U.P.B. Sci. Bull., Series 76, Vol. 1, Iss. 1, 2014 ISSN 2286-3540

A ROLE BASED ACCESS CONTROL SOLUTION FOR LINUX NETWORK

Marius LEAHU[1], Vasile BUZULOIU[2], Dan Alexandru STOICHESCU[3]

Linux networks are widely used nowadays in computers laboratories or clusters as convenient solutions for optimal hardware and software exploitation by the organizations. One of the challenges in managing these networks is the control of users who have different level of expertise and various roles in the organization. This heterogeneity of organization personnel requires a mechanism to control "who accesses what" network node or software application. This article proposes an access control solution for a Linux network built on the Role Based Access Control model which uses well known Linux tools and services: sudo, PAM, LDAP.

Keywords: security, role based access control, hierarchical RBAC, Linux, sudo, PAM, LDAP

1. Introduction

Linux networks are nowadays a common solution in organizations' computers laboratories or clusters for optimal hardware and software exploitation by the organization employees. It allows easy inter communication and data sharing in daily activities performed by the users, but also comes with the risk of damaging or stealing data and resources put in common by the organization through the computers network. It is the goal of computer security field to deal with such risks and the access control domain with its authentication, authorization and audit functions is of prime importance in reducing the security risk.

This article proposes an access control solution for a Linux network which is the result of the author research at CERN[1] in the Trigger and Data Acquisition group of ATLAS Experiment, as member of SysAdmin team[2]. The access control solution implemented in the experiment's computing cluster [3] spreads from operating system level (users login on the cluster's nodes and users access to command line tools) to the application level (the data acquisition

[1]Eng. Ph.D. Student, Faculty of Electronics, Telecommunications and Information Technology, University POLITEHNICA of Bucharest, Romania, e-mail: mleahu@alpha.imag.pub.ro
[2] Prof. Applied Electronics and Informatics Engineering Department, University POLITEHNICA of Bucharest, Romania
[3] Prof., Faculty of Electronics, Telecommunications and Information Technology, University POLITEHNICA of Bucharest, Romania, e-mail: stoich@elia.pub.ro

software running on the computing cluster restricts access to its sensitive functions) and is built on the Role Based Access Control model. This paper focuses on the access control at the level of the operating system, namely the Scientific Linux CERN 5[4] version of Linux.

The following chapters describe the Role Based Access Control model's principles and terminology applied in the solution, the requirements to be fulfilled, then continues with the details of design and implementation of the access control solution.

2. The Role Based Access Control model

An **access control system** regulates the operations that can be executed on data and resources to be protected. Its goal is to control operations executed by subjects in order to prevent actions that could damage or steal data and resources.

Authorization and authentication are fundamental to access control. **Authentication** is the process of determining *who you are* (that user's claimed identity is legitimate), while **authorization** determines *what you are allowed to do*. Note that authorization necessarily depends on proper authentication. If the system cannot be certain of a user's identity, there is no valid way of determining if the user should be granted access.

An access control model in general and the **Role Based Access Control (RBAC)** model in particular as the subject of this article are focused on the authorization part of an access control system.

After a long history of various approaches on the RBAC model, the first step in the direction of standardization was done by Sandhu, Ferraiolo and Kuhn in 2000 [5]. They define consolidated RBAC model for proposed industry standard which has been adopted by the American National Standards Institute, International Committee for Information Technology Standards (ANSI/INCITS) in 2003 as an industry consensus standard INCITS 359:2004.

RBAC provides a valuable level of abstraction to promote security administration at a business enterprise level rather than at the user identity level. The basic role concept is simple: establish permissions based on the functional roles in the enterprise, and then appropriately assign users to a role or set of roles. With RBAC, access decisions are based on the roles individual users have as part of an enterprise. Roles could represent the tasks, responsibilities and qualifications associated with an enterprise. Because the roles within an enterprise are relatively persistent with respect to user turnover and task re-assignment, RBAC provides a powerful mechanism for reducing the complexity, cost and potential for error in assigning user permissions within the enterprise.

The **hierarchical RBAC** (Fig. 1) is one of the RBAC models proposed for standardization [5] and used in the RBAC solution proposed by this article. The drawing below shows three sets of entities called users, roles, and permissions. A **user** in this model is a human being or other autonomous agent such as a process or a computer. A **role** is a job function or job title within the organization with some associated semantics regarding the authority and responsibility conferred on a member of the role. A **permission** is an approval of a particular mode of access to one or more objects in the system. The terms authorization, access right and privilege are also used in the literature to denote a permission. Permissions are always *positive* and confer the ability to the holder of the permission to perform some action(s) in the system. The relationships **user-role assignment** and **permission-role assignment** are *many-to-many* relations. The relationship **role hierarchy** can be also *many-to-many*, but this depends very much on the organization as the role hierarchies are a natural means for structuring roles to reflect an organization's lines of authority and responsibility.

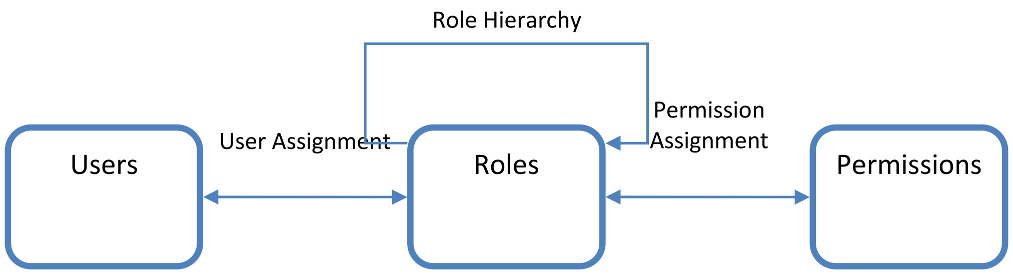


Fig. 1 Hierarchical RBAC

The convention in roles hierarchy diagrams is that more powerful (or *senior*) roles are shown toward the top of diagrams, and less powerful (or *junior*) roles toward the bottom.

