LPI exam prep: Network client management

Intermediate Level Administration (LPIC-2) topic 210

Skill Level: Intermediate

David Mertz (mertz@gnosis.cx)
Developer
Gnosis Software, Inc.

24 May 2006

In this tutorial, the fifth in a series of seven tutorials covering intermediate network administration on Linux, David Mertz continues preparing you to take the Linux Professional Institute Intermediate Level Administration (LPIC-2) Exam 202. By following this tutorial, you will examine several protocols' centralized configuration of network settings on clients within a network. DHCP is widely used to establish basic handshaking to clients machines such as assigning IP addresses. At a higher level, NIS and (more often) LDAP are used for arbitrary shared information among machines on a network. This tutorial also discusses PAM, which is a flexible, networked, user authentication system.

Section 1. Before you start

Learn what these tutorials can teach you and how you can get the most from them.

About this series

The Linux Professional Institute (LPI) certifies Linux system administrators at two levels: *junior level* (also called "certification level 1") and *intermediate level* (also called "certification level 2"). To attain certification level 1, you must pass exams 101 and 102; to attain certification level 2, you must pass exams 201 and 202.

developerWorks offers tutorials to help you prepare for each of the four exams. Each exam covers several topics, and each topic has a corresponding self-study tutorial on developerWorks. For LPI exam 202, the seven topics and corresponding developerWorks tutorials are:

© Copyright IBM Corporation 1994, 2006. All rights reserved.

Table 1. LPI exam 202: Tutorials and topics

LPI exam 202 topic	developerWorks tutorial	Tutorial summary
Topic 205	LPI exam 202 prep (topic 205): Networking configuration	Learn how to configure a basic TCP/IP network, from the hardware layer (usually Ethernet, modem, ISDN, or 802.11) through the routing of network addresses.
Topic 206	LPI exam 202 prep (topic 206): Mail and news	Learn how to use Linux as a mail server and as a news server. Learn about mail transport, local mail filtering, mailing list maintenance software, and server software for the NNTP protocol.
Topic 207	LPI exam 202 prep (topic 207): DNS	Learn how to use Linux as a DNS server, chiefly using BIND. Learn how to perform a basic BIND configuration, manage DNS zones, and secure a DNS server.
Topic 208	LPI exam 202 prep (topic 208): Web services	Learn how to install and configure the Apache Web server, and learn how to implement the Squid proxy server.
Topic 210	LPI exam 202 prep (topic 210): Network client management	(This tutorial) Learn how to configure a DHCP server, an NIS client and server, an LDAP server, and PAM authentication support. See detailed objectives below.
Topic 212	LPI exam 202 prep (topic 212): System security	Coming soon
Topic 214	LPI exam 202 prep (topic 214): Network troubleshooting	Coming soon

To start preparing for certification level 1, see the developerWorks tutorials for LPI exam 101. To prepare for the other exam in certification level 2, see the developerWorks tutorials for LPI exam 201. Read more about the entire set of developerWorks LPI tutorials.

The Linux Professional Institute does not endorse any third-party exam preparation material or techniques in particular. For details, please contact info@lpi.org.

About this tutorial

© Copyright IBM Corporation 1994, 2006. All rights reserved.

Welcome to "Network client management," the fifth of seven tutorials covering intermediate network administration on Linux. In this tutorial, you learn about several protocols' centralized configuration of network settings on clients within a network, look at how DHCP is widely used to establish basic handshaking to clients machines (such as assigning IP addresses), and see how, at a higher level, NIS and (more often) LDAP are used for arbitrary shared information among machines on a network. This tutorial also discusses PAM (Pluggable Authentication Module), a flexible, networked, user-authentication system.

As with the other tutorials in the developerWorks 201 and 202 series, this tutorial is intended to serve as a study guide and entry point for exam preparation, rather than complete documentation on the subject. Readers are encouraged to consult LPI's detailed objectives list and to supplement the information provided here with other material as needed.

is organized according to the LPI objectives for this topic. Very roughly, expect more questions on the exam for objectives with higher weight.

