

PAM

Pluggable Authentication Modules

Markus Korzendorfer Hagen Paul Pfeifer

Software Anwendungs Architektur
Computer-Networking — Fachbereich Informatik
Fachhochschule Furtwangen

<http://pam.0xdef.net>
pam@0xdef.net

17. Dezember 2004

Agenda

1 Einführung

- Prolog
- PAM Architektur

2 Anwendung

- Konfiguration
- Module
- Applicationen
- Anwendungsbeispiele

3 Implementierung (PAM-Unplugged)

- Anwendungsentwicklung
- Modulentwicklung
- pam_blue

4 Epilog

Was ist PAM

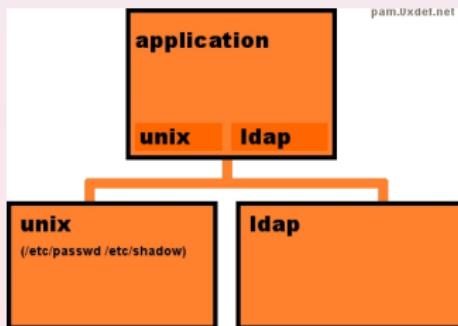
- pluggable authentication modules
- flexibler Mechanismus für Authentification, Session, Passwort und Account Management
- nach OSF-RFC 86.0

PAM Ziele (nach RFC 86.0)

- ① Systemadministrator soll Authentisierungsmechanismus bestimmen
- ② Interaktion mit Anwendung (Darstellung telnet ⇔ xdm)
- ③ Konfigurierbar pro Anwendung
- ④ verschiedene Authentisierungsmechanismen stackbar
- ⑤ mehrere Passwörter möglich
- ⑥ Passwort-, Benutzerkonten-, und Sitzungsmanagement Model
- ⑦ Abwärtskompatibel
- ⑧ Benutzertransparenz

Historisch

- Vergleicht Passwort mit /etc/passwd (/etc/shadow)
- Benutzer ist Benutzer wenn Passwort korrekt
- Anwendungen benötigten mehr: es entstanden Insellösungen



Probleme

- unflexibel
- hoher administrativer Aufwand
- bei neuen Authentication Schema folgt ein rewrite der Application
- keine strikte Trennung Authentifikation von Applicationscode
- Softwareentwickler ist für die Implementierung von sicherheitskritischen Code verantwortlich

PAM

- PAM trennt Applicationscode von Authentificationscode durch Schnittstelle
- Administrator bestimmt Authentifikationsmechanismus
- Sammlung von Modulen
- flexibel durch „Modulstacking“
- spezifiziert in OSF-RFC 86.0
- Unterstützung durch AIX, FreeBSD, HP/UX, GNU/Linux und Solaris

PAM Schichten Modell

ssh login apache gdm ...

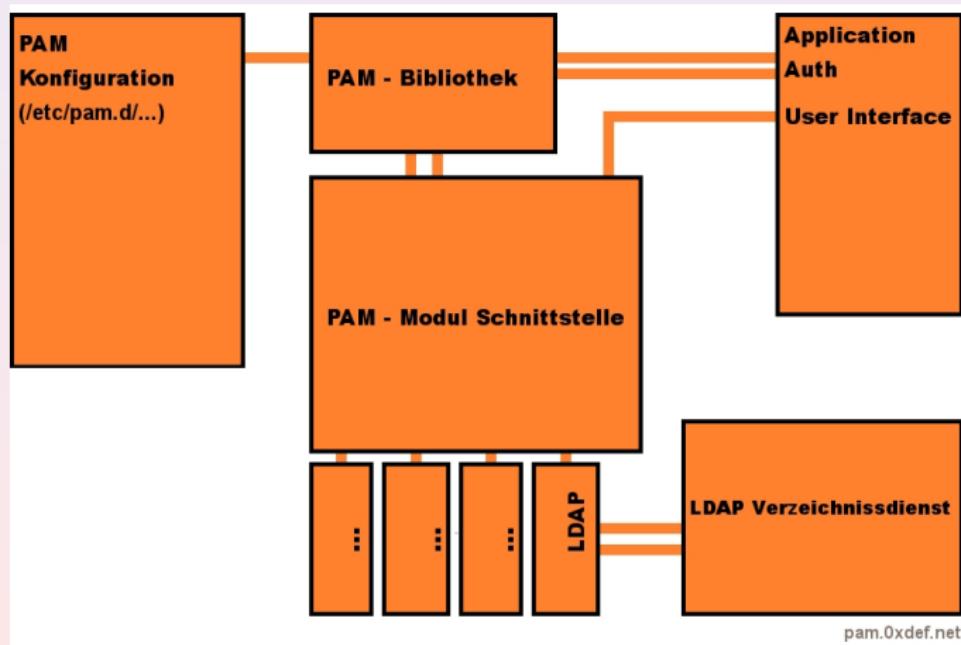
pluggable authentication modules - interface