The roles hierarchy example depicted in Fig. 2 will be used in the next chapters of this article to demonstrate how the access control solution is designed and implemented in a Linux network. The example is composed of 2 types of roles hierarchies:

- *Tree hierarchy* (e.g. Shift Leader, TDAQ Shifter, DCS Shifter): senior roles aggregate the permissions of junior roles. Trees are good for *aggregation* but do not support sharing.
- *Inverted tree hierarchy* (e.g. Observer, TDAQ Shifter, DCS Shifter): senior roles are shown towards the top with edges connecting them to junior roles. The inverted tree facilitates *sharing* of resources. Resources made available to the junior role are also available to senior roles.

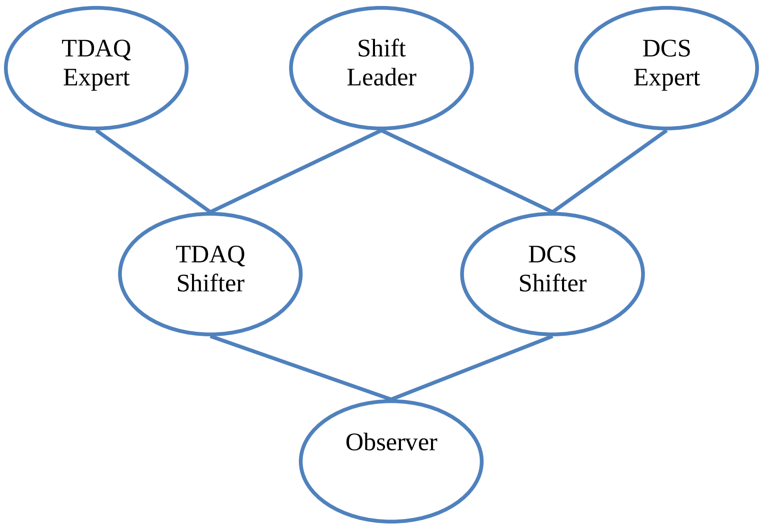


Fig. 2 Example of roles hierarchy

There can be more interpretation of how permissions inheritance goes from a role to another in a role hierarchy. The senior roles in the roles hierarchy example are regarded as inheriting permissions from juniors. This is called the *permission-inheritance* interpretation and the hierarchy is called an inheritance hierarchy. When TDAQ Expert role is activated, the permissions assigned to TDAQ Shifter and Observer roles are all available for use.

3. Access control needs in a Linux network

The access control solution proposed by this article assumes that the Linux network (Fig. 3) is composed of two types of nodes:

- *Servers*: these nodes hosts various services used by all the other nodes in the cluster. For example: a network file sharing service, print service, web, email, directory service etc. There can be one or more server nodes running centralized services for the network. Since the server nodes run services for the whole network, they are critical for the network operation, hence only a limited number of users (SysAdmin team) should have direct access to them.
- *Workstations*: these are client nodes where the users perform their daily work using also the services exposed by the servers. Their main characteristic is the heterogeneity:
    - o The hardware can vary from one node to another (from desktop configuration to “special” hardware, e.g. data acquisition cards installed or connected to the node). It is obvious that only experts or

qualified technicians should be allowed to use the special nodes, while the other nodes for general use can be accessed by more employees.

- Software with limited number of licenses is not installed on all nodes. Also, only a few users may use such software and access only the nodes with that software installed.
- The organization may decide that each department can use exclusively a subset of network nodes but, in the same time, a pool of nodes should be shared by more departments.

