Bezpieczeństwo systemów rozproszonych

ĆWICZENIE PAM

Wprowadzenie

Modularne systemy uwierzytelniania mają za zadanie ułatwić kontrolę dostępu do systemu operacyjnego poprzez uniezależnienie od kodu aplikacji i zautomatyzowanie egzekwowania polityki bezpieczeństwa odnośnie kont użytkowników. Narzędzie PAM (*Pluggable Authentication Modules*) jest przykładem systemu modularnego uwierzytelniania szeroko stosowanym w środowisku Linux i posiada bardzo dużą liczbę dostępnych modułów rozszerzających.

System Unix w początkowych stadiach swojego rozwoju nie oferował jednolitej metody kontroli dostępu do systemu. Aplikacje wymagające dodatkowego uwierzytelniania użytkowników same były odpowiedzialne za parsowanie pliku `/etc/passwd` i podejmowanie decyzji o udzieleniu dostępu. Każda zmiana w systemie zarządzania użytkownikami powodowała poważne skutki – oprogramowanie musiało być zmodernizowane tak, by uwzględnić aktualizacje formatów, wprowadzenie nowych mechanizmów czy ulepszonych algorytmów szyfrowania.

Odpowiedzią na powyższe problemy jest rozdzielenie logiki aplikacji od polityki (i mechanizmów) kontroli dostępu. Rozdzielenie to w prosty sposób umożliwia właśnie narzędzie PAM. Koncepcja i pierwsza implementacja PAM została opracowana dla systemu Solaris w 1995 [SS95], a kilka lat później przeniesiona na Linuksa [M01].

Celem ćwiczenia jest zrozumienie działania PAM i zastosowanie wybranych modułów w praktyce.

1. Pluggable Authentication Modules

System PAM to zestaw bibliotek (modułów, wtyczek), które są wykorzystywane do uwierzytelniania użytkowników w systemie. Odpowiednia biblioteka jest wykorzystywana gdy polityka uwierzytelniania dla danej aplikacji wywołuje procedurę uwierzytelniającą. Moduły określają też dodatkowe wymagania i zadania systemu uwierzytelniającego, jak np. ograniczenie czasu logowania do konkretnych godzin, określenie różnych źródeł danych o użytkownikach (baza LDAP, MySQL i inne).

Wszystkie aplikacje, które chcą realizować uwierzytelnianie poprzez system PAM muszą zostać do tego przystosowane. Muszą posiadać odwołania do biblioteki PAM, która jest odpowiedzialna za komunikację pomiędzy aplikacją a odpowiednim modułem. System PAM po otrzymaniu zgłoszenia od aplikacji czyta plik konfiguracyjny polityki, dotyczący danej aplikacji i wywołuje odpowiednie moduły implementujące wymagane mechanizmy bezpieczeństwa. W zależności od tego czy działanie modułu zakończy się powodzeniem, biblioteka zwróci odpowiednią odpowiedź aplikacji.

1.1 Komponenty PAM

Moduły systemu PAM mogą realizować funkcje z czterech obszarów polityki (*management groups*):

- **Zarządzanie uwierzytelnianiem (ang. *auth*)**

Moduły auth realizują dwa zadania. Najważniejszym zadaniem jest uwierzytelnianie użytkowników. Na polecenie modułu auth aplikacja pyta użytkownika o dane uwierzytelniające (login i hasło lub np. identyfikacja na podstawie klucza, głosu czy danych biometrycznych). Drugą funkcją modułu jest ustalenie uprawnień użytkownika, np. na podstawie przynależności do grup.

- **Zarządzanie kontami (ang. *account*)**

Moduły account odpowiadają za weryfikację, czy można podjąć próbę dostępu do konta użytkownika danych okolicznościach. Moduły te mogą weryfikować porę dnia, sposób dostępu (np. zdalny, lokalny), obciążenie systemu. Mogą również analizować ważność konta oraz reguły dotyczące wygaśnięcia hasła. Komponenty typu account nie mają związku z samym procesem uwierzytelniania.

- **Zarządzanie hasłami (ang. *password*)**

Moduły password odpowiadają za umożliwienie użytkownikowi zmiany hasła lub innej metody uwierzytelniania. Zajmują się również weryfikacją poprawności i złożoności zmienionych poświadczeń.

