

Documentation NSS / PAM /MYSQL

Jutta Horstmann

December 2004

Version 1.0-e

Last Revision: 06 Dec 2004

IT Department
Social Science Center Berlin
Wissenschaftszentrum Berlin für Sozialforschung (WZB)
Reichpietschufer 50
D-10785 Berlin
Germany

Table of Contents

| | |
|--|----|
| 1. Introduction..... | 4 |
| 2. NSS..... | 5 |
| 2.1 Basics..... | 5 |
| 2.2 Architecture..... | 5 |
| 2.3 Performance..... | 6 |
| 2.3.1 nsswitch.conf..... | 6 |
| 2.3.2 MySQL | 6 |
| 2.4 libnss_mysql..... | 6 |
| 2.4.1 Installation (v.1.2)..... | 6 |
| 2.4.2 Configuration..... | 6 |
| 2.4.3 Testing..... | 7 |
| 2.5 NSCD and libnss_mysql..... | 7 |
| 2.6 SuSE and libnss_mysql..... | 7 |
| 3. PAM..... | 8 |
| 3.1 Introduction..... | 8 |
| 3.2 Configuration..... | 9 |
| 3.2.1 Module Types | 9 |
| 3.2.2 Control Flag..... | 10 |
| 3.2.3 Module Path | 10 |
| 3.2.4 Arguments..... | 10 |
| 3.3 The Modules | 11 |
| 3.4 Communication Application <--> Module | 12 |
| 3.5 Control Flow, for example „su“ | 13 |
| 3.6 Projects..... | 14 |
| 4. pam-mysql(im) (Florian Verdet, B.Sc. thesis)..... | 15 |
| 4.1 Extensions to pam_mysql..... | 15 |
| 4.2 Code Organisation..... | 15 |
| 4.3 Installation..... | 16 |
| 4.4 Configuration..... | 16 |
| 4.4.1 Create three pam_mysql users..... | 16 |
| 4.4.2 Create Database..... | 17 |
| 4.4.3 Create Tables..... | 17 |
| 4.4.4 Table Permissions | 17 |
| 4.4.5 Create system users in MySQL tables..... | 18 |
| 4.4.6 PAM-Dateien editieren..... | 18 |
| 4.4.7 Configuration files..... | 19 |
| 4.5 useradd-mysql..... | 19 |
| 4.6 Debug Logging..... | 21 |
| 4.6.1 Test..... | 21 |
| 4.6.2 Debug Logging with Timestamp..... | 21 |
| 4.7 Session Logging..... | 22 |
| 4.8 sqlLog..... | 23 |
| 4.8.1 Configuration..... | 23 |
| 4.8.2 Test..... | 23 |
| 4.9 Account disabling..... | 24 |
| 4.9.1 Configuration..... | 24 |

| | |
|--|----|
| 4.9.2 Test | 24 |
| 4.9.3 Feedback Problem and Patch..... | 25 |
| 4.9.4 Further tests: ssh, ftp..... | 25 |
| 4.10 Password Expiry..... | 25 |
| 4.10.1 Disabled and Expired..... | 25 |
| 4.10.2 Expiry: Not available in pam_mysql(im)..... | 25 |
| 4.10.3 Problems with Expiry and pam_unix2/libnss-mysql..... | 26 |
| 4.10.3.1 lastchange..... | 26 |
| 4.10.3.2 Read „lastchange“ (NSS)..... | 27 |
| 4.10.3.3 Write „lastchange“ (passwd)..... | 27 |
| 4.11 Additional WHERE-Statements using the Config File | 28 |
| 4.11.1 SQL Statements in the Code..... | 28 |
| 4.11.2 Password Check using the WHERE Option..... | 29 |
| 5. Integration of pam_mysql and libnss_mysql..... | 30 |
| 5.1 Comparison of DB schemes..... | 30 |
| 5.2 libnss_mysql: Changed Configuration file..... | 31 |
| 5.3 Test..... | 31 |
| 5.3.1 Problem: passwd gives “free(): invalid pointer” | 32 |
| 6. Miscellaneous remaining questions..... | 33 |
| 6.1 passwd | 33 |
| 6.1.1 Which characters in pam_mysql's login, password?..... | 33 |
| 6.1.2 Passwords in libnss..... | 33 |
| 6.1.3 “Authentication token manipulation error”..... | 33 |
| 6.2 UNIX Sockets of MySQL configurable? - Yes!..... | 33 |
| 6.3 Do SQL statements get properly escaped? - Yes!..... | 34 |
| 6.4 May I change SQL statements using some configuration file? - Yes! .. | 34 |
| 6.5 Does PAM/NSS/MYSQL work with ACLs – Yes!..... | 34 |
| 7. MySQL | 35 |
| 7.1 Subqueries?..... | 35 |
| 7.2 Upgrade to MySQL 4.1..... | 35 |
| 7.3 Problem: “Too many arguments in make_scrambled_password”..... | 36 |
| 8. Excursus: Do we need PAM? Do we need NSS? | 37 |
| 9. Appendix..... | 39 |
| 9.1 Necessary READMEs..... | 39 |
| 9.2 Configuration Files..... | 39 |
| 9.3 Interesting Sources..... | 39 |
| 9.4 Code Changes..... | 40 |
| 9.5 Package pam_nss_mysql_JH..... | 40 |
| 10. References..... | 41 |
| 10.1 PAM/NSS..... | 41 |
| 10.2 NSS..... | 41 |
| 10.3 Project nss-mysql..... | 41 |
| 10.4 Project libnss-mysql..... | 41 |
| 10.5 PAM | 42 |
| 10.6 PAM/NSS/LDAP..... | 43 |
| 10.7 Project PAM-Mysql..... | 44 |
| 10.8 Project PAM-Mysql(im)..... | 44 |
| 10.9 Mysql..... | 44 |

1. Introduction

Linux provides a standardized interface as an extension to the local user administration: "Pluggable Authentication Modules" (PAM) combined with a "Name Service Switch" (NSS).

They offer mechanisms to identify users not only by the local user and group database (normally the `/etc/{passwd | shadow | group}` files) but also by external user administration systems (e.g. LDAP, MySQL...).

The "Name Service Switch" provides information on the user IDs the system has to know about, on the users' logins and the connection between groups and IDs.

PAM is responsible for authenticating the user. Often this is done by entering a password which will be compared with the one stored in the (local) user database.

As it is, the login process on a Linux system may work like this: The user enters her login name and password at the prompt. Then, by using the „Name Service Switch's“ methods, the system checks for the existence of this account, to which group it belongs etc. Afterwards the user is authenticated according to the PAM configuration.

This documentation describes using a MySQL database as backend to PAM and NSS. I will tend to all components in detail, present the installation on a step-by-step base, provide patches to bugs and solutions to problems connected with the installation and configuration.

The installation comprises the PAM module `pam_mysql(im)`, i.e. `pam_mysql` v. 0.5 in its "improved" CVS version by Florian Verdet, `libnss_mysql` of Ben Goodwin as of version 1.2 and MySQL 4.1.2.

NSS and PAM itself are default components of a SuSE 9.1 Professional distribution.

2. NSS

2.1 Basics

NSS provides the interface to connect to various user databases.

You may enter the descriptor(s) of the database(s) in the file `/etc/nsswitch.conf` (e.g. `mysql files`)

NSS will be used if an application calls one of these functions: `getpwnam`, `getpwuid`, `getspnam`, `getpwent`, `getspent`, `getgrnam`, `getgrgid`, `getgrent`, `memsbygid`, `gidsbymem`.

The Name Service Cache Daemon (`nscd`) increases NSS performance. Its configuration file is `/etc/nscd.conf`. As `nscd` caches results of calls to NSS, the frequency of database connections decreases. Access to password data (`shadow`) is not cached.

2.2 Architecture

NSS uses „services“, their code provided by „modules“. Services may be `files`, `mysql`, or `ldap`, i.e. the kind of backend(s) one wants to use for storing user data.

You tell NSS in `nsswitch.conf` which services you want to use by simply entering the descriptor, e.g. `mysql`. The services will be called according to the order used in the config file.

For using MySQL as a backend to NSS, there has to be a module (code) called `libnss_mysql` offering the shared library `libnss_mysql.so.2`.

Figure 1 provides an overview of the architecture.

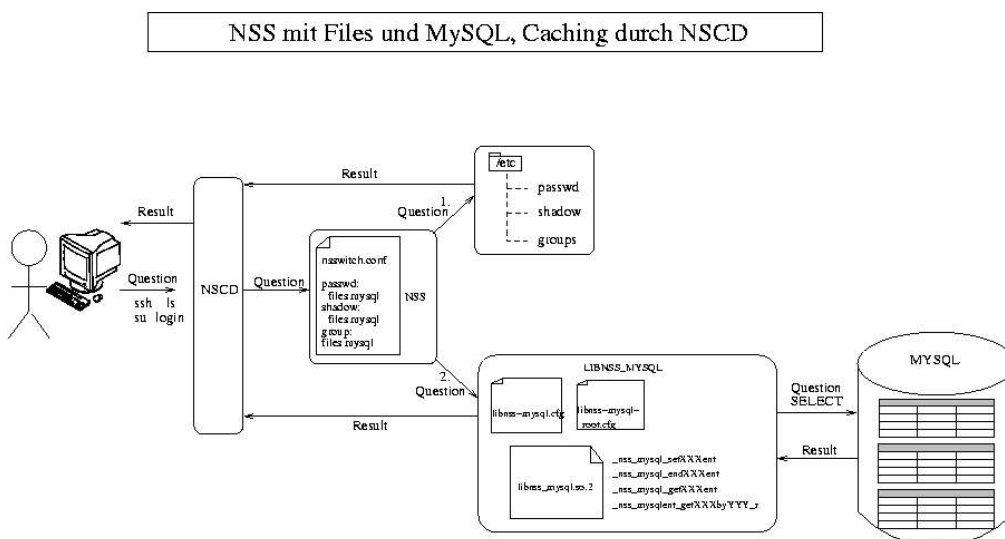


Fig. 1 NSS with "files" and "mysql" as well as NSCD

2.3 Performance

2.3.1 nsswitch.conf

The order of the services in `nsswitch.conf` should be chosen with regard to the installation's environment. Tests of the `libnss_mysql` maintainer show lookups to `/etc/passwd` being much faster compared to the MySQL database as long as it is used by less than 10 000 users. Above, MySQL is generally a better performer, only whole `passwd` scans (e.g. "finger") are even then faster on the file level.

