Systemy Operacyjne:
Pluggable Authentication Modules

Dymitr Pszenicyn
<dp189434@zodiac.mimuw.edu.pl>

03-01-2003

Spis treści

1 Wprowadzenie

Ten dokument ma na celu zapoznać czytelnika z technologią PAM.

PAM – Pluggable Authentication Modules – jest mechanizmem zapewniającym pisanie aplikacji związanych z bezpieczeństwem, który może być łatwo skonfigurowany dla potrzeb systemu i konkretnego programu.

Aplikacje pisane z wykorzystaniem biblioteki PAM mogą być dostosowywane przez administratora do pracy z innym modułem autentykacyjnym bez rekompilacji.

Po raz pierwszy PAM został wykorzystany w systemie operacyjnym Solaris 2.3 firmy Sun i był wykorzystywany przez aplikacje systemowe takie jak: login, passwd, su, rlogind, rshd, telnetd, ftpd. Teraz PAM jest wykorzystywany w wielu programach, także w innych systemach operacyjnych takich jak Linux i Windows NT.

1.1 Definicje

W tym dokumencie będę posługiwać się następującymi pojęciami:

PAM – Pluggable Authentication Modules – mechanizm bezpieczeństwa, zbiór bibliotek wspomagający pisanie programów związanych z bezpieczeństwem;

moduł – moduł autentykacyjny PAM – program który może być wykorzystywany do autentykacji użytkownika w programie korzystającym z PAM;

aplikacja – program wykorzystujący bibliotekę PAM do czynności związanych z bezpieczeństwem;

konfiguracja PAM – przypisanie przez administratora aplikacjom określonych modułów wraz z odpowiednimi argumentami;

funkcja konwersacji – funkcja przekazywana podczas inicjacji PAM z aplikacji do PAM, a później wykorzystywana przez moduły do interakcji z użytkownikiem; taka architektura zapewnia niezależność od interfejsu użytkownika;

2 Cele PAM

Podczas projektowania PAM były postawione następujące cele:

- Administrator systemu powinien móc wybrać domyślny sposób autentykacji.
- Administrator systemu powinien móc ustalić sposób autentykacji dla każdego programu z osobna.
- Interakcja z użytkownikiem nie powinna być ograniczona przez PAM. Ważne jest by zarówno programy działające na konsoli (np. login) jak i aplikacje z graficznym interfejsem użytkownika (np. xdm) mogły wykorzystywać PAM.

- Powinna być możliwość skonfigurowania aplikacji w ten sposób, by korzystała z kilku mechanizmów autentykacji jednocześnie.
- Powinna być możliwość wykorzystania hasła podanego jednemu modułowi autentykacyjnemu w następnym module (stos modułów) bez ponownego wpisywania.
- Moduł powinien móc korzystać więcej niż z jednego hasła.
- Elementy systemu nie powinny wymagać zmiany gdy zmieni się mechanizm na inny.
- Architektura powinna zapewniać „wtyczkowy" model dla autentykacji, a także zarządzania hasłami, kontami i sesjami.
- PAM powinno spełniać wymagania stawiane obecnie używanym w systemie mechanizmom zapewniającym bezpieczeństwo.

PAM nie zajmuje się natomiast następującymi zagadnieniami:

- PAM pomaga tylko ustalić tożsamość użytkownika. Nie zajmuje się natomiast przekazywaniem tej tożsamości innym programom (np. jeśli użytkownik wprowadzi hasło do programu login, to program ftp uruchomiony później nadal będzie wymagał podania hasła).
- PAM nie zajmuje się problemem przesyłania niezakodowanych danych przez sieć (np. przesyłanie niezakodowanego hasła przez klient ftp).
- PAM nie zajmuje się spójnością danych trzymanych w różnych miejscach (np. jeśli hasła trzymane w dwóch bazach danych były identyczne, to PAM nie zapewni automatycznej ich synchronizacji – to zależy od konkretnego modułu).

