Network Security Project

OpenLDAP and OpenVPN
HOWTO

Introduction

LDAP services allows enterprise systems to centralize and store their users' authentication profile. It allows integration of authentication across platforms and systems without the need of maintaining multiple different user databases for all the systems in the organization.

As for OpenVPN, it is a way for remote clients, to connect to organizational private network securely and conveniently. OpenVPN too can connect 2 remote networks together into a bigger network.

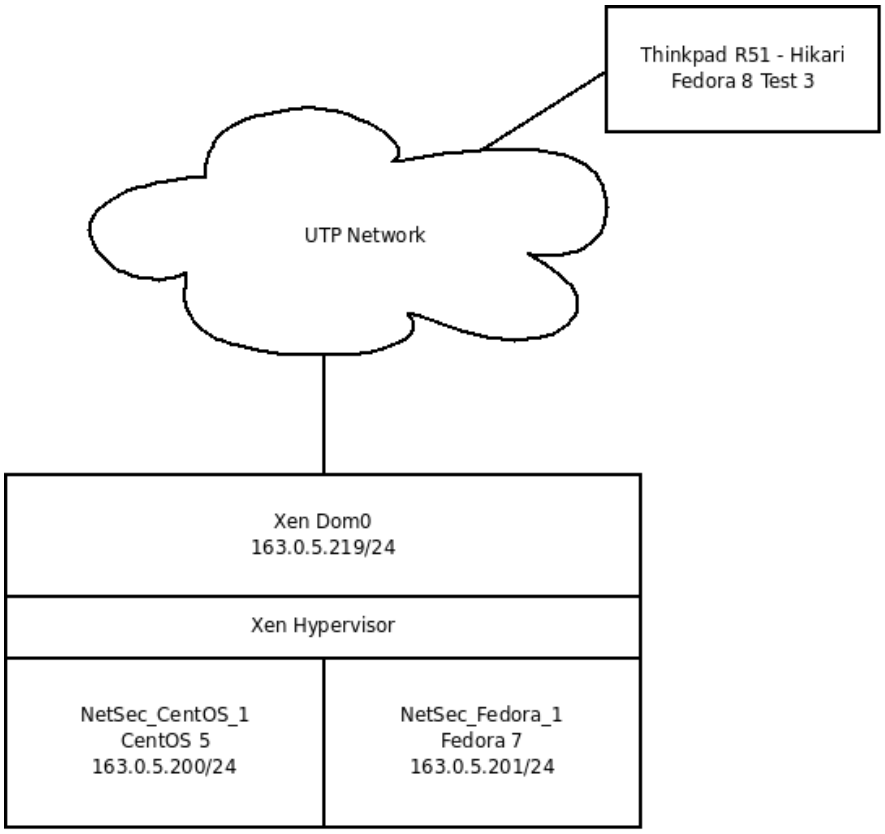
Combined together, these two network services can create a great centralized environment for an organization where services can be provided in one central system and the organization members can communicate, share information, and access services securely from all around the globe.

This Network Security project will cover a basic set-up of a LDAP server using OpenLDAP and setting up a Linux system to allow login using the LDAP informations. This project will also cover a basic setup of an OpenVPN network, complete with routing and certificate based authentication. This project will also tries to demonstrate the use of both services with PETRONAS University of Technology (UTP)'s network as its platform.

The Setup

The infrastructure of this project consist of 3 Xen virtualized systems (a.k.a domains/DomU) which each has their own set of services. The Xen Dom0 (the host server) is a Fedora 7 system with 1.7Ghz processor and 1GB of RAM. The Xen domains consist of 1 CentOS and 2 Fedora 7 servers, of which all running without X (the GUI environment). All of the Xen DomU and Dom0 are running inside Village 3 E IP subnet.

A laptop will be located outside the network so that it can demonstrate the use of OpenVPN.



LDAP Implementation

The LDAP server is hosted by NetSec_CentOS_1, with OpenLDAP as the LDAP service provider.

