

**Scalable Centralized Authentication Services
using Red Hat Directory Server**

Netgroups and its usage

Written by

Unni Krishnan

Edited and Reviewed by

Srinivas Satyavarpu

Scalable Centralized Authentication Services using Red Hat Directory Server

Introduction

One of the challenges faced by system administrators is the management of user access to multiple systems. In a unix environment, providing access based on information stored locally in the system (the `/etc/passwd` and `/etc/shadow` files) becomes unmanageable as the number of systems and users increase. In such environments, storing the user information in an LDAP based directory makes the system scalable, manageable secure, as the user information is not stored locally in the systems.

The Red Hat Directory Server is a robust, scalable server based on the Lightweight Directory Access Protocol (LDAP) protocol. The Directory Server is designed to manage an enterprise-wide directory of users and resources. This white paper discusses using the Red Hat Directory Server to provide centralized authentication to a multi user multi system enterprise. We will show the implementation and configuration of the Red Hat Directory Server and how to use groups to provide restricted access to users based on access rules.

Simple Authentication

Before access can be granted to a system, the user need to be authenticated and authorized to access the system. Authentication answers the question "Who are you?" while Authorization answers the question "What are you allowed to access?" In simple unix login, these tasks are performed using the information stored in `/etc/passwd` and `/etc/shadow` file. The shadow file authenticates the user by verifying the user password and the `passwd` file authorizes by providing the corresponding uid and gid values. These parameters provide access granularity once the user is logged in. In simple authentication, such as the standard unix login, the databases are stored locally which requires the databases to be copied to all the systems. This makes scaling the system difficult. The databases can be moved to a centralized location by using a network based authentication such as NIS. NIS provides the system administrator with a single database, but it does not provide a scalable method for restricted access to systems based on user id and system id.

Advanced Authentication

In a large enterprise, it is very important to restrict access to specific systems for specific users. For example the operation team would have access to the production systems while the developers would not have any access to these systems. The Red Hat Directory Server can be configured to implement this feature in a scalable and manageable way.

There are couple of ways to provide restricted access to the systems. One way is to use the host attribute while defining users in the Directory. The host attribute defines all the systems the user is allowed to access. The systems will then be configured to check the host attribute by pam to deny access if the user doesn't have the system as a host attribute. This method is simple to install and has all the information stored in one record. The disadvantage of this approach is that it is not scalable. Consider the case when a new system is added to the enterprise. Now the system administrator has to update each of the user records to with a new host attribute for the system just added. This becomes difficult and even unmanageable as the number of users or systems become large. It also makes it difficult to remove a system from the enterprise or change its functionality.

A better way to solve this problem is to define groups of users and systems. Access to the systems is provided by configuring pam to check whether a user belongs to an approved user group and the system belongs to an approved system group before granting access. By separating the access rules from the user data and system data, management of access is greatly simplified. This method has the added advantage of being scalable. Addition of a server requires updating of just the groups it belong to, rather than all the users who needs access to

the system.

For the Red Hat Directory Server, the netgroup feature allows the directory administrator to define groups for server access. The netgroup entries are stored as an objectClass of type nisNetgroup in the directory server. The nisNetgroupTriple attribute is used to define a netgroup entry and this attribute can have multiple values. The format of this attribute is as follows

```
nisNetgroupTriple(hostname, user, domain)
```

The netgroup feature also provides a way to include one group within a group. This is a very powerful feature as we can provide access to a group of members to another group. The format of this attribute is as follows.

```
memberNisNetgroup: Group
```

Groups can be nested by this feature. We can have a group included in another group which in turn is included in yet another group. This simplifies management of the users but using nested groups or more than 4 or 5 levels can make it difficult to find all the group members in a group.

Later in this discussion, we will show examples of how to create these groups and how to use them to provide restricted access.

Project Specifications

We are asked to create a Directory Server for the company Theyjas Systems LC. Theyjas Systems is a fast growing company providing software development and production services to its clients. At present they have 8 staff members and 8 systems. They anticipate to increase its staff to about 100 and the number of systems to over 1000 in a year.