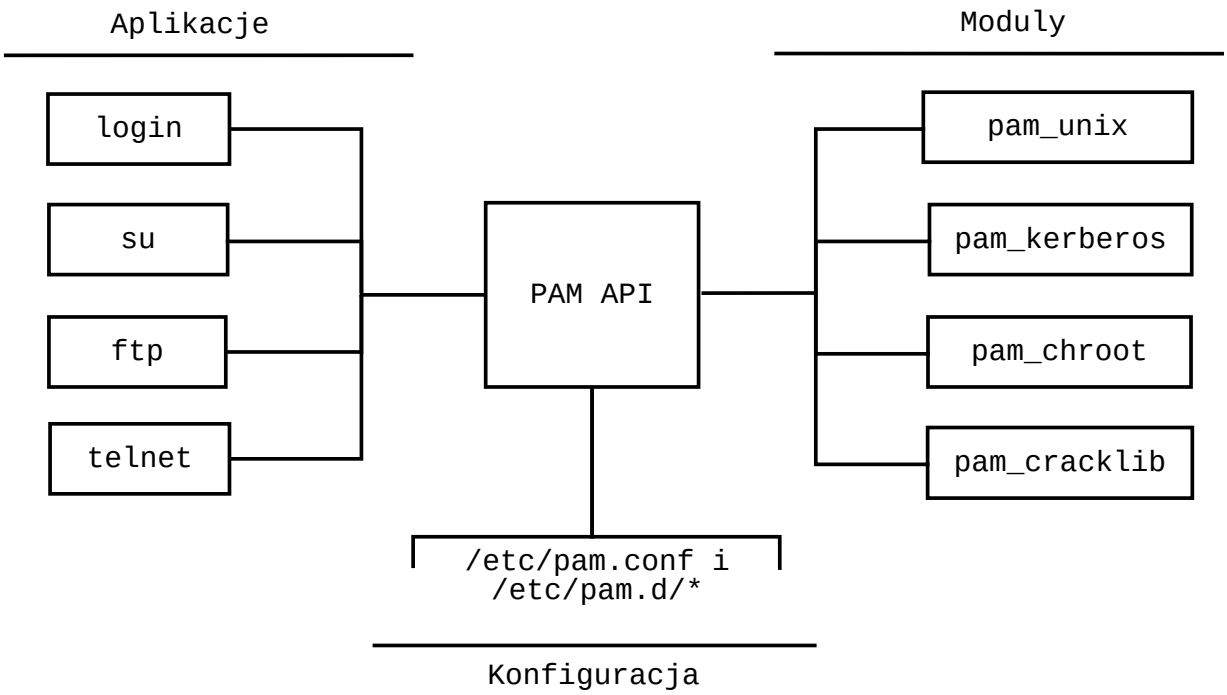
3 Omówienie PAM

W uproszczeniu, powiązania pomiędzy PAM, modułami oraz aplikacją obrazuje diagram 1.

3.1 Wykorzystanie PAM w systemie

Zwykle, wykorzystanie PAM w systemie przebiega następująco:

1. Administrator instaluje w systemie oprogramowanie wykorzystujące PAM (np. sudo lub ssh).
2. Administrator instaluje w systemie moduły PAM które chciałby wykorzystywać w aplikacjach (część modułów, np. pam_unix, jest w standardowym pakiecie PAM).
3. Administrator konfiguruje PAM dla swoich aplikacji (np. sudo lub ssh) „mówiąc" PAM z jakich modułów ma korzystać dana aplikacja do autentykacji.



Rysunek 1: Diagram przedstawiający PAM w uproszczeniu

4. Użytkownik chce skorzystać z aplikacji (np. sudo) i uruchamia ją.
5. Aplikacja prosi użytkownika o nazwę użytkownika (lub np. bierze ją jako argument).
6. Aplikacja wywołuje funkcję pam_start z PAM API przekazując nazwę użytkownika oraz wskaźnik do funkcji „konwersacji” z użytkownikiem.
7. PAM decyduje jakie moduły powinny wziąć udział w autentykacji na podstawie pliku konfiguracyjnego tego programu.
8. Aplikacja wywołuje funkcję pam_authenticate żeby sprawdzić tożsamość użytkownika.
9. Moduły które są skonfigurowane dla tej aplikacji po kolei próbują sprawdzić tożsamość użytkownika i jeśli potrzebują komunikacji z użytkownikiem to wywołują funkcję konwersacji (przekazaną przez aplikację).
10. Po wykonaniu się odpowiednich modułów, do aplikacji zwracany jest wynik pozytywny (PAM_SUCCESS) lub negatywny (PAM_AUTH_ERR lub PAM_ACCT_EXPIRED lub PAM_AUTHOK_ERR – w zależności od kategorii modułu).
11. Aplikacja wykonuje się dalej w zależności od powodzenia autentykacji (np. wychodzi z błędem w razie niepowodzenia).
12. Przed wyjściem, aplikacja wywołuje funkcję pam_end z PAM API.

4 Moduły

Moduły eksportują funkcje należące do czterech kategorii:

auth – moduł sprawdza, czy użytkownik jest tym, za kogo się podaje (czyli na przykład prosi aplikacje o pobranie hasła) oraz może ustawić jakieś dane użytkownika (np. nadaje przynależność do grupy) po uwierzytelnieniu;

account – moduł przeprowadza sprawdzenia nie związane z autentykacją (np. sprawdza godzinę lub konsolę na której jest użytkownik);

session – moduł wykonuje czynności które muszą być wykonane przed/po zalogowaniu (np. zamontowanie/odmontowywanie katalogów lub pisanie do logów);

password – moduł uaktualniania hasła użytkownika;

Moduł może implementować funkcje należące do różnych kategorii na raz.

