

# PAM

Pluggable Authentication Modules

# Fatto

E' responsabilità degli amministratori di sistema stabilire le politiche di sicurezza per le applicazioni che richiedono l'autenticazione dell'utente

# Problema



L'amministratore di un sistema Linux vuole modificare i meccanismi di autenticazione associati con un'applicazione ( ad es. login): deve necessariamente riscrivere il codice dell'applicazione e ricompilarlo?

Prima di PAM non si poteva fare niente di meglio



# Prima di PAM

- Prima di PAM ogni applicazione implementava i meccanismi di autenticazione nel proprio codice
- Ad esempio quando il servizio login doveva identificare un utente...



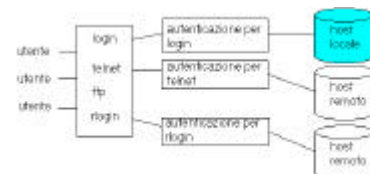
# Prima di PAM

- ...richiamava una sub-routine implementata nel codice...



# Prima di PAM

- ... che si occupava del processo di autenticazione dell'utente
- Anche le altre applicazioni utilizzavano lo stesso meccanismo (telnet, ftp, ...)



## Scopo di PAM

Lo scopo di PAM è separare la scelta dei meccanismi di autenticazione dallo sviluppo di software che richiede servizi di autenticazione

7

## Scopo di PAM

PAM fornisce :

- Un modo per sviluppare programmi indipendenti dallo schema di autenticazione (applicazione PAM-aware)
- Un modo per permettere agli amministratori di sistema di stabilire una politica di sicurezza senza modificare i programmi che effettuano l'autenticazione

8

## Cos'è PAM?

PAM è un complesso di librerie condivise che forniscono all'applicazione un'interfaccia astratta con un generico sistema di autenticazione e una serie di strumenti per interagire con essa



9

## Meccanismo di PAM

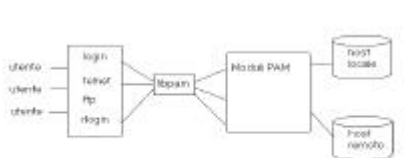
- L'applicazione richiede i servizi alla libreria libpam



10

## Meccanismo di PAM

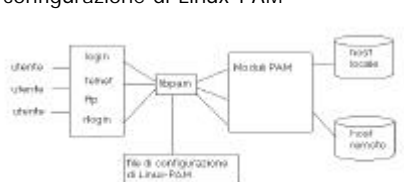
- libpam carica i moduli che eseguono i servizi richiesti dall'applicazione



11

## Meccanismo di PAM

- L'amministratore di sistema indica quale insieme di moduli utilizzare tramite il file di configurazione di Linux-PAM



12

## I moduli

- ❑ Un modulo è un file oggetto caricabile dinamicamente
- ❑ E' compito della libreria libpam chiamare i moduli

13

## I moduli

- ❑ Un modulo implementa i meccanismi del processo di autenticazione dell'utente
- ❑ L'applicazione richiede un servizio al modulo tramite chiamate alle opportune funzioni di libpam
- ❑ Un modulo può fornire quattro tipi di servizi differenti: authentication, password, account e session

14

## Il file di configurazione

L'amministratore può indicare quali moduli utilizzare per il processo di identificazione tramite un file di configurazione per ogni applicazione PAM-aware

15

## Nel file di configurazione

nome_del_servizio	tipo_di_modulo	flag_di_controllo	path_del_modulo	argomenti_per_il_modulo
-------------------	----------------	-------------------	-----------------	-------------------------

Una linea del file di configurazione contiene i seguenti campi:

- ❑ nome\_del\_servizio (es. login,telnet)
- ❑ tipo\_di\_modulo (auth,account,...)
- ❑ flag\_di\_controllo (required,...)
- ❑ path\_del\_modulo (/lib/security/md.so)
- ❑ argomenti\_per\_il\_modulo