ldap unix kerberos secureid ...

Konfiguration - Übersicht

- /etc/pam.d ⇒ Konfigurationsdateien der Applicationen
- /etc/pam.conf ⇒ Konfigurationsdateien der Applicationen (historisch)
- /lib/security ⇒ Modulverzeichniss (Bibliotheksmodule)
- /etc/security ⇒ Konfigurationsdateien der Module

Authentifizierungsmechanismus (visuell)



Typische PAM Konfiguration

Modultyp	Kontrollflag	Modulpfad	Argumente
auth	sufficient	/lib/security/pam_ldap.so	debug
auth	required	/lib/security/pam_unix.so	use_first_pass debug
session	required	/lib/security/pam_unix.so	debug
password	required	/lib/security/pam_cracklib.so	minlen=10
password	required	/lib/security/pam_unix.so	md5 shadow

Modultyp

Spezifiziert welche Managementfunktion erfüllt werden soll

① auth

- Benutzeridentifizierung und -authentifizierung (z.B. Passwortabfrage oder Smartcards)

② account

- Verwaltung des Accounts („Gibt es dieses Benutzer im System und darf er sich anmelden?“)

③ password

- Steuerung der Passwortänderung („Dieses Passwort ist zu kurz!“)

④ session

- Verwaltung der Sitzung (Limits, Berechtigungen, ... während des Zugriffes)

Modulsteuerung (Kontrollflag)

Spezifiziert das Verhalten in Abhängigkeit des Rückgabewertes

① required

- Modul muss zwingend durchlaufen werden

② requisite

- bei Fehler wird sofort zum Anwendungsprogramm zurückgekehrt

③ sufficient

- bei Erfolg des Modul ist dies für eine positive Gesamtmeldung ausreichend

④ optional

- bei Erfolg oder Misserfolg werden trotzdem alle nachfolgende Module abgearbeitet

Modulpfad und Argumente

- Modulpfad
 - ist ein Verweis auf das zu benutzende Modul
- Argumente
 - debug
 - liefert Diagnosemeldungen an Logging Daemon
 - use_first_pass
 - versucht Passwort von vorhergehenden Modul zu übernehmen
 - try_first_pass
 - fordert im Fehlerfall den User auf, sein Passwort erneut einzugeben

PAM Module (kleiner Auszug)

① pam_unix

- bildet historischen Authentifizierungsmechanismus nach (/etc/passwd und /etc/shadow)

② cracklib

- prüft Passwort auf Schwachstellen

③ ldap

- vergleicht Passwort gegen LDAP Verzeichnisdienst

④ time

- setzt zeitgesteuerte Zugangskontrollen

⑤ limits

- teilt Systemressourcen pro Benutzer zu (CPU, Speicher, ...)

PAM enabled applicationsTM

- apache
- login
- samba
- ftp
- imapd
- ssh
- ...

