Modularne systemy uwierzytelniania i kontroli dostępu do systemu operacyjnego: PAM

1. Wprowadzenie

Modularne systemy uwierzytelniania mają za zadanie ułatwić kontrolę dostępu do systemu operacyjnego. Kontrola to zarządzanie kontami użytkowników, a także ustawianie ograniczeń dla tych kont.

Mechanizm PAM jest szeroko stosowanym mechanizmem w systemach Linux i posiada bardzo dużą rozbudowę, poprzez rozbudowę należy rozumieć wiele dostępnych modułów rozszerzających.

Celem ćwiczenia jest zrozumienie mechanizmu PAM i zastosowanie kilku modułów w praktyce.

2. Informacje o mechanizmie PAM

System uwierzytelniania w systemie Linux wykorzystuje mechanizm PAM (Pluggable Authentication Modules – Dołączalne Wtyczki Uwierzytelniające). Implementację dla systemu Linux stworzył i cały czas rozwija Andrew G.Morgan.

System PAM to zestaw bibliotek i wtyczek, które są wykorzystywane do uwierzytelniania użytkowników w systemie. Biblioteka jest wykorzystywana przez aplikację, aby wywołać procedurę uwierzytelniającą. Wtyczki określają możliwości systemu uwierzytelniającego, możliwościami taki są ograniczenie czasu logowania do konkretnych godzin, określenie różnych źródeł danych o użytkownikach (na przykład baza LDAP, MySQL i inne).

Ogólne założenie uwierzytelniania w systemie Linux

Każdy użytkownik w systemie jest jednoznacznie identyfikowany poprzez identyfikator użytkownika (ang. User Identifier – UID), każdy użytkownik należy do conajmniej jednej grupy użytkowników, która również posiada unikalny identyfikator grupy (ang. Group Identifier – GID); oczywiście liczba grup do której należy użytkownik może być dowolnie długa.

Obiekty w systemie również mają przypisane UID i GID, dokładniej właściciela, którego UID jest przypisane plikowi i grupę – GID, do której należą użytkownicy danego obiektu. Oprócz właściciela i grupy wyróżniamy opcje dla pozostałych użytkowników, którzy nie kwalifikują się w żadnej z powyższych grup. Opcjami określanymi dla każdego obiektu i każdego poziomu dostępu są uprawnienia zaprezentowane w mechanizmach lokalnej kontroli dostępu.

System PAM

System uwierzytelniania PAM. Koncepcja i pierwsza implementacja dla systemu Solaris wykonana przez V.Samara i R.Schemersa w dokumencie *OSF RFC 86.0. "Unified Login with Pluggable Authentication Modules (PAM)"* [SS95].

System PAM składa się z czterech komponentów:

- zarządzanie uwierzytelnianiem,

- zarządzanie kontami,
- zarządzanie sesjami,
- zarządzania hasłami.

Każdy z powyższych komponentów może być ustawiony w indywidualny sposób dla różnych aplikacji. Możliwości zarządzania powyższymi komponentami są praktycznie nieograniczone, gdyż zależą od zastosowanych wtyczek systemu PAM. Liczba wtyczek dostarczanych z główną biblioteką PAM jest ograniczone, jednak nie jest problemem napisanie nowej wtyczki systemu PAM w celu stworzenia nowego zastosowania.

Wszystkie aplikacje, które chcą wykorzystywać uwierzytelniania poprzez system PAM muszą zostać do tego przystosowane. Muszą posiadać odwołania do biblioteki systemu PAM, która jest odpowiedzialna za komunikację pomiędzy aplikacją a systemem PAM. System PAM po otrzymaniu zgłoszenia od aplikacji czyta plik konfiguracyjny dotyczący danej aplikacji i wczytuje wszystkie wtyczki, które zostały tam wymienione. Wtyczki te mają rozpocząć proces uwierzytelniania użytkownika i jeśli proces się powiedzie (lub nie) biblioteka zwróci odpowiednią odpowiedź aplikacji.

