ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TH2A005

SECURE ROLE-BASED ACCESS CONTROL FOR RHIC COMPLEX*

J. Morris, A. Sukhanov†, Brookhaven National Laboratory, Upton, NY, USA

Abstract

This paper describes the requirements, design, and preliminary implementation of Role-Based Access Control (RBAC) for the RHIC Complex. This system is designed to protect from accidental, unauthorized access to equipment of the RHIC Complex. It also provides significant protection against malicious attacks. The role assignment is dynamic: device managers always obtains fresh roles for restricted transactions. The authentication is performed on a dedicated role server, which generates an encrypted token, based on user ID, expiration time, and role level. Device managers are equipped with an authorization mechanism which supports either Static or Dynamic assignment of permissions for device parameters. Transactions with the role server take place atomically during secure set() or get() requests. The system has small overhead: ~0.5 ms for token processing and ~1.5 ms for network round trip. A prototype version of the system has been tested at the RHIC complex since 2022. For easy transition, the access to device managers which do not have authorization mechanisms, can be done through dedicated intermediate shield managers.

INTRODUCTION

The Control System of the RHIC complex [1] provides the operational interface to the RHIC collider and to a long chain of particle accelerators (AGS, Booster, Linac, EBIS, Tandem, CeC, LEReC), including beam injection and extraction lines and beam instrumentation systems. The number of controlled and monitored parameters exceeds 1 million.

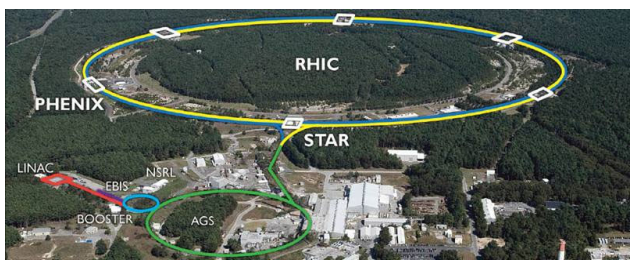

Figure 1: RHIC Complex.

Role-Based Access Control (RBAC) is an approach that limits system access to authorized sets of users [2]. Within an organization, roles are created for various job functions. The permission to perform certain operations is assigned to specific roles. Members of staff (or other system users) are assigned particular roles, and through those role assignments acquire the permissions to perform particular system functions. RBAC is a preventative and therefore inexpensive way to protect accelerator equipment. Other machine protection systems such as interlocks are reactive. Once triggered it can be expensive to recover operations. RBAC can prevent unauthorized users from making incorrect settings which can adversely affect accelerator equipment. RBAC can also be used to ensure machine stability during a run. Once the equipment is fine-tuned and beam is in the machine, an erroneous setting can disrupt operations for hours and valuable data can be lost. RBAC can restrict access to critical settings to a designated set of operators or system experts, reducing the likelihood of an incorrect setting. RBAC role assignments can also be used to determine who is authorized to run control applications.

RHIC CONTROLS SOFTWARE

The RHIC Control System[3] is closed source software, developed at BNL in the 1990s. The original software was all written in C++ and C. The system architecture has stood the test of time with few changes in communications protocols. Java and Python development suites are now part of the Control System. Figure 2 shows the architecture of device control in the RHIC Control System. The accelerator equipment is controlled by Accelerator Device Object (ADO) software modules. ADOs are hosted by dedicated Front-End Computers(FECs) or by ADO Manager processes that can run on many different hardware platforms. An ADO contains a set of related control parameters (similar to EPICS PVs[4]). The communication protocol with clients is RPC[5]. The transport layer is TCPIP. The ADO handles a limited set of requests: info(), get(), set() and subscribe(). The name service, which allows clients to find ADOs of interest in the network, is provided by the ControlsNameServer (CNS).

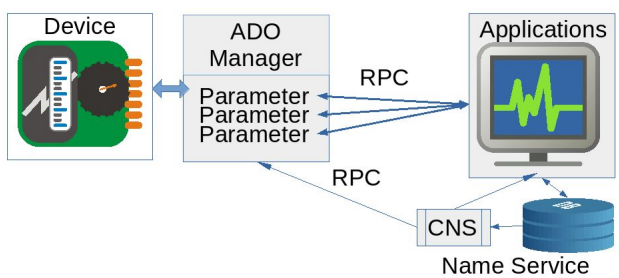

Figure 2: RHIC Controls client-server model.