- **Zarządzanie sesjami (ang. *session*)**

W ramach komponentów session moduły PAM mogą zarządzać sesją użytkownika. Odpowiadają za ustalenie sesji oraz za „porządki" podczas jej zakończenia. Do elementów tworzenia sesji mogą należeć takie zadania jak ustawienie zmiennych środowiskowych, uruchomienie chroot czy ograniczenie dostępnych dla użytkownika zasobów.

Moduły session odpowiadają również za wyświetlanie powiadomień (jak np. motd).

Każdy z powyższych obszarów polityki może być definiowany w indywidualny sposób dla każdej aplikacji. Możliwości polityki są praktycznie nieograniczone, zależą wyłącznie od zastosowanych modułów PAM. Dodatkowo możliwe jest zaimplementowanie własnego modułu.

1.2 Wybrane moduły mechanizmu PAM

Poniższa lista zawiera przykładowe moduły dostępne w bibliotece PAM:

- access – moduł określający kto ma mieć dostęp do systemu oraz w jaki sposób może uzyskać dostęp,
- cracklib oraz pwquality – moduły sprawdzające jakość hasła użytkownika,
- pwhistory – moduł sprawdzający jak bardzo hasło różni się od poprzednich,
- anonymous access – moduł umożliwiający stworzenie anonimowego dostępu do serwera ftp,
- resource limits – określa limity wykorzystania systemu przez użytkownika, limitami takimi można objąć wykorzystanie pamięci operacyjnej dla pojedynczego procesu i dla wszystkich procesów użytkownika, maksymalna czas procesora, maksymalną liczbę logować użytkownika i wiele innych,
- radius session – uwierzytelniania użytkownika przy wykorzystaniu zdalnego serwera Radius,
- time control – określa dostęp użytkownika do systemu w zależności od czasu w którym próbuje on uzyskać dostęp,
- kerberos – uwierzytelnianie użytkownika przy wykorzystaniu systemu Kerberos,
- smart card – uwierzytelnianie użytkownika na podstawie karty chipowej,
- one-time password– uwierzytelnianie użytkownika na podstawie jednorazowych haseł,
- SQL database – uwierzytelnianie użytkownika na podstawie wpisów w bazie danych; istnieją wtyczki do większość baz danych,
- SecurID – uwierzytelnianie użytkownika poprzez tokeny (potrzebny jest do tego serwer, w którym tokeny będą zarejestrowane),
- voiceauth – uwierzytelnianie użytkownika na podstawie jego głosu,
- LDAP – *uwierzytelnianie użytkownika na podstawie wpisów w bazie LDAP.*

2. Wykorzystanie mechanizmu PAM w aplikacji

Poniżej przedstawiono przykładowy schemat wykorzystania PAM w aplikacji [BSD]:

```
/* inicjalizacja */
pamc.conv = &openpam_ttyconv;
pam_start("su", "user", &pam_conversation, &pam_handle);

/* ustawienie takich elementów interakcji jak hostname, username ... */
if ((pam_err = pam_set_item(pam_handle, PAM_OPCJA, wartosc))
       != PAM_SUCCESS)    pam_end(pam_handle, pam_err);
...

/* uwierzytelnienie */
if ((pam_err = pam_authenticate(pam_handle, 0)) != PAM_SUCCESS)
  pam_end(pam_handle, pam_err);

/* weryfikacja, czy nie potrzeba nowego hasła */
if ((pam_err = pam_acct_mgmt(pam_handle, 0))==PAM_NEW_AUTHTOK_REQD)
  pam_err = pam_chauthtok(pam_handle, PAM_CHANGE_EXPIRED_AUTHTOK);
if (pam_err != PAM_SUCCESS)
  pam_end(pam_handle, pam_err);

/* ustawienie żądanych poświadczeń */
if ((pam_err = pam_setcred(pam_handle, PAM_ESTABLISH_CRED))
       != PAM_SUCCESS)  pam_end(pam_handle, pam_err);

/* otwarcie sesji */
if ((pam_err = pam_open_session(pam_handle, 0)) != PAM_SUCCESS)
  pam_end(pam_handle, pam_err);

...

/* zmienne środowiskowe */
if ((pam_envlist = pam_getenvlist(pam_handle)) != NULL) {
       for (pam_env = pam_envlist; *pam_env != NULL; ++pam_env) {
             putenv(*pam_env);
             free(*pam_env);
       }
       free(pam_envlist);
 }

/* logika aplikacji */
...

/* zakończenie sesji i interakcji z PAM */
 pam_err = pam_close_session(pam_handle, 0);
```