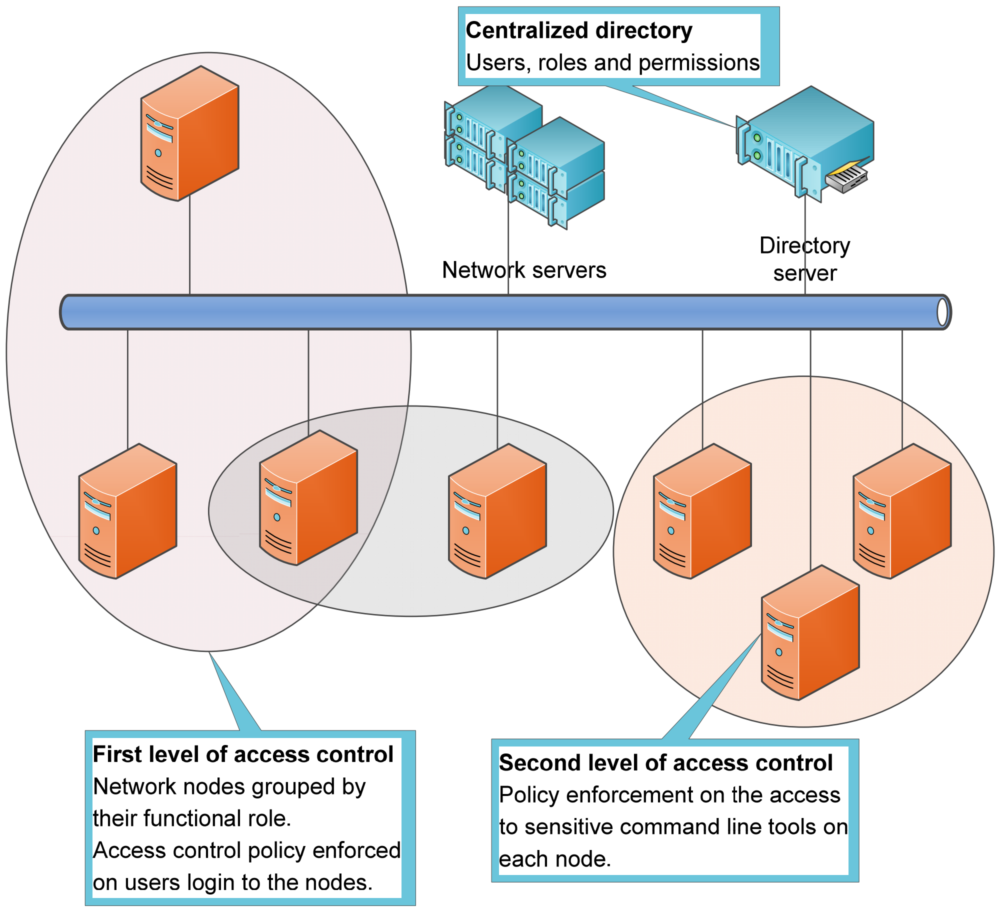

Centralized directory
Users, roles and permissions
Network servers
Directory server
First level of access control
Network nodes grouped by their functional role.
Access control policy enforced on users login to the nodes.
Second level of access control
Policy enforcement on the access to sensitive command line tools on each node.

Fig. 3 Access control levels in a Linux network

The access control solution must be able to enforce the constraints enumerated above, but also should allow sufficient flexibility for the organization in the allocation of its resources to the internal departments. In order to fulfill these requirements, the access control is designed on the RBAC model and structured in two levels of granularity:

- First level of access control regulates the access of the user to the node itself, meaning that a user is allowed to open either a local or a remote shell (ssh [6]) to the node. This allows the organization to allocate nodes exclusively to teams or departments.
- Second level of access control is in the scope of a node. It is assumed that the user passed the first level, so he/she has access to the node. Once logged in, the user may be allowed or not to use all the software available on the node.

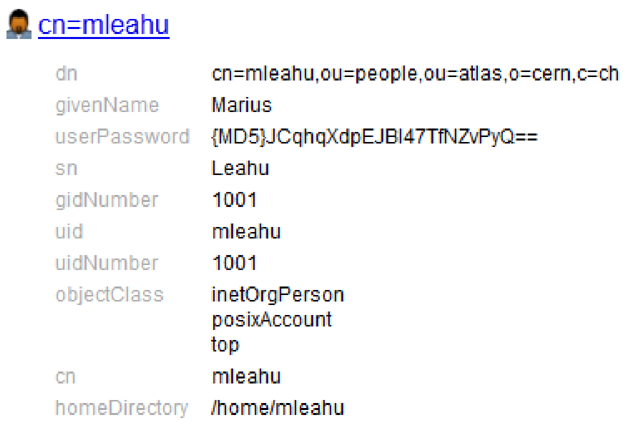
For easier administration of access control solution, the RBAC configuration (users, roles and permissions and assignments) is centralized in Directory Service deployed in the Linux network's servers.

The following chapters describe how the access control solution is implemented on the SLC 5 [4] Linux version.

4. Centralized RBAC configuration

The RBAC configuration is centralized in an LDAP Directory for easier maintenance of nodes configuration. Therefore, all entities specific to Hierarchical RBAC model (Fig. 1) (users, roles, permissions) and relationships (user assignment to roles, permissions assignment to role, role hierarchies) are defined in an OpenLDAP[7] server hosted on one of the network servers.

The **users** are defined in the LDAP directory as *posixAccount* objectclass so that the standard PAM[8] authentication on the Linux nodes to be able to recognize them as user accounts on the node. Fig. 4 shows an example of user definition in LDAP as it can be viewed with phpLDAPAdmin[9] browser.