DEVICE ACCESS CONTROL AT RHIC

The device access policy in the RHIC complex has been based mainly on network restrictions and access

* This work was supported by Brookaven Science Associates, LLC under contract No.DE-SC0012704 with the U.S. Department of Energy.
† sukhanov@bnl.gov

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TH2A005

monitoring rather than access prevention. Though the accelerators in the complex are primarily operated from the Main Control Room, device settings may be sent from other locations in the controls network. Meticulous measures are applied to hardware network security. All wired networking equipment is isolated from the rest of the lab behind the strictly maintained department firewall. Each new device, before being wired to the network, passes a rigorous certification process.

Modifications of editable parameters are logged by a Set History System [6], which stores information about each setting (who, when, what, from where). The system logs several hundred thousand modifications per day with no noticeable impact on application performance. Powerful query tools allow Main Control Room operators to monitor the system and discover unexpected changes.

Software access restriction is provided for certain critical parameters like beam permit system configuration, using a file-based system which only allows unlocking of the parameters only by a small set of operations experts. The system has been very effective but requires non-trivial custom software for each participating system. It is not easily extensible and not considered to be a viable alternative to RBAC for the full control system.

ROLE-BASED ACCESS CONTROL

Role-based access control, as formalized in 1992 by David Ferraiolo and Rick Kuhn [7], has become the predominant model for advanced access control. The RBAC model developed at LHC [8] was widely adopted at other accelerator facilities. The LHC model requires an intermediate layer (Application Server) between the application and device server. The access control restriction process of the LHC model consists of the following steps:

Authentication:
- User sends a request from the Application to be authenticated by the RBAC server.
- RBAC authenticates user using his user name and password or location.
- RBAC returns token to Application.

Authorization:
- Application sends token to the Application Server.
- The Application Server verifies token signature once, and uses the credentials for every subsequent request.
- If access is authorized, the request propagates to device server.

The result of every authorization process, both positive and negative, is logged by the Application Server.
The access rules are managed by a dedicated data base. An equipment specialist has to specify the following fields to define an access rule: 1) device class, 2) property name, 3) device name, 4) role name, 5) application name, 6) location name.

RHIC RBAC

RHIC Controls software allows the implementation of a simpler and more flexible and secure RBAC due to

- full control over communication protocol
- standardized access to all devices through the ADO software modules
- existing Set History Service which already handles the logging function for settings

The following were defined as goals of the RHIC RBAC design:
- Authorization should be primarily based on user login account.
- All controls clients must participate in the RBAC system.
- The restriction mechanism must have minimal impact on performance of sending settings.
- The restriction mechanism can not interfere with reliability of sending settings.
- RBAC protections can be built into ADO Managers in a standard way so that behavior is consistent across ADO Managers and development effort is kept to a minimum.
- The management of the system should be kept as simple as possible, particularly from the point of view of control room operators.
- An emergency override capability must be available to a lead operator in the Main Control Room.

Design Overview

User authentication in the RHIC RBAC design relies on login administration, which is managed and monitored by site system administration and compliant with Cyber Security requirements for national laboratories. No additional authentication is required for RBAC participation. Group access to operational consoles by operator/shift workers is controlled by a ScreenLock process [9] which meets Cyber Security requirements. Once authenticated at a group console, credentials of the shared group account will be used, which is appropriate for operators working in control room environments.

The assignment of user roles is performed by the dedicated role server which is called TokenMan. The determination of whether access to specific device parameters is granted is made at the level of each individual ADO Manager.
The authorization flow of a device setting in our prototype RHIC RBAC system is illustrated in Fig. 3. It involves the following steps:

1. The client Application during its start-up extracts the user ID and group ID of the login account.
2. The user ID and group ID are placed in the authorization credentials structure (Cred) of every RPC packet which is sent from the application to an ADO server. This is in addition

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TH2A005

to the standard fields of the RPC packet which contain information about process ID, program name and version.

3. The ADO Manager retrieves the Cred structure from the RPC packet and sends it to the TokenMan.
4. TokenMan evaluates the user credentials and assigns one or more user roles based on the credentials. It encrypts role information in a token and sends the token back to the ADO Manager.
5. The ADO Manager decrypts the token to retrieve the user roles.
6. The transaction with the device will only be allowed if one of the active roles belongs to the list of roles that are granted permission to access this setting.

