SysAdmin Notes for RHCE

Somenath Sinha

January 2018

Contents

Part I

Advanced System Management

Chapter 1

Configuring Authentication

1.1 Understanding RedHat Identity Management

RedHat Identity Management is based on the FreeIPA (Identity, Policy, Audit) Project. The project bundles together several services in one solution. Some of the services are:

Options	Description
389 Directory Server	This is an **LDAPv3** Directory Server – a replacement for *OpenLDAP*.
Single Sign-on	Provided by MIT **Kerberos** KDC.
Integrated Certificate System	Based on the *Dogtag* project.
Integrated NTP Server	*Chrony* must be disabled to use this!
Integrated DNS Server	Based on *ISC Bind* Service.

Thus, the Identity Management provided by IPA bundles up some pretty complicated projects together and provides an easy interface to manage them all. However, IPA conflicts with other products, such as other *LDAP, Kerberos, Certificate System, NTP or DNS* servers shouldn't be running on the same system. Thus, ideally Identity Management should be set up on a dedicated server.

Kerberos is a Network Authentication Protocol that makes clients prove their identity to the server, and vice versa. Other than the authentication tools, it also supports strong cryptography over the network to keep the data safe in-transit.

1.1.1 IdM Server Components and Requirements

An IdM server needs some from of *Host Name Resolution*, which can be either through a DNS server or via the `/etc/hosts` file. Note that the hostname of the Identity Management server itself must also be specified.

Next we need both the **ipa-server** package, which installs the server components, and the **ipa-client** package which installs the client components. While the client package isn't required to be installed on the server, while configuring a client that talks to an IPA server, then this is one of the solutions available. Another method would be to use **authconfig**.

After the required RPM packages have been installed, we will run **ipa-server-install** which provides an easy, scripted way to install an IPA server, and all we have to do is answer a few questions, at the end of which we get a fully-functional IPA server.

The **ipa** tool is a generic client interface, that's also the administration interface. Thus, it can perform several tasks such as adding users (`ipa user-add <username>`), set the password for an user (`ipa passwd <username>`), see the IPA properties for a user account (`ipa user-find <username>`), etc. *ipa-xxx* can be used instead as well, where *xxx* represents the different tasks. Authentication can also be configured using **authconfig**.

1.1.2 Preparing IdM Installation

First and foremost, the *host name resolution* must be set up, since the installation will fail if the host can't find its own name. Additionally, the DNS name must also be known since the Kerberos domain that we'll configure will be based on the DNS name.

Next, the **nscd** service must be disabled, along with any existing LDAP and Kerberos services. If NTP and ISC Bind must also be disabled if installed (due to possible conflicts). The LDAP, Kerberos, NTP, DNS and certificate system ports must be opened in the firewall.

1.1.3 Installing IdM

The **ipa-server**, **bind** and **nds-ldap** packages must be installed using, following which, we have to run the command **ipa-server-install**, which will perform a wizard-like scripted installation.

```
# yum -y install ipa-server bind nds-ldap
# ipa-server-install
```

If we don't want to enter the information interactively, we can also provide them as options. The hostname, the domain name, a realm name (domain name in upper-case).

```
# ipa-server-install --hostname=vmPrime.somuVMnet.local -n somuVMnet.local -r
↪  SOMUVMNET.COM -p password -a password -U --no-ntp
```

The appropriate flags needed are:

Options	Description
–hostname	The hostname of the server
-n	The Domain name of the server
-r	Realm Name (Domain name in All-Caps)
-p	Password for Directory Manager
-a	Password for admin user
-U	Unattended Install; Doesn't prompt for anything
–no-dns	Do not install the DNS Server

Now, the SSH Daemon must be restarted to ensure that SSH obtains Kerberos credentials:

```
# systemctl restart sshd
```

Then, we generate a new Kerberos ticket and then verify Kerberos authentication for the default admin user by using:

```
# kinit admin
Password for admin@SOMUVMNET.LOCAL:
```

```
# klist
Ticket cache: KEYRING:persistent:0:0
Default principal: admin@SOMUVMNET.LOCAL

Valid starting       Expires              Service principal
2017-12-26T15:44:09  2017-12-27T15:44:05  krbtgt/SOMUVMNET.LOCAL@SOMUVMNET.LOCAL
```

This will show us if we have a valid Kerberos ticket. For any administrative tasks on the IPA server, having a valid Kerberos ticket is mandatory. Finally, we need to verify IPA access using:

```
# ipa user-find admin
```

This will show us the details of the admin user as created in the LDAP directory, along with all of its properties.