Table 2. Web services: Exam objectives covered in this tutorial

LPI exam objective	Objective weight	Objective summary
2.210.1 DHCP configuration	Weight 2	Configure a DHCP server. This objective includes setting default and per client options, adding static hosts and BOOTP hosts. Also included is configuring a DHCP relay agent and maintaining the DHCP server.
2.210.2 NIS configuration	Weight 1	Configure an NIS server. This objective includes configuring a system as an NIS client.
2.210.3 LDAP configuration	Weight 1	Configure an LDAP server. This objective includes working with directory hierarchy, groups, hosts, services, and adding other data to the hierarchy. Also included is importing and adding items, as well as adding and managing users.
2.210.4 PAM authentication	Weight 2	Configure PAM to support authentication using various available methods.

Prerequisites

To get the most from this tutorial, you should already have a basic knowledge of Linux and a working Linux system on which you can practice the commands covered in this tutorial.

© Copyright IBM Corporation 1994, 2006. All rights reserved.

Section 2. Introduction

About DHCP

Dynamic Host Configuration Protocol (DHCP) is a successor to the older BOOTP protocol. The principle role of a DHCP server is to assign IP addresses to client machines that may connect or disconnect from a network. Most IP networks, even those with stable topologies and client lists, use DHCP to prevent conflicts in IP address allocation.

Additionally, a DHCP server provides clients with routing and subnet information, DNS addresses, and in some cases other information. DHCP assignments may have varying durations, ranging from short to infinite, depending on server configuration and client request details. In fact, DHCP is consistent with assigning fixed IP addresses to specific machines (via their MAC hardware addresses), but in any case prevents conflicts among machines.

The formal specification of DHCP is RFC 2131 (see Resources later in this tutorial for a link).

About NIS

The Network Information Service (NIS) is Sun Microsystems' "Yellow Pages" (YP) client-server directory service protocol for distributing system configuration data such as user and host names on a computer network.

NIS/YP is used for keeping a central directory of users, hostnames, and most other useful things in a computer network. For example, in a common UNIX environment, the list of users (for authentication) is placed in /etc/passwd. Using NIS adds another "global" user list which is used for authenticating users on any host.

For the most part, NIS has been superseded by the more general and more secure LDAP for general use.

A good starting point for further information on NIS is the "The Linux NIS(YP)/NYS/NIS+ HOWTO" (see Resources for a link).

About LDAP

The Lightweight Directory Access Protocol (LDAP) is a client-server protocol for accessing directory services, specifically X.500-based directory services.

© Copyright IBM Corporation 1994, 2006. All rights reserved.

An LDAP directory is similar to a database, but tends to contain more descriptive, attribute-based information. As such, LDAP provides enough flexibility for storing any type of network-shared information. The information in a directory is read much more often than it is written, so it is tuned to give quick-response to high-volume lookup or search operations.

LDAP has the ability to replicate information widely in order to increase availability and reliability while reducing response time. When directory information is replicated, any temporary inconsistencies between replicas will become synced over time.

The formal specification of LDAP is RFC 2251 (see Resources for a link).

About PAM

Linux-PAM (Pluggable Authentication Modules for Linux) is a suite of shared libraries that enable the local system administrator to choose how applications authenticate users.

A PAM-aware application can switch at runtime between authentication mechanism(s). Indeed, you may entirely upgrade the local authentication system without recompiling the applications themselves. This PAM library is configured locally with a system file, /etc/pam.conf (or a series of configuration files located in /etc/pam.d/) to authenticate a user request via the locally available authentication modules. The modules themselves will usually be located in the directory /lib/security and take the form of dynamically loadable object files.

The Linux-PAM System Administrators' Guide is a good starting point for further information (see Resources for a link).

Other resources

As with most Linux tools, it is always useful to examine the manpages for any utilities discussed. Versions and switches might change between utility or kernel version or with different Linux distributions. For more in-depth information, the Linux Documentation Project has a variety of useful documents, especially its HOWTOs. A variety of books on Linux networking have been published; I have found O'Reilly's *TCP/IP Network Administration*, by Craig Hunt to be quite helpful. (See Resources for links.)

Section 3. DHCP configuration

The overall protocol

© Copyright IBM Corporation 1994, 2006. All rights reserved.

As with many network protocols, the Dynamic Host Configuration Protocol (DHCP) is a client/server interface. A DHCP client is a much simpler program, both internally and to configure, than is a DHCP server. Essentially, the job of a DHCP client is to broadcast a DHCPDISCOVER message on its local physical subnet, then await a response.