There are four different types of users at Theyjas Systems. The Developers of the Development Team which designs and implements the software projects. The Testers of the Test Team takes the product from the Development Team and tests the functionality along with other software. Once the testing is complete, it is given to the Staging Team which now tests the software in a production clone environment to make sure that the software works as designed and does not impede with the operation of the Production environment. Once the software passes the Stage team, it is ready to be implemented in the Production Environment.

The systems at Theyjas Systems can also be divided into four different types... Development Systems, Test Systems, Stage Systems and Production Systems. If a Tester finds a problem with the software, he would need access to the Development System to report and possibly solve the issue. Similarly, a Stage User need access to Stage Systems, Test Systems, and Development Systems. The Production Users need access to all the Systems.

As per these requirements, we design the following

Four User Groups

1. ProdUsers
2. StageUsers
3. TestUsers
4. DevUsers

Four Server Groups

1. ProdSystems
2. StageSystems
3. TesSystems
4. DevSystems

The rules we need to implement are

1. Only ProdUsers can access ProdSystems.
2. ProdUsers and StageUsers can access StageSystems.
3. TestUsers, StageUsers, and ProdUsers can access TestSystems.
4. Everyone can access DevSystems.

Setting up the Directory Server

The exact hardware specifications needed for the Directory Server is determined by the clients requirements. For this client, we select a Dual P4 server with 2GB of memory and a 120GB disk. The version of the Directory Server we will be using is Version 7.1 SP3 and it will be running on a RHEL4 AS.

Install the server with RHEL4AS. A minimal install should be enough as the system administrator should be able to access the server GUI from his desktop. Configure the network and make sure the server can access the corporate network. It is very important to make sure that the network configuration is correct and the hostname is set properly. Once the Directory Server is installed, it will be very difficult to change these parameters. Make sure that the server is updated with the latest kernel and software updates and reboot the server to activate these updates.

The Directory Server should not be run as root. We need to create an ldap user and group by running the following command.

```
groupadd -g 55 ldap
useradd -c "LDAP User" -u 55 -g ldap -s /bin/false -r -d /var/lib/ldap ldap
```

Now we can start the Directory Server installation. Download the Directory Server rpm to the /tmp directory. Directory Server rpm requires the rpm xorg-x11-deprecated-libs as a prerequisite. If it is not installed, download the rpm to /tmp. Install the rpms on the server by the following command.

```
rpm -Uvh /tmp/xorg-x11-deprecated-libs-6.8.2-1.EL.31.rpm
rpm -Uvh /tmp/redhat-ds-7.1SP3-5.RHEL4.i386.rpm
```

This will install the Directory Server in the /opt/redhat-ds directory. Once the rpm is installed, run the following command to configure the Directory server.

```
/opt/redhat-ds/setup/setup
```

Accept the license. The setup command will now run a performance check on the system and will display any issues it found during this step. You must rectify these errors before continuing with the installation. Once these are corrected, it asks for the install mode. Choose the default, which is 2

Please select the install mode:

- 1 - Express - minimal questions
- 2 - Typical - some customization (default)
- 3 - Custom - lots of customization

Please select 1, 2, or 3 (default: 2) **2**

Choose the hostname to use. This defaults to the hostname of the server. There is no need to change this.

Hostname to use (default: ldap.theyjas.com) **ldap.theyjas.com**

Choose User ID and Group ID to use for the Directory Server as ldap

Server user ID to use (default: nobody) **ldap**

Server group ID to use (default: nobody) **ldap**

Choose defaults for the next 5 questions.

Do you want to register this software with an existing Red Hat configuration directory server? [No]:

Do you want to use another directory to store your data? [No]:

Directory server network port [389]:

Directory server identifier [ldap]:

Red Hat configuration directory server

administrator ID [admin]:

Choose a password for the administrator id. This id will be used for most of the Directory Server operations such as loading the Directory Server, querying it, etc. In our example we are choosing the password as 'red hat'.

Password: **redhat**

Password (again): **redhat**

We must now choose the root of the Directory tree, also known as the Base DN. It will be very difficult to change once this has been set. The default value is probably the best choice and so accept it unless you have a specific reason not to use the default value.