Installation of OpenLDAP 2.3 on CentOS 5

OpenLDAP packages are available from the CentOS software repository. Installation can be done quickly through YUM Update Manager.

```
yum install openldap-servers openldap-clients
```

Initial Setup of OpenLDAP Slapd

After the installation process is done, edit the configuration of the LDAP server to meet the deployment environment settings. The configuration is located at /etc/openvpn/slapd.conf. The initial configuration provided by the distribution package have all basic options to get slapd running without any configuration change. For this project, I have changed several of the lines to make it unique to this project's requirement. Changes are:

```
suffix "dc=kagesenshi,dc=org"
rootdn "cn=Manager,dc=kagesenshi,dc=org"
rootpw {SSHA}VtX51/b3WSgFDRxJWMQjxtbt9WIz9eWW
```

Suffix is a string of Domain Controllers which will be used to identify which domain this LDAP server belongs to. Rootdn is the user who have full access to the LDAP system , similar to *NIX's “root” user. Rootpw is a hash containing the Rootdn's password generated from slappasswd command.

After configuration is done, the LDAP server can be started using this command

```
/sbin/service ldap start
```

More details of configuring the OpenLDAP server can be acquired from the man pages or from lots of resources on the Internet.

```
man slapd.conf
http://www.openldap.org/doc/admin24/slapdconfig.html
```

Adding Users into the LDAP server

Before we can start using the LDAP server to authenticate users, we should add users into the server's database. Adding users into OpenLDAP server is done by the command “ldapadd”. User details are entered in a LDIF file (which is a plain text file with a predefined format) and added into the database through execution of “ldapadd”. Details about LDIF format can be acquired from the manpage of ldif or from in the link below:

```
man ldif
http://www.openldap.org/doc/admin24/dbtools.html#The%20LDIF%20text%20entry
%20format
http://www.rfc-editor.org/rfc/rfc2849.txt
```

The first user we should add into the database is of course, the Rootdn user. Without adding the Rootdn into the database, we are stuck with only local administration of the LDAP server. So, rootdn.ldif file is created with this content

```
dn: dc=kagesenshi,dc=org
objectclass: dcObject
objectclass: organization
o: KageSenshi.Org
dc: kagesenshi

dn: cn=Manager,dc=kagesenshi,dc=org
objectclass: organizationalRole
cn: Manager
```

The user is added through executing

```
ldapadd -x -D "cn=Manager,dc=kagesenshi,dc=org" -W -f rootdn.ldif
```

Adding another user is just a repetition of above process, but with the cn and objectclass set to a different value.

newuser.ldif:

```
dn: cn=Mohd Izhar Firdaus,dc=kagesenshi,dc=org
objectclass: organizationalPerson
cn: Mohd Izhar Firdaus
sn: Ismail
userpassword: {SSHA}UvAJkdlglEovmaOtlsQrNsSPnvAlNZKz
```

Add the user:

```
ldapadd -x -D "cn=Manager,dc=kagesenshi,dc=org" -W -f newuser.ldif
```

This will create user Mohd Izhar Firdaus with password “password”. Like before, the password hash is generated using slappasswd command.

Adding User Remotely

To add user remotely, the same command is used like above but with an additional -H option. The computer which is going to run the command will require openldap-clients package to be installed before it can do this.

```
ldapadd -H ldap://163.0.5.2OO:389 -x \
-D "cn=Manager,dc=kagesenshi,dc=org" -W -f newuser.ldif
```

Testing the LDAP Server

The quickest way to test whether the LDAP is running and queryable is through executing a LDAP search to the directory server. This command is run at NetSec_Fedora_1.

```
ldapsearch -H ldap://163.0.5.2OO -x \
-b 'dc=kagesenshi,dc=org' '(objectclass=*)'
```

If you get a reply which contains all objects under dc=kagesenshi,dc=org , your LDAP setup is running properly. If not, you might misconfigured something (like firewalls, routing tables etc).

Authenticating Login through LDAP

Here I will describe the process of setting up NetSec_Fedora_1 to authenticate users logging into its system through LDAP at NetSec_CentOS_1. First, we will need to configure the system to select LDAP as one of its authentication method. This can be easily done in a Fedora/CentOS/RedHat Enterprise systems through their authentication configurator, system-config-authentication.

```
system-config-authentication --enableldap --enableldapauth --update
```

This basically configures all parameter for ldap authentication automatically. Next, we will need to point the system to the LDAP server. Edit /etc/ldap.conf and /etc/openldap/ldap.conf to point to 163.0.5.200. Now, with that setup, its time to proceed to setting up a user at the LDAP server. There are several fields in the user information which are needed for the authentication to work. So, the entry for the user need to be modified. Modification is also done through the “ldapadd” command.

