

Dokumentation NSS / PAM /MYSQL

Jutta Horstmann

Oktober 2004

DV-Abteilung
Wissenschaftszentrum Berlin für Sozialforschung (WZB)
Reichpietschufer 50
10785 Berlin

Table of Contents

1. Einführung.....	4
2. NSS.....	5
2.1 Grundlagen.....	5
2.2 Architektur.....	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).....	7
2.4.2 Konfiguration.....	7
2.4.3 Testen.....	8
2.5 NSCD.....	8
2.6 SuSE und NSS.....	8
3. PAM.....	9
3.1 Einführung.....	9
3.2 Konfiguration.....	10
3.2.1 Modul-Typen.....	10
3.2.2 Kontroll-Flag.....	11
3.2.3 Modul-Pfad.....	11
3.2.4 Argumente.....	11
3.3 Die Module.....	11
3.4 Kommunikation Anwendung/Modul.....	13
3.5 Kontrollfluss, Beispiel su:.....	14
3.6 Projekte.....	15
4. pam-mysql(im) (Florian Verdet, B.Sc.-Arbeit).....	16
4.1 Erweiterungen von pam_mysql durch FV.....	16
4.2 Organisation des Codes.....	16
4.3 Installation.....	16
4.4 Konfigurieren.....	17
4.4.1 Drei Nutzer anlegen.....	17
4.4.2 Datenbank anlegen.....	17
4.4.3 Tabellen anlegen.....	18
4.4.4 Rechte vergeben für die neuen Tabellen.....	18
4.4.5 Nutzer in DB pam_nss_mysql anlegen.....	18
4.4.6 PAM-Dateien editieren.....	19
4.4.7 Konfigurationsdateien.....	19
4.5 useradd-mysql.....	20
4.6 Debug-Logging.....	21
4.6.1 Testen.....	22
4.6.2 Debug-Logging mit Timestamp.....	22
4.7 Session Logging.....	23
4.8 sqlLog.....	24
4.8.1 Konfiguration.....	24
4.8.2 Testen.....	24
4.9 Account deaktivieren.....	24
4.9.1 Konfiguration.....	25

4.9.2 Test	25
4.9.3 Feedback-Problem und Patch.....	26
4.9.4 Tests: ssh, ftp.....	26
4.10 Password Expiry.....	26
4.10.1 Disabled und Expired.....	26
4.10.2 Expiry: Nicht implementiert in pam_mysql(im).....	27
4.10.3 Probleme mit Expiry und pam_unix2/libnss-mysql.....	27
4.10.3.1 lastchange.....	27
4.10.3.2 lastchange lesen (NSS).....	28
4.10.3.3 lastchange schreiben (passwd).....	28
4.11 Zusätzliche WHERE-Statements per Konfig-Datei.....	29
4.11.1 SQL-Statements im Code.....	29
4.11.2 Passwort-Check mit WHERE-Option.....	30
5. Integration pam_mysql und libnss_mysql.....	32
5.1 Vergleich der DB-Struktur.....	32
5.2 libnss_mysql: Geänderte Konfigurationsdatei.....	33
5.3 Test.....	34
5.3.1 Problem: “free(): invalid pointer” bei passwd.....	34
6. Diverse übrige Fragen.....	36
6.1 passwd	36
6.1.1 Welche Zeichen akzeptiert pam_mysql in einem Usernamen/Pass- wort?.....	36
6.1.2 “Authentication token manipulation error”.....	36
6.1.3 Passwörter in libnss.....	36
6.2 UNIX-Sockets von MySql konfigurierbar? - Ja!.....	36
6.3 Werden SQL-Statements anständig escaped? - Ja!.....	37
6.4 Kann man die SQL-Statements per Konfig-Datei ändern? - Ja!.....	37
6.5 Können PAM/NSS/MYSQL mit ACLs – Ja!.....	37
7. Exkurs: Braucht man PAM? Braucht man NSS?	39
8. Mysql	41
8.1 Upgrade auf 4.1.....	41
8.1.1 Problem: “Too many arguments in make_scrambled_password”...41	41
9. Anhang.....	43
9.1 Notwendige READMEs.....	43
9.2 Konfigurationsdateien.....	43
9.3 Interessante Sourcen.....	43
10. Quellen.....	44

1. Einführung

Linux verfügt über eine standardisierte Schnittstelle zur Erweiterung der lokalen Benutzerverwaltung. Die Rede ist hier von "Pluggable Authentication Modules" (PAM) und dem "Name Service Switch" (NSS). Über diese beiden Mechanismen wird es möglich, Anwender nicht nur über die lokale Benutzer- und Gruppendatenbank (i.d.R. die Dateien `/etc/{passwd | shadow | group}`) zu identifizieren, mit PAM und NSS können auch externe Benutzerverwaltungssysteme "angezapft" werden (z.B. LDAP, MySQL...).

Der "Name Service Switch" stellt hierbei das Wissen zur Verfügung, welche Benutzerkennungen dem System überhaupt bekannt sind, welcher Anwender sich hinter welchem Login verbirgt, welchen Benutzergruppen eine Kennung zugeordnet ist und vieles mehr. Über PAM geschieht dann, wie der Name schon sagt, die Authentifizierung eines Anwenders. In aller Regel erfolgt diese Überprüfung über die Eingabe eines Paßworts, welches dann mit dem in der (lokalen) Benutzerdatenbank hinterlegten Kennwort verglichen wird.

Der Login-Vorgang auf einem Linux-System gestaltet sich demzufolge in etwa folgendermaßen: Der Benutzer gibt seine Login-Kennung und sein persönliches Paßwort auf der Konsole ein. Daraufhin überprüft das System über die im "Name Service Switch" konfigurierten Methoden, ob die Kennung dem System bekannt ist, zu welcher Gruppe sie gegebenenfalls gehört, usw. Anschließend wird der Anwender entsprechend der PAM-Konfiguration authentifiziert.

Die vorliegende Dokumentation beschreibt die Nutzung einer MySQL-Datenbank durch PAM und NSS. Dabei wird auf alle Komponenten im Detail eingegangen, die einzelnen Schritte der Installation durchgegangen, Bugs in den Anwendungen gefixt und diverse Konfigurations- und Installations-Probleme gelöst.

Zur Anwendung kommt das PAM-Modul `pam_mysql(im)`, d.h. `pam_mysql` v. 0.5 in der "improved"-CVS-Version von Florian Verdet, `libnss_mysql` von Ben Goodwin in der Version 1.2 und MySQL, Version 4.1.2. NSS und PAM selbst sind die Standard-Komponenten einer SuSE-9.1-Professional-Distribution.

2. NSS

2.1 Grundlagen

NSS bietet eine Schnittstelle, um auf die Nutzer-Datenbanken zuzugreifen.

In der Datei `/etc/nsswitch.conf` kann eingestellt werden, welche Datenbanken genutzt werden sollen (z.B. `mysql files`)

NSS wird immer dann genutzt, wenn eine Applikation eine der folgenden Funktionen aufruft: `getpwnam`, `getpwuid`, `getspnam`, `getpwent`, `getspent`, `getgrnam`, `getgrgid`, `getgrent`, `memsbygid`, `gidsbymem`.

Zur Performanz-Steigerung bringt NSS den Name Service Cache Daemon (`nscd`) mit, konfigurierbar über `/etc/nscd.conf`. Hier werden Resultate vorhergehender Aufrufe gecached, so dass nicht jedesmal Datenbank-Zugriffe stattfinden müssen. Zugriffe auf Passwort-Daten (`shadow`) werden nicht gecached.

2.2 Architektur

NSS nutzt "Services", der Code dafür ist in "Modulen". Services können z.B. `files`, `mysql`, oder `ldap` sein, also die Art von Backend, die genutzt werden soll.

Man sagt NSS in der Datei `nsswitch.conf`, welche Services man will. Dort steht einfach der passende String, z.B. `mysql`. Die Services werden nach der Reihenfolge in der `conf` ausgewertet.

Um NSS mit MySQL nutzen zu können, muss es ein Modul (Code) geben namens `libnss_mysql`, das die Shared Library `libnss_mysql.so.2` anbietet.

Abbildung 1 zeigt einen Überblick über die gesamte Architektur.

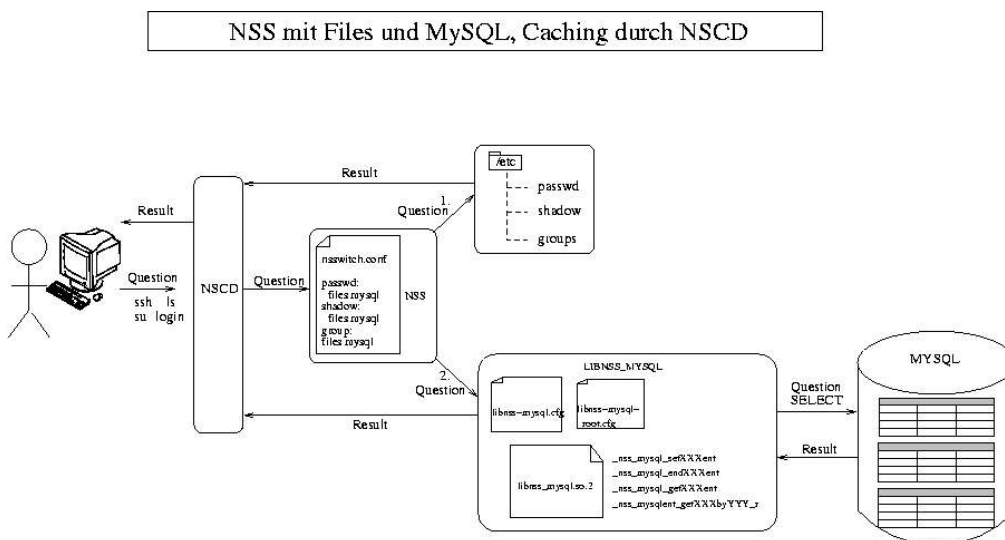


Abb. 1 Überblick NSS mit "files" und "mysql" sowie NSCD

2.3 Performance

2.3.1 nsswitch.conf

Die Reihenfolge der Services in nsswitch.conf sollte unter Berücksichtigung der Installationsumgebung erfolgen. Tests innerhalb des libnss-Projekts haben ergeben, dass Lookups auf `/etc/passwd` wesentlich schneller erfolgen als auf die MySQL-Datenbank, solange es sich um vierstellige Nutzerzahlen handelt. Ab Grössen von 10.000 – 100.000 Nutzer ist MySQL schneller. Komplette `passwd`-Scans (wie z.B. bei "finger") sind jedoch auch dann noch performanter auf `file`-Ebene.