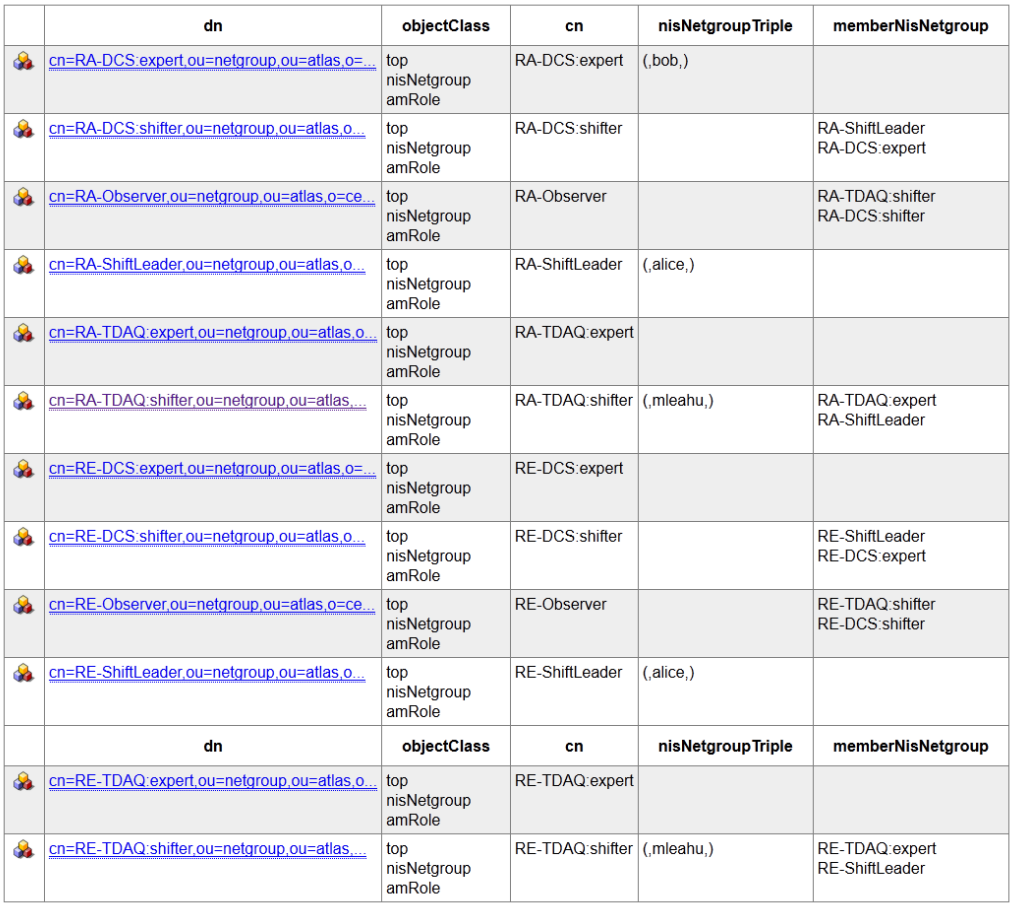
cn=mleahu

dn	cn=mleahu,ou=people,ou=atlas,o=cern,c=ch
givenName	Marius
userPassword	{MD5}JCqhqXdpEJBI47TfNZvPyQ==
sn	Leahu
gidNumber	1001
uid	mleahu
uidNumber	1001
objectClass	inetOrgPerson posixAccount top
cn	mleahu
homeDirectory	/home/mleahu

Fig. 4 User definition in LDAP

The **roles** are mapped in LDAP as *nisNetgroup* object class and *amRole* object class. The use of NIS netgroups[10] brings the following advantages:

- Out of the box integration with many Linux tools (e.g. Sudo[11]) and mechanisms (e.g. various PAM modules are able to work with netgroups), hence the roles are seen as natural components in Linux environment
- Aggregation of more netgroups in one netgroup. This is very helpful to set up the role hierarchies and permission inheritance over the hierarchy.



	dn	objectClass	cn	nisNetgroupTriple	memberNisNetgroup
	cn=RA-DCS:expert,ou=netgroup,ou=atlas,o=...	top nisNetgroup amRole	RA-DCS:expert	(,bob,)	
	cn=RA-DCS:shifter,ou=netgroup,ou=atlas,o...	top nisNetgroup amRole	RA-DCS:shifter		RA-ShiftLeader RA-DCS:expert
	cn=RA-Observer,ou=netgroup,ou=atlas,o=ce...	top nisNetgroup amRole	RA-Observer		RA-TDAQ:shifter RA-DCS:shifter
	cn=RA-ShiftLeader,ou=netgroup,ou=atlas,o...	top nisNetgroup amRole	RA-ShiftLeader	(,alice,)	
	cn=RA-TDAQ:expert,ou=netgroup,ou=atlas,o...	top nisNetgroup amRole	RA-TDAQ:expert		
	cn=RA-TDAQ:shifter,ou=netgroup,ou=atlas,...	top nisNetgroup amRole	RA-TDAQ:shifter	(,mleahu,)	RA-TDAQ:expert RA-ShiftLeader
	cn=RE-DCS:expert,ou=netgroup,ou=atlas,o=...	top nisNetgroup amRole	RE-DCS:expert		
	cn=RE-DCS:shifter,ou=netgroup,ou=atlas,o...	top nisNetgroup amRole	RE-DCS:shifter		RE-ShiftLeader RE-DCS:expert
	cn=RE-Observer,ou=netgroup,ou=atlas,o=ce...	top nisNetgroup amRole	RE-Observer		RE-TDAQ:shifter RE-DCS:shifter
	cn=RE-ShiftLeader,ou=netgroup,ou=atlas,o...	top nisNetgroup amRole	RE-ShiftLeader	(,alice,)	
	dn	objectClass	cn	nisNetgroupTriple	memberNisNetgroup
	cn=RE-TDAQ:expert,ou=netgroup,ou=atlas,o...	top nisNetgroup amRole	RE-TDAQ:expert		
	cn=RE-TDAQ:shifter,ou=netgroup,ou=atlas,...	top nisNetgroup amRole	RE-TDAQ:shifter	(,mleahu,)	RE-TDAQ:expert RE-ShiftLeader

Fig. 5 Roles and role hierarchy definitions in LDAP

The *amRole* object class is specially defined for the RBAC configuration in LDAP for the following reasons:

- Label the NIS netgroups defined in LDAP with the “role” qualifier for easier differentiation in ldap queries
- Attach more properties to netgroups useful in more advanced RBAC models definition, such as: if the role is assignable to users or not (in that case, the role is just internally in the hierarchy just to allow permissions sharing);

constraints like Static Separation of Duties or Dynamic Separation of Duties (not in the scope of this article).