modifyuser.ldif:

```
dn: cn=Mohd Izhar Firdaus,dc=kagesenshi,dc=org
changetype: modify
uid: izhar
objectclass: posixaccount
ou: People
homedirectory: /home/izhar/
loginshell: /bin/bash
uidnumber: 4000
gidnumber: 500
```

Execute this command to apply changes:

```
ldapadd -H ldap://163.0.5.200:389 -x \
-D "cn=Manager,dc=kagesenshi,dc=org" -W -f modifyuser.ldif
```

With that command executed, the LDAP authentication setup is nearly finished. The authentication is tested by logging out of NetSec_Fedora_1 and relogin with the username : izhar with the password: “password” (as setup earlier above). If everything was configured properly, a shell should appear.

The LDAP authentication does not create a home folder for the user, so, a plain shell appeared with an error saying could not find the home folder. To fix this, add this line into /etc/pam.d/system-auth.

```
session required pam_mkhomedir.so skel=/etc/skel/ umask=0077
```

With that added, the LDAP authentication system is now done.

OpenVPN Tunneling

The OpenVPN server is hosted at NetSec_Fedora_1. In the infrastructure, the OpenVPN server allows outside clients outside Village 3 to be able to access the other servers in the infrastructure. The OpenVPN server will act as a gateway to the network which can be tunneled through proxies.

Installation of OpenVPN 2.1 on Fedora 7

Fedora is the upstream for both CentOS and Red Hat Enterprise, so, it uses a similar package manager like the 2 downstreams, which is YUM. OpenVPN can be installed by running this command:

```
yum install openvpn
```

Generating Authentication Keys

OpenVPN uses certificate based authentication for users of the VPN. First, we will need to create a certificate set for the server. The package from Fedora repository provided a certificate generator together with it, complete with some helper scripts to help configuring OpenVPN. The certificate generator is located at /usr/share/openvpn/easy-rsa/2.0/. There is a script named “vars” in the directory. This file contains environment variables for the certificate generator's use. For this infrastructure, I edited these few lines of the code.

```
export KEY_SIZE=1024
# In how many days should the root CA key expire?
export CA_EXPIRE=3650
# In how many days should certificates expire?
export KEY_EXPIRE=3650
# These are the default values for fields
# which will be placed in the certificate.
# Don't leave any of these fields blank.
export KEY_COUNTRY="MY"
export KEY_PROVINCE="Perak"
export KEY_CITY="SriIskandar"
export KEY_ORG="KageSenshi.Org"
export KEY_EMAIL="izhar@fedoraproject.org"
```

Then, these commands are executed to start generating the server side certificates:

```
source ./vars
./clean-all
./build-ca
./build-key-server <servername>
```

The “servername” parameter at build-key-server execution is replaced with a name for the server, in my project, I simply put the name as “server”. Next is to create a Diffie Hellman parameter for the OpenVPN server.

```
./build-dh
```

After the above scripts finished execution, a folder named “keys” will appear in the easy-rsa/2.0 directory. All of the files in the keys folder is then copied to /etc/openvpn/keys/ directory as the openvpn daemon will search for the keys there. Take note of files named starting with <servername> as it will be required for the configuration later.

Next is to create the keys for the clients. Each clients may share the same keys or have one unique key for each of them. Client side keys are generated from these commands:

```
./build-key <clientname>
```

I used “client” as the client name. After generating the keys, several files starting with <clientname> will appear in the keys folder. These files are going to be given to clients which are going to connect to the OpenVPN server. The .csr and .crt files are going to be needed by the server for authenticating the user, so they are copied to /etc/openvpn/keys/ too.

Configuring the Server

OpenVPN allows a lot of options for configuring the VPN service. Due to the complexity, I am not going to explore all of it, but instead uses a very basic working example of a OpenVPN configuration using Tun protocol on TCP listener.

Configuring it is easy enough, just this sample config file is used for it. Configuration files are stored in /etc/openvpn/.

server-tcp-443.conf:

```
port 443
 proto tcp
dev tun
ca keys/ca.crt
cert keys/<servername>.crt
key keys/<servername>.key
dh keys/dh1024.pem
server 10.12.0.0 255.255.255.0
ifconfig-pool-persist ipp.txt
push "redirect-gateway"
keepalive 10 120
comp-lzo
persist-key
persist-tun
status openvpn-status-server-tcp-443.log
verb 3
```

As usual, <servername> is replaced with the server's name, which in this case , simply “server”. This configuration sets openVPN to listen on port 443/TCP, and uses IP 10.12.0.0 as the VPN network's IP. The reason why I used port 443 to listen for OpenVPN connections is because most proxies allows TCP-Connect on that port. So, this VPN server can be accessed virtually from anywhere inside UTP as UTP's proxies allows TCP-Connect on port 443.