16

## Esempio di file di configurazione

Questo file di configurazione indica a libpam di autenticare l'utente tramite il meccanismo delle password standard di Unix con il modulo pam\_unix\_auth.so

```
login auth required /usr/lib/security/pam_unix_auth.so
login account required /usr/lib/security/pam_unix_acct.so
login password required /usr/lib/security/pam_unix_passwd.so
login session required /usr/lib/security/pam_unix_session.so
```

17

## Il file di configurazione

```
login auth required /usr/lib/security/pam_unix_auth.so
login account required /usr/lib/security/pam_unix_acct.so
login password required /usr/lib/security/pam_unix_passwd.so
login session required /usr/lib/security/pam_unix_session.so
```

- ❑ Si può effettuare l'upgrade di un modulo semplicemente modificando una linea nel file

```
login auth required /usr/lib/security/pam_unix_auth.so
login account required /usr/lib/security/pam_unix_acct.so
login password required /usr/lib/security/pam_unix_passwd.so
login session required /usr/lib/security/pam_unix_session.so
```

18

## Il file di configurazione

- ❑ In questo file di configurazione si utilizza il nuovo modulo pam\_unix per effettuare l'autenticazione
- ❑ login non deve essere modificata

```
login auth required /usr/lib/security/pam_unix.so
login account required /usr/lib/security/pam_unix_nocf.so
login password required /usr/lib/security/pam_unix_passwd.so
login session required /usr/lib/security/pam_unix_session.so
```

19

## File di configurazione

- ❑ Il file di configurazione è /etc/pam.conf
- ❑ Un'alternativa al file pam.conf è un file di configurazione distinto per ogni servizio nella directory /etc/pam.d/
- ❑ Nella directory /etc/pam.d/ ogni file di configurazione ha lo stesso nome dell'applicazione a cui è associato

20

## Nome del servizio

Linea del file di configurazione				
nome_del_servizio	tipo_d_modulo	flag_di_controllo	path_d_modulo	argomenti_per_il_modulo

- ❑ Il campo nome\_del\_servizio è il nome del servizio a cui è associata l'entrata
- ❑ Ad esempio, "ftpd", "rlogind", "su" etc

21

## Nome del servizio

- ❑ C'è un nome\_del\_servizio speciale, riservato per definire un meccanismo di autenticazione di default: OTHER
- ❑ Per tutte le applicazioni PAM-aware non presenti nel file di configurazione vengono eseguiti i moduli corrispondenti alle entrate OTHER

22

## Entrate OTHER

Questo file di configurazione nega l'accesso a qualsiasi applicazione che non sia login con il modulo pam\_deny.so

```
login auth required /usr/lib/security/pam_unix.so
login account required /usr/lib/security/pam_unix.so
login password required /usr/lib/security/pam_unix.so
login session required /usr/lib/security/pam_unix.so
OTHER auth required /usr/lib/security/pam_deny.so
OTHER account required /usr/lib/security/pam_deny.so
OTHER password required /usr/lib/security/pam_deny.so
OTHER session required /usr/lib/security/pam_deny.so
```

23

## Tipo di modulo

Linea del file di configurazione				
nome_del_servizio	tipo_d_modulo	flag_di_controllo	path_d_modulo	argomenti_per_il_modulo

- ❑ Ci sono quattro tipi di moduli disponibili: auth, account, password e session
- ❑ Ognuno fornisce un diverso insieme di servizi

24

## Modulo di autenticazione

Ci sono quattro tipi di moduli disponibili: auth, account, password e session

I moduli **auth** forniscono:

- ❑ Servizi per l'autenticazione dell'utente
- ❑ Servizi per modificare le credenziali dell'utente

25

## Moduli gestione account

Ci sono quattro tipi di moduli disponibili: auth, account, password e session

- ❑ I moduli **account** forniscono servizi per la verifica dell'account
- ❑ Ad esempio verificano:
  - ❑ Che l'account dell'utente non sia scaduta
  - ❑ Che all'utente sia concesso accedere alla data e ora attuale
- ❑ Etc...