Anwendungsbeispiel Nr. 1

Beispiel vHost (remote login via ssh (/etc/pam.d/sshd))

```
auth      required    /lib/security/pam_securetty.so
auth      sufficient   /lib/security/pam_ldap.so          debug
auth      required    /lib/security/pam_unix.so         try_first_pass

account   sufficient   /lib/security/pam_ldap.so
account   required    /lib/security/pam_unix.so

password  sufficient   /lib/security/pam_ldap.so
password  required    /lib/security/pam_unix.so         use_first_pass md5 shadow

session   required    /lib/security/pam_unix.so
```

Anwendungsbeispiel Nr. 2

Beispiel für sicherheitskritischen Bereich
(login (/etc/pam.d/login))

```
auth      required  /lib/security/pam_securetty.so
auth      required  /lib/security/pam_secureid.so      debug
auth      required  /lib/security/pam_unix.so        debug

account   required  /lib/security/pam_unix.so

password  required  /lib/security/pam_cracklib.so   minlen=12 retry=3
password  required  /lib/security/pam_unix.so        use_first_pass md5 shadow

session   required  /lib/security/pam_time.so
```

Anwendungsentwicklung - I

- Betrachtung der PAM module als black box
- Schnittstellen sind von primärer Bedeutung
- Synopsis:

```
# include <security/pam_appl.h>
cc -o application object1.o object2.o -lpam -ldl
```

Anwendungsentwicklung - II

Schnittstellen

- `pam_start(const char *, const char *, const struct pam_conv *, pam_handle_t **)`
- `pam_set_item(pam_handle_t *, int, const void *)`
- `pam_authenticate(pam_handle_t *pamh, int flags);`
- `pam_end(pam_handle_t *, int)`

Sicherheitsaspekte

- Module haben die selben Rechte wie Applicationen die sie nutzen
- Rückgabewerte verifizieren
- Servicename hardcoden (`argv[0]` ist gefährlich)
- ...

Modulentwicklung

- Module sind dynamische Programm bibliotheken
- liefern definierte Schnittstelle für PAM
- keine static deklarierte Variablen (`pam_set_data()`)
- Vorsicht vor `free(3)`
- Synopsis:

```
# include <security/pam_modules.h>
cc -fPIC -c object1.c ld -x --shared -o
pam_object.so object1.o
```

Modulentwicklung - II

Schnittstellen

- `int pam_get_item(const pam_handle_t *, int, const void **);`
- `pam_get_user(pam_handle_t *, const char **, const char *);`
- `pam_sm_authenticate(pam_handle_t *, int, int, const char **)`
- `pam_sm_acct_mgmt(pam_handle_t *, int, int, const char **)`
- `pam_sm_open_session(pam_handle_t *, int, int, const char **)`
- `pam_sm_chauthtok(pam_handle_t *, int, int, const char **)`

Modulentwicklung - II

Argumente

- debug
- no_warn
- use_first_pass
- try_first_pass
- use_mapped_pass
- expose_account

Sicherheitsaspekte

- Vorsicht bei Speicheroperationen (`free(3)`, `malloc(3)`, ...)
- **immer** einen sauberen `return()`
- kein `static`
- Achtung auf uid's (`setuid(2)` Programme)
- im Fehlerfall: Meldung an Application
- `syslog(3)` ist dein Freund

pam_blue

- pam_blue ist ein module welches Benutzer gegen ein Bluetooth-device authentifiziert
- es ist generisch ⇒ alle Applicationen können sich gegen pam_blue authentifizieren
-

pam_blue - Konfiguration

Konfiguration (/etc/security/bluescan.conf)

```
general {  
    timeout = 4;  
}  
korzendorfer = {  
    name = tux;  
    bluemac = 54:34:34:34:34:34;  
    timeout = 10;  
}  
pfeifer = {  
    name = AIRBUS;  
    bluemac = 00:0E:07:3B:96:02;  
}  
@users = {  
    name = AirbusA412;  
    bluemac = 54:34:34:34:34:34;  
    timeout = 10;  
}  
gast = {  
    bluemac = 54:34:34:34:34:34;  
}
```

pam_blue - Ausblick

- Möglichkeit des automatischen Ausloggens bei Verlassen der Reichweite
- . . . , your ideas here!

Quellen

- ▶ Manual Pages

man {pam, su, passwd, ...}

- ▶ <http://www.kernel.org/pub/linux/libs/pam>

Documentation, Module,

-  Samar, Vipin and Charlie Lai.

Making Login Services Independent of Authentication Technologies

<http://www.sun.com/software/solaris/pam/pam.external.pdf>

-  Morgan, Andrew G.

The Linux-PAM System Administrator's Guide

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

... use the source, luke!

Fin

- Fragen?