Suffix [dc=theyjas, dc=com]:

Next the Directory Server creates an Administrative User Account. Here also, accept the default value, unless you have a specific reason to choose another. Provide a password for this account. In our example we are choosing the password 'password'.

Directory Manager DN [cn=Directory Manager]:

Password: **password**

Password (again): **password**

Now we need to choose an Administration Domain. Here also, accept the default value, unless you have a specific reason to choose another.

Administration Domain [unni.home.net]:

Red Hat Directory Server provides an Administration Server which listens to a different port than the ldap server. This is a GUI interface which facilitates the easy configuration of the Directory Server. You can use the default or you can choose a different port. In our example we chose the port 9999

Administration port [55335]: **9999**

Choose the default user to run the Administration Server

Run Administration Server as [root]:

The setup program now completes the configuration and starts the directory server daemon slapd.

The installation program does not create a service for the Directory Server. To start and stop the server we need to use the following commands

```
/opt/redhat-ds/slapd-ldap/start-slapd  
/opt/redhat-ds/slapd-ldap/stop-slapd
```

We need to make sure that the Directory Server is started on system boot and shutdown properly during system

shutdown. To best way to accomplish this is to make the Directory server a service by creating the startup script /etc/init.d/slapd as follows

```
#!/bin/sh
# Startup script for Red Hat Directory Server
#
# chkconfig: 2345 99 01
# description: Starts and stop Red Hat Directory Server

# Source function library.
. /etc/rc.d/init.d/functions

RETVAL=0
SLAPD_DIR="/opt/redhat-ds/slapd-ldap"

[ -f /opt/redhat-ds/slapd-ldap/start-slapd ] || exit 0

start() {
    echo -n $"Starting Red Hat Directory Server: "
    $SLAPD_DIR/start-slapd 2>/dev/null 1>&2 && success || failure
    RETVAL=$?
    [ $RETVAL -eq 0 ] && touch /var/lock/subsys/slapd
    echo
    return $RETVAL
}

stop() {
    echo -n $"Stopping Red Hat Directory Server: "
    $SLAPD_DIR/stop-slapd 2>/dev/null 1>&2 && success || failure
    RETVAL=$?
    [ $RETVAL -eq 0 ] && rm -f /var/lock/subsys/slapd
    echo
    return $RETVAL
}

case "$1" in
    start) start ;;
    stop) stop ;;
    restart) stop
            start ;;
    *) echo $"Usage: $0 {start|stop|restart}"
       exit 1
esac

exit $RETVAL
```

Make the Directory Server start on reboot and start the Directory Server by executing the following command.

```
chkconfig --add slapd
chkconfig slapd on
service sldpd start
```

Populating the Directory

Now that we have the Directory Server installed, we need to populate the Directory with data. The Directory Server gets installed in the /opt/redhat-ds directory. To access the ldap commands we need to update the path variable \$PATH and the library path variable \$LD_LIBRARY_PATH. The best way to accomplish this task is by updating the .bash_profile file for the user. The updated .bash_profile is shown below.

```
# .bash_profile
```

```

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
    . ~/.bashrc
fi

# User specific environment and startup programs

export PATH=/opt/redhat-ds/shared/bin:$PATH:$HOME/bin
export LD_LIBRARY_PATH=/opt/redhat-ds/shared/lib:$LD_LIBRARY_PATH

```

LDAP data can be specified in the LDAP Data Interchange Format (LDIF). LDIF files define ldap entries in a standard text based format. Although the LDIF format has many options, for this discussion, the main three features are the following.

- Entries consist of a list of attributes.
- Each attribute is represented in one line.
- Empty line separates the entries.