1.1.4 Understanding Kerberos Tickets

Kerberos tickets are the keys to the proper functioning of Identity Management. To be able to manage the IdM server, we need to log in to the IdM Domain and generate a Kerberos ticket for the admin user, using the command:

```
# kinit admin
```

We can check the validity of the ticket at any time using:

```
# klist
```

1.1.5 Managing the IdM Server

After generating a Kerberos ticket with `kinit admin`, we use the **ipa** command to manage the IdM server. `ipa help commands` shows us a short overview of all the available commands and their usage. For any specific command, we have `ipa help <command>` (such as `ipa help user-add`).

Another method to manage the IdM server is to navigate, using our web browser, to `https://vmPrime.somuVMnet.local` (if our server is named *vmPrime.somuVMnet.local*). This will load the IPA management web interface. Through this interface, after we've authenticated as admin, we will be guided through the various aspects of setting up the IdM environment.

1.1.6 Creating User Accounts

The required commands to create an user called *lisa* and verify the account creation are:

```
# kinit admin
# ipa user-add lisa
# ipa passwd lisa
# ipa user-find lisa
```

1.2 Using authconfig to Setup External Authentication

There are the **authconfig** utilities to setup external authentication (via LDAP), which consist of: *authconfig, authconfig-tui* and *authconfig-gtk*. The GUI utility can be installed using `yum -y install authconfig-gtk`. The utility is started with `authconfig-gtk`.

In the **authconfig-gtk** utility, we have to choose LDAP as the User Account Database in the Identity and Authentication tab. This might prompt for the installation of two packages: *nss-pam-ldapd* (the package that integrates the three) and *pam_krb5* (the package that integrates PAM with Kerberos). Now, we can enter the details for the LDAP server to setup authentication.

In cases of servers which don't have a GUI (or there is some inconvienience with the GUI, such as the apply button hidden by the status bar, etc.), the **authconfig-tui** is a very good alternative. In case of automated scripts, however, the **authconfig** command line utility is the best option.

1.3 Configuring a System to Authenticate using Kerberos

To connect a system for authentication to an LDAP server using Kerberos credentials, a part of the configuration has to be done with authconfig. But even before that, certain things must be ensured. *First*, we need to make sure that the IP address of the server we're trying to connect to can be resolved from the hostname, using `/etc/hosts`:

```
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
90.0.16.100        vmDeux.somuVMnet.com        vmDeux
```

This is important so that we can use the FQDN of the server later while using the `authconfig-tui` utility. Next, the system must be configured to use the DNS component hosted within the IPA server. For this, all we need to do is add the IP address of the IPA server as the first nameserver entry in `/etc/resolv.conf`:

```
# Generated by NetworkManager
search somuvmnet.local
nameserver 90.0.16.100 # IP Address of DNS Server @ vmDeux.somuVMnet.local
nameserver 8.8.8.8
nameserver 202.38.180.7
```

It is important to place the IP address of the DNS server for a nameserver as the first entry because that's the only one configured to *know* the custom FQDNs of the machines on our network. So, this connectivity is essential since the Kerberos client needs to be connected to the Kerberos server.

Finally, we can start the `authconfig-tui` utility, and enter the following details:

```
# In Authentication Configuration:
[*] Use LDAP
[*] User Kerberos Authentication

# LDAP Settings
[*] Use TLS
Server:        ldap://vmdeux.somuvmnet.local
Base DN:        dc=somuvmnet, dc=local
```

```
9
10  # Kerberos Settings
11  Realm:        SOMUVMNET.LOCAL
12  [*] Use DNS to resolve hosts to realms
13  [*] Use DNS to resolve KDCs for realms
```

First, we've just setup the system to use LDAP using Kerberos authentication. Next, we've made it necessary to use a TLS certificate to ensure the security of the connection. Then, the details of the LDAP server have to be entered.