Moduły mają argumenty wywołania które mogą być przekazywane poprzez plik konfiguracyjny PAM.

Najprostszymi modułami są pam_access i pam_deny. Pierwszy z nich zawsze zwraca powodzenie (PAM_SUCCESS), a drugi zawsze porażkę. Oba moduły należą do wszystkich czterech kategorii.

Dokładny opis przydatnych modułów znajduje się w dokumencie Piotra Zuba „Systemy Operacyjne: PAM – opis modułów i konfiguracji”.

5 Pliki konfiguracyjne

Administrator systemu może skonfigurować jakie moduły będą używane przez konkretną aplikację. Do tego celu służy plik /etc/pam.conf lub katalog /etc/pam.d. W pliku pam.conf może być umieszczona konfiguracja dowolnych programów, ale zaleca się używanie katalogu /etc/pam.d w którym każdy program ma osobny plik konfiguracyjny o nazwie odpowiadającej nazwie programu.

W pliku konfiguracyjnym zawarte są następujące informacje:

kategoria modułu – może być auth, account, session lub password;

flaga kontrolna – wskazuje ona jaki wpływ na autentykację ma dany moduł;

moduł – ścieżka i nazwa modułu;

argumenty modułu – argumenty przekazywane do modułu;

Opis dokładnej budowy pliku konfiguracyjnego (wraz z przykładami) znajduje się w dokumencie Piotra Zuba „Systemy Operacyjne: PAM – opis modułów i konfiguracji”.

6 Opis PAM API

Interfejs PAM dla aplikacji składa się z następujących funkcji:

pam_start – pierwsza funkcja PAM która musi być wywołana przez aplikację; jako argumenty dostaje między innymi nazwę użytkownika i wskaźnik do funkcji „konwersacji”; w przypadku sukcesu zwraca na jednym z argumentów uchwyt który powinien być wykorzystywany przy wołaniu innych funkcji PAM;

pam_end – ostatnia funkcja PAM która musi być wywołana przez aplikację; jako argument dostaje uchwyt zwrócony przez pam_start;

pam_set_item – funkcja służąca do ustalania zmiennych PAM takich jak: PAM_SERVICE, PAM_USER, PAM_USER_PROMPT, PAM_TTY, PAM_RUSER, PAM_RHOST, PAM_CONV lub PAM_FAIL_DELAY;

pam_get_item – funkcja służąca do odczytania zmiennej PAM; jako argument dostaje nazwę zmiennej;

pam_strerror – funkcja zwraca słowny opis błędu;

pam_fail_delay – funkcja ustala minimalny czas jaki należy odczekać w przypadku błędu autentykacji;

pam_authenticate – funkcja prosi moduły przypisane do tej aplikacji o próbę autentykacji użytkownika;

pam_setcred – funkcja prosi moduły przypisane do tej aplikacji o ustawienie specyficznych dla każdego modułu danych użytkownika (takimi danymi są na przykład UID i GID w Linux ale zwykle są one obsługiwane bezpośrednio przez aplikację);

pam_acct_mgmt – funkcja prosi moduły przypisane do tej aplikacji o stwierdzenie, czy konto użytkownika jest aktywne, czy może się on zalogować o tej godzinie; zazwyczaj funkcja ta jest wołana przez aplikację po tym jak użytkownik został uwierzytelniony;

pam_chauthtok – funkcja prosi moduły przypisane do tej aplikacji o zmianę haseł użytkownika (poprzez funkcję konwersacji); przy podaniu argumentu PAM_CHANGE_EXPIRED_AUTHTOK tylko moduły w których hasło użytkownika wygasło powinny poprosić o zmienię hasła;

pam_open_session – funkcja prosi moduły przypisane do tej aplikacji o otwarcie sesji (może to na przykład być zamontowanie katalogu);

pam_close_session – funkcja prosi moduły przypisane do tej aplikacji o zamknięcie sesji (może to na przykład być odmontowanie katalogu);

pam_putenv – funkcja wstawia (lub usuwa) do środowiska PAM zmienną postaci NAZWA=wartość; uwaga: to środowisko nie jest tym samym co środowisko ustawiane np. przez aplikację login;