Although any front end ldap GUI could be used to populate the Directory, the most common way to populate a directory is to use ldif files. There is a significant amount of data that need to be typed in to create an ldif file and it follows very strict format rules. A single error in the format would cause the Directory Server to reject the ldif file. To facilitate the managing of the users, lets create a script, UserAdd, to generate our ldif files. The UserAdd script is shown below.

```

#!/bin/bash

function Usage {
    echo "Usage: $0 -f FirstName -i UserID -l LastName -u UserName [ -s Shell -g GroupID -p password]"
    exit 1
}

Shell=/bin/bash
GroupID=100
Password=redhat

while getopts f:g:i:l:i:p:u: OPTION
do
    case $OPTION in
        f) FirstName=$OPTARG;;
        g) GroupID=$OPTARG;;
        i) UserID=$OPTARG;;
        l) LastName=$OPTARG;;
        p) Password=$OPTARG;;
        u) UserName=$OPTARG;;
    esac
done

if [ "$FirstName" == "" ]; then Usage; fi
if [ "$LastName" == "" ]; then Usage; fi
if [ "$UserID" == "" ]; then Usage; fi

if [ "$UserName" == "" ]; then
    FirstInitial=`echo $FirstName | cut -c1`
    UserName=`echo "${FirstInitial}${LastName}" | tr "[:upper:]" "[:lower:]"`
fi

echo "dn: uid=$UserName, ou=People, dc=theyjas,dc=com"
echo "changetype: add"
echo "uid: $UserName"
echo "objectClass: top"
echo "objectClass: person"
echo "objectClass: organizationalPerson"

```

```

echo "objectClass: inetorgperson"
echo "objectClass: posixAccount"
echo "cn: $FirstName $LastName"
echo "sn: $LastName"
echo "givenName: $FirstName $LastName"
echo "gidNumber: $GroupID"
echo "uidNumber: $UserID"
echo "userPassword: {clear}$Password"
echo "loginShell: $Shell"
echo "homeDirectory: /home/$UserName"
echo

```

Use the UserAdd script to represent 8 users in the Idif file Users.Idif as follows

```

UserAdd -u ann -f Ann -l Nelson -i 1001 >> Users.Idif
UserAdd -u dani -f Dani -l Taylor -i 1002 >> Users.Idif
UserAdd -u dave -f Dave -l Meyer -i 1003 >> Users.Idif
UserAdd -u jen -f Jen -l Bailey -i 1004 >> Users.Idif
UserAdd -u john -f John -l Denver -i 1005 >> Users.Idif
UserAdd -u kyle -f Kyle -l Brooks -i 1006 >> Users.Idif
UserAdd -u matt -f Matt -l Hill -i 1007 >> Users.Idif
UserAdd -u sara -f Sara -l Davis -i 1008 >> Users.Idif

```

Now that we have defined the users, we need to define the groups. We use the netgroup feature of the Directory Server to provide the hierarchical group structure requested by the client. We define 4 user groups (ProdUsers, StageUsers, TestUsers, and DevUsers) and 4 system groups (ProdSystems, StageSystems, TestSystems, and DevSystems). We add the users and systems to the corresponding groups by the nisNetgroupTriple attribute.

Our requirements dictate that Production Users should have access to all systems. In other words, Production Users are also Stage Users, Test Users and Dev Users. We could add the name of all Production Users to all the User Groups, but this creates duplicate information in the Directory. It is recommend not to have duplicate information in the Directory to facilitate easy management. To accomplish this, we use the memberNisNetgroup attribute. In our StageUsers group we include the ProdUsers, TestUsers group includes StageUsers group, and DevUsers group includes TestUsers group. Since ProdUsers is part of StageUsers group, TestUsers group will have both StageUsers and ProdUsers. By a similar argument, we can see that DevUsers group has all the users. The Groups.Idif file is created as shown below.

```

dn: cn=ProdUsers,dc=theyjas,dc=com
changetype: add
cn: ProdUsers
objectClass: top
objectClass: nisNetgroup
nisNetgroupTriple: (,ann,)
nisNetgroupTriple: (,dani,)
description: All Production Users

```

```

dn: cn=StageUsers,dc=theyjas,dc=com
changetype: add
cn: StageUsers
objectClass: top
objectClass: nisNetgroup
nisNetgroupTriple: (,dave,)
nisNetgroupTriple: (,jen,)
memberNisNetgroup: ProdUsers
description: All Stage Users

```

```

dn: cn=TestUsers,dc=theyjas,dc=com
changetype: add
cn: TestUsers
objectClass: top
objectClass: nisNetgroup
nisNetgroupTriple: (,john,)