Fig. 5 shows the role definition in LDAP as netgroups for the example of role hierarchy from Fig. 2. The columns from the table represent:

- *dn*: the "path" to the object in the LDAP DB
- *objectClass*: specifies a set of attributes used to describe an object; in our case, the object classes described above are mandatory
- *cn*: the role name prefixed with RA or RE. There are two netgroups defined in LDAP for each role:
    - RA-<rolename> shows that the <rolename> is assigned to an user
    - RE-<rolename> shows that the <rolename> is enabled for an user (more details about assigned/enabled states in user assignment paragraph).
- *nisNetgroupTriple*: used for user assignment relationship
- *memberNisNetgroup*: used for role hierarchy relationship

The **permissions** and **permission assignment** relations are defined in LDAP as specific configuration for the tools to be used to enforce the two levels of access control described in chapter 3. The configuration and enforcement of these two levels of access control is detailed in chapters 5 and 6.

The **user assignment** to roles is accomplished by setting role's netgroup property *nisNetgroupTriple* to the user name in the format (,<username>,). The roles assigned to a user have 2 states:

- *assigned*: this is the initial state when the user is certified for the role. For example, a user who graduated training on DCS technology is recognized as expert in DCS department, hence he gets the DCS expert role assigned.
- *enabled*: a role already assigned to a user is enabled so that the user is able to perform the tasks allowed by this role. The user with DCS expert role can work on DCS hardware only when his role is enabled. In this way, the exclusive access to resources can be regulated by the group leaders.

The example in Fig. 5 has three users with the following roles assignment:

- roles assigned: mleahu – TDAQ:shifter, alice – ShiftLeader, bob – DCS:Expert
- roles enabled: mleahu – TDAQ:shifter, alice – ShiftLeader, bob – no roles enabled.

The **role hierarchy** is mapped to the netgroup aggregation relationship. This is configured in LDAP thanks to the memberNisNetgroup attribute of a role: the value of this attribute represents a *senior* role for the current role. For example, the role DCS shifter has senior roles ShiftLeader and DCS expert as values of its *memberNisNetgroup* attribute; on the other hand, DCS shifter role is senior for Observer, hence Observer's *memberNisNetgroup* value contains it. The role hierarchy managed through netgroups permits *permission inheritance* from junior to senior roles. This means that permission assigned to a role Observer is allowed to users with the role Observer and all Observer's seniors (direct or indirect). The following queries list the users who will get a permission assigned to a certain role for the example in Fig. 5 :

```
[root@localhost]# getent netgroup RA-Observer
RA-Observer          ( , bob, ) ( , alice, ) ( , mleahu, )
[root@localhost]# getent netgroup RE-Observer
RE-Observer          ( , alice, ) ( , mleahu, )
[root@localhost]# getent netgroup RA-TDAQ:shifter
RA-TDAQ:shifter      ( , mleahu, ) ( , alice, )
[root@localhost]# getent netgroup RA-DCS:shifter
RA-DCS:shifter       ( , bob, ) ( , alice, )
[root@localhost]# getent netgroup RE-DCS:shifter
RE-DCS:shifter       ( , alice, )
```

In order to configure the RBAC in LDAP directory as described above, the following configuration checks or adjustments are necessary after the installation of OpenLDAP[7] package on SLC 5 server:

- `nis.schema` must be included in the server configuration file (`slapd.conf`). This schema is necessary to support the NIS netgroups information in the LDAP structure[12].
- `sudo.schema`[13] (shipped also in the sudo package in `/usr/share/doc/sudo-1.7.2p1/schema.OpenLDAP`) must be included in the server configuration to allow the definitions of sudo permissions (more details in the following chapter).
- optionally, the *amRBAC* schema, a custom made LDAP schema which is used to mark the netgroups as roles to be more convenient for shell scripts to identify the RBAC roles in LDAP.

The following chapters will focus on how the policies are defined for the two levels of access control (Fig. 3) and how the enforcement of these policies is done on the network nodes.

5. Controlling the users login on the network nodes

The first level of access control (Fig. 3) makes sure that users are allowed to log in (either locally from the computer's console or remotely via *ssh*[6]) only on the machines which are allocated to groups they belong in the organization.

In order to make a fresh SLC 5 node aware of centralized LDAP configuration, its local services must be adjusted to look up the LDAP server to read their configuration and other information they may need. The LDAP support is enabled on the "Authentication Configuration" (as shown in Fig. 6) for "User Information", "Authentication" and other Options (especially the authorization based on `access.conf`). After this change, the following configuration files are updated:

- the LDAP configuration file `/etc/ldap.conf` used by all tools from the current node that need to know the identifier of central LDAP server. Its content can be like the following one:

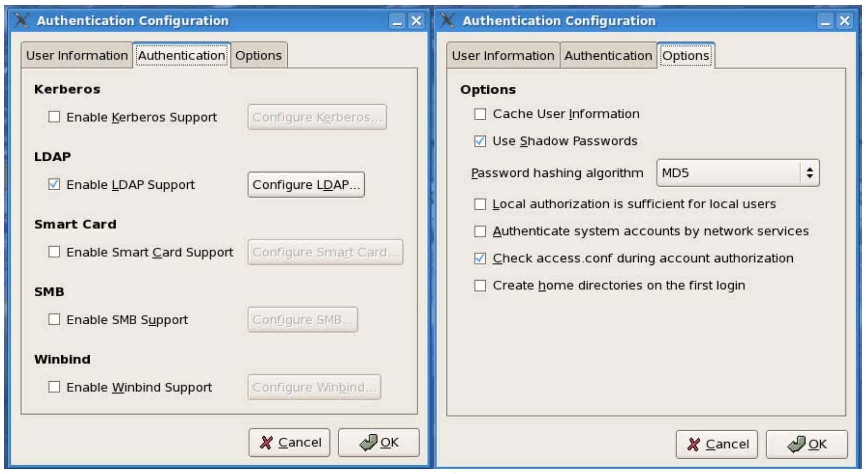
```
# the server hostname where LDAP service is installed
host localhost

# the Base Distinguish Name to use on the LDAP service
base ou=atlas,o=cern,c=ch
```

- the PAM generic configuration `/etc/pam.d/system-auth` includes calls to `pam_ldap.so` plugin in all its stacks (*auth*, *account*, *password*, *session*). Consequently, all system services (e.g. sshd, sudo, login) making use of PAM stack for their user authentication, account information, user password management and user session management will access the LDAP server as a central information repository. The `pam_access.so` is also added in the *account* stack to enforce the authorization rules defined in the `/etc/security/access.conf`. The *account* stack definition looks like this:

```
account  required      pam_access.so
account  required      pam_unix.so broken_shadow
account  sufficient    pam_succeed_if.so uid < 500 quiet
account[default=bad      success=ok       user_unknown=ignore]
pam_ldap.so
account  required      pam_permit.so
```

- the Name Service Switch[14] configuration file `/etc/nsswitch.conf` has LDAP as second source of information for system services. For example, at least the following services will lookup LDAP:

```
passwd:     files ldap
shadow:     files ldap
group:      files ldap
netgroup:   ldap
automount:  files ldap
sudoers:  files ldap
```



Kerberos
- ☐ Enable Kerberos Support — Configure Kerberos...

LDAP
- ☑ Enable LDAP Support — Configure LDAP...

Smart Card
- ☐ Enable Smart Card Support — Configure Smart Card...

SMB
- ☐ Enable SMB Support — Configure SMB...

Winbind
- ☐ Enable Winbind Support — Configure Winbind...

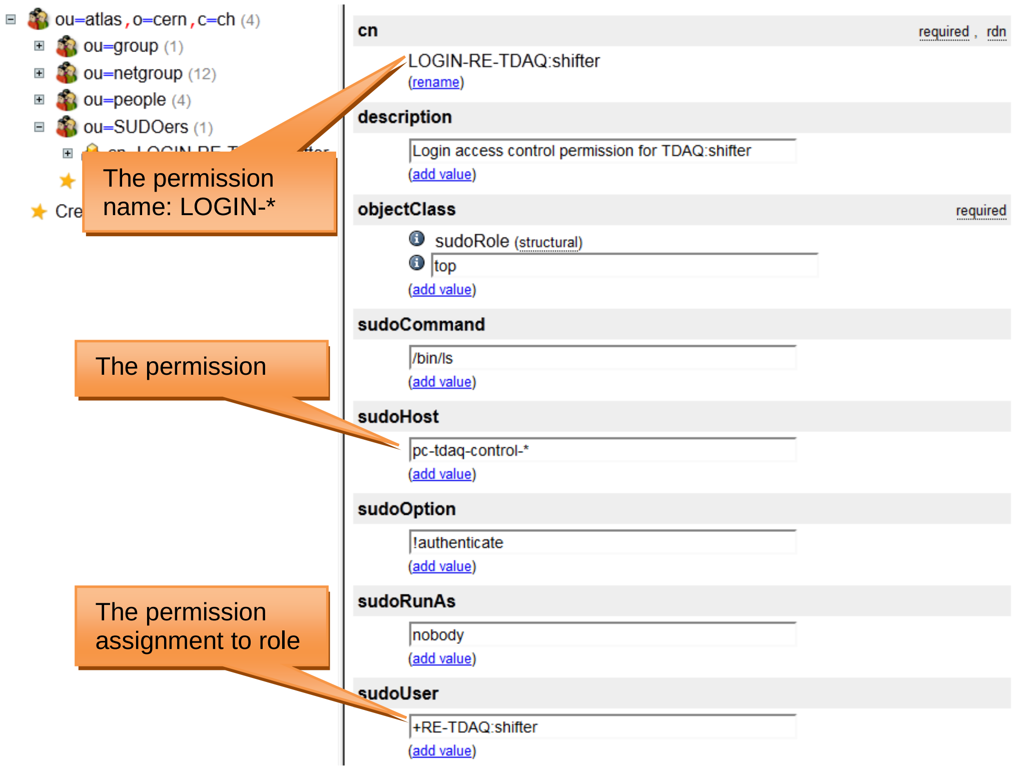
Options
- ☐ Cache User Information
- ☑ Use Shadow Passwords
- Password hashing algorithm: MD5
- ☐ Local authorization is sufficient for local users
- ☐ Authenticate system accounts by network services
- ☑ Check access.conf during account authorization
- ☐ Create home directories on the first login

Fig. 6 "Authentication Configuration" in SLC5

At this stage, the node is ready to enforce login restrictions based on the permissions defined in `/etc/security/access.conf` [15].

