraud by requiring a conspiracy, thereby increasing the risk to the potential perpetrators. In order to protect the most sensitive equipment users must have only one critical role at a specific point in time.

2.4 Dynamic Authorization

The equipment has several modes of operation. While a device is working in the test mode it is often necessary to allow access 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 loosen up on the system security, and secondly it requires significant administrative costs. We believe the introduction of different working modes of authorization assists to problem solving. In this case the authorization algorithm will take into account not only access rules, but also the current working mode of the equipment.

CERN has a lot of exposed equipments due to the size of the LHC, which contains hundreds of thousands of different devices with dozens of properties each. Thus it takes a lot of time to design access rules for all devices. As we cannot force equipment specialists to define these rules in a single day, we have to propose a flexible solution for regulating access to non-protected equipments. This will allow us to deploy the access control system step-by-step, without breaking the existing infrastructure. This is a crucial requirement for the LHC access control system at CERN.

3 THE CMW-RBAC MATHEMATICAL MODEL

3.1 Role-Based Access Control

One of the most challenging problems in managing large networks is the complexity of security administration. Role based access control (RBAC), as formalized in 1992 by David Ferraiolo and Rick Kuhn [6], has become the predominant model for advanced access control because it reduces this cost. In 2000, the Ferraiolo-Kuhn model was integrated with the framework of Sandhu et al. [7] to create a unified model for RBAC, published as the NIST RBAC model [8] and adopted as an ANSI/INCITS standard in 2004. Today, most information technology vendors have incorporated RBAC into their product lines, and the technology is finding applications in areas ranging from health care to defense, in addition to the mainstream commerce systems for which it was designed [9].

Role-based access control is an approach to restrict system access to authorized users. Within an organization, roles are created for various job functions. The permissions to perform certain operations are assigned to specific roles. Members of staff (or other system users) are assigned to particular roles, and through those role assignments acquire the permissions to perform particular system functions. The use of RBAC to manage user privileges within a single system or application is widely accepted as a best practice. Systems including Microsoft Active Directory, Microsoft SQL Server, FreeBSD, Solaris, Oracle DBMS, PostgreSQL 8.1, SAP R/3 and many others effectively implement some form of RBAC [9].

3.2 Related Work

The notion of roles helps simplify controlling access to objects, but it has to be used in conjunction with other information, such as device context, client location and so on [10]. Access decisions must include other factors, in particular, relationships between entities, such as the user and the object to be accessed. The term "dynamic security attributes" is used in contrast to traditional static attributes [11]. Values of dynamic attributes must be obtained at the authorization time while values of static attributes are usually obtained when a session is established.

In [12], Wang et al. present a number of considerations for access control in general middleware systems. Their paper also discusses the need to be placed within a continuum from totally secure, accountable systems, to systems with efficient broadcasting and routing of events largely powered by a lack of need for security-related accounting.

A similar introduction is presented in [13]. Miklos devotes significant attention to specifying maximum and minimum security restrictions by assuming an ordering of events. We feel this approach is too restrictive, and unnecessarily prescriptive. Another problem with his work is the apparent lack of implementation details. Our approach provides a more expressive policy language with which to control security.

3.3 Proposed Model for CMW-RBAC

Role-based access control is a good candidate to provide access control. However, in an equipment control system transactions are often dependent on user context and device context. Unfortunately, classic RBAC cannot be used to handle the above requirements. We propose a new model of RBAC for the distributed control system which we will refer to as CMW-RBAC. This concept is based on the standard model and preserves the advantages of scalable security administration that RBAC-style models offer. Moreover it significantly extends the standard RBAC model according to specific requirements and yet offers the flexibility to specify complex access restrictions based on the dynamic security attributes. The new CMW-RBAC model is quite general and flexible and could be used in many other areas for equipment access control. Below we give a formal mathematical description of the model in terms of sets and relations. Our model can be virtually divided into three subsets:

1. RBAC – contains elements of the standard RBAC model
2. CMW – extension of RBAC with middleware Device Model components
3. Context – user context and device context information.

3.3.1 RBAC Subset

- U – a set of users, $\{u_i | i \in \mathbb{N}\}$. The user is either a human user or a computer program.
- R – a set of roles, $\{r_i | i \in \mathbb{N}\}$. Role is a job function or title which defines an authority level.
- P – a set of permissions, $\{p_i | i \in \mathbb{N}\}$. Permission (access rule) is an approval of a mode of access to a resource.
- UA – user assignment: operation which assigns concrete roles to the users.

$$UA \subseteq U \times R. \tag{1}$$

- PA – permission assignment: $R \to 2^p$ – function, defining a set of access rules for each role. This condition must be met at that:

$$\forall p \in P, \exists r \in R : p \in PA(r). \tag{2}$$

As shown in Figure 2, a user may have many roles, as well as a role may belong to many users; a role may have several permissions, and a permission may belong to many roles. There are also sets of administrative roles ADR and permissions ADP that are used to restrict access to administrative operations within CMW-RBAC. Administrative roles are not used for equipment access, as well as regular roles are not used for administration.