2.3.2 MySQL

Bei hoher Aktivität soll es helfen, die MySQL-Server-Option `'thread_cache_size'` auf einen Wert ungleich Null zu setzen (Connection Threads werden für Wiedernutzung offengehalten, anstatt für jede Verbindung einen neuen Thread zu öffnen).

2.4 libnss_mysql

Die Installation, die hier dokumentiert wird, nutzt `libnss_mysql` (v.1.2) von Ben Goodwin (cinergi), zu finden auf <http://libnss-mysql.sourceforge.net>

`libnss_mysql` wird seit Juli 2004 auch von Sourceforge selbst zur Nutzerverwaltung des Portals eingesetzt (jedoch ohne PAM).

Was kann `libnss_mysql` und was nicht:

- es ist keine PAM-Lösung (kann z.B. keine Nutzerinfos updaten oder anlegen: kein `passwd`, kein `useradd`)
- es kann mit `passwd`, `shadow` und `group` – Informationen umgehen. Nicht: Mail Aliases, Hosts, Netgroups etc.
- die Library ist thread-safe, aber nicht multi-threaded. Zur Performanzsteigerung kann NSCD genutzt werden.
- bietet TCP und UNIX-Sockets
- möglich: Datenbank-Replikation. Aber nicht: Einsatz mehrerer unabhängiger MySQL-Server.
- für Verschlüsselung: Tunnel oder SSL nativ in Mysql (4.x)

2.4.1 Installation (v.1.2)

Einfach per `configure`, `make`, `make install`.

Erzeugt `/lib/libnss_mysql.so.2.0.0` und zwei Links darauf.

2.4.2 Konfiguration

Zunächst Übernehmen der Default-Konfigurationsfiles `/etc/libnss-mysql.cfg`, `/etc/libnss-mysql-root.cfg`.

Anlegen der Default-Datenbank “auth” mit dem Sample-File `sample_database.sql`. Es werden die Tabellen `grouplist`, `groups` und `users` angelegt (mehr dazu in Kapitel 5: Datenbank-Integration).

Eintrag von “mysql” in `/etc/nsswitch.conf`.

Nach Installation und Konfiguration:

SYSTEM KOMPLETT NEU STARTEN!!!

2.4.3 Testen

- funktioniert: `login`, `su`, `groups`, `ssh`, `ftp`, `id`, `getent`, `chown`, `ls`, `cd` (ohne Argumente, ins eigene Home-Verzeichnis), `touch`, `ls`
- nicht: `passwd`, `useradd`, `userdel` (Nutzerinfos ändern ist keine NSS-Aufgabe, sondern gehört zu PAM)

Aber: `useradd` und `userdel` “gucken” schon in die MySQL-Tabellen – ob der Nutzer dort drinsteht:

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

Aber sie können dort nicht reinschreiben, löschen, ändern (nur “SELECT”).

Nach dem Ändern von Userdaten in MySQL (login-Name von Hand geändert): Neuer Username wird von allen Tools erwartet und wird bei `ls -la` auch angezeigt (`uid` mappt auf neuen Namen). Der Name des Home-Verzeichnisses des Users wird natürlich nicht automatisch geändert.

Im Testfalle hat der NSCD die alten Daten nicht gecached; falls so etwas jedoch passieren sollte: Cache löschen mit `/usr/sbin/nscd -i` bzw. `--invalidate=TABLE` (z.B. `/usr/sbin/nscd -invalidate=passwd`)

2.5 NSCD

Der NSCD läuft auf SuSE 9.1 per Default. In der Konfigurationsdatei `/etc/nscd.conf` kann man Logging einschalten (loggt nur Fehler, keinen Erfolg). `/usr/sbin/nscd -g` zeigt den aktuellen Status.

Ein Check mit `lssof` zeigt, dass der NSCD auf die `libnss_mysql`-Bibliothek zugreift.

2.6 SuSE und NSS

Möchte man Nutzer/Gruppen per YaST konfigurieren, kann man nicht auf MySQL zugreifen! -- YaST wertet nur “files” aus, scheint also NSS nicht zu nutzen.

3. PAM

3.1 Einführung

Vor der Einführung des Pluggable Authentication Modules lief UNIX-Authentifizierung wie folgt: Der Nutzer gibt ein Passwort ein und die authentifizierungsbedürftige Applikation checkt, ob es mit dem in `/etc/{passwd|shadow}` übereinstimmt.

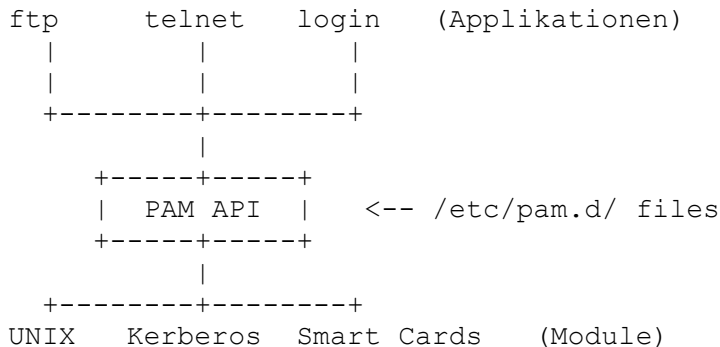
Mit der Entwicklung vieler unterschiedlicher Authentifizierungsmethoden (Bsp. Smartcards) wurde dieses Vorgehen sehr unpraktisch: Die Applikation musste immer wieder so umgeschrieben werden, dass sie die gewünschte Methode unterstützt.

PAM löst dieses Problem dadurch, dass die Applikationen nun nur noch einmal "pamifiziert" werden müssen, während sich dann PAM um die Einbindung der verschiedenen Methoden kümmert.

"Pamifiziert" bedeutet, dass das Programm PAM-Funktionen nutzt (z.B. `pam_start("login", user_name, &pam_conv, &pamh);`
`pam_set_item(pamh, PAM_TTY, tty); pam_authenticate(pamh,`
`0); pam_end(pamh, PAM_SUCCESS);`) und gegen die PAM-Bibliothek linkt (inkludiert `security/pam_modules.h`, `security/pam_appl.h`, `security/pam_misc.h`; Linking checken mittels "ldd").

Jede Authentifizierungsmethode bringt ein Modul mit, das von der Systemadministratorin zu PAM hinzugefügt werden kann.

Die Applikation ruft also die PAM API auf und PAM schickt den Request dann weiter an das (oder die) je nach Konfiguration passende(n) Modul(e). Die Antwort des Moduls leitet PAM wieder zurück an die Applikation.



Worum PAM sich nicht kümmert, ist der Rest der Nutzer-Informationen, der nichts mit Authentifizierung zu tun hat. Das Auflösen einer UID zu einem Nutzernamen (wie es z.B. `ls` benötigt), ist keine Aufgabe für PAM, sondern für NSS.

PAM und NSS müssen nicht auf die selbe Datenbasis zugreifen, obwohl es in den vielen Szenarios praktischer ist, um Inkonsistenzen und Unübersichtlichkeit zu vermeiden. Andererseits gibt es z.B. grosse Systeme oder Netzwerke, wo die meisten Nutzer dem System gar nicht bekannt sein müssen, aber ihre Authentifizierung geregelt werden muss.

3.2 Konfiguration

Die Konfiguration von PAM erfolgt über Dateien, die im Ordner `/etc/pam.d` liegen. Auf dem Test-System:

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

Jede pamifizierte Applikation kann über eine solche gleichnamige Datei bezüglich ihres Authentifizierungsverhaltens eingerichtet werden.

Dabei wird die folgenden Terminologie verwendet: Eine pamifizierte Applikation ist ein Service, ihre Authentifizierungsmethode (z.B. per MySQL oder LDAP) wird als Modul bezeichnet, und die Aufgaben der Module werden in vier Modul-Typen aufgeteilt (auth, account, password, session). Für jeden Modul-Typen können verschiedene Module verwendet werden, deren Reihenfolge und Priorisierung untereinander über Kontroll-Flags bestimmt wird. Das folgende Beispiel zeigt eine mögliche Konfiguration für `su`:

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

account   sufficient      pam_ldap.so
account   sufficient      pam_unix2.so
account   required       pam_deny.so

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

password  required          pam_deny.so

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

In der ersten Spalte steht der Modul-Typ, in der zweiten das Kontroll-Flag, in der dritten der Name des Moduls selbst (Modul-Pfad) und in der vierten optionale Argumente, die an das Modul übergeben werden können.