2.3.2 MySQL

In case of high database activity, setting the MySQL server option `'thread_cache_size'` to something different from zero might be a solution (Connection threads stay open instead of opening a new thread for each connection).

2.4 libnss_mysql

This documentation's installation is using `libnss_mysql` (v.1.2) by Ben Goodwin (cinergi), see <http://libnss-mysql.sourceforge.net>

`libnss_mysql` is used by Sourceforge itself for the portal's user administration since July 2004 (but without `pam_mysql`).

Properties of `libnss_mysql`:

- It is no PAM solution (no updating or inserting of user information: no `passwd`, no `useradd`).
- It is able to handle `passwd`, `shadow` and `group` information. Not: mail aliases, hosts, `netgroups` etc (which other NSS services might provide).
- The library is thread-safe, but not multi-threaded. You may use NSCD for better performance.
- Offers TCP and UNIX sockets
- Database replication is possible. But no usage of several independent MySQL servers.
- encryption: tunneling or native SSL in MySQL (4.x)

2.4.1 Installation (v.1.2)

Simply `configure`, `make`, `make install`.

Creates `/lib/libnss_mysql.so.2.0.0` and two links to it.

2.4.2 Configuration

For a start, we will just use the default configuration files `/etc/libnss-mysql.cfg`, `/etc/libnss-mysql-root.cfg`.

Create the default database „auth“ by using the sample file `sample_database.sql`. The tables `grouplist`, `groups` and `users` will be created (more on the database and the merge with the `pam_mysql` database structure in chapter „Database Integration“).

Enter “mysql” in `/etc/nsswitch.conf`.

After installing and configuration, remember to do a:

COMPLETE SYSTEM SHUTDOWN AND REBOOT!!!

2.4.3 Testing

- works: `login`, `su`, `groups`, `ssh`, `ftp`, `id`, `getent`, `chown`, `ls`, `cd` (no parameters, `cd` to home), `touch`, `ls`
- does not work: `passwd`, `useradd`, `userdel` (changing user information is no NSS feature but belongs to PAM)

Note: `useradd` and `userdel` are checking the MySQL tables for entries of this user:

```
jh@eos:~> sudo useradd testuserin
useradd: User `testuserin' already exists.
```

But they cannot insert, delete or change entries (only “SELECT”).

After changing user data in MySQL (changed the login name manually): All tools expect the new username, which is also displayed by `ls -la` (uid maps to the new name).

The name of the user's home directory obviously does not get updated automatically.

In this test scenario NSCD did not cache the old data; but if it does: You may delete the cache by using `/usr/sbin/nscd -i resp.`

`--invalidate=TABLE` (z.B. `/usr/sbin/nscd -invalidate=passwd`).

2.5 NSCD and `libnss_mysql`

The Name Service Cache Daemon runs on SuSE 9.1 by default. You may switch on logging in it's config file `/etc/nscd.conf` (only errors are logged, no successful caching). `/usr/sbin/nscd -g` shows the daemon's current status.

A check using `lsof` proved NSCD accessing the `libnss_mysql` library.

2.6 SuSE and `libnss_mysql`

You may not use YaST's tools for configuring users and groups, as it does not access `libnss_mysql`. Which means that it does not use NSS at all but just defaults to the `/etc/{passwd | Shadow | group}` files.

3. PAM

3.1 Introduction

Before Pluggable Authentication Modules were used, UNIX Authentication worked like this: The user enters a password and the application which prompted her for it checks it against the one in `/etc/{passwd|shadow}`.

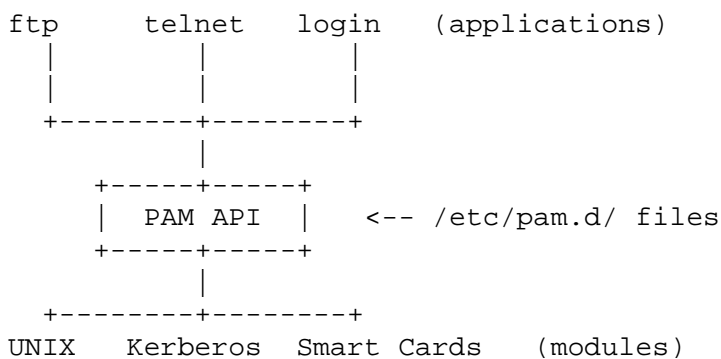
Lately many different authentication methods evolved (e.g. smartcards) which turned this way impractical: The application had to be rewritten for each method to be supported.

The solution to this problem is PAM, with applications only having to be „pamified“ once while PAM cares about integrating the different authentication methods.

„Pamified“ means the application using the PAM functions (e.g. `pam_start("login", user_name, &pam_conv, &pamh); pam_set_item(pamh, PAM_TTY, ttyn); pam_authenticate(pamh, 0); pam_end(pamh, PAM_SUCCESS);`) and linking against the PAM library (including `security/pam_modules.h`, `security/pam_appl.h`, `security/pam_misc.h`; - You may check the linking by “`ldd`”).

There is a PAM module to each authentication module, which the system administrator may add to her PAM installation.

The control flow: The application calls the PAM API and PAM, depending on its configuration, forwards the request to the matching module(s) and its answer back to the calling application.



PAM does not care about the user informations not belonging to authentication. Resolving an UID to a user name (e.g. needed by „`ls`“) is not done by PAM but by NSS.

PAM and NSS do not have to use the same database, even if it is more convenient for most scenarios for avoiding inconsistencies.

On the other hand there may be large systems or networks where most of the users do not have to be known to the system while still their authentication has to be cared for.

3.2 Configuration

Configuring PAM is done by file in the /etc/pam.d directory. These are on the test system:

```
chage, chsh, login, other, rpasswd, shadow, su, sudo,
vsftpd, xdm-np, xscreensaver, chfn, cups, passwd, ppp,
samba, sshd, useradd, xdm, xlock
```

Each pamified application may be configured regarding its authentication behavior by a file of the same name.

We use the following terminology: A pamified application is a service, its authentication method (e.g. password prompt using MySQL or LDAP as backend) is a module and the jobs of the modules are split in four module types (auth, account, password, session).

You may use multiple modules for each module type. Their order and prioritisation is configured by control flags.

The example shows a possible configuration for `su`:

```
##PAM-1.0
auth      sufficient      pam_rootok.so
auth      sufficient      pam_ldap.so
auth      sufficient      pam_unix2.so          nullok
auth      required       pam_denied.so

account   sufficient      pam_ldap.so
account   sufficient      pam_unix2.so
account   required       pam_denied.so

password  required          pam_pwcheck.so      nullok
password  sufficient      pam_ldap.so
password  sufficient      pam_unix2.so nullok use_first_pass
                                use_authok

password  required          pam_denied.so

#session  required          pam_homecheck.so
session   required       pam_unix2.so      debug #none or trace
```

The first column is the module type, the second the control flag, third comes the name of the module itself (the module path) and the fourth features optional arguments which may be passed to the module.

3.2.1 Module Types

By adding a module to a module type it will be used in executing jobs of this type. The jobs of the module type in detail:

- **auth**: Modules entered here are responsible for authenticating the user (e.g. by a password prompt) and allow or deny access to the service.
- **account**: Modules entered here care for restrictions to resources, things the user is allowed to do, e.g. use some features only at certain times of the day or e.g. root: is only allowed to log in at the CLI. Additionally: Account management, i.e. password expiry and account disabling.
- **password**: Modules entered here control the way of password changing

- **session:** Modules entered here determine what is to be done before/after the user is allowed to use the service (e.g. logging, mounting of file systems)

3.2.2 Control Flag

Modules of the same type will be processed by their order in the file. The control flag states what to happen after one of them was processed successfully or if there was an error.

- **required:** This module has to be processed successfully (but control flows back to the application only after all modules of this type got processed).
- **requisite:** like „required“, but control gets back to the application immediately after an error
- **sufficient:** if a module flagged „sufficient“ is processed successfully, no more modules of this type's stack will be used. If it returns an error then not the whole module type will fail, instead just the following module in the stack will be processed.
- **optional:** the module flagged likewise is of no concern to the success of this module type. It will be processed if before no „requisite“ module of this module type stack failed and no „sufficient“ module was successful.

3.2.3 Module Path

Here you enter the applied modules, i.e. the path to their location. The example above featured `pam_rootok.so`, `pam_ldap.so`, `pam_unix2.so`, `pam_deny.so`, `pam_pwcheck.so`, `pam_homecheck.so`.

As these files sit in the default path `/lib/security`, you only have to enter the file name instead of the whole path.

More on this in the next chapter „Modules“.

3.2.4 Arguments

Arguments are passed in calling the module. How to process them is up to the module. There are generic arguments, e.g. `debug` (debug logging to the system logfiles) or `use_first_pass` (the module will not ask again for a password but use one entered before).

3.3 The Modules

Beside the module `pam_mysql` which is the concern of this documentation, there are several other PAM modules which shall be presented briefly. They are either functional equivalents to `pam_mysql` (e.g. `pam_ldap`, `pam_unix`) or serve a special purpose (e.g. `pam_rootok`, `pam_pwcheck`).

pam_rootok

This module is used if the super user wants to use a service without having to enter a password. You know this behaviour from „su“: root may su to any user without having to know her password. It is only checked for the caller's UID being 0.

pam_unix2

This is SuSE's implementation of `pam_unix`, the module for traditional authentication via `/etc/{passwd|shadow|group}`.