- AUA – administrative user assignment: operation which assigns concrete administrative roles to the users.

$$AUA \subseteq U \times ADR. \tag{3}$$

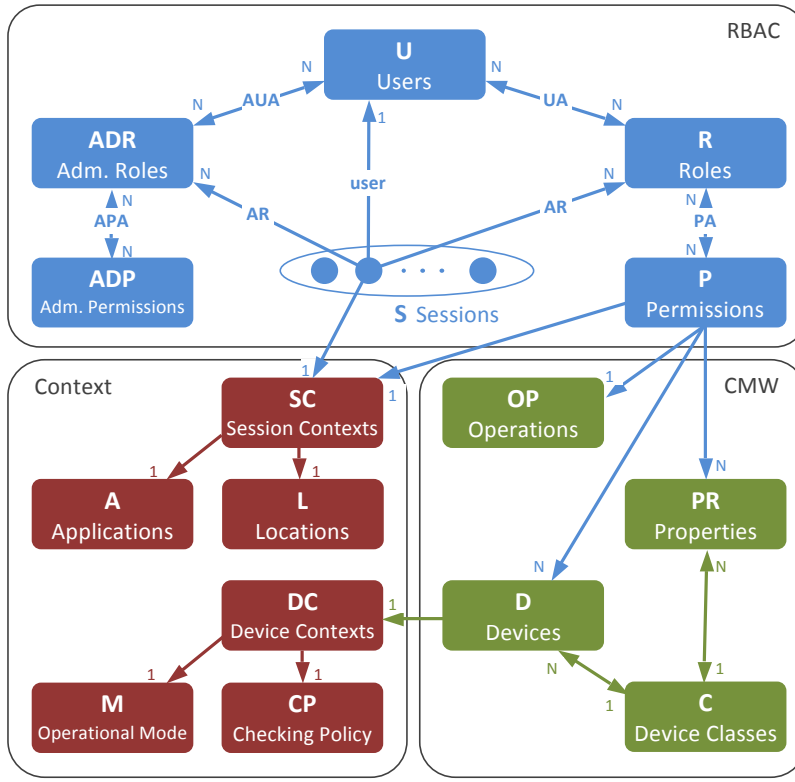

Fig. 2. Mathematical model of CMW-RBAC

- S – a set of sessions, $\{s_i | i \in \mathbb{N}\}$. Session (subject) is mapping of one user to possibly many roles, i.e., a user establishes a session during which the user activates some subset of roles that he or she is a member of. The double-headed arrow from the session to R in Figure 2 indicates that multiple roles are simultaneously activated. The permissions available to the user are the union of permissions from all roles activated in that session. Each session is associated with a single user, as indicated by the single-headed arrow from the session to U in Figure 2. This association remains constant for the life of a session.

$$user(s_i) : S \to U \tag{4}$$

- AR – a set of the subject's active roles (which can change with time). For each subject, the active role is the one that the subject is currently using.

$$AR(s_i) : S \to 2^{R \cup ADR} \tag{5}$$

$$AR(s_i) \subseteq \{r \in R | (user(s_i), r) \in UA \cup AUA\} \tag{6}$$

Sessions are under the control of individual users. As far as the model is concerned, a user can create a session and choose to activate some subset of the user's roles. Roles active in a session can be changed at the user's discretion. The session terminates at the user's initiative. A system may also terminate a session if it is inactive for too long.

3.3.2 CMW Subset

Within the model device is a named entity in the control system. A device can be controlled via its properties. Each property has a name and value. The set of properties specific for the group of homogenous devices forms a device class. The model also defines operation types for work with all environments in the system. These are the get, set and monitor types, which allow to read a value from a device, write a value to a device and monitor a device property.

- T – a set of operation types, $\{get, set, monitor\}$.
- PR – a set of device properties, $\{pr_i | i \in \mathbb{N}\}$.
- C – a set of device classes, $\{c_i | i \in \mathbb{N}\}$.

Function defining a set of properties for each device class:

$$props(c_i) : C \to 2^P R. \tag{7}$$

- D – a set of devices (equipments), $\{d_i | i \in \mathbb{N}\}$.