The DHCPDISCOVER message MAY include options that suggest values for the network address and lease duration. Servers that receive a DHCPDISCOVER message should respond to the requesting MAC address with a DHCPOFFER message. The client, in turn, responds with a DHCPREQUEST message to one of the offering servers, usually to the first (and only) responding server.

The actual configuration parameters a client uses are received in a DHCPACK message. At that point, the client has received an allocated IP address and its communications will move, so to speak, from the Data Link Layer (Ethernet) to the Network Layer (IP).

The client process

A DHCP client typically only needs to be configured with the set of information it wishes to obtain. For example, Debian-based distributions typically use the DHCP client, dhclient, which is configured with the /etc/dhcp3/dhclient.conf file. The sample file that is distributed with the dhcp3-client package has all but one configuration option commented out. The one enabled option might look like:

Listing 1. Option for dhclient.conf

```
        request subnet-mask, broadcast-address, time-offset, routers,
          domain-name, domain-name-servers, host-name,
          netbios-name-servers, netbios-scope;
```

In this example, the default configuration, the client essentially just says "ask for everything possible." In the negotiation messages, the DHCPACK message from the server will contain information for all these requested values which the client will use once they are received. Client IP address is implied in this list since that configuration is always negotiated.

As well as specifying timeout and lease time options (and a few others), a client *may*, but need not in most cases, put some restrictions on the IP addresses it wishes to use. To exclude a particular one, you can use `reject 192.33.137.209;`. To specify the explicit address the client wishes to use, then use `fixed-address 192.5.5.213;`.

A client may reject a lease offer with the DHCPDECLINE message, but servers will try to fulfill requests where possible. A DHCP server may also make a fixed assignment of a particular IP address to a requesting MAC address; configuring a per-machine IP address is more often done with server configuration than with client configuration.

In order to keep track of acquired leases, dhclient keeps a list of leases it has been

© Copyright IBM Corporation 1994, 2006. All rights reserved.

assigned in the /var/lib/dhcp3/dhclient.leases file (the path may vary across distros); this way a non-expired DHCP lease is not lost if a system disconnects from the physical network and/or reboots.

The server process

A DHCP server needs to know a bit more about its options since it provides various information to clients in DHCP leases and also must assure that IP addresses are uniquely assigned per client. The DHCP server usually runs as the daemon, dhcpd, and takes its configuration information from /etc/dhcpd.conf (this path may vary across distros). A single dhcpd daemon may manage multiple subnets, generally if multiple physical networks connect to a server; most frequently however, one server manages one subnet. Listing 2 is a fairly complete example of a server configuration.

Listing 2. dhcpd.conf configuration options

```
# default/max lease in seconds: day/week
default-lease-time 86400;
max-lease-time 604800;
option subnet-mask 255.255.255.0;
option broadcast-address 192.168.2.255;
option routers 192.168.2.1;
# DNS locally, fallback to outside local domain
option domain-name-servers 192.168.2.1, 151.203.0.84;
option domain-name "example.com";
# Set the subnet in which IP address are pooled
subnet 192.168.2.0 netmask 255.255.255.0 {
   range 192.168.2.100 192.168.2.199;
}
# Assign a fixed IP to a couple machines by MAC address
group {
   use-host-decl-names true;
   host first.example.com {
      hardware ethernet 00:00:c1:a2:de:10;
      fixed-address 192.168.2.200;
   }
   host second.example.com {
      hardware ethernet 00:dd:cc:a1:78:34;
      fixed-address 192.168.2.201;
   }
```

When a client sends out a broadcast message to a server running with this configuration, it will either receive a lease on 192.168.2.200 or 192.168.2.201 if it has the specified MAC address, or it will receive a lease on an available address in the pool 192.168.2.100 through 192.168.2.199.

A client may also use the DHCPINFORM message to tell a server that it already has an assigned IP address (by manual configuration), but wishes to obtain other configuration information. Notice that informing a server that a client *is* using a particular IP address is not the same as a requesting a specific IP address; in the latter case, the server may or may not grant the request depending on existing leases. In the former case, the server has no voice in the decision and no lease is granted per se at all (however, servers will try to avoid assigning IP addresses known to be in use to new requesting clients).