But as this module is using the glibc NSS calls, `pam_unix2` is checking for users at the locations entered in `nsswitch.conf`, which may not only be „files“ but also „mysql“ or anything else likewise.

In the above example of `/etc/pam.d/su`, the argument „nullok“ is passed to the module. Accounts without password are deactivated by default. „nullok“ alters this behaviour by allowing the user to change the password of such an account.

pam_deny

The only purpose of this module is to deny access. It returns a `PAM_ACCT_EXPIRED` error.

Which means in the case of our „su“ example: The stack consists of several „sufficient“ modules and at last `pam_deny` (required).

So if none of the previous modules of this type processed successfully, the execution of this stack ends at `pam_deny` and returns the error to the calling application.

pam_pwcheck

The module checks a newly entered password's quality by calling „cracklib“ and executing some more tests. You may configure it by `/etc/login.defs`.

More popular modules:

(<http://www.kernel.org/pub/linux/libs/pam/Linux-PAM-html/pam-6.html>)

- `pam_access` (The access module)
- `pam_chroot` (Chroot)
- `pam_cracklib` (Cracklib pluggable password strength-checker)
- `pam_env` (Set/unset environment variables)
- `pam_filter` (The filter module)
- `pam_ftp` (Anonymous access module)
- `pam_group` (The group access module)
- `pam_issue` (Add issue file to user prompt)
- `pam_krb4` (The Kerberos 4 module)
- `pam_lastlog` (The last login module)
- `pam_limits` (The resource limits module)

- pam_listfile (The list-file module)
- pam_mail (The mail module)
- pam_mkhomedir (Create home directories on initial login)
- pam_motd (Output the motd file)
- pam_nologin (The no-login module)
- pam_permit (The promiscuous module)
- pam_pwdb (The Password-Database module)
- pam_radius (The RADIUS session module)
- pam_rhosts_auth (The rhosts module)
- pam_securetty (The securetty module)
- pam_tally (The login counter (tallying) module)
- pam_time (Time control)
- pam_unix (The Unix Password module)
- pam_userdb (The userdb module)
- pam_warn (Warning logger module)
- pam_wheel (The wheel module)

3.4 Communication Application <--> Module

The „pamified“ application links against the PAM libraries and employs the following PAM calls: pam_start, conv, pam_get_item, pam_set_item, pam_get_data, pam_set_data, pam_authenticate, pam_setcred, pam_acct_mgmt, pam_open_session, pam_close_session, pam_chauthtok, pam_end.

pam_start

- initiates authentication
- initialises PAM library
- reads PAM config file

conv

PAM modules do not communicate directly with the user but leave this to the application. The conv function is used to transfer data between user, application and PAM. This way the module may prompt the user for data and output errors or other text.

pam_get_item, pam_set_item

Reading and writing of status information concerning the PAM transaction. Independent modules may use those for sharing information (e.g. pam_mysql and pam_pwcheck sharing the password).

pam_get_data, pam_set_data

Access to module specific informations.

pam_authenticate

Identifies the user by her password or another authentication token.

pam_setcred

Sets the credentials for this process.

pam_acct_mgmt

Checks the account's and password's validity (account disabled, password expired, user is only authorised at certain time intervals...)

pam_open_session, pam_close_session

Informs the modules of a new session's start or a running session's end.

pam_chauthtok

Cares for changing the authentication token.

pam_end

Ends the PAM transactions, frees memory, cleans up.

3.5 Control Flow, for example „su“

The control flows from the user to the application, passing to NSS and then to PAM and back. Figure 2 shows this for su with libnss_mysql and pam_unix2.

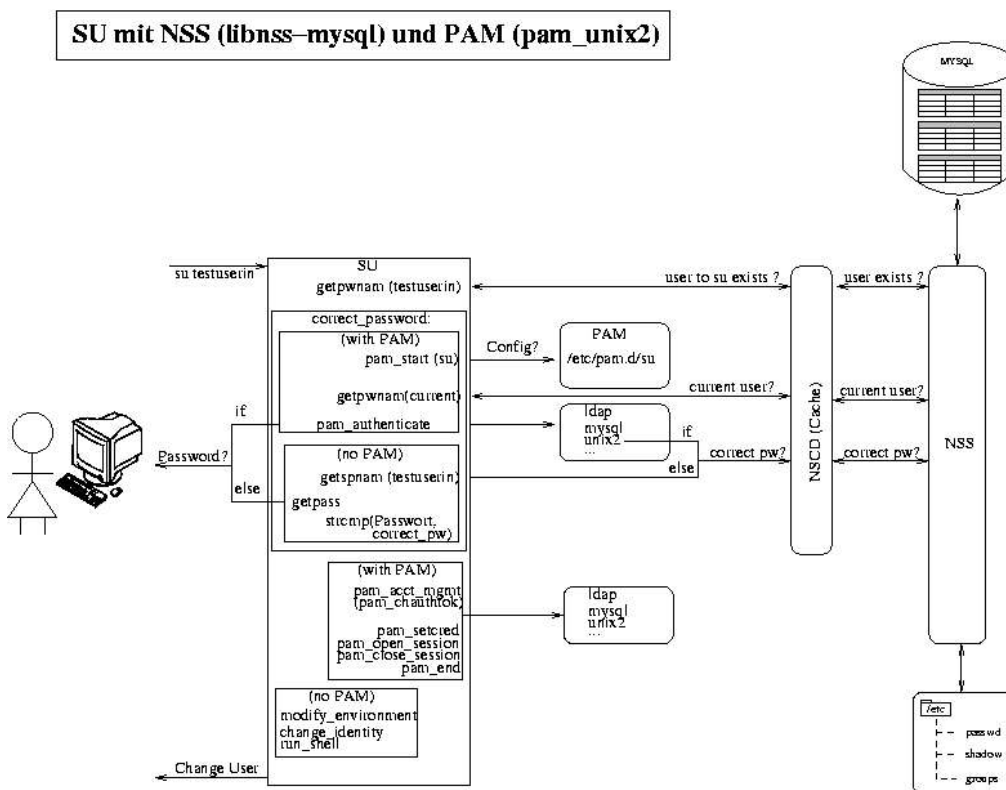


Fig. 2 Control flow between application (su), PAM, NSS

I will detail the flow once again:

1. User: su testuserin
2. su is linked against...
 - >ldd /bin/su
 - linux-gate.so.1 => (0xffffe000)
 - libcrypt.so.1 => /lib/libcrypt.so.1 (0x4002d000)
 - libpam.so.0 => /lib/libpam.so.0 (0x4005e000)
 - libpam_misc.so.0 => /lib/libpam_misc.so.0 (0x40066000)
 - libdl.so.2 => /lib/libdl.so.2 (0x4006a000)
 - libc.so.6 => /lib/tls/libc.so.6 (0x4006d000)
 - /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
3. Does user „testuserin“ exist?
 - su: pw = getpwnam (new_user);
 - getpwnam: calls NSS
4. NSS files are checked and MySQL Statement executed
5. get password of user „testuserin“
 - su: correct_password -> pam_start
 - (if PAM is available; else: getsppnam: get password via libnss_mysql from /etc/shadow or MySQL)
6. /etc/pam.d/su:
 - pam_authenticate -> auth: rootok (no), ldap (no), unix2 (yes)
7. pam_unix2: get password by NSS, user prompt, compare passwords
8. Process rest of PAM options (acct_mgmt, setcred, session)

3.6 Projects

I found three pam_mysql projects, of which two were inactive:

- pam_mysql_auth, Sourceforge, Joshua T. Hogle, 2002
- pam_mysql2, Sourceforge, Filipe Brandenburger, 2002

The last update of SF's pam_mysql dates from 2002 (v. 0.5). Florian Verdet, a swiss student, continued work on this project (then called pam_mysql(im)) in his bachelor's thesis 2003 and now acts co-maintainer of the SF project.

This documentation's installation will use pam_mysql(im), SF CVS version.

4. pam-mysql(im) (Florian Verdet, B.Sc. thesis)

4.1 Extensions to pam_mysql

- Account Management (Accounts disabling)
- Conversion scripts (to convert user information from `/etc/{passwd, shadow, group}` to MySQL. Written in Perl.)
- Session Logging to a MySQL Table
- “Housekeeping and code cleaning”
- “Code Splitting” (into smaller modules)
- Merge PAM-MySQL & NSS-MySQL config files (which was not applied to this installation, as `libnss_mysql` differs from `nss_mysql`)
- database connection via SSL (optional)

4.2 Code Organisation

`pam_mysql.c`, `pam_mysql.h`:

Database connection, interaction with PAM, calling of `pam_sm_authenticate`, `pam_sm_chauthtok`, `pam_sm_acct_mgmt`, `pam_sm_setcred`, `pam_sm_open_session` and `pam_sm_close_session`.

`pam_mysql_{auth,passwd,acct,sess}.c`:

Functions for the „real work“, e.g. processing of MySQL queries

`lib.c`, `lib.h`:

Debugging, Makro for malloc checks and structure for parsed values of the configuration files.

`mysql.c`, `mysql.h`:

Database connection

`options.c`, `options.h`:

Parsing of the options in `/etc/pam.d/<service>`

`parser.c`, `parser.h`:

Parsing of the options in the configuration files.

`pwlib.c`, `pwlib.h`:

Helper functions for password prompting and creation.

4.4.2 Create Database

```
$ create database pam_nss_mysql;
```

4.4.3 Create Tables

pam_mysql comes with SQL scripts for creating the required tables (mysql -u mysql -p pam_nss_mysql < scriptname.sql):

- auth (like /etc/passwd)
- shadow (like /etc/shadow)
- groups (like /etc/groups)
- groupusers (table for assigning groups to users)
- session_log (see chapter “Session Logging”)
- disabled (see chapter “Account Disabling”)

Furthermore there may be a table „sqllog“, but pam_mysql features no script for this one (see chapter „sqlLog“).

pam_mysqlim offers scripts for converting /etc/{passwd|shadow|groups} entries to MySQL tables (scriptname.conv). I did not test this feature, as it was irrelevant to my installation.