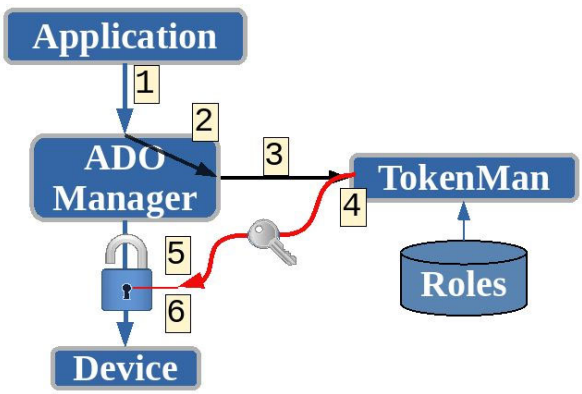

Figure 3: Authorization flow of the RHIC RBAC.

User Roles

The translation from login credentials to user roles is performed by TokenMan, a Python-based ADO Manager which has access to role tables (internal or remote). It generates encrypted tokens based on the credentials passed in requests from ADO Managers.

Users may have multiple roles. The roles are stored in the user-keyed dictionary and have the following fields: 'Basic Roles', 'Elevated Roles' and 'Expiration'. The 'Basic Roles' are the set of roles which users are assigned for routine operations. These are assigned based on user login credentials. To simplify management, it is expected that common roles will be defined that align with login group assignments (e.g. *operator* and *developer* roles). Special roles can be assigned to system experts or lead operators. It is envisioned that these role assignments will be maintained in a database that will be accessed by TokenMan.

An extremely limited *guest* role may be assigned to allow guest or novice users something close to a read-only view of the control system. It is envisioned that guest level protection will be applied at the application level rather than by preventing each setting at the ADO Manager level using the RBAC permission mechanism defined in this paper. This can be done by restricting which applications can be run or by putting applications into read-only mode based on the RBAC defined role of the user.

'Elevated Roles' are roles that are assigned for special operations that take place during a limited period of time. It is envisioned that Main Control Room operators would manage the assignment of Elevated roles to a selected set of users as well as the determination of when those elevated roles expire. For example, a *developer* might be elevated to the role of *operator* or system expert during a short term system commissioning period.

Permission handling in ADO Manager

Permissions are the rules which define which user roles will be granted access to the restricted parameters of an ADO manager. Two types of permissions are supported by the ADO Manager API:

- Static: Permissions are built into the manager code or held in static device configuration.
- Dynamic: A list of permitted roles may be held in a special manager parameter named 'permissions'. An authorized user can change the permitted user roles by changing the values in this list. The 'permissions' parameter of the ADO manager could contain a reference to a parameter of another ADO, which acts as a centralized permission server. This would allow the permitted roles for a large group of managers to be managed in a centralized location. Note that any parameters that are used for dynamic control of permissions (local or centralized) must themselves be protected by RBAC.

In order to keep system management simple, it is expected that static permission assignment will be used for most protected parameters. Elevating a user's role is expected to be the more common method of dynamically providing access.

In testing to date, permission evaluation time is less than 2 ms, which includes 0.5 ms of token processing and ~1.5 ms of network round-trip to TokenMan. That performance impact is considered acceptable but this will be evaluated further as RBAC is deployed more widely. The potential impact of TokenMan transactions on the reliability of sending settings also has to examined.

Comparison with the LHC RBAC model

The primary difference between our proposed RHIC RBAC system and the LHC model is the fact that there is no dedicated authentication server. Client applications extract user, group and program ID using system calls. This is integrated into the RHIC Controls client API.

Acquisition of the role definition token is done on the server side (ADO Manager). This slightly increases the

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TH2A005

transaction time at the server by the round-trip time to the TokenMan but the performance degradation appears to be acceptable. The process of gaining permission in the ADO Manager is stateless and dynamic. The manager is not tracking the expiration time of the token. It simply gets the user roles which are actual at the time of transaction.

The RHIC RBAC provides enhanced security because it does not rely on a middleware (Application Server) layer between applications and device servers. Clients can not bypass the RBAC and exploit low level access to device servers.

In RHIC RBAC, ADO Managers also offer the option of dynamic permission modification based on the value of the dedicated parameter 'permission'. An authorized equipment specialist or operator can specify permissions for individual parameters.

Logging of settings of restricted parameters need not be handled by the RHIC RBAC system because it is handled by the existing Set History system. The very mature and reliable Set History system logs settings and provides operator tools for searching the setting history. Additional logging may be added at the ADO Manager level using the existing ADO message logging system.