26

## Moduli gestione della password

Ci sono quattro tipi di moduli disponibili: auth, account, password e session

- ❑ I moduli **password** forniscono servizi per la modifica dei token di autenticazione
- ❑ Servono per modificare la password

27

## Moduli gestione della sessione

Ci sono quattro tipi di moduli disponibili: auth, account, password e session

- ❑ I moduli **session** sono usati dopo l'autenticazione per rendere possibile l'uso dell'account all'utente
- ❑ Ad esempio rendendogli disponibile la home directory, la casella postale etc...



28

## Flag di controllo

Linea del file di configurazione

nome_del_servizio	tipo_del_modulo	flag_di_controllo	path_del_modulo	argomenti_per_il_modulo
-------------------	-----------------	-------------------	-----------------	-------------------------

- ❑ Più moduli possono essere utilizzati per eseguire lo stesso compito
- ❑ Ad esempio più moduli possono essere utilizzati per l'autenticazione

```
ftpd auth sufficient /usr/lib/security/pam_ftp.so  
ftpd auth required /usr/lib/security/pam_unix_auth.so  
ftpd auth required /usr/lib/security/pam_listfile.so
```

29

## Flag di controllo

- ❑ I moduli dello stesso tipo sono inseriti in uno stack e vengono eseguiti nell'ordine in cui compaiono nel file

```
ftpd auth sufficient /usr/lib/security/pam_ftp.so  
ftpd auth required /usr/lib/security/pam_unix_auth.so  
ftpd auth required /usr/lib/security/pam_listfile.so
```

Stack dei moduli

1. pam\_ftp.so
2. pam\_unix\_auth.so
3. pam\_listfile.so

30

## Flag di controllo

- ❑ Le funzioni fornite dai moduli che implementano i servizi di identificazione restituiscono un unico valore, combinazione dei valori di ritorno di ogni modulo
- ❑ Il campo `flag_di_controllo` determina l'importanza di ogni modulo per il successo o il fallimento globale

31

## Flag di controllo

I flag di controllo sono i seguenti:  
`required`, `requisite`, `sufficient` e `optional`

- ❑ `required`: indica che il successo del modulo è indispensabile per il successo globale. Il fallimento non sarà visibile all'utente finché tutti i moduli rimanenti dello stesso tipo non siano stati eseguiti

32

## Flag di controllo

I flag di controllo sono i seguenti:  
`required`, `requisite`, `sufficient` e `optional`

- ❑ `requisite`: come `required` ma il controllo viene restituito subito all'applicazione

33

## Flag di controllo

I flag di controllo sono i seguenti:  
`required`, `requisite`, `sufficient` e `optional`

- ❑ `sufficient`: il successo di questo modulo è ritenuto sufficiente per il successo dell'autenticazione

34

## Flag di controllo

I flag di controllo sono i seguenti:  
`required`, `requisite`, `sufficient` e `optional`

- ❑ `optional`: il fallimento o il successo di questo modulo non influiscono sul processo di autenticazione

35

## Flag di controllo

Il valore di ritorno all'applicazione è:

- ❑ `PAM_SUCCESS`
- oppure
- ❑ Il valore di ritorno del primo modulo `required` o `requisite` che ha fallito

36

## Esempio di uso del flag di controllo

```
ftpd auth sufficient /usr/lib/security/pam_ftp.so
ftpd auth required /usr/lib/security/pam_unix_auth.so
ftpd auth required /usr/lib/security/pam_listfile.so
```

Con questo file di configurazione libpam effettuerà le seguenti azioni:

- Se il modulo pam\_ftp.so restituisce PAM\_SUCCESS libpam restituisce subito il controllo all'applicazione (restituendo successo)

37

## Esempio di uso del flag di controllo

```
ftpd auth sufficient /usr/lib/security/pam_ftp.so
ftpd auth required /usr/lib/security/pam_unix_auth.so
ftpd auth required /usr/lib/security/pam_listfile.so
```

- Altrimenti esegue i due moduli successivi
- Il successo di entrambi i moduli è richiesto affinché l'autenticazione riesca
- Un fallimento del modulo pam\_unix\_auth.so non sarà visibile all'utente fino alla terminazione del modulo pam\_listfile.so