The **permissions** and **permission assignment** to role from our RBAC model are centralized in LDAP in the form of sudo roles. The example in Fig. 7 shows the main characteristics of a sudo role definition that transforms it in a login restriction definition:

- sudo role name which represents the *permission name* must start with LOGIN- keyword.
- *permission* is defined as the hostname where the login is allowed. The hostname can be also a simple regular expression (for example, all the nodes with the hostname starting with `pc-tdaq-control-`), thus making the permission valid for a group of machines obeying a hostname naming convention (e.g. the hostname can have the structure <type of machine: pc/sbc>-<group name>-<subgroup>-<index>).
- *permission assignment to role* is given by the netgroup name specified in the sudoUser field.
- the other attributes of a sudo role in LDAP are set to default values as in the example below.



```
cn                                   required, rdn
LOGIN-RE-TDAQ:shifter
(rename)
description
Login access control permission for TDAQ:shifter
(add value)
objectClass                          required
sudoRole (structural)
top
(add value)
sudoCommand
/bin/ls
(add value)
sudoHost
pc-tdaq-control-*
(add value)
sudoOption
!authenticate
(add value)
sudoRunAs
nobody
(add value)
sudoUser
+RE-TDAQ:shifter
(add value)
```

Fig. 7 Example of login restriction definition in LDAP

The `access.conf` file is generated by a shell script which runs periodically and looks up in LDAP all permissions defined for the current node. An example of its execution output is shown below:

```
[root@localhost  am_scripts]#  ./amLoginRestriction   -n  pc-tdaq-
control-001 -v
>>> ======= LOGIN RESCTRICTION CONFIGURATION ==========
ldapsearch  -h  localhost  -b  ou=atlas,o=cern,c=ch  -x  -LLL  -S  cn
(&(|(sudoHost=pc-tdaq-control-001)(|(sudoHost=pc-tdaq-control-
\*)(|(sudoHost=pc-tdaq-\*)(sudoHost=pc-\*)))))(&(cn=LOGIN-
*)(objectClass=sudoRole))) sudoUser cn sudoHost
dn: cn=LOGIN-RE-TDAQ:shifter,ou=SUDOers,ou=atlas,o=cern,c=ch
cn: LOGIN-RE-TDAQ:shifter
sudoHost: pc-tdaq-control-*
sudoUser: +RE-TDAQ:shifter

>>> PAM ACCESS configuration generated from LDAP information!
#
# Login access control table.
# Generated automatically by the script './amLoginRestriction'
#
```

```
# SUDO RULE DN:
# cn=LOGIN-RE-TDAQ:shifter,ou=SUDOers,ou=atlas,o=cern,c=ch
+ | @RE-TDAQ:shifter | ALL

- | ALL EXCEPT root | ALL
```

We can check also the users who get the access to this node and it can be observed that *alice* is also granted the permission through the permission inheritance over role hierarchy:

```
[root@localhost  am_scripts]#  ./amLoginRestriction   -n  pc-tdaq-
control-001 -s

>>> Users allowed to login to [pc-tdaq-control-001]:
>>> Netgroup [RE-TDAQ:shifter]:
alice
mleahu
```

6. Controlling the access to sensitive command line tools

The second level of access control (Fig. 3) comes on top of the first level by increasing the granularity of access control to the sensitive tools available on the node.

The prerequisite of enabling this level of access control is the presence (by default in SLC 5) of *sudo*[11] application on each node of the Linux network. The application description provided in [11] states: “Sudo (su "do") allows a system administrator to delegate authority to give certain users (or groups of users) the ability to run some (or all) commands as root or another user while providing an audit trail of the commands and their arguments”. Since we are interested in having a centralized LDAP configuration of the second level of access control, the *sudo* tool must be available with support for LDAP – by default in SLC 5. The *sudo* configuration must be also adjusted to point to the right LDAP server and base DN:

- */etc/ldap.conf* must contain the Base DN where the sudo roles are defined besides the server identifiers already mentioned in the previous chapter:

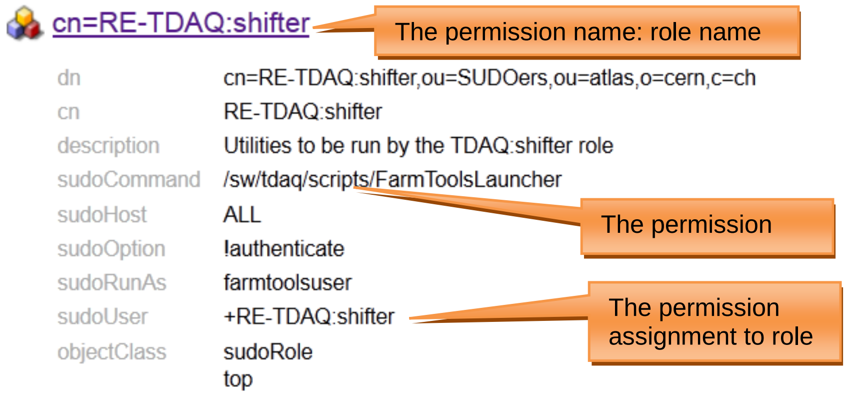
```
sudoers_base    ou=SUDOers,ou=atlas,o=cern,c=ch
```

The protection of tools (e.g. shell scripts or binaries) subject to access control must be done in two steps:

- restrict the file access permissions of the targeted tool to a generic Linux user created only for this purpose. For example, the tool `FarmToolsLauncher` is owned and allowed to be run only by *farmtoolsuser*:

```
chown farmtoolsuser /sw/tdaq/scripts/FarmToolsLauncher
chmod u=rwx /sw/tdaq/scripts/FarmToolsLauncher
chmod og-rwx /sw/tdaq/scripts/FarmToolsLauncher
```