Shield ADO Manager

A large number of FECs and ADO managers are built using the C++ toolchain, which does not currently provide server level RBAC support. There are also several EPICS-controlled devices in the RHIC complex. A mechanism will likely be needed to protect devices that do not directly participate in RBAC. The Shield Manager is a Python-based manager with full set of RBAC features. It can act as a bridge to parameters of other managers/FECs or EPICS devices that require protection. The RBAC protection in this model is not as complete as the protection that is built in at the device ADO Manager level. The full strategy for deployment of Shield Managers has yet to be defined.

IMPLEMENTATION AND FUTURE WORK

A prototype version of RBAC functionality has been integrated into the client/server API of the Python development suite of the RHIC Control System. RBAC has been used in a test environment with Python-based ADO managers. Python client applications can all participate in RBAC. The C++ development suite has been upgraded to supply user credentials in the dedicated field of the RPC packet. This means that newly released C++ client applications can also participate in RBAC. Implementation of RBAC support in C++ ADO manager software has just begun.

Future development.

RBAC needs to be more widely tested in Python managers. Support in C++ ADO managers needs to be implemented and tested. Based on the results of testing, some adjustments in RBAC design may be made.

The policy for user role assignment needs to be developed. Though our goal is to keep role management as simple as possible, tools for operations monitoring and management of roles will be required. Development of both policies and tools will have to be done in coordination with operations staff. More experience is needed to determine what will work best for operations.

A commissioning plan for deployment in operational systems needs to be developed. Note that RBAC can be introduced incrementally at the server level with only selected ADO managers participating. In fact, participation may be limited to managers which have a particular need for parameter protection. All client applications must participate, however, for RBAC to be effective.

Performance will need to reassessed as RBAC is deployed on a wider scale. The critical dependency of ADO Managers on the TokenMan servers needs to be carefully reviewed to ensure that failures can not delay or prevent critical settings from being delivered. The emergency override mechanism for lead operators has to be defined.

CONCLUSION

- Role-based access control (RBAC) infrastructure is part of the RHIC Control Python API for clients and servers. It is partially integrated into the C++ API for clients and servers.
- The RBAC system is designed to require little management effort once permissions and roles have been established, but the design allows for dynamic changing of user roles and device permissions when necessary.
- RBAC token transactions have been observed to have a very small effect on server and client performance(2ms per transaction), which would not be noticeable in most system use cases.
- More experience is needed to fully evaluate the performance, reliability and manageability of the RHIC RBAC design.

ACKNOWLEDGEMENTS

The authors would like to thank Sam Clark, Ted D'Ottavio, Peggy Harvey and Seth Nemesure of the Collider-Accelerator Department Controls Software Group and James Jamilkowski of Electron-Ion Collider Controls for their contribution to the RHIC RBAC design process.

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TH2A005

REFERENCES

[1] https://www.bnl.gov/rhic/complex.php

[2] https://en.wikipedia.org/wiki/Role-based_access_control

[3] D.S. Barton *et al.*, "RHIC Control System", *Nucl. Instrum. Methods Phys. Res., Sect. A,* vol. 499, no. 2-3, pp. 356-371, Mar. 2003. doi:10.1016/S0168-9002(02)01943-5

[4] Getting started with EPICS, https://docs.epics-controls.org/en/latest/getting-started/EPICS_Intro.html

[5] RPC: Remote Procedure Call Protocol Specification Version 2, https://datatracker.ietf.org/doc/html/rfc1831

[6] W. Fu, D. P. Ottavio, and T. D, "Tracking Accelerator Settings", in *Proc. ICALEPCS'07*, Oak Ridge, TN, USA, Oct. 2007, paper RPPB22, pp. 653-655.

[7] D.F. Ferraiolo and D.R. Kuhn, "Role-Based Access Control", in *Proc. 15Th National Comput. Secur. Conference,* Baltimore,MD, USA, Oct. 1992. pp. 554–563. doi:10.48550/arXiv.0903.2171

[8] W. Sliwinski. Introduction to RBAC. https://indico.cern.ch/event/625527/attachments/1482155/2298902/RBAC-Overview-BE-ICS-23-June-2017.pdf

[9] S. Binello, T. D, R. A. Katz, and J. Morris, "Cybersecurity and User Accountability in the C-AD Control System", in *Proc. ICALEPCS'07*, Oak Ridge, TN, USA, Oct. 2007, paper WPPB30, pp. 457-459.