3.2.1 Modul-Typen

Indem man ein Modul zu einem Modul-Typen einträgt, zeigt man an, dass dieses Modul zur Ausführung der Aufgaben dieses Typs genutzt werden soll. Die Aufgaben der Modul-Typen im Detail:

- **auth:** Hier eingetragene Module kümmern sich um Authentifizieren von Nutzern per Passwort-Abfrage, dem Nutzer Zugang zu dem Service geben oder verweigern.
- **account:** Hier eingetragene Module kümmern sich um Restriktionen auf Ressourcen: Was dieser Nutzer darf, z.B. nur spezielle Features zu bestimmten Tageszeiten nutzen oder z.B. root: Nur über Konsole einloggen. Ausserdem:

Account-Management: Zeitbeschränkungen für Passwörter (password expiry) und abschalten von Accounts (account disabling)

- **password:** Hier eingetragene Module kümmern sich darum, wie das Passwort-Ändern ausschauen soll
- **session:** Hier eingetragene Module kümmern sich darum, was erledigt werden soll bevor/nachdem der Nutzer den Service nutzen darf (z.B. Logging, mounten von Filesystemen)

3.2.2 Kontroll-Flag

Die Module vom selben Typ werden der Reihenfolge nach abgearbeitet, wie sie in der Datei stehen. Das Kontroll-Flag gibt an, was passieren soll, wenn eines davon erfolgreich abgearbeitet wurde oder wenn es einen Fehler gab.

- **required:** Dieses Modul muss erfolgreich abgearbeitet werden (Kontrolle geht aber erst an die Anwendung zurück, wenn alle Module dieses Typs abgearbeitet wurden)
- **requisite:** wie required, aber die Kontrolle geht sofort bei Fehler an die Applikation zurück
- **sufficient:** wenn das entsprechende Modul erfolgreich abgearbeitet wurde, werden keine weiteren aus dem Stack desselben Typs verwendet. Gibt es einen Fehler, scheitert nicht der gesamte Modultyp, sondern es wird einfach das nächste Modul ausprobiert
- **optional:** ist für Erfolg des Modultyps egal. Wird abgearbeitet, wenn vorher in diesem Modultyp-Stack kein "requisite" gescheitert und kein "sufficient" erfolgreich war

3.2.3 Modul-Pfad

Hier stehen die verwendeten Module, bzw. die Pfadangabe zu ihrem Speicherort. Im obigen Beispiel sind das `pam_rootok.so`, `pam_ldap.so`, `pam_unix2.so`, `pam_deny.so`, `pam_pwcheck.so`, `pam_homecheck.so`. Die Dateien liegen im Default-Pfad `/lib/security`, daher muss nur der Dateiname und nicht der komplette Pfad angegeben werden. Mehr zu dem Thema im folgenden Kapitel "Module".

3.2.4 Argumente

Argumente werden bei Aufruf an das Modul übergeben. Was das entsprechende Modul damit anfängt, bleibt diesem überlassen. Generische Argumente sind z.B. `debug` (Debug-Logging in die System-Logfiles) und `use_first_pass` (Modul soll nicht nocheinmal nach Passwort fragen, sondern vorher eingetipptes Passwort verwenden)

3.3 Die Module

Neben dem Modul `pam_mysql`, um das sich diese Dokumentation dreht, gibt es eine Reihe weiterer PAM-Module, die im folgenden kurz vorgestellt werden sollen. Sie bilden entweder funktionale Äquivalente zu `pam_mysql` (z.B. `pam_ldap`, `pam_unix`) oder erfüllen einen speziellen Zweck (z.B. `pam_rootok`, `pam_pwcheck`).

pam_rootok

Das Modul wird verwendet, wenn der superuser einen Service nutzen möchte, ohne ein Passwort eingeben zu müssen. Standardanwendung `su`: `root` kann sich als beliebiger Nutzer anmelden, ohne dessen Passwort wissen zu müssen. Gecheckt wird nur, ob die UID des Aufrufers 0 ist.

pam_unix2

Hierbei handelt es sich um die SuSE-Variante von `pam_unix`, das Modul zur traditionellen Authentifizierung über `/etc/{passwd|shadow|group}`. Da das Modul jedoch die `glibc`-NSS-Calls verwendet, greift `pam_unix2` auf den Speicherort zu, der in NSS definiert ist, im vorliegenden Falle also nicht nur auf die "files" sondern auch auf "mysql".

Das Argument "nullok": Normalerweise sind Accounts ohne Passwort deaktiviert. Wenn aber "nullok", dann darf der Nutzer das Passwort eines solchen Accounts ändern.

pam_deny

Der einzige Sinn des Moduls ist es, Zugang zu verweigern. Es gibt den Fehler `PAM_ACCT_EXPIRED` zurück.

Das heisst z.B. im Falle von `su`: In einem Stack stecken immer mehrere sufficient-Module und als letztes `pam_deny` (required). Sollte also keines der vorhergegangenen Module dieses Typs erfolgreich gewesen sein, so endet die Abarbeitung auf jeden Fall bei `pam_deny` und der Applikation wird ein Fehler zurückgegeben.

pam_pwcheck

Das Modul checkt die Qualität eines neu eingegebenen Passworts, indem es zunächst "cracklib" aufruft und dann noch eine Reihe weiterer Tests ausführt. Es ist über die Datei `/etc/login.defs` konfigurierbar.

Weitere bekannte/offizielle Module:

(<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 Kommunikation Anwendung <--> Modul

Die "pamifizierte" Applikation linkt gegen die PAM-Bibliotheken und nutzt die folgenden PAM-Aufrufe: 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

- initiiert Authentifizierung
- initialisiert PAM-Bibliothek
- liest PAM Konfig-Datei

conv

PAM-Module kommunizieren nicht direkt mit dem User, darum kümmert sich die Applikation. Zum Datentransfer User-Applikation-PAM dient die conv-Funktion. Darüber kann das Modul Daten vom User abfragen, Fehlermeldungen oder sonstige Teste ausgeben.

pam_get_item, pam_set_item

Lesen und Schreiben der Status-Informationen betreffend die PAM-Transaktion. Voneinander unabhängige Module können darüber Informationen teilen (z.B. pam_mysql und pam_pwcheck das Passwort)

pam_get_data, pam_set_data

Zugriff auf Modul-spezifische Informationen.

pam_authenticate

Identifizieren des Nutzers durch Eingabe eines Passworts oder anderem Identitätsnachweis.

pam_setcred

Gibt dem Prozess die passenden Berechtigungen.

pam_acct_mgmt

Checkt, ob Account und Passwort des Nutzers valide sind (Account deaktiviert, Passwort abgelaufen, ist Nutzer nur zu bestimmten Zeiten berechtigt...)

pam_open_session, pam_close_session

Informiert die Module darüber, dass eine neue Session gestartet wurde bzw. eine laufende beendet wurde.

pam_chauthtok

Kümmert sich um Passwort-Änderung.

pam_end

Beendet PAM-Transaktionen, gibt Speicher frei, räumt auf.

3.5 Kontrollfluss, Beispiel su:

Der Kontrollfluss gestaltet sich wie folgt (ohne pam_mysql): Nutzer – Applikation – NSS/MySql – PAM/pam_unix2.

Abbildung 2 zeigt dies im Überblick.

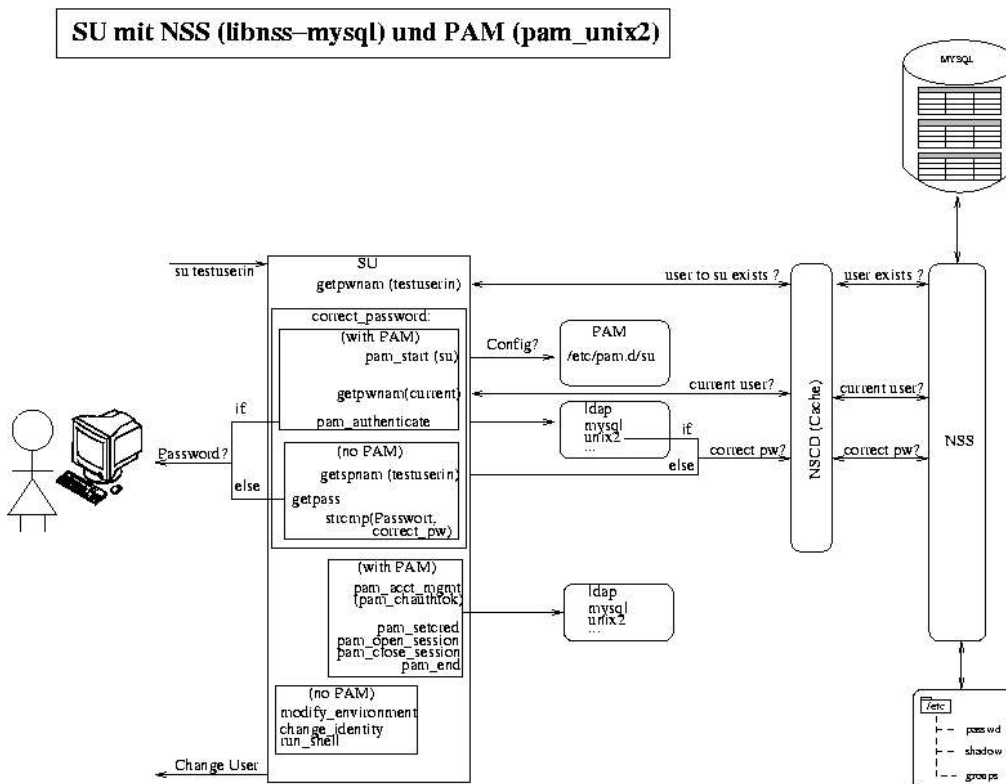


Abb.2 Kontrollfluss zwischen Applikation, PAM und NSS am Beispiel su

Der Verlauf noch einmal im Detail:

1. Nutzer: su testuserin
2. su ist gelinkt mit...

```
>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. Existiert der Nutzer, zu dem gewechselt werden soll?