```

nisNetgroupTriple: (,kyle,)
memberNisNetgroup: StageUsers
description: All Test Users

dn: cn=DevUsers,dc=theyjas,dc=com
changetype: add
cn: DevUsers
objectClass: top
objectClass: nisNetgroup
nisNetgroupTriple: (,matt,)
nisNetgroupTriple: (,sara,)
memberNisNetgroup: TestUsers
description: All Development Users

dn: cn=ProdSystems,dc=theyjas,dc=com
changetype: add
cn: ProdSystems
objectClass: top
objectClass: nisNetgroup
nisNetgroupTriple: (prod00.theyjas.com,,)
nisNetgroupTriple: (prod01.theyjas.com,,)
description: All Production systems

dn: cn=StageSystems,dc=theyjas,dc=com
changetype: add
cn: StageSystems
objectClass: top
objectClass: nisNetgroup
nisNetgroupTriple: (stg00.theyjas.com,,)
nisNetgroupTriple: (stg01.theyjas.com,,)
description: All Stage systems

dn: cn=TestSystems,dc=theyjas,dc=com
changetype: add
cn: TestSystems
objectClass: top
objectClass: nisNetgroup
nisNetgroupTriple: (test00.theyjas.com,,)
nisNetgroupTriple: (test01.theyjas.com,,)
description: All Test systems

dn: cn=DevSystems,dc=theyjas,dc=com
changetype: add
cn: DevSystems
objectClass: top
objectClass: nisNetgroup
nisNetgroupTriple: (dev00.theyjas.com,,)
nisNetgroupTriple: (dev01.theyjas.com,,)
description: All Development systems

Load the Users.ldif and Groups.ldif to the directory server by the following command.

```
ldapmodify -h ldap.theyjas.com -D "cn=Directory Manager" -w password -f Users.ldif  
ldapmodify -h ldap.theyjas.com -D "cn=Directory Manager" -w password -f Groups.ldif
```

Verify that the Directory is populated by the following command

```
ldapsearch -h ldap.theyjas.com -D "cn=Directory Manager" -w password -b "dc=theyjas,dc=com" -s sub  
"objectclass=*" uid=testuse r1
```

The output should be like

```
version: 1
dn: dc=theyjas.dc=com
dn: cn=Directory Administrators, dc=theyjas.dc=com
dn: ou=Groups, dc=theyjas.dc=com
dn: ou=People, dc=theyjas.dc=com
dn: ou=Special Users,dc=theyjas.dc=com
dn: cn=Accounting Managers,ou=groups,dc=theyjas.dc=com
dn: cn=HR Managers,ou=groups,dc=theyjas.dc=com
dn: cn=QA Managers,ou=groups,dc=theyjas.dc=com
dn: cn=PD Managers,ou=groups,dc=theyjas.dc=com
dn: uid=ann, ou=People, dc=theyjas.dc=com
dn: uid=dani, ou=People, dc=theyjas.dc=com
dn: uid=dave, ou=People, dc=theyjas.dc=com
dn: uid=jen, ou=People, dc=theyjas.dc=com
dn: uid=john, ou=People, dc=theyjas.dc=com
dn: uid=kyle, ou=People, dc=theyjas.dc=com
dn: uid=matt, ou=People, dc=theyjas.dc=com
dn: uid=sara, ou=People, dc=theyjas.dc=com
dn: cn=ProdUsers,dc=theyjas.dc=com
dn: cn=StageUsers,dc=theyjas.dc=com
dn: cn=TestUsers,dc=theyjas.dc=com
dn: cn=DevUsers,dc=theyjas.dc=com
dn: cn=ProdSystems,dc=theyjas.dc=com
dn: cn=StageSystems,dc=theyjas.dc=com
dn: cn=TestSystems,dc=theyjas.dc=com
dn: cn=DevSystems,dc=theyjas.dc=com
```

Now we have loaded all the data into the Directory Server.