38

## Path del modulo

Linea del file di configurazione				
nome_del_servizio	tipo_di_modulo	flag_di_controllo	path_del_modulo	argomenti_per_il_modulo

Il path\_del\_modulo indica il nome del modulo da caricare e la directory in cui si trova (solitamente è /lib/security/)

39

## Argomenti per il modulo

Linea del file di configurazione				
nome_del_servizio	tipo_di_modulo	flag_di_controllo	path_del_modulo	argomenti_per_il_modulo

- Il campo argomenti\_per\_il\_modulo è una lista di argomenti passati al modulo quando è chiamato
- Sono opzionali e sono specifici di ogni modulo
- Gli argomenti invalidi sono ignorati

40

## Applicazioni PAM-aware

- Per rendere un'applicazione PAM-aware si include il file security/pam\_appl.h nel codice sorgente C
- libpam mette a disposizione sei funzioni da utilizzare nel codice C dell'applicazione
- Queste funzioni forniscono l'interfaccia con il generico sistema di autenticazione
- Sono tutto ciò di cui l'applicazione necessita per effettuare i servizi di identificazione dell'utente

41

## Applicazioni PAM-aware

Le sei funzioni sono:

- pam\_authenticate() autentica l'utente
- pam\_setcred() modifica le credenziali dell'utente
- pam\_acct\_mgmt() controlla se l'accesso all'utente è permesso
- pam\_open\_session() inizia la sessione
- pam\_close\_session() termina la sessione
- pam\_chauthtok() modifica il token di autenticazione

42

## Moduli

- Un modulo può fornire quattro tipi di servizi differenti: authentication, password, account e session
- Per essere correttamente definito, un modulo deve definire tutte le funzioni di almeno uno dei quattro tipi di servizio
- Un singolo modulo può contenere le funzioni necessarie per tutti i quattro tipi di servizio

43

## Moduli

Nel codice C:

- Un modulo **auth** deve definire le funzioni `pam_sm_authenticate()` e `pam_sm_setcred()`
- Un modulo **account** deve definire la funzione `pam_sm_acct_mgmt()`
- Un modulo **password** deve definire la funzione `pam_sm_chauthtok()`
- Un modulo **session** deve definire le funzioni `pam_sm_open_session()` e `pam_sm_close_session()`

44

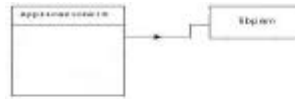
## Interazione applicazione-libreria-modulo

- L'applicazione di servizio X deve autenticare l'utente



## Interazione applicazione-libreria-modulo

- L'applicazione effettua una chiamata alla funzione `pam_authenticate()` della libreria `libpam`



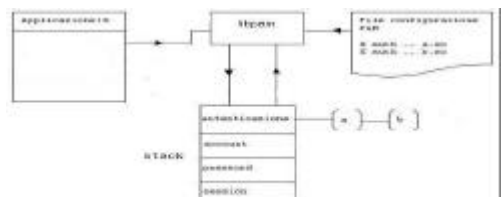
## Interazione applicazione-libreria-modulo

- La libreria legge il file di configurazione



## Interazione applicazione-libreria-modulo

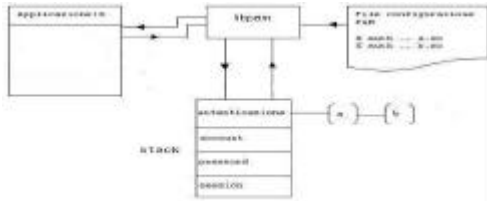
- La libreria chiama la funzione di autenticazione di ognuno dei moduli di tipo `auth` elencati nel file e li inserisce nello stack





## Interazione applicazione-libreria-modulo

- Il valore di ritorno della funzione `pam_authenticate()` indica all'applicazione se l'autenticazione è riuscita



## Esempio di applicazione

- La seguente applicazione richiede di autenticare un utente delegando la scelta del meccanismo di autenticazione all'amministratore di sistema
- L'applicazione resta all'oscuro di come l'utente viene autenticato, ma, a seconda del valore di ritorno della funzione `pam_authenticate()`, può decidere di concedere l'accesso all'utente o meno.