```
su: pw = getpwnam (new_user);
getpwnam: ruft NSS
```
4. NSS-Dateien werden gecheckt und MYSQL-Statement ausgeführt
5. Passwort des neuen Nutzers holen

```
su: correct_password -> pam_start
(falls es PAM gibt, ansonsten: getsppnam: Passwort per libnss_mysql aus /
etc/shadow oder Mysql holen)
```
6. /etc/pam.d/su:

```
pam_authenticate -> auth: rootok (nein), ldap (nein), unix2 (ja)
```
7. pam_unix2: Passwort über NSS beziehen, User-prompt, Vergleich der Passwörter
8. Abarbeiten der restlichen PAM-Optionen (acct_mgmt, setcred, session)

3.6 Projekte

Es gibt drei pam_mysql-Projekte, von denen zwei inaktiv sind:

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

Der letzte Stand des Sourceforge-Projekts pam_mysql datiert von 2002 (v.0.5). Florian Verdet, ein Schweizer Student, hat dieses Projekt 2003 in seiner Abschlussarbeit als pam_mysql(im) weiterentwickelt und fungiert jetzt als Co-Maintainer des Sourceforge-Projekts.

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

4.1 Erweiterungen von pam_mysql durch FV

- Account Management (Accounts deaktivieren)
- Konvertierungsskripte (um Nutzerinfos z.B. aus /etc/{passwd, shadow, group} nach MySQL zu portieren. Geschrieben in Perl.)
- Session Logging in eine MySQL-Tabelle
- “Housekeeping and code cleaning”
- “Code Splitting” (aufteilen des Codes in überschaubare Module)
- Merge PAM-Mysql & NSS-Mysql Config-Files (das konnte hier nicht verwendet werden, da libnss_mysql anders aufgebaut ist als nss_mysql)
- Datenbank-Verbindung über SSL optional

4.2 Organisation des Codes

pam_mysql.c, pam_mysql.h:

Datenbankverbindung, Interaktion mit PAM, Aufruf von

pam_sm_authenticate, pam_sm_chauthtok, pam_sm_acct_mgmt,
pam_sm_setcred, pam_sm_open_session und pam_sm_close_session.

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

Enthalten Funktionen für die “eigentliche Arbeit”, z.B. Ausführen von MySQL-Anfragen.

lib.c, lib.h:

Debugging, Makro für malloc-Checks und Struct für gepackte Werte aus Konfigurations-Dateien.

mysql.c, mysql.h:

Datenbankverbindung

options.c, options.h:

Parsing der Optionen in /etc/pam.d/<service>

parser.c, parser.h:

Parsing der Optionen in Konfigurationsdateien

pwlib.c, pwlib.h:

Hilfs-Funktionen zur Passwort-Abfrage und -Erzeugung.

4.3 Installation

Verwendet wird der von Florian Verdet als aktuellster Code bezeichnete aus dem Sourceforge-CVS:

```
cvs -z3
-d:pserver:anonymous@cvs.sourceforge.net:/cvsroot/pam-mysql
co -r pam_mysql-0_5-fvg pam_mysql
```

Kompilieren mit `make`; falls es nicht das erste Mal ist: Vorher `make clean`.

Warnings der folgenden Art ignorieren:

```
warning: traditional C rejects automatic aggregate
initialization
warning: traditional C rejects ISO C style function
definitions
```

Die Library an die richtige Stelle kopieren:

```
sudo cp pam_mysql.so /lib/security/
```

4.4 Konfigurieren

Eine ausführliche Anleitung existiert in Verdets "Mini-Howto". Hier einige Anmerkungen zur getesteten Installation.

Anlegen von neuen MySQL-Nutzern, als DB-Administrator (`root`, `mysql`) in der `mysql`-System-Datenbank:

```
$ mysql -u mysql -p mysql
```

4.4.1 Drei Nutzer anlegen

Standard (`pam_nss`), Root (`pam_nss_shadow`) und Admin (`pam_nss_admin`). Standard und Root sind identisch, ausser dass Root alleine Rechte auf die `shadow`-Tabelle hat (entspricht den Rechten auf die Dateien (`etc/passwd` und `etc/shadow`)). Admin ist nicht notwendig, wenn man die Administration sowie als DB-Admin ausführt.

```
$ INSERT INTO user (Host,User,Password) VALUES
('localhost','pam_nss',PASSWORD('pamPass'));
```

```
$ INSERT INTO user (Host,User,Password) VALUES
('localhost','pam_nss_shadow',PASSWORD('PMshadow'));
```

```
$ INSERT INTO user (Host,User,Password) VALUES
('localhost','pam_nss_admin',PASSWORD('nssAdmin'));
```

4.4.2 Datenbank anlegen

```
$ create database pam_nss_mysql;
```

4.4.3 Tabellen anlegen

pam_mysql bringt sql-Skripte mit, um die folgenden Tabellen anzulegen

```
(mysql -u mysql -p pam_nss_mysql < skriptname.sql):
```

- auth (wie /etc/passwd)
- shadow (wie /etc/shadow)
- groups (wie /etc/groups)
- groupusers (Zuordnungstabellen Gruppen/Nutzer)
- session_log (s. Kapitel "Session Logging")
- disabled (s. Kapitel "Account deaktivieren")

Hinzu kommt eine Tabelle "sqllog", für die keine sql-Datei mitgeliefert wird (s. Kapitel "sqlLog").

pam_mysqlim bringt Skripte mit, um /etc/{passwd|shadow|groups} als MySQL-Tabellen zu konvertieren (skriptname.conv). Dieses Feature wurde nicht getestet, da es für die Installation nicht relevant war.

4.4.4 Rechte vergeben für die neuen Tabellen

```
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;
```

Falls Session-Logging genutzt wird:

```
$ 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;
```

Falls disabled-tabelle genutzt wird:

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

4.4.5 Nutzer in DB pam_nss_mysql anlegen

Die Verwendung von “useradd” funktioniert nicht. Eine Variante ist das Tool “useradd_mysql” von Florian Verdet, dass in pam_mysqlim enthalten ist (s. Kapitel useradd_mysql).

Ansonsten: Von Hand in MySQL anlegen, am Besten wie folgt:

```
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');
```

Hierbei wird für das Passwort ein leerer Dummy eingefügt, der danach mit “passwd” geändert werden muss.

Ausserdem muss das HOME-Verzeichnis von Hand angelegt und mit den passenden User-Rechten versehen werden.

4.4.6 PAM-Dateien editieren

In allen Services, die man mit pam_mysql nutzen möchte, muss dieses Modul passend eingetragen werden (Details s. unten). In der vorliegenden Installation handelt es sich um /etc/pam.d/{login, passwd, sshd, su, sudo, vsftpd}.

Da pam_mysql eine grosse Vielfalt von Optionen bietet, ist es besser, diese in Konfigurationsdateien zusammenzufassen, die man dann wiederum in der Service-Datei inkludiert.

Beispiel /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_authtok
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 Konfigurationsdateien

In den Dateien `pam_mysql.conf`, `pam_mysql_shadow.conf` und `pam_mysql_session.conf` kann eine Vielfalt von Einstellungen vorgenommen werden. Die verwendeten Dateien finden sich im Anhang.

Einen Überblick über die Optionen in den Konfigurationsdateien bietet das `pam_mysqlim`-“Readme”. Die Defaults können weitgehend übernommen werden, in diesem Falle ist kein extra Eintrag in eine Datei notwendig.

Kurzüberblick über die Optionen:

- Zugangsdaten zur Datenbank (user, password, host, db)
- Name und Spalten der Tabelle, die `/etc/passwd` entspricht
- Art der Verschlüsselung (hierbei ist ein Fehler im Readme zu bemerken: `crypt=N`, `Y` oder `P` funktioniert nicht, es muss `0`, `1` oder `2` verwendet werden!)
- `sqlLog` oder nicht, wenn ja: Name der Tabelle und ihrer Spalten
- `session-Log` oder nicht, wenn ja: Name der Tabelle und ihrer Spalten
- “WHERE”-Statement für eine Nutzer-Abfrage (s. auch Kapitel “Zusätzliche WHERE-Statements”)
- Account-Deaktivierung durch Spalte in “user” oder durch eigene Tabelle; Name dieser Tabelle und ihrer Spalten

Einträge in die Datei erfolgen immer in der Form `Name=Value`.

`pam_mysql.conf` und `pam_mysql_shadow.conf` werden getrennt gehalten, da sie die jeweiligen DB-Nutzer-Passwörter im Klartext enthalten (`pam_nss` und `pam_nss_shadow`). Die Datei `pam_mysql_shadow.conf` ist daher nur für `root` lesbar (Rechte auf `600`; `700` funktioniert nicht!). Die Einträge in `pam_mysql_session.conf` können auch in `pam_mysql.conf` stehen, eine Trennung in zwei Dateien ist eher bei vielen verwendeten Optionen sinnvoll, weil übersichtlicher.

4.5 useradd-mysql

Da das Standard-Tool `useradd` (genauso `userdel` etc.) nicht mit MySQL interagieren kann, wird von `pam_mysqlim` ein eigenes `useradd-mysql` angeboten.

Dies muss extra kompiliert werden, es ist nicht in das Makefile das Standrad-Moduls integriert.

Die folgenden Tabelle zeigt in einer Übersicht einen Vergleich der Möglichkeiten von `useradd` und `useradd_mysql`.

	<i>useradd (UNIX)</i>	<i>useradd_mysql (PAM_MYSQL)</i>
Optionen	[-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	gecos (default <account>) homedir (default /home/<account>) = uid (automatisch) höchste bisherige + 1 shell (default /bin/bash) account
ändert	/etc/shadow, passwd, group	PAM-Tabellen: shadow, auth, groups
liest	/etc/shadow, passwd, group NSS-Tabellen: users, groups, grouplist	PAM-Tabellen: shadow, auth, groups
home-Vz.	wird nicht angelegt (auch nicht mit -d)	wird nicht angelegt
Passwort	ohne -p: "!" in shadow	muss nachträglich mit passwd (inkl. pam_mysql) gemacht werden

Test:

- Kompilieren mit make
- Kopieren der Konfig-Datei useradd_mysql.conf nach /etc/security; die Defaults können übernommen werden.
- wenn der Nutzer schon in den pam_mysql-Tabellen ist, gibt es eine Fehlermeldung. Es werden aber nur diese gecheckt; nicht /etc/passwd etc. und auch nicht NSS
- die Argumente für useradd_mysql werden ohne Flag eingegeben und müssen daher in der richtigen Reihenfolge eingegeben werden:
- ./useradd_mysql jh "mein Name" "" /home/bla funktioniert, aber wegen des "" wird die "shell"-Spalte leer gelassen und nicht mit dem default-Wert besetzt

4.6 Debug-Logging

Per Default ist Debug-Logging eingeschaltet und zwar auf stderr (Kommandozeile). Existiert jedoch die Datei /tmp/pam_mysql-debug.log, so wird dorthin geschrieben.

Diese Einstellungen finden sich in der Datei lib.h und können nur dort geändert werden (danach kompilieren mit make).