Wybrane wtyczki mechanizmu PAM

- access – wtyczka określająca kto ma mieć dostęp do systemu oraz w jaki sposób może uzyskać dostęp,
- cracklib – wtyczka sprawdzająca siłę hasła użytkownika, który dokonuje zmiany hasła; wtyczka dodatkowo może sprawdzać czy hasło nie jest którymś z poprzednich lub jak bardzo różni się od poprzedniego,
- locking-out – wtyczka blokująca dostęp, zawsze zwraca nie poprawność logowania,
- anonymous access – wtyczka umożliwiająca stworzenie anonimowego dostępu do serwera ftp,
- resource limits – określa limity wykorzystania systemu przez użytkownika, limitami takimi można objąć wykorzystanie pamięci operacyjnej dla pojedyńczego 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 authentication – uwierzytelnianie użytkownika na podstawie jednorazowych haseł,
- SQL database based authentication – 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 są zarejestrowane,
- voiceauth – uwierzytelnianie użytkownika na podstawie jego głosu,
- LDAP – uwierzytelnianie użytkownika na podstawie wpisów w bazie LDAP,

Wpływ działania wtyczki na proces uwierzytelniania

- wymagana *(ang. required)* – wtyczka musi zwrócić sukces,
- wymagana *(ang. requisite)* – wtyczka musi zwrócić błąd,
- wystarczająca *(ang. sufficient)* – wtyczka powinna zwrócić sukces, jednak jej porażka nie oznacza nie przyznania dostępu,
- opcjonalna *(ang. optional)* – wtyczka może zwrócić dowolną wartość, jednak znaczenie może być przy przekazywaniu uprawnień do aplikacji.

Podsumowanie mechanizmu PAM

Możliwości systemu PAM są bardzo duże, umożliwiają ustawienie innych systemów uwierzytelniających różnym aplikacjom lub dla wszystkich aplikacji zastosowanie jednego systemu uwierzytelniania.

Oczywiście uwierzytelnianie to jedna z części potrzebna do działania, potrzebne są jeszcze uprawnienia do obiektów, listy kontroli dostępu zostały omówione wcześniej.

3. Zadania

Wykorzystać zaprezentowane powyżej wtyczki w celu:

- ograniczenia czasowego logowania się użytkownika *guest* do systemu do godzin 8:00 – 9:00 w poniedziałki (lub inny termin, którego weryfikację uda się przeprowadzić),
- ograniczenie dostępu do systemu poprzez zdalne logowanie (na przykład przy użyciu SSH) użytkownika *root* i zweryfikowanie poprawności działania,
- ustawienie weryfikacji haseł oraz zablokowanie ustawianie identycznego hasła jak poprzednie 2 – zweryfikować poprawność działania,
- ustawić wymuszanie limitów zasobowych (polecenie ulimit) dla użytkowników grupy *guest* i zweryfikować poprawność tego wymuszania,
- ustawienie więzienia (chroot) dla użytkownika *guest* i sprawdzenie czy działa poprawnie,
- ustawić system do wykonania uwierzytelniania przy użyciu zdalnego źródła użytkowników, na przykład usługi katalogowej LDAP, bazy danych MySQL.

4. Problemy do dyskusji

- czy mechanizm PAM posiada wady, jakie?
- jakie są zalety mechanizmu PAM?
- czy mechanizm PAM jest skalowalny?
- dlaczego mechanizm PAM stał się podstawowym systemem uwierzytelniania w systemie Linux?
- czy aplikacje mogą pomijać mechanizm PAM podczas uwierzytelniania użytkowników? jeśli tak to w jaki sposób? oraz czy jest to słuszne podejście?
- czy istnieją inne rozwiązania uwierzytelniania w systemach Linux/Unix oraz czy dają lepsze/gorsze możliwości uwierzytelniania?

5. Bibliografia

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

[PAM] Linux-PAM http://www.kernel.org/pub/linux/libs/pam/