50

## Esempio di applicazione

```
#include <security/pam_appl.h>
#include <security/pam_misc.h>
#include <stdio.h>

static struct pam_conv conv = {
    misc_conv
    NULL};

int main(int argc, char *argv[])
{
    pam_handle_t *pamh=NULL;
    int ritorno;
    const char *utente=NULL;
    if(argc == 2)
        utente = argv[1];
    if(argc > 2) {
        printf("Numero argomenti errati. Uso: ver_utente [nome utente]\n");
        exit(1);
    }
}
```

51

## Esempio di applicazione

```
ritorno = pam_start("ver_utente",utente, &conv, &pamh); // Legge il file di configurazione
if (ritorno == PAM_SUCCESS) ritorno = pam_authenticate(pamh, 0); // Richiede l'autenticazione
else printf("%s\n", pam_strerror(pamh, ritorno));

if(ritorno==PAM_USER_UNKNOWN) printf("Utente inesistente\n");

if (ritorno == PAM_SUCCESS) printf("L'utente è stato autenticato\n"); // Controlla il valore restituito da
else printf("L'utente non è stato autenticato\n"); // pam_authenticate() per verificare se
// l'autenticazione è riuscita

if (pam_end(pamh,ritorno) != PAM_SUCCESS) {
    pamh = NULL; // Fine iterazione con la libreria
    printf("%s\n", pam_strerror(pamh, ritorno));
    exit(1);
}
```

52

## Esempio di modulo

Il seguente modulo di autenticazione effettua due controlli:

- Si assicura che l'utente sia elencato nel file delle password
- Richiede una password all'utente e verifica che la sua lunghezza sia almeno di otto caratteri

Come richiesto dalle convenzioni della libreria PAM, il modulo definisce le due funzioni `pam_sm_authenticate()` e `pam_sm_setcred()`

53

## Esempio di modulo

```
#include <string.h>
#include <stdio.h>
#include <pwd.h>
#define PAM_SM_AUTH
#include <security/pam_appl.h>
#include <security/pam_misc.h>

// Questa è la funzione che libpam chiama quando un'applicazione richiede l'autenticazione di un utente
PAM_EXTERN int pam_sm_authenticate(pam_handle_t *pamh, int flags, int argc, const char **argv)
{
    int ritorno;
    const char *utente;
    const struct pam_conv *conv;
    struct passwd *pw;
    struct pam_message *messaggi[1];
    const struct pam_message *mes[1];
    struct pam_response *risposta;

    ritorno = pam_get_user(pamh, &utente, "Inserire nome utente: "); // Ottiene il nome dell'utente
    pw = getpwnam(utente); // Verifica se l'utente è presente nel file delle password

    if (!pw) return(PAM_USER_UNKNOWN); // Ritorna utente sconosciuto se non è presente nel file password
}
```

54

## Esempio di modulo

```
ritorno = pam_get_item(pamh,PAM_CONV.(const void **) &conv);
mes[0] = messaggio;
messaggio[0].msg_style=PAM_PROMPT_ECHO_OFF;
messaggio[0].msg="Inserire la password\n";
conv->conv(1,(const struct pam_message **) mes,&risposta,NULL); // Richiedi la password all'utente

if(strlen(risposta->resp)<8) return PAM_AUTH_ERR; //Verifica che la password è almeno di otto caratteri
else return PAM_SUCCESS;
}

//Questa è la funzione che libpam chiama quando un'applicazione richiede di modificare le credenziali dell'utente
PAM_EXTERN int pam_sm_setcred(pam_handle_t *pamh, int flags,
{
return PAM_SUCCESS;
}
```

55

## Presentazione a cura di

- Melucci Giovanni Matr. 56/00588
- Ullo Nicola Matr. 56/00552

56