```
#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
```

Geloggt wird in der Form “[<aufrufende Datei>:<aufrufende Funktion>] Aktion”. Im Rahmen des vorliegenden Projekts wurde noch ein Zeitstempel eingefügt.

4.6.1 Testen

Setting in /etc/pam.d/ immer wie bei su, ausser anderweitig angegeben.

- funktioniert: login, passwd, su, sudo, vsftpd, ssh
- ftp funktioniert nur, wenn pam_mysqlim die Debugging-Ausgaben nicht auf stderr loggt (Default-Einstellung), sondern in eine Datei.

4.6.2 Debug-Logging mit Timestamp

Das Debugging wird durch das D()-Makro der PAM-API übernommen. Ausgaben in der folgenden Form:

```
[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 'pamnss_shadow'@'localhost' for table
'disabled' - ev. table does not exist
[mysql.c:db_close(92)] called.
```

Der Teil in eckigen Klammern wird automatisch geschrieben, der Teil danach als Argument für D().

Um das Debug-Log effizienter nutzen zu können, füge ich an die D(("called.")); immer noch ein Timestamp an.

Das betrifft die Dateien: 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

Statt "D ("called. ");" wird jetzt immer das Makro LOGTIME (); aufgerufen, das steht in lib.h:

```
#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)
```

Es ging nicht als *Funktion* in lib.c, weil sonst der Eintrag im log: "[lib.c:logtime (65)] called -- Aug 19 11:47:02" gewesen wäre. Mit *Makro* funktioniert es: [mysql.c:db_connect(48)] called -- Aug 19 12:00:17

4.7 Session Logging

Hier kann in eine MySQL-Tabelle geloggt werden, wer wann einen Service genutzt hat und wann dieser beendet wurde. Insofern ssh oder ftp benutzt wurde, wir auch der "Remote Host" (dessen IP) eingetragen.

Zum Anlegen der Tabelle mit den Feldern sessionID, service, login, remote_host, remote_user, tty, opened, closed dient die Datei sessionlog.sql.

(NB: "remote_user" ist nicht der Username auf dem Remote System, sondern die UID der aufrufenden Applikation).

In allen Services, die das nutzen sollen, muss Sessionlogging "eingeschaltet" werden, Beispiel /etc/pam.d/su:

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

In der Datei pam_mysql_session.conf muss dann mindestens die Option sessionlog=Y eingetragen werden. Ausserdem müssen die DB-Nutzer die folgenden Rechte auf die sessionlog-Tabelle besitzen:

```
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:

- funktioniert: ftp, ssh, su, login
- sudo hat keine session-Option
- passwd unterstützt keine Sessions.
- ftp: in /etc/vsftpd.conf hinzufügen: "session_support=YES", der Parameter ist in der SuSE-Datei überhaupt nicht vorhanden und steht per default auf NO (s. man vsftpd.conf)

4.8 sqlLog

Vor FVs Implementation von session_log gab es schon sqlLog. Das hat aber nichts mit Session-Logging zu tun, sondern loggt erfolgreiche Passwort-Authentifizierungen (auch wenn dann der Dienst gar nicht genutzt werden konnte, weil u.U. der account deaktiviert oder das Passwort abgelaufen war).

4.8.1 Konfiguration

Es gibt keine passende Tabelle und keine sql-Datei -> Anlegen:

```
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;
```

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

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

4.8.2 Testen

Aufgerufen wird sqlLog() von db_checkpasswd() in pam_mysql_auth.c
--> Modul-Typ auth in den Service-Dateien in /etc/pam.d

- Es werden nur Aktionen von Usern geloggt, die in MySQL geführt werden, nicht die, die nur in files stehen!
- funktioniert alles, nur bei ssh wird der host nicht geloggt (bei ftp schon)

4.9 Account deaktivieren

pam_mysqlim bringt eine “disabled”-Tabelle und Funktion mit, über die man Accounts deaktivieren kann, anstatt sie zu löschen (und man kann auch reinschreiben, warum sie deaktiviert wurden).

Es gibt zwei Möglichkeiten: Entweder eine Spalte `disabledcolumn` direkt in der `auth`-Tabelle bei den Nutzerinformationen (dann steht dort aber nur Y oder N, kein “date”, kein “reason”).

Oder mit eigener Tabelle, das wird in der getesteten Installation eingesetzt. Diese hat die Felder: `login`, `date`, `reason`, `until`, `until_action`. Dabei sind die Felder `until` und `until_action` nur informativ, der Account wird nicht zum `until`-Datum automatisch wieder aktiviert!

4.9.1 Konfiguration

Muss gar nicht extra in den `conf`-Dateien angeschaltet werden, `disabled`-Tabelle wird immer gelesen!

Keine weiteren Options nötig, es werden die Defaults verwendet (s. `Readme`).

Es gibt kein Skript, um die `disabled`-Tabelle zu füllen, muss von Hand geschehen.

```
mysql> insert into disabled values ('neutest', NOW(), 'War immer fies', '2004-09-01', 'wenn er bettelt');
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

Der User “neutest” wird per `disabled`-Tabelle deaktiviert.

Das bedeutet, PAM schreibt ins `syslog`:

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

Behandelt wird die Situation von `db_checkaccount` in `pam_mysql_acct.c`. Wenn diese Funktionen einen `disabled`-Eintrag vorfindet, gib das PAM-Modul “`PAM_ACCT_EXPIRED`” zurück.

Nun muss berücksichtigt werden, wie der Stack in `/etc/pam.d/<servicedatei>` aussieht!

Zunächst lag dies im Beispiel `su` wie folgt:

```
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
```

Das bedeutet, wenn `pam_mysql` scheitert, übernimmt `pam_unix2` die Kontrolle! Das weiss natürlich nichts über die `disabled`-Tabelle.

Daher der korrekte Stack mit `account requisite pam_mysql.so` statt “`sufficient`”. “`Requisite`” kommt auch mit dem Fall zurecht, dass der User nur in

/etc/passwd steht und nicht in der Mysql-Tabelle. Dann übernimmt pam_unix2 korrekt. Nach PAM_ACCT_EXPIRED von pam_mysql steigt der Stack aber komplett aus, anstatt an pam_unix2 zu übergeben.

(Würde man "required" statt "requisite" verwenden, würde das nicht funktionieren, pam_unix2 würde übernehmen, wie bei "sufficient").

Das entsprechende Stacking muss in allen betroffenen Service-Dateien eingetragen werden!

4.9.3 Feedback-Problem und Patch

In dem Falle, dass der Account deaktiviert ist, schreibt pam-mysql eine korrekte Fehlermeldung ins syslog (s. oben). Der Nutzer bekommt davon aber nichts mit, sondern nur die Ausgabe "incorrect password".

Das liegt an su (andere Applikationen dito): Wenn pam nicht PAM_SUCCESS sagt, bringen die ihre eigenen Fehlermeldungen, in diesem Falle eben "incorrect password".

Dagegen lässt sich nichts machen, ich habe aber in pam_mysql_acct.c noch die Nachricht eingefügt:

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

Das bedeutet, es gibt jetzt zunächst die korrekte Fehlermeldung von pam_mysql und dann danach noch ein "incorrect password" von su.

4.9.4 Tests: ssh, ftp

(passwd zu testen macht nicht viel Sinn, da man sich dafür ja erstmal als derjenige User einloggen müsste – was man nicht darf, und wenn man es als root macht, dann darf man sowieso das Passwort ändern, ob der Account deaktiviert ist oder nicht.)

ftp und ssh erkennen beide den Account als deaktiviert und lassen kein Login zu. ftp bringt die ("gepatchte") Fehlermeldung, ssh leider nicht. Da gibt man dann dreimal das Passwort ein und bekommt danach "Permission denied", ohne weitere Auskünfte, warum das so ist.

4.10 Password Expiry

4.10.1 Disabled und Expired

Disabled und Expired sind zwei grundsätzlich unterschiedliche Konzepte!

Expire ist ein Feature von shadow, wo die Nutzerin oder die Admine festlegen kann, wann das Passwort abläuft und der Nutzer aufgefordert wird, ein neues einzugeben – wenn er das erledigt hat, kann er den Account weiter nutzen.

Disabled hingegen bedeutet (s. oben), dass die Admine einen Account unbrauch-

bar macht, ohne ihn gleich zu löschen – die Nutzerin kann nichts dagegen tun.

Für pam_mysql bedeutet das:

Expiry: Immer dann, wenn ein Service genutzt werden soll, wird gecheckt, ob das Passwort abgelaufen ist und in diesem Falle der Nutzer aufgefordert, ein Neues zu setzen (Standard, kein Extra-Feature notwendig). Zugegriffen wird dabei auf die Felder lastchange, min und max in der Tabelle “auth”.

Disabling: Wird dieses Feature verwendet (die Tabelle angelegt und gefüllt), so kann der Administrator per Tabelleneintrag einen Account deaktivieren und der User wird dann beim Nutzen eines Services abgewiesen. Zugegriffen wird auf die Tabelle “disabled”.

4.10.2 Expiry: Nicht implementiert in pam_mysql(im)

Die aktuelle (Standard)-pam-mysql-Version (0.5) implementiert überhaupt keine Account-Management-Funktionen. Das Modul gibt hier immer PAM_SUCCESS zurück und das nächste Modul im Stack wird abgearbeitet (pam_unix2). Florian Verdet hat nun in pam_mysqlim das Account Deaktivieren implementiert, d.h. das Modul checkt, ob es eine disabled-Tabelle gibt, wenn nicht oder der Account nicht eingetragen ist, gibt das Modul wiederum PAM_SUCCESS zurück und pam_unix2 übernimmt. Password Expiry wird nicht unterstützt.

Dies übernimmt dann pam_unix2, an das die Kontrolle in jedem Falle übergeht (ausser, man deaktiviert dieses Modul).

4.10.3 Probleme mit Expiry und pam_unix2/libnss-mysql

Wenn pam_unix2 die Kontrolle übernimmt, greift es auf NSS zu, um User-Daten zu erhalten. Das bedeutet: Auch hier (wegen libnss_mysql) wird noch einmal zunächst in “mysql” und dann in “files” geguckt.

Das passiert 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;
```

Also: Das Passwort ist abgelaufen, wenn der aktuelle Timestamp now grösser ist als der lastchange-Wert + max-Wert (in Tagen).

Dabei wird now berechnet als `time (NULL) / SCALE`, wobei gilt: `#define SCALE (24L*3600L)`

4.10.3.1 lastchange

lastchange selbst ist in der struct spwd (shadow.h) als long definiert:

```
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 */
};
```

- Und wird in /etc/shadow lstchg mit `lstchg = now / (60*60*24)`; gesetzt, wobei now Sekunden seit 1.1.1970 liefert.

Das Feld lastchange in der Tabelle shadow ist jedoch per Default char(50). (Daneben gibt es noch ein lastchange in der Tabelle auth im Timestamp-Format, das jedoch nirgendwo verwendet wird.)

Die Anpassungen in der Datenbank:

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

4.10.3.2 lastchange lesen (NSS)

Der MySQL-Datentyp timestamp zeigt das Datum im Format YYYY-MM-DD hh:mm:ss. Daher muss es in der libnss-mysql.cfg umgewandelt werden:

```
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
```

Erklärung:

- (24*3600) um auf Tage zu kommen
- *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.*

Man könnte natürlich auch gleich den UNIX-Timestamp in die Tabelle eintragen

(wie in /etc/shadow), aber formatiert lässt es sich hübscher lesen.

4.10.3.3 lastchange schreiben (passwd)

Sobald eine Nutzerin ihr Passwort ändert (per passwd), muss lastchange auf das aktuelle Datum gesetzt werden.

Da das Feld lastchange von pam_mysql jedoch nicht verwendet wird, schreibt die passwd-Funktion von pam_mysql auch nichts in das entsprechende Tabellenfeld.

Der notwendige Patch lautet wie folgt (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'",
options.table, "lastchange", options.passwdcolumn,
escNew, options.usercolumn, escUser);
```

(Dabei darauf achten, dass man die Änderung nicht nur in das SQL-Statement schreibt, sondern auch im malloc dafür extra Speicher alloziert!)

4.11 Zusätzliche WHERE-Statements per Konfig-Datei

Es gibt in pam_mysqlim 7 SQL-Statements:

- (Verschlüsseltes) Passwort holen für Check, ob Passwort stimmt (SELECT)
neues Passwort setzen und lastchange ändern (UPDATE)
- Wenn Session geschlossen wird: im Session-Log die entsprechende Uhrzeit in closed einfügen (UPDATE)
- Eintrag in sessionLog (INSERT)
- Eintrag in sqlLog (INSERT)
- Disabled-Infos aus der auth-Tabelle holen, falls vorhanden (SELECT)
- Disabled-Infos aus der disabled-Tabelle holen, falls vorhanden (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 im Code

Disabled-Infos aus der auth-Tabelle holen, falls vorhanden:

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

Disabled-Infos aus der disabled-Tabelle holen, falls vorhanden:

```
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);
```

Neues Passwort setzen und lastchange ändern:

```
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);
```

Wenn Session geschlossen wird: im Session-Log die entsprechende Uhrzeit in closed einfügen:

```
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);
```

Eintrag in sessionLog:

```
pam_mysql_sess.c:
snprintf(sql, querysize, "INSERT INTO %s (%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);
```

Eintrag 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);
```

(Verschlüsseltes) Passwort holen für Check, ob Passwort stimmt:

```
pam_mysql_auth.c:
snprintf(sql, querysize,
```

```
"SELECT %s FROM %s WHERE %s='%s'",
options.passwdcolumn, options.table, options.usercolumn,
escapeUser);
```

Hierfür gibt es die "where"-Option in der Konfig-Datei:

Dort kann man where=<irgendeine Bedingung> angeben, das wird dann an obiges Statement angehängt.

Für die vorliegende Installation ist diese WHERE-Option ausreichend.

4.11.2 Passwort-Check mit WHERE-Option

Hier ist zu beachten, dass die Option in der Konfigurationsdatei "where_clause" heisst, und nicht "where", wie in der pam_mysql-Doku angegeben. Beispiel:

```
where = uid=15001 AND min=0
```

So wird das Statement im Code verarbeitet (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)));
}
```

Problem: Wenn pam_mysql einen Fehler zurückgibt, dann übernimmt NSS die Authentifizierung. Darum muss dasselbe WHERE-Statement auch in die /etc/libnss-mysql.cfg eingefügt werden:

```
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
```

Dabei ist zu beachten, dass die WHERE-Klausel dann natürlich immer noch nicht für Nutzer gilt, die nur in den Standard-UNIX-Dateien stehen. Control-Flow: pam_mysql -> pam_unix2 -> libnss_mysql -> nss-"files". Nutzer, die nur in /etc/{passwd|shadow} auftauchen, können sich weiterhin erfolgreich authentifizieren, auch wenn die WHERE-Klausel nicht auf sie zutrifft.

5. Integration pam_mysql und libnss_mysql

pam_mysqlim von Florian Verdet kommt mit einer PAM/NSS-Integration bezüglich gemeinsamer Tabellen in einer gemeinsamen Datenbank und einer gemeinsamen Konfigurationsdatei. Es basiert jedoch auf dem alten nss_mysql-Modul. Um libnss_mysql zu integrieren (gemeinsam nutzen geht auch so), ist folgendes notwendig:

- Datenbanken integrieren
- libnss_mysql-SQL-Statements anpassen, so dass er mit neuer Datenbank zu-rechtkommt (oder View verwenden?)

Optional (hier nicht implementiert):

- conf-Dateien integrieren und entsprechend in /etc/pam verwenden (in options angeben)
- Syntax in conf-Dateien anpassen, damit auch NSS es versteht (vgl. nss-mysql-Integration von Florian Verdet)

5.1 Vergleich der DB-Struktur

<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	

<i>libnss_mysql</i>	<i>pam_mysqlim</i>
	disabled: `login` varchar(30) `date` datetime `reason` varchar(127) `until` date `until_action` varchar(127)

In der vorliegenden Installation wird die pam_mysql-Datenbank verwendet und NSS greift auf diese zu. Dafür sind Änderungen in libnss-mysql.cfg notwendig.

5.2 libnss_mysql: Geänderte Konfigurationsdatei

/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

getspwent   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: klappt
- getent shadow (als root) funktioniert nur, wenn das Flag 0 mit selektiert wird (betrifft getsnam und getspent). In struct spwd in shadow.h:
unsigned long sp_flag; /* not used */
sp_flag wird zwar nirgendwo im libnss-Code verwendet, aber offensichtlich muss da was drinstehen.
- su: User pam_nss_shadow auch Rechte an den Nicht-shadow-Tabellen haben.
- passwd (s. unten)
- groups , id, chown, touch, ls, cd

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

Das folgende Problem trat auf:

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

Das Passwort-Setzen funktioniert nichtsdestotrotz, es kommt nur eben diese störende Fehlermeldung. Das Problem tauchte erst nach der DB-Integration auf! (getestet: mit getrennter NSS-DB kommt es nicht).

Ursache ist der folgende 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);
```

Das Problem ist, dass tmp ge-free-t wird, obwohl es nicht mallocated wird.