When leases expire, clients and servers must negotiate new leases for configuration

© Copyright IBM Corporation 1994, 2006. All rights reserved.

parameters to remain valid. Shorter leases may be used where configuration information on a server is likely to change (for example, with dynamic DNS via an external WAN). A client may gracefully terminate a lease by sending the DHCPRELEASE message, but this is not required for correct operation (clients sometimes crash, reboot, or become disconnected without the opportunity to send this message, after all).

Absent a release message, a lease is maintained by the server for whatever time terms it was granted on, so a rebooted machine will often continue using its prior lease (which will be stored in dhclient.leases on both server and client).

Section 4. NIS configuration

When to use NIS

Most of the utilities associated with NIS are still named with a *yp* prefix because it was formerly known as "Sun Yellow Pages"; trademark issues forced the name change to NIS. NIS is used to let a network of machines share information such as users and groups (the contents of /etc/passwd and /etc/group, respectively), allowing users rights on any machine within a NIS domain.

NIS operates in a manner similar to DNS in defining domains where information is distributed and also in allowing master and slave servers to hierarchically distribute information within a domain. In fact, in principle NIS could be used in place of DNS by distributing domain name information found in /etc/hosts, but this is rarely done in practice. NIS has a certain flexibility in that any type of information can, in principle, be put into a NIS database (which is in DBM format, though the tool makedbm in the NIS server package converts flat files to this format, generally "behind the scenes").

There is also a service called NIS+ which was intended to supersede NIS and includes data encryption and authentication; however, NIS+ is not backwards compatible with NIS and is not widely used.

Before you start

To run any of the NIS utilities you need to run the daemon /sbin/portmap which converts RPC program numbers into TCP/IP (or UDP/IP) protocol port numbers since NIS clients make RPC calls. Most Linux distributions launch /sbin/portmap in their startup scripts, but you should check that this daemon is running with `% ps -ax | grep portmap`.

If not already running, install /sbin/portmap and include it in your system startup scripts.

© Copyright IBM Corporation 1994, 2006. All rights reserved.

NIS client utilities (ypbind daemon)

A NIS client includes the tools **ypbind**, **ypwhich**, **ypcat**, **yppoll**, and **ypmatch**. The daemon ypbind must run as root, and is normally launched as part of system startup scripts (though it is not required to do so).

The other tools rely on the services of ypbind but run at a user level. Old version of ypbind broadcast a binding request on the local network, but this allows a malicious NIS server to answer the request and provide bad user and group information to clients. It is preferable to configure specific servers for ypbind to connect to using the /etc/yp.conf file. If multiple servers are configured (or if broadcast is used despite the danger), ypbind may switch bound servers each 15 minutes according to which is able to respond most quickly. These various servers should be arranged in master/slave configuration, but the client does not need to know or care about that detail. For example, a ypbind configuration may look like:

Listing 3. /etc/yp.conf

```
ypserver 192.168.2.1
ypserver 192.168.2.2
ypserver 192.168.1.100
ypserver 192.168.2.200
```

Before /usr/sbin/ypbind runs, you must set the NIS domainname of your network. This may be whatever name the NIS server is configured to use and should generally be chosen as something different from the DNS domainname. Set the NIS domainname using (substitute the actual name): `% ypdomainname my.nis.domain`.

NIS client utilities (other configuration)

If you want to use NIS as part of your domain name lookup, you should modify /etc/host.conf to include NIS in the lookup order; for instance, to check a name first in /etc/hosts, then in NIS, then in DNS:

Listing 4. Modifying lookup order

```
% cat /etc/host.conf
order hosts,nis,bind
```

To enable NIS distributed users, you should modify the client /etc/passwd file to include `+::::::`.

The NIS database information acts as a template for login attempts with this "unfilled" pattern. You may fine-tune the user information if you like, for example:

Listing 5. Detailed /etc/passwd

```
+user1::::::
+user2::::::
+user3::::::
+@sysadmins::::::
-ftp
```

© Copyright IBM Corporation 1994, 2006. All rights reserved.

```
        +:*::::::/etc/NoShell
```

This allows login access only to `user1`, `user2`, and `user3`, and all members of the sysadmin netgroup, but provides the account data of all other users in the NIS database.

NIS sources themselves are configured in /etc/nsswitch.conf. The name might suggest this is strictly for name server lookup, but a variety of information types are described. Basically, this configuration describes the order in which information sources are searched. The name `nis` in an order means information obtained from a NIS server; the name `files` means to use an appropriate local configuration file. The name `dns` is used for the `hosts` option.