- prepare a sudo role in the LDAP to allow the execution of the protected tool. Fig. 8 shows an example of sudo role for the tool `FarmToolsLauncher` where:
    - the *permission name* is by convention the role name
    - the *permission* is represented by the tool identifier and the user to run as
    - the *permission assignment to role* is defined as the netgroup corresponding to the role allowed to execute the tool



cn=RE-TDAQ:shifter

dn	cn=RE-TDAQ:shifter,ou=SUDOers,ou=atlas,o=cern,c=ch
cn	RE-TDAQ:shifter
description	Utilities to be run by the TDAQ:shifter role
sudoCommand	/sw/tdaq/scripts/FarmToolsLauncher
sudoHost	ALL
sudoOption	!authenticate
sudoRunAs	farmtoolsuser
sudoUser	+RE-TDAQ:shifter
objectClass	sudoRole top

The permission name: role name
The permission
The permission assignment to role

Fig. 8 Example of sudo role in LDAP

Let's check now the sudo permissions for user *alice*:

```
# sudo -l -U alice

User alice may run the following commands on this host:
(farmtoolsuser) NOPASSWD: /sw/tdaq/scripts/FarmToolsLauncher
```

Hence, the *alice* user is able to run the *FarmToolsLauncher* application thanks to the permission defined in LDAP and permission inheritance over the role hierarchy.

7. Conclusions

This paper presented an access control solution for Linux networks to regulate the access of the organization employees to the network nodes and tools running available on the nodes. The access control management is flexible thanks to the RBAC model used in its design which maps naturally the organization structure to the roles and roles hierarchy defined in the access control configuration. The solution implementation makes use of the Linux standard tools and mechanisms to control the user's login to the machines and the access to the sensitive software tools on each network node. The access control configuration is uniform over the network nodes and managed centrally in an LDAP repository.

The solution described by this paper can be further improved by:

- Preparation of a notification mechanism on the LDAP server which triggers configuration updates on the network nodes when the configuration in LDAP changes. In this way, the nodes won't be forced to periodically poll the LDAP server to check their *pam_access* configuration, but they'll be notified when changes occurred. Also, the changes propagation in the network will be faster.
- Development of a dedicated policy management tool with Graphical User Interface to replace the management done through the generic LDAP administration tools.
- Centralization of logs generated by *pam_access* and *sudo* for an easier audit of access control on the Linux network. An alarm system can be also envisaged to notify organization administrators about events that occurred in the network: for example, users which are repeatedly denied access to a node (perhaps the user really needs access to that node and his reasons must be understood) or users who attempt to run software which they are not supposed to.

The access control granularity can be increased by moving policy enforcement in the application themselves: if an application performs sensitive tasks (e.g. changing voltages on a peripheral device used for data acquisition), the access to them to be regulated by checking the permissions and roles defined in the central LDAP (e.g. enable the button to increase/decrease the voltage only if the user running the application has appropriate role and permissions). This would require dedicated policy management tailored to the application needs and a set of libraries which offer Application Programming Interface to interface the application with the network access control infrastructure (access to LDAP roles, permissions).

The solution described in this article has been successfully deployed in the large Linux cluster (~3000 nodes) used for data acquisition in the CERN ATLAS experiment and improvements, including those mentioned above, are being analyzed and developed by the SysAdmin team.

REFERENCES

[1] CERN, "European Organization for Nuclear Research," CERN, [Online]. Available: http://www.cern.ch.

[2] CERN, "ATLAS TDAQ SysAdmin Team," [Online]. Available: http://atlas-tdaq-sysadmin.web.cern.ch.

[3] *M. C. Leahu, M. Dobson and G. Avolio*, "Role Based Access Control in the ATLAS," IEEE Transactions on Nuclear Science, **vol. 55**, no. 1, pp. 386 - 391, February 2008.

[4] CERN, "Scientific Linux CERN 5," [Online]. Available: http://linux.web.cern.ch/linux/scientific5/.

[5] *R. Sandhu, D. Ferraiolo and R. K. D*, "The NIST Model for Role Based Access Control: Towards a Unified Standard," in 5th ACM Workshop Role-Based Access Control, Berlin, 2000.

[6] "OpenSSH," [Online]. Available: http://www.openssh.org/.

[7] "OpenLDAP," [Online]. Available: http://www.openldap.org/.

[8] "Linux PAM," [Online]. Available: http://www.linux-pam.org/.

[9] "php LDAP Admin," [Online]. Available: http://phpldapadmin.sourceforge.net.

[10] ORACLE, "System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)," [Online]. Available: http://docs.oracle.com/cd/E19082-01/819-3194/anis2-14244/index.html.

[11] "Linux Sudo," [Online]. Available: http://www.sudo.ws/.

[12] ORACLE, "NIS Extension Guide: NIS Information in the LDAP Directory," [Online]. Available: http://docs.oracle.com/cd/E19513-01/806-4251-10/mapping.htm.

[13] "Sudo - OpenLDAP schema," [Online]. Available: http://www.sudo.ws/repos/sudo/file/dacfad7e7a95/doc/schema.OpenLDAP. [Accessed September 2012].

[14] "Name Service Switch - The GNU C Library," [Online]. Available: http://www.gnu.org/software/libc/manual/html_node/Name-Service-Switch.html.

[15] "PAM Access," [Online]. Available: http://linux.die.net/man/8/pam_access.