Signatur von 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.

Lösung: free(tmp) auskommentiert.

crypt() wird auch in pam_mysql_auth.c verwendet, aber der Rückgabewert direkt benutzt und nicht in Variable gespeichert --> keine Änderung.

6. Diverse übrige Fragen

6.1 passwd

6.1.1 Welche Zeichen akzeptiert pam_mysql in einem Usernamen/Passwort?

Beliebig, auch in der Länge (bzw. bis zu 35 Zeichen, s. DB)

6.1.2 “Authentication token manipulation error”

Wenn man das neue Passwort beim zweiten Mal falsch eingibt, bekommt man “Authentication token manipulation error” (PAM_AUTHTOK_ERR) statt einer passenden Fehlermeldung. Der Code:

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

Die passende Fehlermeldung wird von cracklib produziert, das von pam_pwcheck aufgerufen wird (steht per SuSE-Default im Modul-Stack):

```
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"
```

6.1.3 Passwörter in libnss

libnss kann nicht mit Passwörtern umgehen, die Sonderzeichen enthalten, Zahlen gehen auch nicht (nur kleine und grosse Buchstaben)!

--> Egal, da Authentisierung ja über PAM geregelt ist

6.2 UNIX-Sockets von MySql konfigurierbar? - Ja!

NSS:

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

Wobei socket verwendet wird, wenn man bei host “localhost” eingibt und port, wenn man bei host eine IP angibt.

Wenn beide auskommentiert sind, werden die MySQL-Defaults verwendet (das sind die, die auch oben stehen). Diese könnte man dann in my.cnf konfigurieren.

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 Werden SQL-Statements anständig escaped? - Ja!

NSS:

libnss_mysql verwendet `mysql_real_escape_string()`.

Aus der MySQL-Doku:

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:

Auch pam_mysql verwendet seit v.0.4.7 `mysql_real_escape_string()`.

Aus dem 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 Kann man die SQL-Statements per Konfig-Datei ändern? - Ja!

NSS: Ja: /etc/libnss_mysql.conf

PAM: siehe Kapitel "Zusätzliche WHERE-Statements per Konfig-Datei"

6.5 Können PAM/NSS/MYSQL mit ACLs – Ja!

setfacl, getfacl kommen sowohl mit Nutzern in MySQL als auch in /etc/passwd zurecht.

Die files-Nutzer können mysql-Nutzern extended-Rechte geben und umgekehrt, die Rechte werden auch von beiden Seiten gesehen und akzeptiert.

7. MySQL

Zu MySQL ist an sich wenig zu sagen, da sich aus der PAM/NSS-Installation keine Auswirkungen auf die Datenbankkonfiguration ergeben.

Damit libnss_mysql auf die selbe Datenbank wie pam_mysql zugreifen kann, wurden in der vorliegenden Installation jedoch u.a. ein Sub-Select verwendet, die erst ab MySQL 4.1 unterstützt werden. Dazu im folgenden noch ein paar Worte.

7.1 Sub-Selects?

libnss_mysql nutzt die folgenden Statements, um die Gruppenzugehörigkeit eines Nutzers zu bestimmen und die Mitglieder einer Gruppe herauszufinden:

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

Die gemeinsame Datenbank pam_nss_mysql nutzt jedoch eine eigene Zuordnungstabelle "groupusers". Um den Zugriff darauf zu gewährleisten, wurden Subqueries verwendet:

```
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')
```

Einige Anmerkungen:

Subqueries werden erst ab MySQL Version 4.1 unterstützt. Daher wurde diese Version für die beschriebene Installation verwendet.

Statt Subqueries sind jedoch auch die folgenden Varianten möglich und werden auch von älteren MySQL-Versionen unterstützt:

```
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))
```

Nutzt man Subqueries, so ist zu beachten, dass das Subselect in memsbygid mehrere Zeilen zurückgeben kann. Daher muss die Syntax "where uid in (...)" oder "where uid = any (...)" verwendet werden.

7.2 Upgrade auf 4.1

Ist schon eine MySQL-Version installiert und man führt das Upgrade auf 4.1 aus, so ist es notwendig, "mysql_fix_privilege_tables" auszuführen (<http://dev.mysql.com/doc/mysql/en/Upgrading-from-4.0.html>).

Ausserdem ist zu beachten, dass pam_mysql das mysql-devel-rpm-Paket benötigt!

7.3 Problem: “Too many arguments in make_scrambled_password”

Nach der Installation von MySQL 4.1 erschien die Fehlermeldung “Too many arguments in make_scrambled_password”. Grund ist der folgende Code in pam_mysql_auth.c:

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

Dies basiert offensichtlich auf der Annahme, MySQL hätte mit Version 4.1.0 die Signatur der make_scrambled_password()-Funktion geändert.

In Version 4.1.3 werden jedoch sowohl im Header als auch in der Sourcen nur zwei Argumente verwendet:

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

Lösung: In pam_mysql_auth.c und pam_mysql_passwd.c nur die Funktion mit zwei Argumenten verwendet (Code geändert).

8. Exkurs: Braucht man PAM? Braucht man NSS?

Im Laufe des Projekts tauchte immer wieder die Frage auf, ob eine Installation wirklich sowohl PAM als auch NSS benötige. Und des weiteren: Wenn PAM und NSS, braucht man dann `pam_mysql`? Reicht nicht `libnss_mysql` mit `pam_unix2`?

Die unterschiedlichen Einsatzszenarien für NSS und PAM wurden in den ersten drei Kapiteln ja schon ausführlich dargestellt. Zur Wiederholung:

NSS dient zur Extraktion von Nutzerinformationen aus einer beliebigen Quelle, sobald eine Applikation diese benötigt: Login, UID, Passwort, Gruppenname, GID, Mitglieder der Gruppe, Gruppenzugehörigkeiten eines Nutzers oder alle zu dem Nutzer oder der Gruppe gehörenden Informationen auf einmal. Es kommt also immer dann zum Einsatz, wenn eine Applikation eine der folgenden Funktionen nutzt: `getpwnam`, `getpwuid`, `getspnam`, `getpwent`, `getspent`, `getgrnam`, `getgrgid`, `getgrent`, `memsbygid`, `gidsbymem`.

PAM kann diese Aufgaben nicht übernehmen.

Benötigt eine Applikation die Authentifizierung des Nutzers, so checkt sie das Vorhandensein von PAM auf dem System. Ist keines installiert, so kümmert sie sich selbst um die Authentifizierung und bietet dafür eigene Dialoge und Funktionen an, die u.a. auch auf NSS zugreifen. Zum Beispiel fordert sie das Passwort vom Nutzer, lässt das hinterlegte über NSS aus der Nutzerdatenbank/-datei holen, vergleicht es mit dem eingegebenen und akzeptiert dann den Nutzer oder nicht. Mehr bietet die Applikation nicht.

Ist PAM installiert, so nutzt die Applikation die verfügbaren PAM-Funktionen zur Authentifizierung, anstatt die in der Applikation eingebauten.

Der entscheidende Vorteil von PAM ist damit klar: Es bietet die Möglichkeit unterschiedlichster Authentifizierungsvarianten (Datei, Datenbank, Biometrie, Smartcard...), ohne dass der Code der Applikation dafür geändert werden muss.

Ist es jedoch auch notwendig, wenn ein Passwort-Dialog eigentlich ausreichen würde? - Dies bietet PAM noch zusätzlich:

- Password Expiry
- Account-Deaktivierung
- Session Management (was erledigt werden soll, bevor/nachdem der Nutzer den Service nutzen darf)
- Restriktionen auf Ressourcen
- Spezifizieren von Bedingungen für Änderung oder Setzen des Passworts
- ... beliebige weitere Möglichkeiten je nach Vorhandensein passender Module
- ... und das Anpassen dieser Optionen für jeden Service (jede Applikation) unterschiedlich, nach Bedarf

Zusammenfassend: NSS braucht man auf jeden Fall, ohne PAM geht es im Notfall, aber es macht das Leben wesentlich angenehmer.

Nun bleibt zuletzt die Frage nach dem Sinn von `pam_mysql` statt `pam_unix2`. Hintergrund: Mit `libnss_mysql` und PAM, aber ohne `pam_mysql` läuft der Kontrollfluss wie folgt:

Eine Applikation möchte authentifizieren. PAM wird aktiv und findet im Stack das Standard-Modul `pam_unix2`. Dieses ist zwar an sich für die Nutzer-Dateien `/etc/{passwd|shadow|groups}` gedacht. Aber das heisst: Zur Abfrage der Nutzerdaten verwendet es NSS, welches wiederum die Daten aus der MySQL-Datenbank holt. Man bekommt also PAM-Funktionalität und die Nutzerverwaltung in MySQL, ohne extra `pam_mysql` installieren zu müssen.

Hinzu kommt, dass `pam_unix2` Features unterstützt, die `pam_mysql` bis dato nicht anbietet, z.B. Password Expiry.

Ob der Einsatz von `pam_mysql` also Sinn macht, hängt davon ab, inwieweit man die Möglichkeiten der MySQL-Datenbank in Verbindung mit PAM nutzen möchte. Geht es nur um das Abfragen von Nutzer-Informationen aus der Datenbank, so reicht die Kombination `libnss_mysql` und `pam_unix2` aus. Doch schon das Schreiben in die Datenbank (Ändern oder Anlegen von Passwörtern) funktioniert natürlich nicht mehr, `pam_unix2` würde in diesem Falle in `/etc/shadow` schreiben!