Now we give a function that for every device from the set D defines the associated device class:

$$class(d_i) : D \to C. \tag{8}$$

The set of properties for a concrete device is defined as:

$$APR(d_i) = \{pr \in PR | (class(d_i), pr) \in props(class(d_i))\}. \tag{9}$$

Hereby one device class may have many properties, and many devices may belong to the same device class. The set of transactions T, which can be executed by user working within the control system:

$$T = \{(ot, pr, d) | ot \in OT, pr \in APR(d), d \in D\}. \tag{10}$$

This implies that the interaction of a user with the control system comes to the execution of operations from the OT set over the properties of some devices. The

function defining a set of transactions for each permission hereafter referred to as transaction assignment TA is:

$$TA(p_i) : P \rightarrow 2^T. \tag{11}$$

Now we define a predicate indicating whether a transaction is protected by access rules:

$$\forall t \in T, (protected(t) = 1 \Leftrightarrow \exists p \in P : t \in TA(p)); \tag{12}$$

that is a transaction is considered to be protected if there is at least one access rule which restricts access to it.

- CP – a set of checking policies implementing a dynamic authorization algorithm, $\{no\_check, lenient, strict\}$. Later we will show how checking policies impact authorization.

3.3.3 Context Subset

In an equipment control system transactions are often dependent on user context and device context. Unfortunately, classic RBAC cannot be used to handle the above requirements. Therefore the RBAC model needs to be extended with context information. In our model capabilities and privileges of a subject not only depend on its identity but also on its current context and context of the target device.

- L – a set of user locations, $\{l_i | i \in \mathbb{N}\}$.
- A – a set of client applications, $\{a_i | i \in \mathbb{N}\}$.

  Elements of these two sets form a Session Context set SC:

$$SC = L \times A \tag{13}$$

$$loc(sc_i) : SC \rightarrow L \tag{14}$$

$$app(sc_i) : SC \rightarrow A. \tag{15}$$

Each session and permission has its own session context.

$$sc(s_i) : S \rightarrow SC \tag{16}$$

$$sc(p_i) : P \rightarrow SC \tag{17}$$

- M – a set of operational modes, $\{operational, non - operational\}$.
- DC – a set of device contexts, $\{dc_i | i \in \mathbb{N}\}$. Each device context has its own operational mode:

$$mode(dc_i) : DC \rightarrow M. \tag{18}$$

Each device and permission has its own device context.

$$dc(d_i) : D \to DC \tag{19}$$

$$dc(p_i) : P \to DC \tag{20}$$

Now we define a special predicate context which determines whether the transaction is potentially executable for the given session:

$$\begin{aligned} \forall s \in S, t \in T, context(s,t) &= 1 \Leftrightarrow \exists p \in P : \\ ((loc(p) &= \emptyset) \vee (loc(p) = loc(sc(s)))) \wedge \\ ((app(p) &= \emptyset) \vee (app(p) = app(sc(s)))) \wedge \\ ((mode(p) &= \emptyset) \vee (mode(p) = mode(dc(d)))). \end{aligned} \tag{21}$$

3.4 Dynamic Separation of Duty

According to the described model, the system allows to create sessions and choose a concrete set of active roles for each session. In the standard model this concept is called dynamic separation of duty [14, 15]. This approach puts additional restrictions on the set of active roles. In particular there are mutually exclusive roles. The same user can be assigned to at most one role in a mutually exclusive set. In the implementation of the CMW-RBAC we define the set of critical roles, which represents one set of all mutually exclusive roles. Critical roles are used to protect the most sensitive equipment. In order to perform an operation permitted for a critical role, the user has to provide an additional certificate.

3.5 Dynamic Authorization Mechanism

Authorization is the function of specifying access rights to resources or services [16]. In our case the object of authorization is the set of transactions T, which a user can execute working within the control system. Dynamic authorization is the algorithm of authorization that takes into account not only access rules, but also the checking policy of the authorization subject. Within CMW-RBAC we developed 3 different checking policies. Each of them represents an authorization algorithm. A checking policy is defined at the level of every device and can be changed at runtime. The checking policy reflects the internal state of the device in terms of authorization.

The function mapping each device from the set D to the associated checking policy is as follows:

$$policy(d_i) : D \to CP \tag{22}$$

We define the predicate *exec* which is true if a subject can execute a transaction at the current time, otherwise it is false. Now we specify the authorization algorithm behavior for each checking policy.