As well, you may specify what to do if an initial source does not contain the information desired: `return` means to give up (and unless NIS was simply altogether unresponsive, `continue` means to try the next source for that datum). For example:

Listing 6. /etc/nsswitch.conf

```
        passwd:         compat
        group:          compat
        shadow:         compat
        hosts:          dns [!UNAVAIL=return] files
        networks:       nis [NOTFOUND=return] files
        ethers:         nis [NOTFOUND=continue] files
        protocols:      nis [NOTFOUND=return] files
        rpc:            nis [NOTFOUND=return] files
        services:       nis [NOTFOUND=return] files
```

NIS client user utilities

The utilities **ypwhich**, **ypcat**, **yppoll**, and **ypmatch** are used at a user level to query NIS information:

- ypwchich prints the name of a NIS server.
- ypcat prints the values of all the keys in a NIS database.
- yppoll prints the version and master server of a NIS map.
- ypmatch prints the values of one or more keys from a NIS map.

See the corresponding manpages of each utility for more usage information.

NIS server utilities (ypinit, ypserv)

A NIS server uses the ypserv daemon to provide NIS databases to clients and is configured with the /etc/ypserv.conf file. As I mentioned, you may run both master and slave NIS servers within a domain. The set of NIS databases is initialized on a master server using (just the first time you run it; after that use `make -C /var/yp` like this: `% /usr/lib/yp/ypinit -m`.

© Copyright IBM Corporation 1994, 2006. All rights reserved.

A slave server is really just a NIS client that gets its databases from the master server (and runs ypserv). To copy master server information to the locally running slave server, use `% /usr/lib/yp/ypinit -s my.nist.domain`.

On the master server, the NIS databases are built from information in (some of) the following familiar configuration files:

- /etc/passwd,
- /etc/group,
- /etc/hosts,
- /etc/networks,
- /etc/services,
- /etc/protocols,
- /etc/netgroup,
- /etc/rpc.

Exactly what databases are exported is configured in /var/yp/Makefile which also propagates changes when it is rebuilt.

Slave servers will be notified of any change to the NIS maps when they are rebuilt (via the yppush program) and automatically retrieve the necessary changes in order to synchronize their databases. NIS clients do not need to do this since they continually talk to the NIS server to read the information stored in its DBM databases.

Section 5. LDAP configuration

When to use LDAP

In principle, the Lightweight Directory Access Protocol is similar in purpose to NIS. Both distribute some structured information about network configurations from client to server; however, LDAP goes further in hierarchically structuring which information is distributed to which clients, redirecting requests to other LDAP servers where necessary and in building in security mechanisms. Moreover, LDAP provides mechanisms and tools for clients to update information held in LDAP servers which in turn distribute that information to other clients requesting it (subject to permissions, of course).

Installation

© Copyright IBM Corporation 1994, 2006. All rights reserved.

Before you run OpenLDAP (the Free Software implementation generally used on Linux, though some commercial implementations exist), you will need to install or verify the existence of several requisite libraries:

- The OpenSSL Transport Layer Security (TLS) services may be obtained from the OpenSSL Project (or via the install mechanisms of your Linux distribution).
- Kerberos Authentication Services are optionally supported, but this is very desirable. Either MIT Kerberos or Heimdal Kerberos will work.
- Simple Authentication and Security Layer may be installed as part of your base distro, but may be obtained as Cyrus SASL as well.
- Sleepycat Software Berkeley DB is recommended, though other DBM implementations are probably compatible.
- Posix threads and TCP wrappers are desirable, if not strictly required.

Given that you have satisfied these prerequisites, download the OpenLDAP library and perform the more-or-less usual dance:

Listing 7. The usual OpenLDAP install dance

```
        % ./configure
        % make depend
        % make
        % make test  # make sure things went OK
        % su root -c 'make install'
```

After basic installation, you need to configure the slapd configuration, usually at /usr/local/etc/openldap/slapd.conf. Setup should include your domain components:

Listing 8. Domain components to include with slapd.conf

```
        database bdb
        suffix "dc=eng,dc=uni,dc=edu,dc=eu"
        rootdn "cn=Manager,dc=eng,dc=uni,dc=edu,dc=eu"
        rootpw <secret>
        directory /usr/local/var/openldap-data
```

In order to find a value for `<secret>`, use the utility `slappasswd` and use this encrypted base64 encoded string for your "`<secret>`":

Listing 9. Discovering your "secret"

```
        % slappasswd
        New password: *******
        Re-enter new password: ********
        {SSHA}YzPqL5Jio2+17NFIy/pAz8pqS5Ko13fH
```

To get more information on permission, replication, and other options you can configure in slapd.conf, take a careful look at the detailed manpages.