In the Kerberos authentication step, the *Realm* refers to the Kerberos realm that the server is a part of. If we've setup the DNS component of the IPA server properly, then the system is able to detect the KDCs properly for each realm, as well as assign hosts to their realm appropriately. Now, the TLS Certificate for the IPA Server have to be downloaded and put in the `/etc/openldap/cacerts` directory (from whichever location the IPA Server stored them in, typically `/root/cacert.p12` for the root user):

```
1  # cd /etc/openldap/cacerts/
2  # scp vmdeux.somuvmnet.local:/root/cacert.p12 .
```

At this point, we should be good to go. We can verify the LDAP connectivity by trying to login as an LDAP user. For this we use (for an LDAP user lisa):

```
1  # su - lisa
2  Last login: Tue Dec 26 18:52:05 IST 2017 on pts/0
3  su: warning: cannot change directory to /home/lisa: No such file or directory
4  -sh-4.2$
```

The warning is natural if no home directory has been configured yet.

1.3.1 Troubleshooting Authentication

When authentication doesn't work, for some reason related to the certificates, then there is an easy fix as well. Depending on whether our LDAP and Kerberos credentials are being cached by **nslcd** or **sssd**, we can edit their configuration file to ignore the validity of the certificate. This is because the *self-signed cacert* may not meet the standards dictated and required by the program. For this, we can add to `/etc/nslcd.conf`:

```
1  tls_reqcert never
```

If SSSD is used instead, then we can edit `/etc/sssd/sssd.conf` and add the following line:

```
1  ldap_tls_reqcert = never
```

When using Certificates that are well signed from an External Certificate Authority, this of course becomes unnecessary.

1.4 Understanding authconfig Configuration Files

1.4.1 Authconfig Configuration

The primary configuration of the authconfig utitlity is located at `/etc/sysconfig/authconfig`. The contents of this file is used by other config files, such as `USELDAP=yes`.

```
CACHECREDENTIALS=yes
FAILLOCKARGS="deny=4 unlock_time=1200"
FORCELEGACY=no
FORCESMARTCARD=no
IPADOMAINJOINED=no
IPAV2NONTP=no
PASSWDALGORITHM=sha512
USEDB=no
USEECRYPTFS=no
USEFAILLOCK=no
USEFPRINTD=no
USEHESIOD=no
USEIPAV2=no
USEKERBEROS=yes
USELDAP=yes
USELDAPAUTH=no
USELOCAUTHORIZE=yes
USEMKHOMEDIR=no
USENIS=no
USEPAMACCESS=no
USEPASSWDQC=no
USEPWQUALITY=yes
USESHADOW=yes
USESMARTCARD=no
USESSSD=yes
USESSSDAUTH=no
USESYSNETAUTH=no
USEWINBIND=no
USEWINBINDAUTH=no
WINBINDKRB5=no
```

These are the settings we provided to the **authconfig** utility.

1.4.2 SSSD Configuration

Things like the Kerberos password, the LDAP search base, etc. and other IPA specific settings are stored in the `/etc/sssd/sssd.conf` file, to ensure that the connection to the IPA Server is successfully initiated and it's possible to login and use the services provided by it. Typical contents of this file look like:

```
[sssd]
config_file_version = 2
services = nss, pam
# SSSD will not start if you do not configure any domains.
# Add new domain configurations as [domain/<NAME>] sections, and
# then add the list of domains (in the order you want them to be
# queried) to the "domains" attribute below and uncomment it.
; domains = LDAP