No-check policy grants access for each transaction without any verification. Typically this policy is used at the design stage, when the device interface is not fixed and there are no access rules yet. This policy is also used during the testing phase if needed to permit equipment access for some additional users for a short period of time. This mode could be useful for system debugging or for other activities when it is required to disable CMW-RBAC authorization checks.

$$\forall s \in S, t \in T, exec(no\_check, s, t) = \begin{cases} 1 & protected(t) = 1 \\ 1 & protected(t) = 0 \end{cases}$$

Lenient checking policy implements relaxed authorization. For the protected transactions algorithm access is granted only if the corresponding access rule permits so. That is, in order to deal with protected transactions the user must be authenticated in the system. For unprotected transactions access is not restricted. Typically this policy is used at the testing stage, when access rules exist only for the most critical settings of the equipment. Some of the devices work in this mode permanently, because sometimes it is desirable to restrict access only to some critical settings while keeping others unprotected. This checking policy provides good protection. Switching from no-check to lenient policy is considered as the step towards better security, and all equipment specialists are encouraged to do so. This is an intermediate stage between no-check and strict policy, which allows us to propagate CMW-RBAC in successive steps.

$$exec(lenient, s, t) = \begin{cases} t \in TA(PA(AR(s))) \wedge (context(s, t)) & protected(t) = 1 \\ 1 & protected(t) = 0 \end{cases}$$

Strict checking policy is the most detailed authorization algorithm. It always requires users to be authenticated in the system; otherwise user's requests will be blocked. Access to the protected transactions is granted only if there is an associated access rule. For unprotected transactions access is permitted only for operations of reading and monitoring. Setting new values with an unprotected transaction is not allowed. This checking policy is the strictest in existence at CERN. Our final goal is to propagate this mode as wide as possible, because it provides the highest security. All critical equipments are supposed to work in this mode.

$$exec(strict, s, t) = \begin{cases} t \in & TA(PA(AR(s))) \wedge (context(s, t)) \text{ if } protected(t) = 1 \\ 0 & s = \emptyset \\ 1 & ot \in get, monitor \wedge protected(t) = 0 \\ 0 & ot = set \wedge protected(t) = 0 \end{cases}$$

By introducing a dynamic authorization algorithm we obtained a flexible security system whose behavior can be easily changed at runtime. One of the main

advantages of this approach is that it allows deploying CMW-RBAC step-by-step without interruption of the existing software. Our final goal is to protect every single device, but it is impossible to accomplish this job even in one year taking into account the number of environments. We believe that our CMW-RBAC model is a good example of a flexible authorization system which can be applied to many other very large distributed systems. Based on this new model we developed a software implementation for securing the LHC controls middleware. In the following chapters we provide an overview of the software technical requirements, and describe the implementation of the most important components.

4 TECHNICAL REQUIREMENTS

To distinguish authentication from the closely related term authorization, the shorthand notations A1 (authentication) and A2 (authorization) are occasionally used. In the present paper we list only the most important requirements of the authentication and authorization components of the CMW-RBAC system [17].

4.1 Authentication Requirements

1. Encryption: the credentials used to authenticate the user shall be encrypted when sent over the network.
2. Hardware independence: the programming interface of authentication shall be independent of specialized hardware such as a card reader, fingerprint reader etc.
3. Quick and simple: the operators must be able to log in quickly and easily. Therefore, the method of authentication must be straightforward for the users.
4. Generic Application Programming Interface (API) for CMW-RBAC: to avoid duplication of specific business logic in different applications and stay independent of changes to the data model, the CMW-RBAC software shall provide a common, platform independent API to all the interested systems.
5. Authentication method: authentication shall be done via user name and password from a personal CERN account. The software should allow flexibility for future implementations of Kerberos and/or X.509 certificates.
6. Authentication by location should be implemented as an additional authentication method. This will allow users to be authenticated without providing credentials from a very limited set of trusted machines located in the CERN Control Centre (CCC).
7. Role activation mechanism: must be implemented to allow users to pick up roles from the set of available roles.

4.2 Authorization Requirements

1. Safeguards for authenticated users: CMW-RBAC shall be used for granting authority to read, monitor, and write to devices. CMW-RBAC shall not grant the permission to make a setting when it is denied by the Management of Critical Settings (MCS).
2. Object of authorization: it shall be possible to restrict access, i.e. define access privileges for the following operations on each device property: read, monitor, and write. In practice, it is the write operation and exceptionally the monitor operation which will be restricted.
3. Permission administration: the permissions shall be defined during the design/deployment phase, and authorized administrators shall be able to edit the permissions at any time
4. Logging/tracking: CMW-RBAC shall keep track of all write actions. It shall keep a log of the action stating the user name, the location, the day and time, the property and any parameters used to grant permission.
5. Performance: authorization shall be fast and shall not hinder the performance of the middleware.

5 AUTHENTICATION

The purpose of the CMW-RBAC authentication system is to verify the digital identity of a principal (which is either a human user or a program). This can be accomplished in several ways, described below. In any case, if the authentication succeeds its result is a digitally signed authentication token that is returned to the application [18]. The program can use the token whenever it needs to interact with various parts to the control system. For example, the token can be provided as one of the arguments in a Remote Method Invocation (RMI) call to set a device. Front-ends and the middleware that are receiving such calls will verify the token, thus confirming the identity of the remote party, and can use it as a basis for authorization.