4.4.4 Table Permissions

```
mysql -u root -p mysql
$ GRANT SELECT,UPDATE ON pam_nss_mysql.auth TO pam_nss;
$ GRANT SELECT,UPDATE ON pam_nss_mysql.groups TO pam_nss;
$ GRANT SELECT,UPDATE ON pam_nss_mysql.groupusers TO
pam_nss;

$ GRANT SELECT,UPDATE ON pam_nss_mysql.shadow TO
pam_nss_shadow;

$ GRANT SELECT,UPDATE,DELETE,INSERT,CREATE,DROP,LOCK TABLES
  ON pam_nss_mysql.* TO pam_nss_admin;

$ FLUSH PRIVILEGES;
```

If you use session logging:

```
$ grant update, select insert on pam_nss_mysql.session_log
to pam_nss;
$ grant update, select insert on pam_nss_mysql.session_log
to pam_nss_shadow;
```

If you use account disabling by table:

```
$ grant select on pam_nss_mysql.disabled to pam_nss_shadow;
$ grant select on pam_nss_mysql.disabled to pam_nss;
```

4.4.5 Create system users in MySQL tables

You cannot use your standard „useradd“ with pam_mysql. Florian Verdet wrote „useradd_mysql“, which is part of pam_mysqlim (see chapter useradd_mysql).

Otherwise you have to add users manually in MySQL, like this:

```
mysql -u pam_nss_admin -p pam_nss_mysql

mysql> INSERT INTO auth VALUES('testuserin', 'x', '5002',
    '5000', 'Testa Testuserin', '/home/testuserin',
    '/bin/bash', NOW());

mysql> INSERT INTO shadow VALUES ('5002', 'testuserin',
    '$1$12345678$-----|-----|---', NOW(),
    '0', '99999', '7', '0', '-1');
```

Instead of a password an empty dummy will be inserted which you have to change afterwards with „passwd“.

The HOME directory has to be created manually and given the fitting user permissions.

4.4.6 PAM-Dateien editieren

To any services you want to use by authenticating via pam_mysql, you have to add this module (see below for details). For my installation these are /etc/pam.d/{login, passwd, sshd, su, sudo, vsftpd}.

As pam_mysql offers plenty of options, you may collect them into config files which you include in the service file.

The example of /etc/pam.d/su:

```
auth sufficient pam_rootok.so
auth sufficient pam_mysql.so config_file=/etc/security/pam_mysql.conf
                                config_shadow=/etc/security/pam_mysql_shadow.conf
auth sufficient pam_unix2.so nullok
auth required pam_denial.so

account requisite pam_mysql.so config_file=/etc/security/pam_mysql.conf
                                config_shadow=/etc/security/pam_mysql_shadow.conf
account sufficient pam_unix2.so
account required pam_denial.so

password required pam_pwcheck.so nullok
password sufficient pam_mysql.so config_file=/etc/security/pam_mysql.conf
                                config_shadow=/etc/security/pam_mysql_shadow.conf
password sufficient pam_unix2.so nullok use_first_pass use_authok
password required pam_denial.so

session sufficient pam_mysql.so config_file=/etc/security/pam_mysql.conf
                                config_file=/etc/security/pam_mysql_session.conf
session required pam_unix2.so debug
```

4.4.7 Configuration files

There is a plentitude of configuration options that you may collect into the files `pam_mysql.conf`, `pam_mysql_shadow.conf` and `pam_mysql_session.conf`. Please find the files of this installation in the appendix.

The `pam_mysqlim` README presents all parseable options. You may stick to the defaults most of the time, which means that in these cases you don't have to enter anything to the config files.

A short intro to the options:

- access data to the database (user, password, host, db)
- name and columns of the table matching `/etc/passwd`
- crypto method (note that there is an error in the README: `crypt=N`, Y or P does not work, you will have to use 0, 1, or 2!)
- using `sqlLog` or not, if yes: Name of the table and its columns
- using session logging or not, if yes: Name of the table and its columns
- additional „WHERE“ statements in user query (see also chapter „Additional WHERE statements“)
- account disabling by using a column in the „user“ table or by a proper table; name of this table and its columns

You enter options in the file in `name=value` format.

One should separate the options in two files, `pam_mysql.conf` and `pam_mysql_shadow.conf`. The latter will contain the password in plaintext of the DB user who is allowed to read from the „shadow“ table. Therefore it must be only root-readable (set permissions to 600, it will not work with 700!).

To separate the session options to its own file `pam_mysql_session.conf` is more concise in case of lots of options.

4.5 useradd-mysql

As `useradd` (`userdel` as well) cannot interact with MySQL, `pam_mysqlim` offers its own tool `useradd_mysql`.

You have to compile it separately as it is not integrated with the makefile of the module.

The following table compares the options and potentials of `useradd` and `useradd_mysql`.

| | <i>useradd (UNIX)</i> | <i>useradd_mysql (PAM_MYSQL)</i> |
|----------|---|---|
| Options | [-D binddn] [-P path] [-c comment] [-d homedir] [-e expire] [-f inactive] [-G group,...] [-g gid] [-m [-k skeldir]] [-o] [-p password] [-u uid] [-r] [-s shell] [--service service] [--help] [--usage] [-v] account | gecost (default <account>) homedir (default /home/<account>) = uid (automatically) highest set uid + 1 shell (default /bin/bash) account |
| changes | /etc/shadow, passwd, group | PAM tables: shadow, auth, groups |
| reads | /etc/shadow, passwd, group NSS tables: users, groups, grouplist | PAM tables: shadow, auth, groups |
| homedir | not created (as well when using the -d parameter) | not created |
| password | without -p: “!” in shadow, otherwise pw in shadow | has to be set afterwards by passwd (which uses pam_mysql) |

Test:

- Compile with make
- Copy the config file useradd_mysql.conf to /etc/security; you may stay with the defaults
- You get an error message if the user already exists in the pam_mysql tables. But only these are checked, there is no call to NSS and such no access to the /etc/passwd etc. files
- the arguments to useradd_mysql are entered without flags and such have to be entered in the correct order

4.6 Debug Logging

Debug logging is active per default and outputs to stderr (command line). Only if there exists a file /tmp/pam_mysql-debug.log, logging will go there.

This behaviour as well as the path mentioned above is hardcoded to lib.h and may only be changed there (you have to compile the module again afterwards).

```
#define DEBUG
/* If DEBUG is defined, debugging info is sent to following
file.
*/
#ifndef _PAM_LOGFILE
#define _PAM_LOGFILE "/tmp/pam_mysql-debug.log"
#endif
```

The log entries are formatted like „[<calling file>:<calling function>] action“. I patched this by also providing a timestamp (see below).

4.6.1 Test

- debug logging works for login, passwd, su, sudo, vsftpd, ssh
- ftp only gets debug logging, if pam_mysql logs to a file instead of stderr.

4.6.2 Debug Logging with Timestamp

Debugging is done by the PAM API's D() macro. It outputs like this:

```
[mysql.c:db_connect(28)] called.
[mysql.c:db_connect(59)] _nss_mysql_db_connect: using default
port
[mysql.c:db_connect(65)] hostname [port] set to: localhost
[3306].
[mysql.c:db_connect(86)] returning 0 .
[pam_mysql_acct.c:db_checkaccount(24)] called.
[pam_mysql_acct.c:db_checkaccount(118)] SELECT
login,date,reason FROM disabled WHERE login='nobody'
[pam_mysql_acct.c:db_checkaccount(127)] MySQL error: select
command denied to user 'pam_nss_shadow'@'localhost' for table
'disabled' - ev. table does not exist
[mysql.c:db_close(92)] called.
```

The part in brackets is written automatically, the rest comes as argument to D(). For a more efficient use of debug logging I added a timestamp to each D (“called.”);.

This had to be added to these files: lib.c, mysql.c, options.c, pam_mysql_acct.c, pam_mysql_auth.c, pam_mysql.c, pam_mysql_passwd.c, pam_mysql_sess.c, pwlib.c

Instead of “D (“called.”);” the macro LOGTIME(); in lib.h gets called:

```

#define LOGTIME()
do
{
    time_t now;
    struct tm *tm_now;
    char buff[81];
    now = time ( NULL );
    tm_now = localtime ( &now );
    strftime ( buff, (sizeof (buff)-1), "%b %d %H:%M:%S",
    tm_now );
    D(("called -- %s", buff));
} while(0)

```

I could not implement this as a function in lib.c, as this had always resulted in log entries like: “[lib.c:logtime(65)] called -- Aug 19 11:47:02”. By using a macro the log now looks like:

```
[mysql.c:db_connect(48)] called -- Aug 19 12:00:17
```

4.7 Session Logging

You may log into a MySQL table which user used a service, when it started and when it stopped. If the service is ssh or ftp, also the „remote host“ (its IP) will be logged.

The file sessionlog.sql may be used for creating the table with columns sessionID, service, login, remote_host, remote_user, tty, opened, closed.

(note: “remote_user” is not the user name on the remote system, but the UID of the calling application).

You have to „switch on“ session logging in any service that shall be monitored.

E.g. /etc/pam.d/su:

```

session sufficient      pam_mysql.so
config_file=/etc/security/pam_mysql.conf
config_file=/etc/security/pam_mysql_session.conf

```

The pam_mysql_session.conf file must at least contain the sessionlog=Y

option. Plus you have to grant these permissions to the sessionlog table:

```

grant update, select, insert on pam_nss_mysql.session_log
to pam_nss;
grant update, select insert on pam_nss_mysql.session_log to
pam_nss_shadow;

```

Tests:

- works: ftp, ssh, su, login
- sudo got no session option
- passwd does not support sessions
- ftp: you have to add to /etc/vsftpd.conf: “session_support=YES”. In the file coming with SuSE Pro 9.1, this parameter is not mentioned at all, but „NO“ per default (see man vsftpd.conf)

4.8 sqlLog

Before Florian Verdet implemented `session_log`, there was `sqlLog`. This one does not care about session logging, but monitors successful password authentications (even if the service could not be used afterwards).