Warto zwrócić uwagę na kilka szczegółów. Po pierwsze, od momentu ustalenia sesji, aplikacja działa z uprawnieniami użytkownika docelowego. Po drugie, aplikacja nie otrzymuje (i nie przekazuje do PAM) danych uwierzytelniających użytkownika. W wywołaniu `pam_start` przekazywany jest wskaźnik na strukturę `pam_conversation` – zawiera ona wskaźnik na funkcję, która stanowi interfejs między PAM a użytkownikiem. Funkcja ta może być wywołana przez PAM do następujących celów:

- wyświetlenie komunikatu zachęty,
- wyświetlenie komunikatu błędu,
- pobranie tekstu jawnego (np. login),
- pobranie tekstu ukrytego (np. hasło).

2.1 Moduł PAM

Przygotowanie modułu PAM wymaga zaimplementowania co najmniej jednej z poniższych funkcji:

- `pam_sm_authenticate()` – moduł auth,
- `pam_sm_acct_mgmt()` – moduł account,
- `pam_sm_chauthtok()` – moduł password,
- `pam_sm_open_session()` oraz `pam_sm_close_session()` – moduł session.

Aby dowiedzieć się jakie obszary polityki obsługuje dany moduł można skorzystać z jego podręcznika lub z polecenia:

```
nm --dynamic --defined-only nazwa_modułu.so
```

3. Konfiguracja PAM

Konfiguracji polityki PAM dokonuje się albo za pomocą pojedynczego pliku konfiguracyjnego `/etc/pam.conf` lub, według nowszych standardów, poprzez zestaw oddzielnych plików polityk w katalogu `/etc/pam.d/`. Obecność katalogu automatycznie wyłącza obsługę pojedynczego pliku polityki. Istnieje możliwość zawierania w sobie plików konfiguracyjnych za pomocą poleceń `include` i `substack`.

Polityka składa się z listy (tzw. stosu) reguł analizowanych w kolejności występowania w pliku. Pojedyncza reguła polityki wygląda następująco:

```
[nazwa aplikacji] komponent reakcja wtyczka parametry_wtyczki
```

W katalogu `/etc/pam.d/` nazwy plików konfiguracyjnych są zgodnie z nazwami aplikacji, więc nazwy te są pomijane w treści reguł zapisywanych w owych plikach.

Parametr `reakcja` określa sposób, w jaki system PAM powinien zinterpretować wynik działania wywołanego modułu. Istnieją dwa formaty zapisu pola reakcji – pełny oraz skrócony. Format pełny pozwala na określenie akcji dla każdej możliwej wartości zwróconej przez moduł w następujący sposób:

```
[wartość1=akcja1 wartość2=akcja2 … ]
```

Możliwe wartości wymienione są w podręczniku pam.d. Poniżej opisano dostępne akcje:

`ignore` – ignorowanie wyniku działania modułu,

`bad` – niepowodzenie całego stosu, ale dalsza kontynuacja jego przetwarzania,

`die` – niepowodzenie całego stosu i przerwanie przetwarzania,

`ok` – powodzenie całego stosu, o ile wcześniej nie wystąpiła akcja `bad`,

`done` – powodzenie całego stosu, o ile wcześniej nie wystąpiła akcja `bad` i przerwanie przetwarzania,

`N` (dodatnia liczba) – analogicznie do `ok` oraz pominięcie kolejnych N modułów w stosie,

`reset` – wyczyszczenie stanu stosu i kontynuacja od następnego modułu.

Format skrócony definiuje najczęstsze reakcje, odpowiednie dla większości reguł:

`requisite` ([success=ok new_authtok_reqd=ok ignore=ignore default=die])

`required` ([success=ok new_authtok_reqd=ok ignore=ignore default=bad])

`sufficient` ([success=done new_authtok_reqd=done default=ignore])

`optional` ([success=ok new_authtok_reqd=ok default=ignore]).