The CMW-RBAC authentication token is a short-term uniform substitute of the real credentials. It is issued by a central service that can reliably verify the user's identity. The central authentication service always signs tokens with a private key to prevent any further modifications. Various recipients of the tokens can validate them quickly and easily with the public key, and use them for making authorization decisions [19].

The following diagram demonstrates the authentication process.

1. The application sends a login request to the CMW-RBAC.
2. CMW-RBAC checks the user location in the database.
3. If location is trusted, then the new token is immediately created and returned to the user.

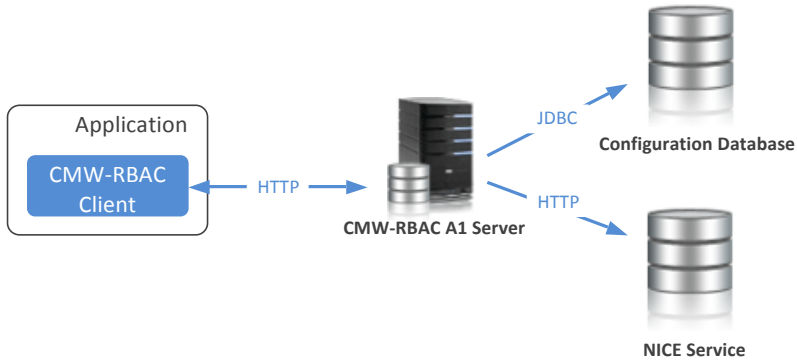

Fig. 3. Authentication process

4. Otherwise the user is required to enter his credentials or choose the certificate and send it to CMW-RBAC.
5. CMW-RBAC verifies the credentials with the CERN Account System (NICE).
6. If the check is successful CMW-RBAC retrieves roles from the database, generates a new token and signs it with private key to prevent any modifications.
7. CMW-RBAC returns a signed token containing the user roles to the application.

The A1 server is implemented in Java. It receives authentication requests via HTTP from multiple clients, returning back either an authentication token, or an error code. Each request from a user contains its credential in some form. All requests are atomic, so no session information is cached by the server. The SSL/TLS protocol is generally used over HTTP to protect the communication between the two parties, and to authenticate the client's X.509 certificate, if provided.

The client side is organized as a library implemented both in C++ and Java, which can be used by other applications or application frameworks. This library provides a function that should be called in order to obtain the authentication token from the server. The Java implementation also provides several standard GUI components, such as a login dialog, role picker dialog, and others. Basically, that client-side authentication library can be used in most applications without changes. The C++ version of the client library is more lightweight because it does not provide any GUI.

5.1 Types of Authentication

Currently CMW-RBAC supports four types of authentication:

1. By user name and password: the user names and passwords are checked against the central NICE account database, via a dedicated web service. No user account information is stored in the CMW-RBAC own database.

2. By X.509 certificate: if the user's X.509 certificate is available, it can be used to perform standard TLS/SSL authentication. Then, the certificate information is used to look up the user name in the CMW-RBAC database.
3. By the network address (also called Authentication by location): certain clients can be authenticated by their IP addresses, using a lookup table in the CMW-RBAC database. Normally, the address authentication is permitted only for a very limited number of machines, such as control room consoles.
4. By using an existing authentication token: any existing token can be used to request a new one, providing that the original token is not expired, bears valid signature, and was issued to the same location address. The lifetime of the new token will not exceed the validity time of the original one.

5.2 Dynamic Separation of Duty

This section describes the implementation of the session mechanism in the CMW-RBAC system. There is a special token type called master-token. This token may be generated by the A1 server, as any other type of the token. The difference is that a master-token does not contain roles at all and that is why it cannot be used for executing operations directly. Instead of roles a master-token contains a description of the possible roles which allows the user to know the list of available roles and activate some of them. When a user picks some roles, a master-token is sent to the A1 server together with the list of the selected roles.

The A1 server verifies both the master-token and the roles and in case of success returns to the user a newly generated token which contains only selected roles. Typically the lifetime of the master-token is longer than regular token (application token). A master-token may be valid up to one year. Thereby the usage of master-tokens allows to activate some roles and helps stay logged in the system longer. This is important for many operator programs which are supposed to stay authenticated permanently, in spite of the several interruptions that occur along the year.

In most cases tokens are stored in the application memory to prevent any usage by other clients. However, in some cases master-tokens are stored on hard disc allowing several applications to use them.