4.8.1 Configuration

The table you will have to create for `sqlLog` looks like this:

```
CREATE TABLE `sqllog` (  
  `pid` int(11) NOT NULL, `message` varchar(30) character  
    set latin1 NOT NULL default '*unknown*',  
  `user` varchar(30) character set latin1 default '',  
  `host` varchar(30) character set latin1 default '',  
  `time` datetime NOT NULL default '0000-00-00 00:00:00'  
) TYPE=MyISAM;
```

```
grant select,insert,update on pam_nss_mysql.sqllog to  
pam_nss_shadow;  
grant select,insert,update on pam_nss_mysql.sqllog to  
pam_nss;
```

Activation: in `/etc/security/pam_mysql{,_shadow}.conf`:

```
sqllog=Y  
logtable=sqllog  
logmsgcolumn=message  
logpidcolumn=pid  
logusercolumn=user  
loghostcolumn=host  
logtimecolumn=time
```

4.8.2 Test

`sqlLog()` is called by `db_checkpasswd()` in `pam_mysql_auth.c`
--> Module type `auth` in the service files in `/etc/pam.d`

- Only actions of users get logged who are found in MySQL, not those in files
- `sqlLog` works for all services, only in `ssh` the host is not logged (it does in `ftp`)

4.9 Account disabling

pam_mysqlim offers a „disabled“ table and function for disabling accounts without deleting them (and you may enter why the account got disabled).

You may either use a column called `disabledcolumn` inside the „auth“ table (only Y and N flags, no date, no reason). Or its own table, which I used in my installation. It comes with the columns: `login`, `date`, `reason`, `until`, `until_action`. The columns `until` and `until_action` are only for your information – the account will not get automatically activated at the `until` date!

4.9.1 Configuration

You do not have to switch on account disabling in the config files: the `disabled` table gets always read. I did not use any other options, but stayed with the defaults (see also README).

You may disable accounts with entries to the „disabled“ table like this:

```
mysql> insert into disabled values ('newtest', NOW(),
'stepped on my cat', '2004-09-01', 'ring twice');
```

Remember to set the correct permissions to the table:

```
mysql> grant select on pam_nss_mysql.disabled to
pam_nss_shadow;
mysql> grant select on pam_nss_mysql.disabled to pam_nss;
```

4.9.2 Test

The user „newtest“ got disabled by the above entry. Afterwards she is not able to use any service which requires authentication. PAM writes to the syslog:

```
Aug 23 17:59:45 eos su: FAILED SU (to newtest) jh on /
dev/pts/2
Aug 23 17:59:54 eos su: pam_mysql: attempt to access
account for newtest, disabled since 2004-08-23 17:57:16,
reason: stepped on my cat.
```

The situation is tackled by `db_checkaccount` in `pam_mysql_acct.c`. If this function finds a matching entry in the „disabled“ table, the PAM module returns “PAM_ACCT_EXPIRED”.

Now the `/etc/pam.d/<servicefile>` stack is important!

First I had „su“ like this:

```
account sufficient pam_mysql.so
                    config_file=/etc/security/pam_mysql.conf
                    config_shadow=/etc/security/pam_mysql_shadow.conf
account sufficient pam_unix2.so
```

Which means: If `pam_mysql` fails, `pam_unix2` will take over which does not know anything about some „disabled“ table.

Therefore you have to change the stack to `account requisite pam_mysql.so` instead of “sufficient”. “Requisite” does also work if the user exists only in `/etc/passwd` and not in a MySQL table. `pam_unix2` will then take

over. Only if pam_mysql throws PAM_ACCT_EXPIRED the stack will end instead of switching to pam_unix2.

(It will not work if you use “required” instead of “requisite”: pam_unix2 would take over, like in “sufficient”).

This stacking has to be entered into the files of all services that shall provide discount enabling.

4.9.3 Feedback Problem and Patch

If the account is disabled, pam_mysql will write a corresponding message to syslog (see above). Only the user does not know anything about being disabled, she only gets “incorrect password”.

The problem is in the application: If PAM does not return PAM_SUCCESS, the applications will print their own error messages, as in this case „su“ said “incorrect password”.

You cannot change this without delving into the application's source. Anyway I patched pam_mysql_acct.c with this message:

```
fprintf (stderr, "Your account has been disabled. Please contact your system administrator.\n");
```

Now first pam_mysql will give the correct error message – and afterwards the application will throw “incorrect password”. Well...

4.9.4 Further tests: ssh, ftp

ftp and ssh both recognize the account as being disabled and deny login. ftp gives the (patched) error message while ssh doesn't. There you have to enter your password thrice and afterwards get „Permission denied“ without knowing why this has happened to you.

4.10 Password Expiry

4.10.1 Disabled and Expired

„Disabled“ and „Expired“ are essentially different concepts!

„Expire“ is one of shadow's features: The user or the admin may set an expiration date – then the user is prompted to change her password and may continue using her account afterwards.

„Disabled“ on the other hand means the admin rendered the account unusable without deleting it completely (see above) – the user can't do anything about it.

4.10.2 Expiry: Not available in pam_mysql(im)

pam_mysql v0.5 does not implement any account management functionality. The module always returns PAM_SUCCESS and the next module in the stack takes over (pam_unix2).

Florian Verdet then implemented account disabling, but there is still no password expiration functionality.

So pam_unix2 takes over (unless you disable this module there).

4.10.3 Problems with Expiry and pam_unix2/libnss-mysql

If pam_unix2 takes over, it will call NSS to get user data. Which means: At this point there is not only a lookup to „files“, but also once again to „mysql“ (because we are using libnss_mysql)

This is happening in pam_unix2 (unix_acct.c):

```
else if (sp->sp_lstchg > 0 && sp->sp_max >= 0 &&
        (now > sp->sp_lstchg + sp->sp_max))
[... ]
    __write_message (pamh, flags, PAM_TEXT_INFO,
                    "Your password has expired. Choose a
                    new password.");
    return PAM_NEW_AUTHTOK_REQD;
```

Which means: The password expires, if the current timestamp „now“ is larger than „lastchange“+„max“ (in days). „now“ is computed as `time (NULL) / SCALE`, with „SCALE“ being: `#define SCALE (24L*3600L.)`

4.10.3.1 lastchange

lastchange itself is defined in `struct spwd` (shadow.h) as a „long integer“:

```
struct spwd {
    char *sp_namp;           /* Login name */
    char *sp_pwdp;          /* Encrypted password */
    long sp_lstchg;         /* Date of last change */
    long sp_min;            /* Min #days between changes */
    long sp_max;            /* Max #days between changes */
    long sp_warn;          /* #days before pwd expires
                           to warn user to change it */
    long sp_inact;          /* #days after pwd expires
                           until account is disabled */
    long sp_expire;         /* #days since 1970-01-01
                           until account is disabled */
    unsigned long sp_flag; /* Reserved */
};
```

- And it is set in `/etc/shadow` as `lstchg = now / (60*60*24)`; with „now“ being seconds elapsed since 01.01.1970.

The „lastchange“ column of the „shadow“ table, on the other hand, is typed as `char(50)`. (There is also lastchange in the auth table, which comes in timestamp format, but this column does not get used anywhere.)

I modified the database like this:

```
mysql> alter table shadow modify lastchange timestamp not
null default current_timestamp;
mysql> alter table auth drop column;
```

4.10.3.2 Read „lastchange“ (NSS)

MySQL's „timestamp“ type formats the date as YYYY-MM-DD hh:mm:ss.

Therefore the following changes to libnss-mysql.cfg:

```
getspnam      SELECT  login,passwd,floor(UNIX_TIMESTAMP
(lastchange)/(24*3600)),min,max,warn,inactive,expire,0 FROM
shadow WHERE login='%s' LIMIT 1
getspent     SELECT login,passwd,floor(UNIX_TIMESTAMP
(lastchange/(24*3600)),min,max,warn,inactive,expire,0 FROM
shadow
```

Notes:

- (24*3600) to get days
- *unix_timestamp()*: If called with no argument, returns a Unix timestamp (seconds since '1970-01-01 00:00:00' GMT) as an unsigned integer. If *UNIX_TIMESTAMP()* is called with a date argument, it returns the value of the argument as seconds since '1970-01-01 00:00:00' GMT.

You could as well just enter the UNIX timestamp to the table (like in / etc/shadow), but this way the table is more human readable.

4.10.3.3 Write „lastchange“ (passwd)

If a user changes her password (using passwd), lastchange has to be updated to the current date.

But as pam_mysql does not use lastchange (does not do password expiry at all), pam_mysql's passwd will not write into it either.

The necessary patch (updatePasswd() in pam_mysql_passwd.c):

```
sql = malloc(sizeof(char) *
(strlen("update set = now(), = ' ' where = ' '") +
strlen(options.table) + strlen(options.passwdcolumn) +
strlen(escNew) + strlen(options.usercolumn) +
strlen(escUser)) + strlen("lastchange") + 2);
[...]
sprintf(sql, "UPDATE %s SET %s=NOW(), %s = '%s' WHERE %s='%s'
s", options.table, "lastchange", options.passwdcolumn,
escNew, options.usercolumn, escUser);
```

(Here one has to care for not only changing the SQL statement but also allocating the additional memory!)