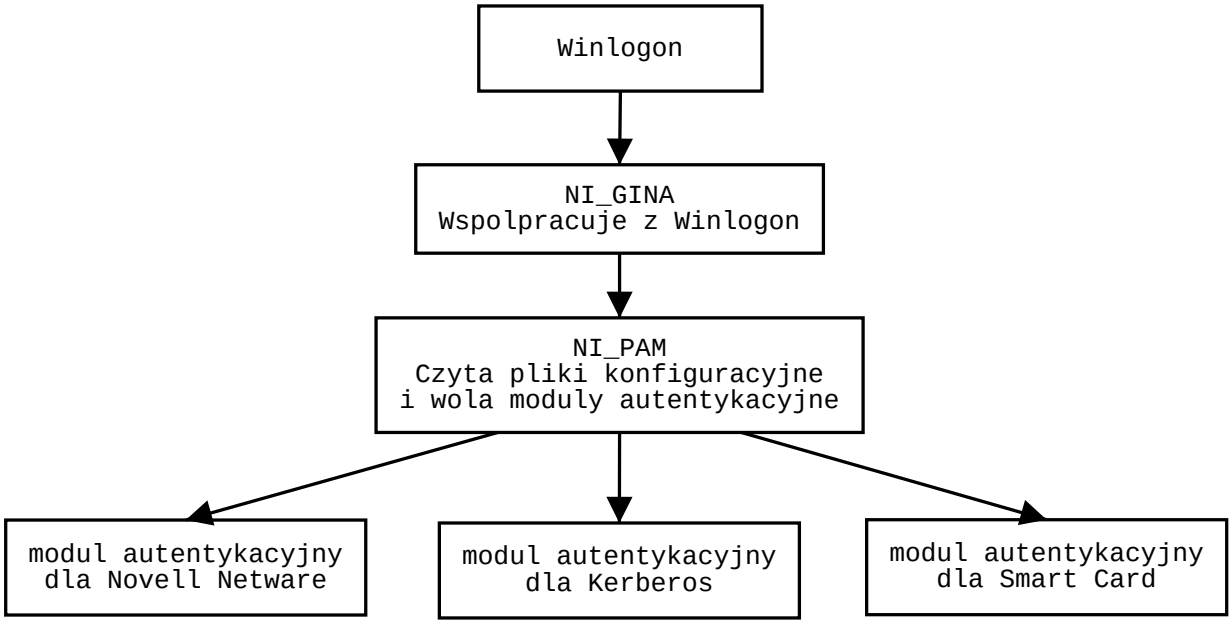
pam_getenv – funkcja pobiera zmienną ze środowiska PAM; uwaga: to środowisko nie jest tym samym co środowisko ustawiane np. przez aplikację login;

pam_getenvlist – funkcja zwraca całe środowisko PAM;

7 PAM w Windows NT

Istnieje projekt stworzenia PAM w Windows NT.

Ogólny pomysł przedstawia diagram 2.



Rysunek 2: Diagram przedstawiający PAM w Windows NT (na podstawie strony http://www.citi.umich.edu/projects/singlesignon/poster2.html)

Autorzy pomysłu planują zastąpić mechanizm autentykacji w Windows – GINA przez moduł NI_GINA. Moduł ten współpracowałby z programem NI_PAM będącym implementacją PAM pod Windows NT. Wtedy NI_PAM mógłby korzystać z dowolnych modułów autentykacyjnych (np. dla Novell Netware, Kerberos albo Smart Card).

A PAM w Internecie

Niektóre strony WWW związane z PAM:

http://www.kernel.org/pub/linux/libs/pam/ – Główna strona projektu Linux-PAM. Znajdują się tu źródła i dokumentacja.

http://www.sun.com/software/solaris/pam/ – Główna strona projektu PAM w systemie operacyjnym Solaris firmy Sun. Znajduje się tu dokumentacja i źródła przykładowych modułów PAM.

http://www.kernel.org/pub/linux/libs/pam/modules.html – Lista odnośników do modułów PAM tworzonych przez rożnych ludzi.

http://www.usenix.org/publications/libraryproceedings/usenix-nt98/itoi_slides/ – Strona poświęcona projektowi stworzenia PAM dla Windows NT. Są na niej slajdy wyjaśniające zalety stworzenia PAM dla Windows.

http://www.citi.umich.edu/projects/singlesignon/poster2.html – Strona przedstawiająca w skrócie zalety i sposób realizacji PAM dla Windows NT.

B Bibliografia

- Andrew G. Morgan „The Linux-PAM Systems Administrators' Guide v0.77"
- Andrew G. Morgan „The Linux-PAM Application Developers' Guide v0.76"
- Andrew G. Morgan „The Linux-PAM Module Writers' Guide v0.76"
- Open Source Foundation RFC 86.0 „Unified Login with Pluggable Authentication Modules (PAM)"
- Sun Microsystems „Making Login Services Independent of Authentication Technologies"
- Peter Honeyman „Pluggable Authentication Module for Windows NT"