Launching the slapd daemon is pretty much like staring any daemon; you can test to see it worked with `ldapsearch`:

© Copyright IBM Corporation 1994, 2006. All rights reserved.

Listing 10. Testing to see if slapd worked

```
        su root -c /usr/local/libexec/slapd
        ldapsearch -x -b '' -s base '(objectclass=*)' namingContexts
```

If all went well, you should see something like this:

Listing 11. Working slapd response

```
        dn:
        namingContexts: dc=eng,dc=uni,dc=edu,dc=eu
```

Adding data with an LDIF file

The data format used in LDAP is a binary format, but a ASCII serialization called LDAP Data Interchange Format (LDIF) is used for exporting and importing data to an LDAP database. Binary data in LDIF is represented as base64 encoding. OpenLDAP includes tools for exporting data from LDAP servers to LDIF (**ldapsearch**), importing data from LDIF to LDAP servers (**ldapadd**), and applying a set of changes described in LDIF to LDAP servers (**ldapmodify** and **ldapdelete**).

Moreover, LDIF is one of the formats for importing and exporting address book data for the Mozilla Application Suite and other user application-level tools. Even Microsoft Windows 2000 Server and Windows Server 2003 include an LDIF tool, **LDIFDE**, to transfer data to and from Active Directory.

To manually add information to an LDAP server, first create an LDIF file:

Listing 12. Creating the sample LDIF file, example.ldif

```
        dn: dc=example,dc=com
        objectclass: dcObject
        objectclass: organization
        o: Example Company
        dc: example

        dn: cn=Manager,dc=example,dc=com
        objectclass: organizationalRole
        cn: Manager
```

Then add it using `% ldapadd -x -D "cn=Manager,dc=example,dc=com" -W -f example.ldif`.

Obviously, you replace the example.com domain with one appropriate to your site. As a rule, LDAP domain hierarchies and names match those used by familiar DNS names. You will required to enter the `rootpw` value you specified in slapd.conf.

Querying LDAP databases

There is a tool, **slurpd** (Standalone LDAP Update Replication Daemon), that replicates a complete database of information; but for individual data values, you will either use a tool like ldapsearch or more likely LDAP support will be built into some application you run. The **slapcat** tool is also available for dumping an LDAP

© Copyright IBM Corporation 1994, 2006. All rights reserved.

database into LDIF. For example, many Mail User Agents (MUAs) can use LDAP to locate address and contact information.

Within applications, including ones you might program yourself using application or scripting languages, LDAP resources may be accessed with LDAP URLs. These have the form of `ldap://host:port/dn?attributes?scope?filter?extensions`.

Most of these fields are optional. The default hostname is localhost; the default port is 389. The default root distinguished name is the empty string. If you need authentication information, specify it in the extensions portion of the URL.

In addition to LDAP URLs, many LDAP servers and clients also support the non-standard but widely used LDAPS URLs. LDAPS URLs use SSL connections instead of plaintext connections and use a default port of 636: `ldaps://host:port/dn?attributes?scope?filter?extensions`.

Section 6. PAM authentication

When to use PAM

The first thing to keep in mind about Pluggable Authentication Modules (PAM) is that it is not an application or protocol itself. Rather, this is a collection of libraries that applications may have been compiled to utilize. If an application is PAM-enabled, the security policy of that application can be configured by a system administrator without modifying or upgrading the application itself. Many Linux tools, especially daemons and servers, are PAM-enabled.