4.11 Additional WHERE-Statements using the Config File

pam_mysqlim uses 7 SQL-Statements:

- get the (encrypted) password for checking against the entered one (SELECT)
- set new password, change „lastchange“ (UPDATE)
- if session gets closed: enter timestamp in „closed“ of the sessionLog table (UPDATE)
- entry in sessionLog (INSERT)
- entry in sqlLog (INSERT)
- get „disabled“ information from „auth“ table, if available (SELECT)
- get „disabled“ information from „disabled“ table, if available (SELECT)

In der Konfig-Datei gibt es die "where"-Option. Dort kann man mittels where=<irgendeine Bedingung> eine ebensolche formulieren. Abgefragt wird dieser Parameter nur beim Passwort-Select (s. oben).

4.11.1 SQL Statements in the Code

Get „disabled“ information from „auth“ table, if available:

```
pam_mysql_acct.c:
snprintf(sql, querysize,
"SELECT %s FROM %s WHERE %s='%s'",
options.disabledcolumn, options.table, options.usercolumn,
escapeUser);
```

Get „disabled“ information from „disabled“ table, if available:

```
pam_mysql_acct.c:
snprintf(sql, querysize,
"SELECT %s,%s,%s FROM %s WHERE %s='%s'",
options.acctmgmt_login_column, options.acctmgmt_since_column,
options.acctmgmt_reason_column, options.acctmgmt_table,
options.acctmgmt_login_column, escapeUser);
```

Set new password, change „lastchange“:

```
pam_mysql_passwd.c:
sprintf(sql, "UPDATE %s SET %s=NOW(), %s = '%s' WHERE %s='%s'",
options.table, "lastchange", options.passwdcolumn,
escNew, options.usercolumn, escUser);
```

If session gets closed: enter timestamp in „closed“ of the sessionLog table:

```
pam_mysql_sess.c:
snprintf(sql, querysize, "UPDATE %s SET %s = NOW() WHERE %s='%s'",
options.sessionlog_table, options.sessionlog_closed_column,
options.sessionlog_sessionid_column, sessId);
```

Entry in sessionLog:

```
pam_mysql_sess.c:
snprintf(sql, querysize, "INSERT INTO %s (%s,%s,%s,%s,%s,%s,%s)
VALUES ('%s','%s','%s','%s','%s',NOW(),NULL)",
options.sessionlog_table, options.sessionlog_service_column,
```

```
options.sessionlog_login_column,
options.sessionlog_ruser_column,
options.sessionlog_rhost_column, options.sessionlog_tty_column,
options.sessionlog_opened_column,
options.sessionlog_closed_column, service, escapeUser, ruser,
rhost, tty);
```

Entry in sqlLog:

```
mysql.c:
sprintf(sql, "insert into %s (%s, %s, %s, %s, %s) values('%s',
'%s', '%s', '%i', NOW())",
options.logtable, options.logmsgcolumn,
options.logusercolumn, options.loghostcolumn,
options.logpidcolumn, options.logtimecolumn, escapeMsg,
escapeUser, remote_host, pid);
```

Get the (encrypted) password for checking against the entered one:

```
pam_mysql_auth.c:
snprintf(sql, querysize,
"SELECT %s FROM %s WHERE %s='%s'",
options.passwdcolumn, options.table, options.usercolumn,
escapeUser);
```

For this last statement exits a „where“ option in the configuration file. You may there enter „where_clause = <some condition>“, which will be appended to the above statement.

4.11.2 Password Check using the WHERE Option

Note that the option in the config file has to be entered as “where_clause”, not just “where”, as the pam_mysql documentation mentions.

Example:

```
where_clause = uid=15001 AND min=0
```

The statement is processed as follows (pam_mysql_auth.c):

```
if (strlen(options.where) > 0){
    strncat(sql, " AND (", (querysize - strlen(sql)));
    strncat(sql, options.where, (querysize - strlen(sql)));
    strncat(sql, ")", (querysize - strlen(sql)));
}
```

Note: If pam_mysql returns an error, pam_unix2 and libnss_mysql will take over authentication. Therefore you have to enter the same WHERE statement in / etc/libnss-mysql.cfg:

```
getspnam          SELECT  login,passwd,floor(UNIX_TIMESTAMP
(lastchange)/(24*3600)),min,max,warn,inactive,expire,0 FROM
shadow WHERE login='%s'and (uid=15001 or min=0) LIMIT 1
```

And last but not least, please note that the WHERE clause does not apply to users who only show up in the UNIX files. Control-Flow: pam_mysql -> pam_unix2 -> libnss_mysql -> nss-”files”.

5. Integration of pam_mysql and libnss_mysql

pam_mysqlim of Florian Verdet comes with a PAM/NSS integration regarding shared tables in a shared database as well as a common configuration file. The drawback: It is based on the older nss_mysql module.

The following steps are necessary to integrate libnss_mysql with pam_mysql (just using both together will work as well):

- integrate databases
- adapt libnss_mysql SQL statements to the new database

optional (not in my installation):

- integrate configuration files

5.1 Comparison of DB schemes

| <i>libnss_mysql</i> | <i>pam_mysqlim</i> | |
|--|---|--|
| grouplist: rowid (int 11) gid (int 11) username (char 16) | groupusers: gid (int 11) uid (int 11) | |
| groups: name (varchar 16) password (varchar 34) gid (int 11) | groups: gname (char(35)) passwd (char(35)) gid (int 11) | |
| users: username (varchar 16) uid (int 11) gid (int 11) gecos (varchar 128) homedir (varchar 255) shell (varchar 64) password (varchar 34) lstchg (bigint 20) min (bigint 20) max (bigint 20) warn (bigint 20) inact (bigint 20) expire (bigint 20) flag (bigint 20) | shadow: login (char(35)) uid (int 11) passwd (char(35)) lastchange (char 50) min (int(11)) max (int(11)) warn (int(11)) inactive (int(11)) expire (int(11)) | auth: login (char(35)) uid (int 11) gid (int 11) gecos (char(63)) home (char(35)) shell (char(35)) passwd (char(35)) lastchange (timestamp14) |
| | session_log: `sessionID` int(11) `service` varchar(30) `login` varchar(30) `remote_host` varchar(30) `remote_user` varchar(30) `tty` varchar(10) `opened` datetime `closed` datetime | |
| | disabled: `login` varchar(30) `date` datetime `reason` varchar(127) `until` date `until_action` varchar(127) | |

My installation uses the pam_mysql database with libnss_mysql accessing it as well. This necessitates changes to libnss-mysql.cfg.

5.2 libnss_mysql: Changed Configuration file

/etc/libnss-mysql.cfg

```
[queries]
getpwnam    SELECT login,'x',uid,gid,gecos,home,shell FROM auth WHERE
            login='%s' LIMIT 1

getpwuid    SELECT login,'x',uid,gid,gecos,home,shell FROM auth WHERE
            uid='%u' LIMIT 1

getspnam    SELECT login,passwd,floor(UNIX_TIMESTAMP(lastchange)/
            (24*3600)),min,max,warn,inactive,expire,0 FROM shadow
            WHERE login='%s' LIMIT 1

getpwent    SELECT login,'x',uid,gid,gecos,home,shell FROM auth

getspent    SELECT login,passwd,floor(UNIX_TIMESTAMP(lastchange)/
            (24*3600)),min,max,warn,inactive,expire,0 FROM shadow

getgrnam    SELECT gname, passwd, gid FROM groups WHERE gname='%s'
            LIMIT 1

getgrgid    SELECT gname,passwd,gid FROM groups WHERE gid='%u' LIMIT 1

getgrent    SELECT gname,passwd,gid FROM groups

memsbygid   SELECT login FROM auth WHERE uid = any (SELECT uid FROM
            groupusers WHERE gid='%s')

gidsbymem   SELECT gid FROM groupusers where uid=(SELECT uid FROM auth
            WHERE login='%s')

[server]
host        localhost
database    pam_nss_mysql
username    pam_nss
password    pamPass
timeout     3
compress    0
```

/etc/libnss-mysql-root.cfg

```
[server]
username    nss-root
password    rootpass
```

5.3 Test

- getent passwd, getent group: works
- getent shadow (as root): only works if flag 0 gets selected as well (w/r/t getspnam and getspent). „struct spwd“ in shadow.h:
unsigned long sp_flag; /* not used */
sp_flag is as well not used in the libnss code, but obviously there has to be

- entered something
- su: User pam_nss_shadow has to have access to the other tables, not only „shadow“
- passwd (s. below)
- groups , id, chown, touch, ls, cd: works

5.3.1 Problem: passwd gives “free(): invalid pointer”

This problem occurred:

```
sudo passwd nochntest
Changing password for nochntest.
(New) Password:
Retype (New) Password:
free(): invalid pointer 0x404a53a0!
```

The password got set nevertheless, but the error message was irritating. The problem started after the DB integration.

It is caused by the following code in updatePasswd (pam_mysql_passwd.c):

```
char *tmp = NULL;
tmp = crypt(newpass, salt);
encNew = malloc(sizeof(char) * (strlen(tmp) + 2));
strncpy(encNew, tmp, strlen(tmp) + 1);
free(tmp);
```

The problem being the „free“ing of „tmp“ which was not mallocated before.

Signature of crypt (crypt.h):

```
extern char *crypt (__const char *__key, __const char
*__salt) __THROW;
```

man crypt:

A pointer to the encrypted password is returned. On error, NULL is returned.

Hier steht nichts, ob der Pointer statisch oder dynamisch ist (im Gegensatz zur manpage der BSD-Implementierung: statisch). Da aber in der Fehlermeldung immer wieder die selbe Speicheradresse angegeben wird: wohl statisch.

Solution: comment out „free (tmp)“

crypt() is also used in pam_mysql_auth.c, but there the return value is used directly without being saved to a variable. Therefore this code does not have to be patched.

6. Miscellaneous remaining questions

6.1 passwd

6.1.1 Which characters in pam_mysql's login, password?

Any characters, any length (no limit in the code, but the database field only permits 35 characters).

6.1.2 Passwords in libnss

libnss does not work with passwords containing special characters or numbers!
--> Does not matter for my installation, as authentication is handled by PAM.