Setting up the Systems

Although any ldap enabled operating systems would work, in our example we are going to use RHEL5 systems. Install the systems with RHEL5 Server and update it to the latest packages. Make sure that the network is configured properly and the hostname is configured correctly. Reboot the systems to activate the new packages. By default, ldap authentication is disabled. To enable ldap based authentication we need to know the ip address of the ldap server and the base DN used by the ldap server. In our example, the Directory Server named ldap.theyjas.com with ip address 10.10.1.100. The base DN for the LDAP server is dc=theyjas,dc=com. Configure ldap authentication on the systems by the following commands

```
system-config-authentication --enableldap --update
system-config-authentication --ldapserver=10.10.1.100 --update
system-config-authentication --ldapbasedn=dc=theyjas,dc=com --update
```

These commands update the /etc/nsswitch.conf and /etc/ldap.conf. It also configures the pam configuration files which enables the systems to authenticate against all the users on the ldap database. To allow only a specific group of people to access the system we need to use the pam_access module to provide selective access. To accomplish this, we need to update the config files located in the /etc/pam.d directory. Be extra careful while updating files in the /etc/pam.d directory. The changes made to pam configuration files is instantaneous and any error may lockout access to the system. Always make sure you have a root window open to the system while editing the pam files to recover from any mistakes.

We need to add the following line to the /etc/pam.d/system-auth file

```
account    required    pam_access.so accessfile=/etc/security/group.conf
```

This line instructs pam to look into the file /etc/security/group.conf for access rules. The /etc/pam.d/system-auth file for our example is shown below.

```

auth    required    pam_env.so
auth    sufficient  pam_unix.so nullok try_first_pass
auth    requisite   pam_succeed_if.so uid >= 500 quiet
auth    sufficient  pam_ldap.so use_first_pass
auth    required    pam_deny.so

account required    pam_unix.so broken_shadow
account required    pam_access.so accessfile=/etc/security/netgroup.conf
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

password requisite   pam_cracklib.so try_first_pass retry=3
password sufficient  pam_unix.so md5 shadow nullok try_first_pass use_authtok
password sufficient  pam_ldap.so use_authtok
password required    pam_deny.so

session optional    pam_keyinit.so revoke
session required    pam_limits.so
session [success=1 default=ignore] pam_succeed_if.so service in crond quiet use_uid
session required    pam_unix.so
session optional    pam_ldap.so

```

The access file `/etc/security/group.conf` is created in the access configuration file format. Each line of the access file has three fields separated by a ":". The first field, the permission field, is either a "+" to grant access or a "-" to deny access. The second field, the users field, denotes the groups that are allowed/denied accesses. The third field, the origins field, denotes where the access is allowed from. The rules are checked from top to bottom. So to make sure that only the allowed users get access, make the default rule a deny by adding `- : ALL :ALL` as the last line. Our `/etc/security/group.conf` is shown below.

```

+ : root          : LOCAL
+ : @ProdUsers@@ProdSystems : 10.
+ : @StageUsers@@StageSystems : 10.
+ : @TestUsers@@TestSystems : 10.
+ : @DevUsers@@DevSystems : 10.
- : ALL          : ALL

```

The first line gives access to root from the local console. The next line states that if the user belongs to the ProdUsers group, this system belong to the ProdSystems group and the request is coming from the 10 network, grant access. The next 3 lines state the same rule but for StageUser/StageSystems, TestUser/Test Systems and DevUser/DevSystems. The system checks these rules from top to bottom. If any of them match, access is granted. If none of these match, the default rule is applied, which denies the user access.

By separating the users and systems to different groups, we can create a standard access file which can be propagated to all servers in the enterprise. Whether a particular server belongs to a specific group, say production, is determined by the Directory Server administrator which eliminates the need for updating configuration files in multiple machines. Identical servers, except for their individual network configurations, can be used for different tasks by designing these groups efficiently. This will facilitate creating standard images for servers, thereby simplifying the deployment process.

Testing the Configuration

To simplify the testing process, we assumed that there are only 8 machines in the enterprise while populating our Directory Server. We create following 8 machines.

1. prod00.theyjas.com
2. prod01.theyjas.com
3. stg00.theyjas.com

4. stg01.theyjas.com
5. test00.theyjas.com
6. test01.theyjas.com
7. dev00.theyjas.com
8. dev01.theyjas.com

All of these are installed with identical images of RHEL5 base install. Each of them are setup as mentioned in the above section. Our rules have been set as follows.