A quick way to check if a given tool is *probably* PAM-enabled is to use `ldd` to check which libraries it uses. For example, I might wonder whether my login utility is PAM-enabled:

Listing 13. Is my login PAM-enabled?

```
% ldd /bin/login | grep libpam
libpam.so.0 => /lib/libpam.so.0 (0xb7fa8000)
libpam_misc.so.0 => /lib/libpam_misc.so.0 (0xb7fa5000)
```

The use of `libpam.so` and `libpam_misc.so` by `login` does not fully guarantee that the PAM facilities are actually being used and used correctly by this tool, but it is a good suggestion. Likewise, perhaps I wonder similarly about my Apache and FTP servers:

Listing 14. How about Apache and FTP servers?

```
% ldd /usr/sbin/apache2 | grep libpam
% ldd /bin/login | grep libpam
libpam.so.0 => /lib/libpam.so.0 (0xb7fa8000)
```

© Copyright IBM Corporation 1994, 2006. All rights reserved.

```
        libpam_misc.so.0 => /lib/libpam_misc.so.0 (0xb7fa5000)
```

So I know my particular Apache installation is not PAM-enabled (though versions are available that include this).

To check more thoroughly if PAM is fully working with a given tool, you can create a PAM configuration file for the program. For example, to test the login utility, you might create a file /etc/pam.d/login (but notice that it probably already exists on your system with a more meaningful setting, so do not delete the existing copy):

Listing 15. Checking login for PAM with etc/pam.d/login

```
        auth      required    pam_permit.so
        auth      required    pam_warn.so
```

Now running a properly PAM-enabled `login` will let anyone login, but it will log the action to the system log. If syslog shows an entry, PAM is enabled for this application. Readers will notice that this is about the worst configuration you could invent for `login` since it gives *anyone* shell access. Having noticed that, be warned that PAM should be configured with a certain caution.

PAM configuration

PAM works with two different types of configuration files. Most `libpam.so` libraries are compiled in the "use the better one if available, but settle for the worse one" mode. However, you might also have a PAM library compiled as "use the better one, but also check the worse one." Let me explain that.

The currently preferred way to configure PAM is with files in the directory /etc/pam.d/ that are named the same as the service whose security they describe. An older and less preferred style used a single file, /etc/pam.conf, to set security policy for all applications. From a maintainability point of view, the per-application configuration files are just easier to work with and may also be symlinked to "duplicate" a security policy from one application in another. Both configuration styles look basically the same. The single /etc/pam.conf file contains lines of the form:

Listing 16. Both configuration files contain

```
        <service>  <module-type>  <control-flag>  <module-path>  <args>
```

In per-application configuration files, the first field is omitted since it is already the same as the filename. In the older style, the test-only login configuration we saw would look like:

Listing 17. /etc/pam.conf

```
        login     auth      required    pam_permit.so
        login     auth      required    pam_warn.so
```

The `<module-type>` field may have one of four values:

- `auth` (authentication),

© Copyright IBM Corporation 1994, 2006. All rights reserved.

- `account` (non-authentication permissions based on system of user status),
- `session` (perform actions before/after service used), and
- `password` (update user authentication tokens).

The `<control-flag>` field is used to "stack" modules which allows you rather subtle control of when a method is required, whether it's performed at all, and when some other fallback is acceptable. Its options are

- `required`,
- `requisite`,
- `sufficient`,
- `optional`, and
- `include`.

I will discuss these in the next panel.

The `<module-path>` we have seen in our examples; it names a shared library either in the expected module location if no path is given or at an explicit location if it starts with a "/". For example, in Listing 17, you might have specified `/lib/security/pam_warn.so`. The `<args>` might be anything, depending on what a particular module needs to configure its operation, though a few generic arguments should be supported by most PAM modules. Notice that PAM modules are extensible. If someone writes a new PAM module, you can simply drop it into /lib/security and all your applications can use it once their configuration file is updated to indicate that.

A PAM permission example

To see how the `<control-flag>` works, let's develop an example that is moderately complex. First thing we should do is create a special *OTHER* service. If this exists and no PAM policy is defined for a service, OTHER's policy is used. A safe default might be like Listing 18:

Listing 18. /etc/pam.d/other

```
        auth      required    pam_warn.so
        auth      required    pam_deny.so
        @include safe-account
        @include safe-password
        @include safe-session
```

In this example, an attempt at authentication is first logged to syslog and is then denied. The `@include` statements just include contents from elsewhere, such as /etc/pam.d/safe-account and friends, where these "safe" definitions might contain similar warn-then-deny instructions for the other `<module-type>`'s.

© Copyright IBM Corporation 1994, 2006. All rights reserved.