6.1.3 "Authentication token manipulation error"

If you enter the new password differently the first and second time, you get an "Authentication token manipulation error" (PAM_AUTHTOK_ERR) instead of the correct error message. The code:

```
if( newpass == NULL || strcmp(sTmp, newpass) ) {
    D(("error getting password after second try"));
    free(sTmp);
    db_close();
    return PAM_AUTHTOK_ERR;
}
```

The fitting error message is produced by cracklib, called by pam_pwcheck (in the module stack by SuSE's default). If you keep it in the stack, you will get the correct message.

```
Linux-PAM-0.77/modules/pam_cracklib/pam_cracklib.c:#define
MISTYPED_PASS "Sorry, passwords do not match"
Linux-PAM-0.77/modules/pam_unix/support.h:#define
MISTYPED_PASS "Sorry, passwords do not match"
```

6.2 UNIX Sockets of MySQL configurable? - Yes!

NSS:

```
In /etc/libnss_mysql.cfg:
host          localhost
#socket       /var/lib/mysql/mysql.sock
#port         3306
```

Socket gets used if you enter „localhost“ for host while port will be used if you enter some IP.

If both are commented out (as by default), the MySQL defaults will be used. You may configure these in my.cnf.

PAM:

In /etc/security/pam_mysql.conf:

host(Default: localhost)

Machine that is running the sql server

Syntax: <host>, <host>:<port>, inet:<host>:<port> ,

inet:<host>, unix:</path/to/mysql.sock>

6.3 Do SQL statements get properly escaped? - Yes!

NSS:

libnss_mysql uses mysql_real_escape_string().

From the MySQL documentation:

The string in from is encoded to an escaped SQL string, taking into account the current character set of the connection. The result is placed in to and a terminating null byte is appended. Characters encoded are NUL (ASCII 0), '\n', '\r', '\', '\", '\'', and Control-Z.

PAM:

pam_mysql is too using mysql_real_escape_string(), starting with version 0.4.7.

From the Changelog:

Version 0.4.7 7/9/2000 <delancie@users.sourceforge.net>

URGENT! This release fixes a SERIOUS security hole in the authentication mechanism and is one I am deeply to ashamed to admit was there, but must.

The SQL statement was never being escaped, so your users can effectively 'break out' of the query, add their own SQL and get authentication.

Whichever version of PAM-MYSQL you are running, you should upgrade immediately to fix this problem. ANYONE can get authenticated on your system without needing to know the password of the user they are trying to be authenticated as. This means root too. And it is easy... Specify the username as root. Specify the password as; ' and user='SomeKnownUser' and whammo, you have root access to the machine because PAM authorised you. UPGRADE NOW!

6.4 May I change SQL statements using some configuration file? - Yes!

NSS: Yes: /etc/libnss_mysql.conf

PAM: see chapter “Additional WHERE statements using the Config File”

6.5 Does PAM/NSS/MYSQL work with ACLs – Yes!

You may use setfacl, getfacl on users from MySQL as well as from /etc/passwd.

The „files“ user may grant extended permissions to „mysql“ users and vice versa, the permissions are recognized and accepted from both sides.

7. MySQL

I will not have to comment much on MySQL, as the PAM/NSS installation has no impact on the database configurations.

Only did my installation use a subquery to allow `libnss_mysql` access to `pam_mysql`'s database, which is only supported from MySQL 4.1 onward. Some words to this in the following.

7.1 Subqueries?

`libnss_mysql` uses the following statement to get a user's group membership and the members of a group:

```
memsbygid    SELECT username FROM grouplist WHERE gid='%u'  
gidsbymem    SELECT gid FROM grouplist where username='%s'
```

On the other hand, the shared database `pam_nss_mysql` uses an own relationship table „`groupusers`“. I used subqueries to let `libnss_mysql` access this one:

```
memsbygid    SELECT login FROM auth WHERE uid= any(SELECT  
uid FROM groupusers WHERE gid='%u')  
gidsbymem    SELECT gid FROM groupusers where uid=(SELECT  
uid FROM auth WHERE login='%s')
```

Note: The sub-select in `memsbygid` may return multiple rows. Therefore you have to use this syntax: “where uid **in** (...)” or “where uid = **any** (...)”.

Subqueries are only supported from MySQL 4.1 onwards. Therefore I used this version for my installation (which led to some problems, see below).

Instead of subqueries you may as well use the following statements which are also supported by older MySQL versions:

```
memsbygid    SELECT auth.login FROM auth, groupusers WHERE  
((groupusers.gid='%u') AND (auth.uid=groupusers.uid))  
gidsbymem    SELECT groupusers.gid FROM groupusers, auth WHERE  
((auth.login='%s') AND (groupusers.uid=auth.uid))
```

7.2 Upgrade to MySQL 4.1

If you created your installation's databases with an older MySQL version and are later upgrading to 4.1, you will have to execute “`mysql_fix_privilege_tables`” (<http://dev.mysql.com/doc/mysql/en/Upgrading-from-4.0.html>).

Also note that `pam_mysql` needs the `mysql-devel` package!

7.3 Problem: “Too many arguments in make_scrambled_password”

After installing MySQL 4.1 I got the error message “Too many arguments in make_scrambled_password”. It is caused by this code in pam_mysql_auth.c:

```
#ifdef HAVE_MYSQL_41
make_scrambled_password(encryptedPass, passwd, 1, NULL);
#else
make_scrambled_password(encryptedPass, passwd);
#endif
```

Which is based on the assumption that MySQL would change the signature of make_scrambled_password() in 4.1.0.

I was using version 4.1.3 and this one still only wanted two arguments:

```
mysql_com.h:
void make_scrambled_password(char *to, const char *password);
```

Solution: I patched the code in pam_mysql_auth.c and pam_mysql_passwd.c to use the two arguments function.

8. Excursus: Do we need PAM? Do we need NSS?

In the course of this project the question surfaced once and again, if an installation really needs PAM as well as NSS.

And more: If PAM and NSS, do we need `pam_mysql`? Can't we get by with using `libnss_mysql` combined with `pam_unix2`?

I detailed the different scenarios for using NSS and PAM in the first three chapters. A short reminder:

NSS is used to extract user information from any source, when some application is in need of this data: login, UID, password, group name, GID, group members, group membership of some user, or all of these informations belonging to a user or a group together. To make a long story short: NSS is called when an application uses one of these functions:

`getpwnam`, `getpwuid`, `getspnam`, `getpwent`, `getspent`, `getgrnam`, `getgrgid`, `getgrent`, `memsbygid`, `gidsbymem`.

PAM is not able to fulfill these tasks.

If an application needs the user's authentication, it checks for a PAM instance on the system. If none is to be found, the application will care for authentication itself, having its own dialogues and functions, part of which will use the NSS. For example „su“ will prompt the user for her password, lets NSS get the saved one from the user database or file, compares it to the entered password and then accepts or denies the user. „su“ does not offer any more options for authentication, only via password.

If PAM is installed, the application will use the available PAM function for authentication instead of its own built-ins.

So this is the pivotal advantage of PAM: You may use any authentication method (file, database, biometrics, smartcard...) without having to adapt the application's code to each.

But do we need PAM too, if a password check would suffice? These are PAM's additional features:

- Password Expiry
- Account Disabling
- Session Management (what has to be done before/after the user is granted access to the service)
- Restrictions to resources
- Specify conditions for changing or setting of passwords
- ... any further possibilities depending on available modules
- ... adapting of these options to each service (application) differently, if needed

Conclusion: You will need NSS in any case, while you may live without PAM, which will ease your life nevertheless.

Last but not least remains the question on using `pam_mysql` instead of `pam_unix2`. The background: Using `libnss_mysql` and PAM without `pam_mysql`,

the control flow will work like this:

An application wants to get authentication. PAM kicks in and the stack tells it to use `pam_unix2`. This one was written to authenticate using `/etc/{passwd|shadow|groups}`.

But as it calls NSS, `libnss_mysql` will try to fetch the data also from the MySQL database. That is, you get PAM functionality with user management in MySQL without having to install `pam_mysql`!

Add to this `pam_unix2` supporting features that `pam_mysql` does not, like password expiry.

So whether it is necessary to use `pam_mysql` at all depends on your interest in using the features of a MySQL database in combination with PAM. If it's only about querying user information from the database, you may be satisfied with using `libnss_mysql` and `pam_unix2`. But there will be no writing to the database (as in setting and changing passwords for example), as `pam_unix2` would write to `/etc/shadow`!

As well there would be no additional features of `pam_mysql` like managing disabled accounts in its own table or logging to tables (`Sessionlog`, `sqlLog`). These would not be used by `pam_unix2`.

And last but not least one could imagine plenty of other features not yet implemented in `pam_mysql`, which could use MySQL's possibilities even more.

For example as part of the session management: Querying of script names from a table which should be executed when starting a session.

Conclusion: `libnss_mysql` + `pam_unix2` will suffice in a small setting that does not depend on writing of passwords (or other informations) to the database.

In any other case you should use `pam_mysql`. (But note that your stack should combine `pam_mysql` with `pam_unix2`, first for managing password expiry, second for user data not found in the database, e.g. system users).