The OpenVPN server is started using this command:

```
/sbin/service openvpn start
```

NOTE: If the server uses SELinux Enforcing , the service might not able to start. A solution is to switch from enforcing to permissive using system-config-securitylevel

Setting up the Client

As for the client side, this configuration is used.

myserver-tcp-443.ovpn

```
client
dev tun
proto tcp
remote 163.0.5.201 443
float
resolv-retry infinite
nobind
persist-key
persist-tun
ca keys/ca.crt
cert keys/<clientname>.crt
key keys/<clientname>.key
ns-cert-type server
comp-lzo
verb 3
http-proxy 160.0.234.1 8080
```

As usual, <clientname> is replaced with the client name during key generation. The keys generated at the server are copied to the client's /etc/openvpn/keys/ directory.

Connecting to the OpenVPN Network

Connecting to the network is fairly easy, simply run this command at the client.

```
/sbin/openvpn  myserver-tcp-443.ovpn
```

OpenVPN connection has been successfully established after this line appears in the output.

```
Initialization Sequence Completed
```

Testing the Connection

Depending on the network settings at the server's configuration file, ping the server's IP. In this setup, the server's IP is 10.12.0.1 as the settings uses 10.12.0.0/24 as the VPN network. If the server replies, the OpenVPN is now working.

Routing The VPN Connection to the Server's Network

Once the VPN successfully tested running properly, its time to set up the routing system for the server so that the client can connect to the other computers in the server's network. Routing in Linux is done using several Iptables rules, and these are the rules which need to be setup at the server.

- Allow masquerade traffic to Physical network from VPN connection
- Allow masquerad traffic from Physical network to VPN connection

These rules can be applied quickly using these iptables command.

```
iptables -t nat -A POSTROUTING -s 10.12.0.0/24 -o eth0 -j MASQUERADE
iptables -t nat -A POSTROUTING -s 163.0.5.0/24 -o tun0 -j MASQUERADE
```

To make this rules always starts on boot , they need to be added to /etc/sysconfig/iptables. Adding them is pretty simple, just insert these lines into the configuration files.

```
*nat
:PREROUTING ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A POSTROUTING -s 10.12.0.0/255.255.255.0 -o eth0 -j MASQUERADE
-A POSTROUTING -s 163.0.5.0/255.255.255.0 -o tun0 -j MASQUERADE
COMMIT
```

Another Iptables rule which will be required for masquerading are these (inserted below :FORWARD ACCEPT[0:0]).

```
-A FORWARD -s 10.12.0.0/24 -j ACCEPT
-A FORWARD -d 10.12.0.0/24  -j ACCEPT
```

End Note

The design was originally planned to use Fedora Directory Server as the LDAP service provider as it provides several GUIs for helping LDAP administration. However, my assumption of GUI as easier to use was proven wrong when I spent several weeks trying to configure Fedora-ds properly. The GUI of Fedora-ds is designed for help experienced admins rather than for a quick setup, which left me struggling to get. After several weeks of pain without gain, the project took a different direction by implementing LDAP using OpenLDAP instead.

References

LDAP

- http://www.linuxtopia.org/online_books/centos_linux_guides/centos_linux_reference_guide/s1-ldap-pam.html
- http://www.linuxtopia.org/online_books/centos_linux_guides/centos_linux_reference_guide/s1-ldap-pam.html
- http://publib-b.boulder.ibm.com/Redbooks.nsf/RedbookAbstracts/tips0410.html?Open
- http://publib.boulder.ibm.com/infocenter/wsdoc400/v6r0/index.jsp?topic=/com.ibm.websphere.iseries.doc/info/ae/ae/tsec_seccltpa.html
- http://linux.derkeiler.com/Mailing-Lists/RedHat/2006-01/msg00312.html

OpenVPN

- http://www.imped.net/oss/misc/openvpn-2.0-howto-edit.html
- http://www.smallwhitecube.com/php/dokuwiki/doku.php?id=howto:openvpn
- http://www.howforge.com/how-to-install-openvpn-on-fedora-core-2