Now let's configure access for our hypothetical classified-db application. Being rather concerned about access, for a user to use this application, he or she needs to provide either a matched retinal print or a fingerprint and also enter a password. The password, however, might be stored in either the local /etc/passwd and /etc/shadow configurations or available via one of two outside database servers.

None of the security modules I use in this example actually exist (to my knowledge), except `pam_unix.so` which is old-fashioned UNIX-style password access.

Listing 19. /etc/pam.d/classified-db

```
       auth      optional    pam_unix.so
       auth      optional    pam_database1.so    try_first_pass
       auth      optional    pam_database2.so    use_first_pass
       auth      requisite   pam_somepasswd.so
       auth      sufficient  pam_fingerprint.so  master=file1 7points
       auth      required    pam_retinaprint.so
```

The flow through this configuration is modestly complex. First we try password authentication by three `optional` module types. Since these are `optional`, failing one does not stop the authentication process nor satisfy it. The standard UNIX authentication password is tried first (the user is prompted to enter a password). After that, we check passwords in `database1`, but we first use the generic module argument `try_first_pass` to see if the UNIX password is the same one in the database; only if it is not do we prompt for an additional password. For `database2` however, we only try to authenticate using the UNIX password that the user entered (generic argument `use_first_pass`).

Having checked against some `optional` password systems, we use a hypothetical `pam_somepasswd.so` that needs to determine whether any of the earlier password checks succeeded (perhaps using a semaphore; but how that is done is left open for hypothetical modules). The point is that since this check is `requisite`, if it fails no further authentication is done and the failure is reported.

The final two authentication checks (if they are reached) are `sufficient`. That is, satisfying one of them returns an overall success status to the calling application. So first we try a fingerprint check using `pam_fingerprint.so` (notice some hypothetical arguments are passed to the module). Only if this fails -- maybe because of the absence of a fingerprint scanner as well as for a bad fingerprint -- is the retinal scan attempted. Likewise, if the retinal scan succeeds, that is `sufficient`. However, just to demonstrate all the flags, we actually use `required` here which means that even if retinal scanning succeeds, we would continue checking other methods (but no more exist in this example, so `sufficient` would do the same thing).

There is also a way of specifying more fine-tuned compound flags for the `<control-flag>` using bracketed `[value1=action1 ...]` sets, but the basic keywords usually suffice.

© Copyright IBM Corporation 1994, 2006. All rights reserved.

Resources

Learn

- Review the entire LPI exam prep tutorial series on developerWorks to learn Linux fundamentals and prepare for system administrator certification.
- At the LPIC Program, find task lists, sample questions, and detailed objectives for the three levels of the Linux Professional Institute's Linux system administration certification.
- Learn about DHCP (RFC 2131) by reading the formal specifications.
- Read "The Linux NIS(YP)/NYS/NIS+ HOWTO" to learn about NIS.
- The formal specification of LDAP is RFC 2251.
- *The Linux-PAM System Administrators' Guide* is a good starting point for more on the Pluggable Authentication Modules.
- *TCP/IP Network Administration, Third Edition* by Craig Hunt (O'Reilly, April 2002) is an excellent resource on Linux networking.
- This list of more than 700 Linux User Groups around the world can help you find local and distance study groups for LPI exams.
- For more in-depth information, the Linux Documentation Project has a variety of useful documents, especially its HOWTOs.
- In the developerWorks Linux zone, find more resources for Linux developers.
- Stay current with developerWorks technical events and Webcasts.

Get products and technologies

- Order the SEK for Linux, a two-DVD set containing the latest IBM trial software for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
- With IBM trial software, available for download directly from developerWorks, build your next development project on Linux.

Discuss

- Check out developerWorks blogs and get involved in the developerWorks community.

About the author

David Mertz
David Mertz thinks that artificial languages are perfectly natural, but natural languages seem a bit artificial. You can reach David at mertz@gnosis.cx; you can investigate all aspects of his life at his personal Web page. Check out his book, Text Processing in Python. Suggestions and recommendations on past or future columns

© Copyright IBM Corporation 1994, 2006. All rights reserved.

are welcome.

Trademarks

DB2, Lotus, Rational, Tivoli, and WebSphere are trademarks of IBM Corporation in the United States, other countries, or both.
Linux is a trademark of Linus Torvalds in the United States, other countries, or both.