6 AUTHORIZATION

Because the CMW-RBAC project begun when all other parts of the LHC Control Software were already completed, its design was subject to a number of requirements and limitations dictated by the infrastructure in place. Basically, we couldn't change much how the system operated, so CMW-RBAC was built as an additional part on top of the existing components (in particular CMW) [19].

The CMW-RBAC A2 library is part of each and every device server. It is implemented in C++ and compiled for all platforms used at CERN. The A2 server library is called by the CMW server to perform authorization for equipment transactions.

Every operation (get, set or subscription demand) is subject to an authorization process and its execution can be denied when the issuer has insufficient privileges. Other types of operations (e.g. reboot, configuration change) can be also controlled through the RBAC by introducing additional server properties and limiting their access with appropriate access rules.

Each transaction request is made from the application via the CMW client to the CMW server. The CMW-RBAC token obtained at authentication is passed to the CMW client. There the digital signature is verified, and if valid, the token is sent to the CMW server. If the token is not valid a meaningful exception message is returned to the sender. The CMW looks up the permission in the access map, and depending on access rights either grants access or blocks the request. The authorization process is demonstrated in Figure 4.

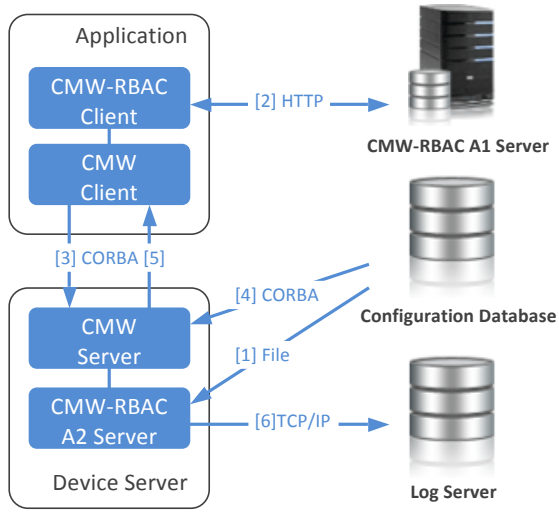

Fig. 4. Authorization process

1. The access map is loaded from a local file upon device server startup.
2. The client application authenticates to the CMW-RBAC A1 server and obtains a valid token.
3. The token is passed to the CMW client and then to the device server via CMW protocol.
4. The CMW server receives the accelerator mode from a timing source.
5. The dynamic authorization algorithm verifies the user permissions and either allows the operation or throws an exception.
6. The result of the authorization process is logged for auditing.

7 ACCESS RULES

Deciding whether a particular operation is valid or not depends on a set of access rules. They are specified by an equipment specialist for every device class, stored and managed centrally in the Controls database. Every CMW server can read access rules (referred to as access map), relative to device classes it is providing access to, through a tab-separated text file located in the Network File System. This file mirrors the access rules located centrally in the database, and is digitally signed by the server to prevent any modifications.

The access map is read by the CMW server on its start-up. In addition, access map can be reloaded upon a remote call from CMW client. It usually happens when the set of related access rules has been altered by an equipment specialist. Access rules are parameterized using all factors related to the authorization process. Apart from specific values, an equipment specialist can put a wildcard '*' in any of the fields except device class and operation. This is interpreted as "all values fit".

The proposed structure of access rules allows straightforward and natural definition of access patterns to devices. Furthermore, we expect an average access map to contain no more than 20-30 rules, which is an easily manageable number.

The authorization process is performed for all transactions, therefore it must be fast and must not penalize significantly the performance of the system. As the time spent in the authorization process was a concern for the CMW operation, it was decided to implement the access map as a tree structure with a separate tree for each operation type. Figure 5 presents the structure of access map tree.

8 PERFORMANCE AND TESTS

An important concern with any design is its performance. CMW-RBAC performs authorization for every transaction within control system and thus authorization algorithm must work as fast as possible. It should not hinder the performance of the middleware. In order to estimate how much different factors affect the performance we made several experiments.

8.1 Architecture

The LHC controls can run in 2-tier mode, meaning that the application and the client are on one machine and the database and devices are accessed directly on the second machine. This configuration is used for development and testing. However when the system is in operation the configuration is usually 3-tier, with the client in a middle tier. A common client is shared by several applications and consolidates requests. In 2-tier, a dedicated client ensures that each request is made from the same application. Therefore the CMW-RBAC token can be passed once per session and the credentials can be used for each subsequent request.