Hinzu kommen Zusatz-Funktionalitäten von `pam_mysql`, wie die Disabled-Tabelle zum Deaktivieren von Passwörtern sowie die verschiedenen Logging-Tabellen (Sessionlog, sqlLog). Diese könnten natürlich von `pam_unix2` auch nicht genutzt werden.

Daneben kann man sich beliebige zusätzliche Features für `pam_mysql` ausmalen, die bis dato nicht implementiert sind, aber die Möglichkeiten der MySQL-Anbindung stärker nutzen. Ein Beispiel wäre – im Rahmen des Session-Managements – das Abfragen von Skriptnamen aus einer Tabelle, die beim Starten einer Session ausgeführt werden sollen.

Zusammenfassung: `libnss_mysql` + `pam_unix2` funktioniert in einem sehr schmalen Setting, das ohne Schreiben von Passwörtern (oder jeglichen Informationen) in die Datenbank auskommt. Ansonsten ist die Nutzung von `pam_mysql` zu empfehlen. (Dabei ist zu berücksichtigen, dass `pam_mysql` nicht alleine, sondern im Stack mit `pam_unix2` genutzt werden sollte, einerseits wegen der Password Expiry, andererseits für die Nutzerdaten, die nicht in der Datenbank gehalten werden, z.B. root, Systemnutzer.)

9. Anhang

9.1 Notwendige READMEs

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

9.2 Konfigurationsdateien

```
/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 Interessante Sourcen

- **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-Änderungen

lib.c	Zeitstempel in Debug-Log
lib.h	#define LOGTIME()
mysql.c	Zeitstempel in Debug-Log
options.c	Zeitstempel in Debug-Log
pam_mysql_acct.c	Zeitstempel in Debug-Log, Fehlermeldung bei Disabled
pam_mysql_auth.c	Zeitstempel in Debug-Log, make_scrambled_password
pam_mysql.c	Zeitstempel in Debug-Log, make_scrambled_password, free(tmp) auskommentiert, "lastchange" auch updaten
pam_mysql_passwd.c	Zeitstempel in Debug-Log,
pam_mysql_sess.c	Zeitstempel in Debug-Log
pwlib.c	Zeitstempel 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/          (Die Original-Dateien aus dem CVS)  
    code_jh/      (Der Code mit Änderungen 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
```

Ausserdem verwendet:

```
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. Quellen

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.dji.state.va.us&rnum=4&prev=/groups%3Fq%3Dpam%2Bnss%2Bcommon%26hl%3Den%26lr%3D%26ie%3DUTF-8%26selm%3D19990802084306.A9221%2540lappy.dji.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 Projekt nss-mysql

NSS-Projektseite auf 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 Projekt 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 Projekt PAM-Mysql

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

10.8 Projekt 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

11. Index

Alphabetical Index

/etc/pam.d 9f., 15f., 19, 22ff., 43
/etc/security 19, 21, 23ff., 37
/tmp/pam_mysql-debug.log 22
account 10f., 19, 21, 24ff., 28, 37
Account deaktivieren 2, 18, 24, 27
Accounts deaktivieren 16, 24
ACL 3, 37
auth 7, 10, 13, 15f., 18f., 21f., 24, 26, 28ff., 35, 39f., 44, 46f.
Authentication token manipulation error 3, 36
Cache 5f., 8
conf-Datei 25, 32
Connection Thread 6
conv 9, 13, 18
cracklib 12, 36
crypt 20, 34f.
D("called."); 22f.
Datenbank-Replikation 7
db_checkaccount 22, 25
db_checkpasswd 24
db_close 22, 36
db_connect 22f.
Debug-Logging 2, 11, 21f.
disabled 18, 22, 24ff., 33
Fehlermeldung 13, 26, 34, 36, 40
Florian Verdet 2, 4, 15f., 18, 27, 32
free(): invalid pointer 3, 34
getfacl 37
getgrent 5, 33, 41
getgrgid 5, 33, 41
getgrnam 5, 33, 41
getpwent 5, 33, 41
getpwnam 5, 15, 33, 41
getpwuid 5, 33, 41
getspent 5, 28, 33f., 41
getspnam 5, 15, 28, 31, 33f., 41
gidsbymem 5, 33, 39, 41
GRANT 18
groupusers 18, 32f., 39
Home 8
incorrect password 26
Konfig-Datei 3, 13, 21, 29f., 37
Konfigurationsdatei 2f., 8, 16, 19, 30, 32f., 43
Kontrollfluss 2, 14, 41

Konvertierungsskript 16
 lastchange 3, 26ff.
 lib.c 16, 22f.
 libnss_mysql **3ff., 6, 7f., 15f., 27, 31ff., 36f., 39, 41f.**
 libnss_mysql.so.2 5, 7
 libnss-mysql-root.cfg 7, 34
 libnss-mysql.cfg 7, 28, 31, 33
 LOGTIME() 23
 make_scrambled_password 3, 40
 Makro 16, 22f.
 malloc 16, 29, 34
 memsbygid 5, 33, 39, 41
 Modul 2, 4f., 9ff., 16, 19f., 24f., 27, 32, 36, 41f., 45f.
 Modul-Typ 2, 10
 multi-threaded 7
 MySQL 4.1 39f.
 mysql_fix_privilege_tables 39
 mysql_real_escape_string 37
 mysql.c 16, 22f., 30
 Name Service Cache Daemon 5
 nscd 2, 5ff.
 NSS **1ff., 8f., 12, 14ff., 27f., 31ff., 36f., 39, 41f., 44, 46f.**
 nss_mysql 2, 16ff., 22ff., 32f., 39
 nsswitch.conf 2, 5ff., 44
 optional 11, 16
 options.c 16, 22
 PAM_ACCT_EXPIRED 12, 25
 pam_acct_mgmt 13
 pam_authenticate 9, 13, 15
 PAM_AUTHOK_ERR 36
 pam_chauthtok 13f.
 pam_close_session 13f.
 pam_deny 10ff., 19
 pam_end 9, 13f.
 pam_get_data 13
 pam_get_item 13
 pam_mysql 2ff., 11ff., 39ff.
 pam_mysql_acct.c 22, 25f., 29
 pam_mysql_auth.c 22, 24, 30f., 35, 40
 pam_mysql_passwd.c 22, 29f., 34, 40
 pam_mysql_sess.c 22, 30
 pam_mysql_session.conf 19f., 23
 pam_mysql_shadow.conf 19f., 25
 pam_mysql.conf 19f., 23, 25, 37
 pam_mysqlim 18ff., 22, 24, 27, 29, 32, 43
 PAM_NEW_AUTHOK_REQD 27
 pam_nss_mysql 2, 17ff., 23ff., 33, 39
 pam_open_session 13f.

pam_pwcheck 10ff., 19, 36
 pam_rootok 10ff., 19
 pam_set_data 13
 pam_set_item 9, 13
 pam_setcred 13
 pam_sm_acct_mgmt 16
 pam_sm_authenticate 16
 pam_sm_chauthtok 16
 pam_sm_close_session 16
 pam_sm_open_session 16
 pam_sm_setcred 16
 pam_start 9, 13, 15
 PAM_SUCCESS 9, 26f.
 pam_unix2 3, 10ff., 14f., 19, 25, 27, 31, 41f., 46
 pamifiziert 9f., 13
 passwd 3f., 6ff., 12, 16ff., 25f., 28ff., 36f., 40, 42f.
 password 3, 10ff., 15, 19ff., 26f., 32ff., 36f., 40
 Password Expiry 3, 26f., 41f.
 Patch 3, 26, 29
 Performance **2, 6**
 Performanz 5, 7
 Permission denied 26
 PRIVILEGES 18
 pwlib.c 16, 22
 Readme 19f., 25, 43
 required 10ff., 19, 25
 requisite 11, 19, 25
 service 4ff., 10ff., 16, 19, 21, 23ff., 30, 32, 41, 44, 46f.
 session 10f., 13ff., 18ff., 23f., 32
 Session Logging 2, 16, 18, 23
 session_support=YES 23
 Session-Logging 18, 24
 setfacl 37
 shadow 4f., 7, 9f., 12, 15ff., 31ff., 42f.
 Smartcard 9, 41
 Sourceforge 6, 15, 17
 sqlLog 2, 18, 20, 24, 29f., 42
 SSL 7, 16
 Stack 11f., 25, 27, 36, 42
 stderr 21f., 26
 struct spwd 27, 34
 Sub-Select 39
 Subqueries 39
 sufficient 10ff., 19, 23, 25
 SuSE 2, 4, 8, 12, 23, 36
 syslog 25f.
 TCP 7
 thread_cache_size 6

thread-safe 7
 Timestamp 2, 22, 27f.
 Too many arguments in make_scrambled_password 3, 40
 unix_timestamp() 28
 UNIX-Socket 3, 7, 36
 updatePasswd 29, 34
 updatePasswd() 29
 useradd 2, 7f., 10, 18, 20f., 43
 useradd_mysql 18, 20f., 43
 useradd_mysql.conf 21, 43
 userdel 8, 20, 43
 Verschlüsselung 7, 20
 vsftpd.conf 23
 warning: traditional C rejects automatic aggregate initialization 17
 warning: traditional C rejects ISO C style function definitions 17
 where_clause 31
 WHERE-Klausel 31
 WHERE-Statement 3, 20, 29, 31, 37
 YaST 8
 Your account has been disabled. Please contact your system administrator 26
 Your password has expired. Choose a new password 27
 Zeitstempel 22