```

```
[nss]

[pam]

# Example LDAP domain
; [domain/LDAP]
; id_provider = ldap
; auth_provider = ldap
# ldap_schema can be set to "rfc2307", which stores group member names in the
# "memberuid" attribute, or to "rfc2307bis", which stores group member DNs in
# the "member" attribute. If you do not know this value, ask your LDAP
# administrator.
; ldap_schema = rfc2307
; ldap_uri = ldap://ldap.mydomain.org
; ldap_search_base = dc=mydomain,dc=org
# Note that enabling enumeration will have a moderate performance impact.
# Consequently, the default value for enumeration is FALSE.
# Refer to the sssd.conf man page for full details.
; enumerate = false
# Allow offline logins by locally storing password hashes (default: false).
; cache_credentials = true

# An example Active Directory domain. Please note that this configuration
# works for AD 2003R2 and AD 2008, because they use pretty much RFC2307bis
# compliant attribute names. To support UNIX clients with AD 2003 or older,
# you must install Microsoft Services For Unix and map LDAP attributes onto
# msSFU30* attribute names.
; [domain/AD]
; id_provider = ldap
; auth_provider = krb5
; chpass_provider = krb5
;
; ldap_uri = ldap://your.ad.example.com
; ldap_search_base = dc=example,dc=com
; ldap_schema = rfc2307bis
; ldap_sasl_mech = GSSAPI
; ldap_user_object_class = user
; ldap_group_object_class = group
; ldap_user_home_directory = unixHomeDirectory
; ldap_user_principal = userPrincipalName
; ldap_account_expire_policy = ad
; ldap_force_upper_case_realm = true
;
; krb5_server = your.ad.example.com
; krb5_realm = EXAMPLE.COM
```

This is probably one of the most important configuration files when **SSSD** is being used. If **nslcd** is being used instead, then the config file of interest is `/etc/nslcd.conf`.

1.4.3 Kerberos Configuration File

The Kerberos configuration file (for connecting to a Kerberos Server) is stored in `/etc/krb5.conf` and typically has contents like:

```
# Configuration snippets may be placed in this directory as well
includedir /etc/krb5.conf.d/

includedir /var/lib/sss/pubconf/krb5.include.d/
```

```
[logging]
default = FILE:/var/log/krb5libs.log
kdc = FILE:/var/log/krb5kdc.log
admin_server = FILE:/var/log/kadmind.log

[libdefaults]
dns_lookup_realm = true
ticket_lifetime = 24h
renew_lifetime = 7d
forwardable = true
rdns = false
# default_realm = EXAMPLE.COM
default_ccache_name = KEYRING:persistent:%{uid}

dns_lookup_kdc = true
default_realm = SOMUVMNET.LOCAL
[realms]
# EXAMPLE.COM = {
#  kdc = kerberos.example.com
#  admin_server = kerberos.example.com
# }

SOMUVMNET.LOCAL = {
}

[domain_realm]
# .example.com = EXAMPLE.COM
# example.com = EXAMPLE.COM
somuvmnet.local = SOMUVMNET.LOCAL
.somuvmnet.local = SOMUVMNET.LOCAL
```

Here, the DNS domain to realm mapping is specified, to tell us which domain on the DNS belongs to which Kerberos realm.

1.4.4 NSSwtich Configuration

This file specifies the locations and the order in which passwords are searched for authentication. This includes the order in which passwords, shadow and groups are searched. The order is typically like:

```
passwd:     files sss ldap
shadow:     files sss ldap
group:      files sss ldap
```

This instructs the system to look for passwd files in the local file system first, then SSS and finally LDAP. The same is true for the two following categories of shadow and group.

1.4.5 NSLCD Configuration

While this file may be missing from newer versions of RHEL, this is an older version of LDAP configuration file. This file is supposed to be replaced by the `/etc/sssd/sssd.conf` file, and thus, all relevant settings should be provided in that file.

Chapter 2

Configuring iSCSI Target and Initiator

2.1 Understanding iSCSI Target and Initiator

SCSI (Small Computer System Interface) [read as *scuzzy*] is an alternative to ATA (a.k.a. IDE) Hard drives, which most consumer computers stick to. While SCSI drives provide significantly more throughput for certain scenarios, IDE suffices for most home computer usage. However, in case of servers, SCSI proves to be a much better alternative, since they provide more reliability and data transfer speed (much higher than ATA), owing to the fact that data transfer occurs in full-duplex mode (i.e., data can be read and written at the same time at full speeds). They also boast higher speeds (such as 15,000 RPM) as compared to ATA speeds (7200 RPM). Another reason servers tend to use SCSI (or related technologies, such as Serially Attached SCSI or SAS) is that the protocol makes it easy to *daisy-chain* several SCSI devices to the same controller, several times that of IDE devices. In fact, in the pre-USB era, SCSI was the go-to common interface for connecting peripherals or even devices such as printers.