1. Only production users can log in to production servers.
2. Both production and stage users can log in to the stage machines.
3. Production, Stage, and Test users can log into Test Machines.
4. All users can log into Dev Machines.

Try logging into the machines as different users. The results will be as follows

	prod00	prod01	stg00	stg01	test00	test01	dev00	dev01
ann	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS
dani	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS
dave	FAIL	FAIL	PASS	PASS	PASS	PASS	PASS	PASS
jen	FAIL	FAIL	PASS	PASS	PASS	PASS	PASS	PASS
john	FAIL	FAIL	FAIL	FAIL	PASS	PASS	PASS	PASS
kyle	FAIL	FAIL	FAIL	FAIL	PASS	PASS	PASS	PASS
matt	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	PASS	PASS
sara	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	PASS	PASS

Updating the Directory Server

The Directory is a dynamic entity. The data in the directory need to be updated with the changes occurring with the enterprise. All the update operations can be accomplished by Ldif files. The three main tasks are add, delete, and update. All of these can be accomplished by Ldif files and the ldapmodify command.

Addition

Zach Newman joined Theyjas Systems and he need to be added to the Directory. We first create a new Ldif file Zach.Ldif by using our UserAdd and load the data into the Directory with the ldapmodify command as follows.

```
UserAdd -u zach -f Zach -l Newman -i 1009 >> Zach.Ldif
ldapmodify -h ldap.theyjas.com -D "cn=Directory Manager" -w password -f Zach.Ldif
```

Running the following ldapsearch command will show that the entry has been added to the Directory.

```
ldapsearch -h ldap.unni.home.net -D "cn=Directory Manager" -w password -b
"dc=unni,dc=home,dc=net" uid=zach
```

Deletion

Sara Davis has left Theyjas Systems and need to be deleted from the directory. To accomplish this task we create the Sara.Ldif file as follows

```
dn: uid=unni, ou=People, dc=unni,dc=home,dc=net
changetype: delete
```

We now update the Directory with the `ldapmodify` command as follows

```
ldapmodify -h ldap.theyjas.com -D "cn=Directory Manager" -w password -f Sara.ldif
```

Running the `ldapssearch` command will show that the entry has been deleted from the Directory.

```
ldapssearch -h ldap.unni.home.net -D "cn=Directory Manager" -w password -b "dc=unni,dc=home,dc=net" uid=sara
```

Change

Theyjas Systems purchased a new Development server `dev02.theyjas.com` and added it to the Development Servers. We need to update the Directory with this information. We create the `Update.ldif` file as follows

```
dn: cn=DevSystems,dc=theyjas,dc=com
changetype: delete
```

```
dn: cn=DevSystems,dc=theyjas,dc=com
changetype: add
cn: DevSystems
objectClass: top
objectClass: nisNetgroup
nisNetgroupTriple: (dev00.theyjas.com,,)
nisNetgroupTriple: (dev01.theyjas.com,,)
nisNetgroupTriple: (dev02.theyjas.com,,)
description: All Development systems
```

We now update the Directory with the `ldapmodify` command as follows

```
ldapmodify -h ldap.theyjas.com -D "cn=Directory Manager" -w password -f Update.ldif
```

Running the following `ldapssearch` command will show that the entry has been updated in the Directory.

```
ldapssearch -h ldap.unni.home.net -D "cn=Directory Manager" -w password -b "dc=unni,dc=home,dc=net" cn=DevSystems
```

Conclusion

The Red Hat Directory Server provides a scalable method for Centralized Authorization and Authentication services to a large enterprise. The `netgroup` feature provided by the Directory Server provides a way to group users and systems into different groups. By configuring `pam` to use these groups, we can implement user access control enterprise wide with minimal changes to the systems. Since the user, system, and group information are stored centrally at the Directory Server, the information is secure and updates are instantaneous and enterprise wide. By eliminating system and user specific information (except the network configuration) on the systems, system deployment is simplified by creating standard images. Further, addition, deletion and role change of users or systems is greatly simplified.