9. Appendix

9.1 Necessary READMEs

- libnss_mysql/README.txt
- libnss_mysql/DEBUGGING
- pam_mysqlim/Readme
- pam_mysqlim/Mini-HowTo
- pam_mysqlim/useradd_mysql/Readme

9.2 Configuration Files

```
/etc/pam.d/  
    login  
    passwd  
    sshd  
    su  
    sudo  
    vsftpd  
/etc/security/  
    pam_mysql.conf  
    pam_mysql_session.conf  
    pam_mysql_shadow.conf  
/etc/  
    libnss-mysql.cfg  
    libnss-mysql-root.cfg
```

9.3 Interesting Sources

- **Header** in /usr/include, /usr/include/linux
- **coreutils**: basename cat chgrp chmod chown chroot cksum comm cp csplit cut date dd df dir dircolors dirname du echo env expand expr factor false fmt fold install groups head id join kill link ln logname ls md5sum mkdir mkfifo mknod mv nice nl nohup od paste pathchk pinky pr printenv printf ptx pwd readlink rm rmdir seq sha1sum shred sleep sort split stat stty su sum sync tac tail tee test touch tr true tsort tty uname unexpand uniq unlink uptime users vdir wc who whoami yes
- **pwdutils**: shadow, pam_rpasswd.so, chfn, chsh, newgrp, passwd, groupadd, groupdel, groupmod, useradd, userdel, usermod
- **mysql-4.1.3-beta**
- **Linux-PAM-0.77**
- **pam_unix2-1.15**

9.4 Code Changes

| | |
|--------------------|--|
| lib.c | time stamp in Debug Log |
| lib.h | #define LOGTIME() |
| mysql.c | time stamp in Debug Log |
| options.c | time stamp in Debug Log |
| pam_mysql_acct.c | time stamp in Debug Log, error messages in Disabled |
| pam_mysql_auth.c | time stamp in Debug-Log, make_scrambled_password |
| pam_mysql.c | time stamp in Debug-Log, make_scrambled_password, free(tmp) commented out, also update "lastchange" |
| pam_mysql_passwd.c | time stamp in Debug Log, |
| pam_mysql_sess.c | time stamp in Debug Log |
| pwlib.c | time stamp in Debug Log |

9.5 Package pam_nss_mysql_JH

```
package_jh/  
  libnss_mysql/  
    libnss-mysql-1.2.tar.gz  
    libnss-mysql.cfg  
    libnss-mysql-root.cfg  
  pam_mysqlim_jh  
    orig/          (the original files from CVS)  
    code_jh/       (code including patches JH)  
    pam_config/  
      etc_pamd/  
        login passwd sshd su sudo vsftpd  
      etc_security/  
        pam_mysql.conf  
        pam_mysql_session.conf  
        pam_mysql_shadow.conf  
    db_user.sql  
    README_JH
```

Also used:

MySQL-devel-4.1.3-0.i386.rpm

MySQL-client-4.1.3-0.i386.rpm

MySQL-server-4.1.3-0.i386.rpm

10. References

10.1 PAM/NSS

Treffen der Generationen

http://www.rz.uni-augsburg.de/connect/2000-02/pam_nss/

pam/nss/ldap

<http://www.faveve.uni-stuttgart.de/it/auth/ldap-client.php>

[Proposal] Forget PAM, stick with NSS

<http://groups.google.com/groups?hl=en&lr=&ie=UTF-8&threadm=19990802084306.A9221%40lappy.djj.state.va.us&rnum=4&prev=/groups%3Fq%3Dpam%2Bnss%2Bcommon%26hl%3Den%26lr%3D%26ie%3DUTF-8%26selm%3D19990802084306.A9221%2540lappy.djj.state.va.us%26rnum%3D4>

NSS and PAM

<http://groups.google.com/groups?hl=en&lr=&ie=UTF-8&threadm=xzpllpwceay.fsf%40dwp.des.no&rnum=11&prev=/groups%3Fq%3Dpam%2Bnss%2Bcommon%26start%3D10%26hl%3Den%26lr%3D%26ie%3DUTF-8%26selm%3Dxzpllpwceay.fsf%2540dwp.des.no%26rnum%3D11>

10.2 NSS

gclib - NSS

http://www.gnu.org/software/libc/manual/html_node/Name-Service-Switch.html

NameServiceSwitch - Naked Ape Wiki

<http://nakedape.cc/wiki/ApplicationNotes/NameServiceSwitch>

UNIX man pages : nss (4)

<http://bama.ua.edu/cgi-bin/man-cgi?nss+4>

man 5 nsswitch.conf

http://dpobel.free.fr/man/html/affiche_man.php?id=2004

10.3 Project nss-mysql

NSS project page on Savannah

<http://savannah.nongnu.org/projects/nss-mysql>

NSS-MySQL homepage

<http://www.nongnu.org/nss-mysql/>

freshmeat.net: Project details for NSS-MySQL

<http://freshmeat.net/projects/nss-mysql/>

10.4 Project libnss-mysql

libnss-mysql (SF)

<http://libnss-mysql.sourceforge.net/>

SourceForge.net: Project Info - NSS MySQL Library

<http://sourceforge.net/projects/libnss-mysql/>
libnss-mysql: FAQ
<http://libnss-mysql.sourceforge.net/libnss-mysql/FAQ>

10.5 PAM

LDAP Authentication HOWTO (PAM)
<http://ldots.org/ldap/>

Linux-PAM modules etc. page
<http://www.kernel.org/pub/linux/libs/pam/modules.html>

A Linux-PAM page
<http://www.kernel.org/pub/linux/libs/pam/>

SourceForge.net: Pluggable Auth Modules
<http://sourceforge.net/projects/pam>

PAM FAQ
<http://www.kernel.org/pub/linux/libs/pam/FAQ>

PAM System Administrators' Guide
<http://www.kernel.org/pub/linux/libs/pam/Linux-PAM-html/pam.html>

PAM Module Writers' Guide
http://www.kernel.org/pub/linux/libs/pam/Linux-PAM-html/pam_modules.html

PAM Application Developers' Guide
http://www.kernel.org/pub/linux/libs/pam/Linux-PAM-html/pam_appl.html

Re: PAM modules that reset the user name -
<http://archives.neohapsis.com/archives/pam-list/2001-04/0124.html>

PAM Module Writers' Guide: Programming notes
http://www.kernel.org/pub/linux/libs/pam/Linux-PAM-html/pam_modules-5.html

Linux: PAM (Pluggable Authentication Modules) - Authentication
<http://www.linuxforum.com/linux-user-authentication/x101.html>

Chapter 25: PAM - Pluggable Authentication Modules
<http://www.bb-zone.com/SLGFG/chapter25.html>

RFC PAM
<http://www.opengroup.org/tech/rfc/rfc86.0.html>

Introduction to PAM
<http://www.phrack.org/show.php?p=56&a=13>

Unofficial SUSEFAQ - PAM Pluggable Authentication Module
<http://scott.exti.net/susefaq/howto/pam.html>

Securing Applications on Linux with PAM
<http://www.linuxjournal.com/article.php?sid=5940>

docs.sun.com: Solaris Security for Developers Guide
<http://docs.sun.com/db/doc/816-4863/6mb20lvfc?a=view>

pam_ldap
http://www.padl.com/OSS/pam_ldap.html

pam_unix2(8) manual page
http://sman.informatik.htw-dresden.de/man/ALL/pam_unix2.html

The Linux-PAM System Administrators' Guide: A reference guide for available modules
<http://www.muehlgasse.de/doc/packages/pam/html/pam-6.html>

PAM Configuration
http://www.freebsd.org/doc/en_US.ISO8859-1/articles/pam/pam-config.html

10.6 PAM/NSS/LDAP

LDAP authentication using pam_ldap and nss_ldap
<http://www.tldp.org/HOWTO/LDAP-Implementation-HOWTO/pamnss.html>

LDAP Implementation HOWTO
<http://linux.co.uk/Pages/howtos/LDAP-Implementation-HOWTO.html>

nss_ldap
http://www.padl.com/OSS/nss_ldap.html

pam_ldap
http://www.padl.com/OSS/pam_ldap.html

Security with LDAP
<http://www.skills-1st.co.uk/papers/security-with-ldap-jan-2002/security-with-ldap.html>

System Authentication using LDAP
http://quark.humbug.org.au/publications/ldap/system_auth/sage-au/system_auth.html

Red Hat Linux 7.2: The Official Red Hat Linux Reference Guide
Chapter 15. Lightweight Directory Access Protocol (LDAP)
<http://www.redhat.com/docs/manuals/linux/RHL-7.2-Manual/ref-guide/s1-ldap-redhattips.html>

Using OpenLDAP For Authentication
<http://www.mandrakesecure.net/en/docs/ldap-auth.php>

Installation Guidelines for Connexitor Naming Services
<http://www.symas.net/download/connexitor/cns/INSTALL.txt>

OpenLDAP with libnss-ldap and libpam-ldap
<http://www.debianplanet.org/node.php?id=1048>

[pamldap] ldap.conf
<http://www.netsys.com/pamldap/2002/10/msg00067.html>

Using LDAP for name resolution
<http://people.debian.org/~torsten/ldapnss.html>

LDAP Authentication for Linux
<http://www.metaconsultancy.com/whitepapers/ldap-linux.htm?PHPSESSID=3f4f722fe77476db603400c54ab24dba>

10.7 Project PAM-Mysql

<http://pam-mysql.sourceforge.net/>

10.8 Project PAM-Mysql(im)

Website

<http://users.linuxbourg.ch/fvgoto/informatica/tbsc/>

Dokumentation

http://users.linuxbourg.ch/fvgoto/informatica/tbsc/doc/final/pam_mysqlim.pdf

10.9 Mysql

MySQL Manual | 13.8.2 Encryption Functions

http://dev.mysql.com/doc/mysql/en/Encryption_functions.html

MySQL Manual | A.4.1 How to Reset the Root Password

http://dev.mysql.com/doc/mysql/en/Resetting_permissions.html

MySQL Manual | 2.5.2 Upgrading from Version 4.0 to 4.1

<http://dev.mysql.com/doc/mysql/en/Upgrading-from-4.0.html>

MySQL Manual | 14.5.1.2 GRANT and REVOKE Syntax

<http://dev.mysql.com/doc/mysql/en/GRANT.html>

MySQL Manual | 13.5 Date and Time Functions

http://dev.mysql.com/doc/mysql/en/Date_and_time_functions.html

MySQL Manual | 21.2.3.44 mysql_real_escape_string()

http://dev.mysql.com/doc/mysql/en/mysql_real_escape_string.html