Poniżej przedstawiono przykładową politykę `/etc/pam.d/login` omówioną na wykładzie:

```
auth        requisite     pam_nologin.so
auth        requisite     pam_securetty.so
auth        sufficient    pam_rhosts_auth.so
auth        sufficient    pam_ldap.so
auth        required      pam_unix.so try_first_pass nullok
account     required      pam_unix.so
password    required      pam_cracklib.so minlen=8 dcredit=-2 retry=3
password    required      pam_unix.so use_authtok remember=4
session     required      pam_env.so
session     required      pam_syslog.so
session     required      pam_mail.so
session     optional      pam_limits.so
```

🕮 Literatura

[SS95] V.Samara i R.Schemersa, OSF RFC 86.0, "Unified Login with Pluggable Authentication Modules (PAM)", October 1995

[M01] PAM – draft specification https://www.kernel.org/pub/linux/libs/pam/pre/doc/draft-morgan-pam-08.txt

[Lin] Linux-PAM http://www.linux-pam.org/

[PAM] PAM Tutorial http://wpollock.com/AUnix2/PAM-Help.htm

[BSD] PAM Guide http://www.netbsd.org/docs/guide/en/chap-pam.html

🗒 Problemy do dyskusji

- Jakie są zalety mechanizmu PAM? Jakie wady?
- Czy aplikacje mogą pomijać politykę PAM podczas uwierzytelniania użytkowników? Jeśli tak to w jaki sposób?

💻 Zadania:

Kontrola logowania

1. Zidentyfikuj moduły PAM zdefiniowane dla demona sshd.
2. Wykorzystaj moduł pam_nologin.so. Przetestuj jego działanie.
3. Wycofaj wprowadzone zmiany.

💻 *Dla potrzeb dalszych ćwiczeń utwórz użytkowników* test *oraz* test2 *i przypisz ich do grupy* test_group.

4. Który moduł PAM odpowiada za ograniczanie dostępu użytkowników o określonych nazwach ze wskazanych lokalizacji?
5. Dodaj ten moduł do stosu demona sshd i wykorzystaj w zad. 6–8.

6. Zablokuj możliwość logowania przez ssh dla użytkownika test ze zdalnego systemu.
7. Wprowadź w tym module regułę domyślnej odmowy dostępu.
8. Zezwól sąsiadowi na dostęp przez SSH dla wszystkich użytkowników spoza grupy test_group.
9. Który moduł PAM odpowiada za zarządzanie czasem logowania użytkowników?
10. Zezwól użytkownikowi test na logowanie przez `ssh` we wszystkie robocze dni tygodnia z wyjątkiem poniedziałku, w wybranym 10-minutowym przedziale czasu (czas dobierz tak, by można zweryfikować efekt działania polecenia).

Kontrola haseł

11. Który moduł odpowiada za kontrolę złożoności haseł?
12. Stwórz regułę złożoności haseł dla polecenia `passwd`. Hasło musi posiadać co najmniej 2 cyfry, 2 litery małe, 2 litery wielkie, 2 inne znaki, nie może zawierać nazwy użytkownika oraz cztero- i więcej literowych sekwencji kolejnych znaków (np. 4567).
13. Przetestuj to ograniczenie.
14. Który moduł pozwala na weryfikację historii haseł?
15. Dodaj do uprzednio skonfigurowanych ograniczeń weryfikację, czy wprowadzone hasło jest różne od dwóch poprzednich. Zapewnij, by moduł nie pytał ponownie o podanie już raz wcześniej wprowadzonego hasła.
16. Sprawdź gdzie przechowywane są stare hasła? W jakiej formie?

Ograniczenie powłoki

17. Który moduł pozwala na zdefiniowanie ograniczeń środowiska użytkownika?
18. Ustaw maksymalny rozmiar pliku zrzutu dla grupy test_group na `unlimited` oraz maksymalną ilość procesów dla użytkownika test2 na 100.
19. Czym różnią się limity typu „hard” od „soft”?
20. Jak zweryfikujesz relację między ograniczeniami „hard” i „soft”? Skorzystaj z ustawień z zad. 18.

Uwaga! Pozostawienie błędnych ustawień w modułach PAM może utrudnić kolejnym grupom przeprowadzenie zajęć laboratoryjnych. Po zakończeniu zajęć koniecznie uruchom system Linux Lokalny Czysty.