Traditional SCSI devices use a long cable and a SCSI **Command Descriptor Block (CDB)** command to interact with the SCSI devices. In case of iSCSI, the same CDBs are used, but they're transmitted over IP packets over a network, instead of the cable. Thus, the SCSI devices are emulated by using a storage backend and presenting them on the network using iSCSI targets. SCSI targets are typically storage devices, while the hosts they're connected to are the initiators. Thus, this technology enables us to share PVs or LVs on the network, represented by iSCSI targets.

A **Storage Area Network (SAN)** is a network that provides access to a consolidated, block level data storage. *Block devices* provide a buffered data storage method, where data is transferred from the kernel buffer to the physical device. Also, data can be read and written in entire blocks. SANs thus present devices such as disk arrays as locally attached storage to servers. **Fiber Channel** or **FC** is a high speed network technology developed to enable fast data transfers between servers and SANs. Ethernet structures utilizing iSCSI technology can be as fast as their FC structure counterparts, thus making the technology enterprise ready for SAN creation.

2.1.1 iSCSI Operation

In the case of iSCSI storage, we have the SAN, on which runs a *iSCSI target* which can provide access to the storage backend. For any server that needs to access the files hosted

by the SAN, it needs to run an **iSCSI initiator**, which performs a discovery operation first. During this, the SAN tells it about the iSCSI devices it has to offer. Once this is complete, the iSCSI initiator can login to the devices.

2.1.2 iSCSI Components

Both the iSCSI targets and the storage backends need to be set up for the SAN to operate. The storage backend can be an entire disk, a dedicated partition, a logical volume or even a file! The servers, running the iSCSI initiators, will see the iSCSI targets as new storage devices after successfully logging in to them. This can be verified by viewing the output of the `/proc/partitions` file. A tool called *lsscsi* can also alternatively used, although it is not installed by default.

2.1.3 Basic iSCSI Terminology

Terms	Description
IQN	iSCSI Qualified Name - an unique name assigned to each iSCSI target and initiator, used to identify them.
Initiator	The iSCSI client that is identified by its IQN.
Target	The service on the iSCSI server that gives access to the storage backend.
ACLs	Access Control Lists that are based on the node's IQNs.
Portal	Also known as **nodes**, this is the combination of the IP address and the port that are used by both targets and initiators to establish connections.
discovery	The process through which an indicator finds the available targets that are configured for a given portal.
LUNs	The *Logical Unit Number* is a number used to identify the logical unit (i.e., block devices shared through the target) being addressed by the iSCSI Controller.
login	The act which gives an initiator the relevant LUNs.
TPG	The *Target Portal Group* is a collection of IP Addresses and TCP ports to which a particular iSCSI Target will listen.

So, there can be more than one portals per server, and more than one targets per portal.

2.1.4 After connecting an initiator to an iSCSI Target

The new block devices thus accessed will appear as local devices (/dev/sdb, /dev/sdc etc.) Note that if a LUN is available and used by multiple servers, multiple devices can access the LUN post connection, i.e., multiple servers can use the disk at the same time. This is a bit dangerous, since it requires using clustering, for providing multiple servers to use the storage. Otherwise for a file system like XFS or Ext4, two servers writing to the same file can cause data loss.

To avoid this, shared file systems such as GFS2 can be used. In GFS2, the file system cache is shared among all the nodes. Thus, all nodes writing to the file system know what all the other nodes are doing.

Chapter 3

System Performance Reporting

Chapter 4

System Optimization Basics

Chapter 5

Configuring Logging

Part II

Networking and Apache

Chapter 6

Configuring Advanced Networking

Chapter 7

Managing Linux Based Firewalls

Chapter 8

Configuring Apache Virtual Hosts

Chapter 9

Managing Advanced Apache Features

Part III

DNS and File Sharing

Chapter 10

Configuring a Cache-only DNS Server

Chapter 11

Configuring NFS File Sharing

Chapter 12

Managing SMB File Sharing

Part IV

Essential Back-end Services

Chapter 13

Setting up an SMTP Server

Chapter 14

Managing SSH

Chapter 15

Managing MariaDB

Chapter 16

Managing Time Services

Chapter 17

Shell Scripting