In 3-tier, the requests can come from any application at random times. Thereby the token must be passed for each transaction. This slows down processing be-

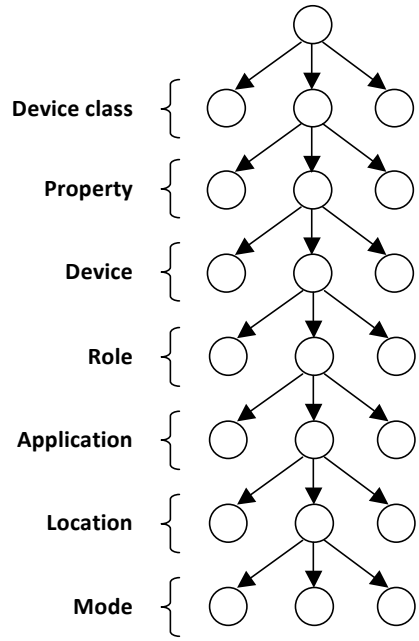

Fig. 5. Access rules structure

cause of additional operations of token serialization and de-serialization. In order to measure the performance we accessed test device server from remote client. The authorization time is the time for the following operations: token serialization, de-serialization and authorization process. The request time is the duration of the whole equipment call. Table 1 demonstrates difference in terms of performance between 2-tier and 3-tier architecture.

	Linux 4 × 2.7 GHz	LynxOS 400 MHz	WinXP 3 GHz
CMW Authorization 2-tier	12.83 μs	201.5 μs	13.44 μs
CMW Request 2-tier	190 μs	2 000 μs	200 μs
CMW Authorization 3-tier	171.73 μs	2 111 μs	182.4 μs
CMW Request 3-tier	400 μs	4 100 μs	410 μs

Table 1. Comparison of 2-tier vs. 3-tier

8.2 Logging

The purpose of this test is to estimate how logging of authorization impacts on performance. In order to evaluate the impact we measured the whole time of the equipment call when logging was disabled and when it was enabled. When access

is granted only some user information (token, location, transaction) is being sent to the log server; but when access is denied then device server also logs the extract of the access map with the access rules that protect given transaction. Throwing a detailed exception to the client is also expensive. The results of this test are presented in Table 2.

	Linux 4×2.7 GHz	LynxOS, 400 MHz
Policy: no-check. Log: off. Access: granted	140 μs	1 900 μs
Policy: no-check. Log: on. Access: granted	200 μs	2 400 μs
Policy: strict. Log: off. Access: granted	150 μs	2 000 μs
Policy: strict. Log: on. Access: granted	220 μs	2 500 μs
Policy: strict. Log: on. Access: denied	390 μs	2 630 μs

Table 2. Logging of authorization result

8.3 Conclusions

Experimental performance evaluation results for the CMW-RBAC system at CERN lead to the following conclusions:

1. 3-tier token verification on every request has a larger impact on performance than the other concerns. In 3-tier configuration the CMW authorization takes 2.1 ms at most. At this time, this is acceptable according to the requirements. The performance decrease is caused mainly by additional operations of token de-serialization and digital signature verification.

2. Logging of each request has a significant effect on the performance. Our tests show the difference between having logging on versus having it off is roughly 20 % when access is granted and even more when access was denied. However, typical device server configuration disables log output for authorized requests, this is only enabled during server debugging. Logging of unauthorized requests is always enabled and cannot be turned off.

3. The size of the access map has little effect on performance due to sophisticated and optimized search algorithms. Time complexity of the authorization algorithm is logarithmic, because of efficient binary search in the access tree. Our test result shows a double increase in authorization time between one rule and 200 rules access map. Further increasing of the access map size has no significant impact on authorization time.

4. The number of roles in the client token influences the performance more significantly than the access map size. Authorization time almost doubled when number of roles increased from 1 to 20.

9 CONCLUSIONS

We have outlined the security design for the Controls Middleware at CERN and and presented CMW-RBAC a novel access control model. This new model incorporates the advantages of role-based permission administration together with a fine-grained control of dynamic context attributes. We also propose a new technique called dynamic authorization that allows phased introduction of access control policies in large distributed systems.

The CMW-RBAC approach was successfully implemented based on the proposed model. The system successfully passed many centrally organized tests. The feasibility, performance and overheads of CMW-RBAC were experimentally evaluated. The results show that the overhead is reasonable and the model can be effectively used for dynamic context-aware access control for distributed controls middleware.

This allows us to assume that the proposed model of the CMW-RBAC could be used in many other areas where access control is needed for large distributed systems. Currently the RBAC system is released in a production version and used by virtually every equipment device at CERN. Thanks to the dynamic authorization algorithm we could propagate the CMW-RBAC step-by-step, without interruption of the legacy subsystems. In the future we expect to extend the CMW-RBAC functionality and applicability domain.

REFERENCES

[1] CERN: Why the LHC. Available on: `http://public.web.cern.ch/public/en/LHC/WhyLHC-en.html`, 2008.

[2] Wikipedia: Large Hadron Collider. Available on: `http://en.wikipedia.org/wiki/Large_Hadron_Collider`, 2008.

[3] CERN: What is LHCb. CERN FAQ LHC: The Guide. CERN Communication Group, available on: `http://cdsmedia.cern.ch/img/CERN-Brochure-2008-001-Eng.pdf`, 2008.

[4] Wenninger, J.: Operational Challenges of the LHC. Available on: `http://irfu.cea.fr/Phocea/Seminaires/1595/Dapnia-Novc07-partB.ppt`, 2007.

[5] Trofimov, N.—Baggiolini, V.—Jensen, S.—Kostro, K.—Di Maio, F.—Risso, A.: Remote Device Access in the New Accelerator Controls. ICALEPCS 2001, San Jose, USA, 27–30 Nov. 2001.

[6] Ferraiolo, D. F.—Kuhn, D. R.: Role Based Access Control. 15th National Computer Security Conference, Baltimore, USA 1992.

[7] Sandhu, R.—Coyne, E. J.—Feinstein, H. L.—Youman, C. E.: Role-Based Access Control Models. IEEE Computer, Vol. 29, 1996, No. 2, p. 3847.

[8] Sandhu, M. R.—Ferraiolo, D. F.—Kuhn, D. R.: The NIST Model for Role Based Access Control: Toward a Unified Standard. 5th ACMWorkshop Role-Based Access Control 2000, p. 4763.

[9] Wikipedia: Role Based Access Control. Available on: `http://en.wikipedia.org/wiki/Role_based_access_control`, 2009.

[10] Beznosov, K.: Requirements for Access Control: US Healthcare Domain. Proceedings of the Third ACM workshop on Role-Based Access Control, October 1998, Fairfax, VA, USA.

[11] Barkley, J.—Beznosov, K.—Uppal, J.: Supporting Relationships in Access Control Using Role Based Access Control. Proceedings of the Fourth ACM Workshop on Role-Based Access Control, October 1999, Fairfax, VA, USA.

[12] Wang, C.—Carzaniga, A.—Evans, D.—Wolf, A.: Security Issues and Requirements in Internet-Scale Publish-Subscribe Systems. In Proceedings of the Thirty-Fifth Annual Hawaii International Conference on System Sciences (HICSS 02), page 303, IEEE, 2002

[13] Miklos, Z.: Towards an Access Control Mechanism for Wide-Area Publish/Subscribe Systems. In International Workshop on Distributed Event-Based Systems, July 2002.

[14] Ahn, G.-J.—Sandhu, R.: The RSL99 Language for Role-Based Separation of Duty Constraints. Proceedings of 4th ACM Workshop on Role-Based Access Control, 1999, pp. 43–54.

[15] Ferraiolo, D. F.—Gilbert, D. M.—Lynch, N.: An Examination of Federal and Commercial Access Control Policy Needs. Proceedings of the 16th NIST-NSA National Computer Security Conference, Baltimore, MD 1993, pp. 20–23.

[16] Wikipedia: Authorization. Available on: `http://en.wikipedia.org/wiki/Authorization`, 2009.

[17] Gysin, S.—Kostro, K.—Kruk, G.—Lamont, M.—Lueders, S.—Sliwinski, W.—Charrue, P.: Role-Based Access for the Accelerator Control System in the LHC Area Requirements, EDMS Id 769302, 2006.

[18] Charrue, P. et al.: Role Based Access Control for the Accelerator Control System in the LHC Era – Design. EDMS Id 805654, 2007.

[19] Petrov, A.—Schumann, C.—Gysin, S.: User Authentication for Role-Based Access Control. Proceedings of ICALEPCS 2007.

[20] Kostro, K.—Gajewski, W.—Gysin, S.: Role-Based Authorization in Equipment Access at CERN. Proceedings of ICALEPCS 2007.

Ilia Yastrebov is the senior software engineer at European Organisation for Nuclear Research. He works in the Beams Department within Controls Middleware team that is providing a common software communication infrastructure for the real-time and distributed control system of CERN accelerators. He received M. Sc. and Ph. D. degrees, both in computer science. He is also (co-)author of scientific books and numerous scientific papers, contributions at international scientific conferences and workshops.

Natalia Yastrebova worked as scientific engineer at European Organisation for Nuclear Research until 2011, now she is a full-time parent. She received M. Sc. in economics in 2004 and Ph. D. degrees in computer science in 2008. She is (co-)author of scientific books and numerous scientific papers, contributions and invited lectures at international scientific conferences and workshops. She has several patents. She also gave lectures at Ulyanovsk State Technical University.