sicurezza dei sistemi unix e linux

sommario

- controllo di accesso
- sicurezza nel filesystem
- autenticazione e login
- hardening
- log management e auditing

unix – controllo di accesso

controllo di accesso in Unix

- un utente opera sulle risorse tramite i “suoi processi”
  - a partire dalla shell che usa per dare comandi
- in Unix il controllo di accesso basato su **credenziali** associate ai processi
- i processi accedono alle risorse tramite il kernel (system call)
- il controllo di accesso viene eseguito dal kernel

credenziali di un processo unix

- a ciascun processo unix è associato…
  - l'identificativo dell'**utente** "**reale**" che ha lanciato il processo (**UID**)
  - l'identificativio dell'**utente** "**effettivo**" che si deve considerare per l'accesso a servizi e risorse forniti dal kernel. (**EUID**)
  - l'identificativo del **gruppo** "**reale**" dell'utente che ha lanciato il processo (**GID**)
  - l'identificativo del **gruppo** "**effettivo**" che si deve considerare per l'accesso a servizi e risorse forniti dal kernel. (**EGID**)
  - supplementary group list

processi privilegiati e non

- processi privilegiati (EUID=0, cioè root)
  - il kernel non applica alcun controllo
- processi non priviliegiati (EUID?0)
  - accesso ristretto alle risorse e ai servizi offerti dal kernel
- per quasi tutti i processi UID=EUID e GID=EGID ma non sempre...

diritti dei processi non privilegiati

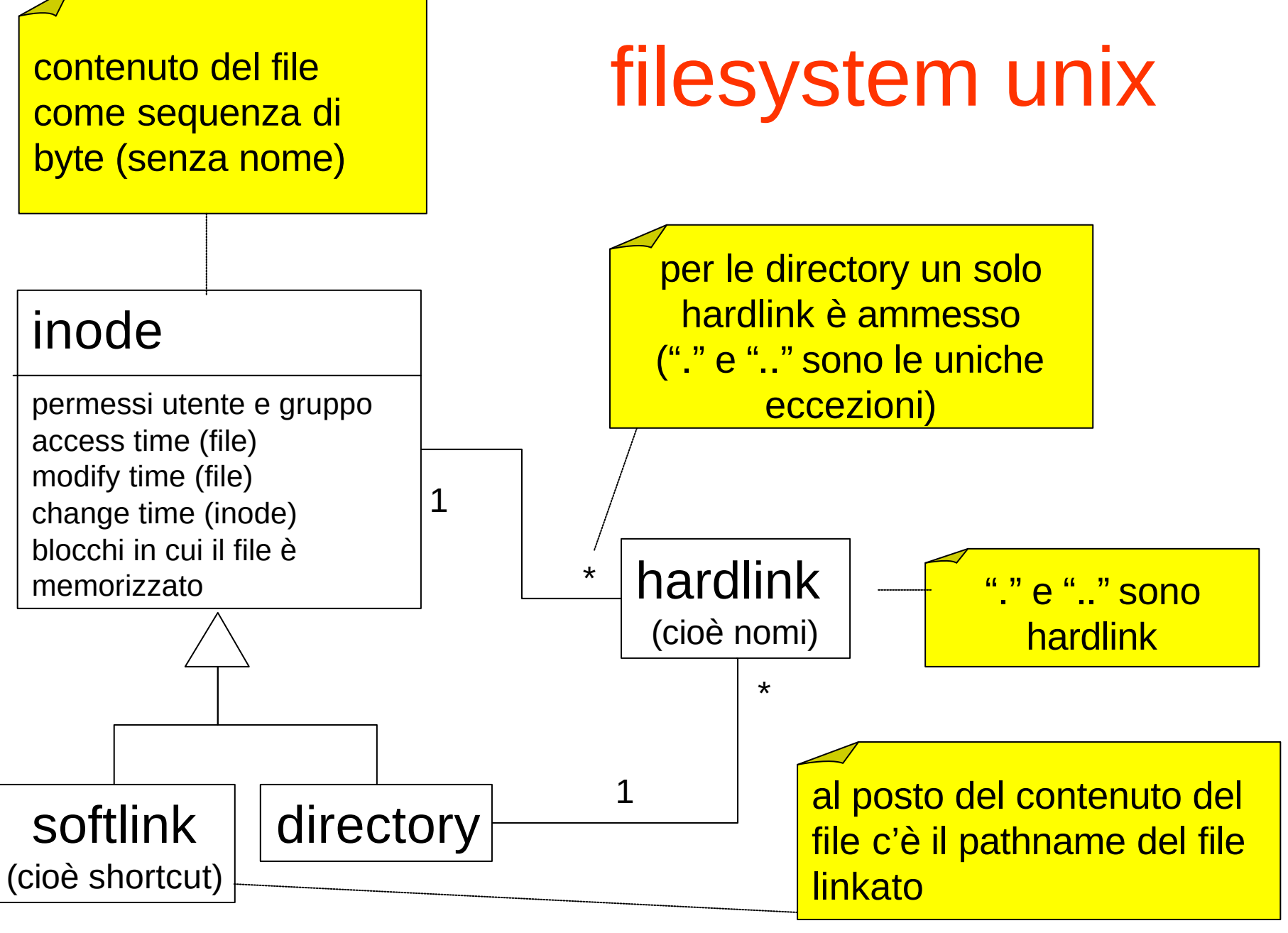
- “l'utenza” è identificata da EUID?0
- processi
  - kill o trace su processi dello stesso utente
  - comprende altri segnali come stop, cont, ecc.
- filesystem
  - **creare e cancellare file (links) in directory in accordo con i permessi configurati per la directory**
  - **aprire** file in accordo con i permessi configurati per il file
    - per qualsiasi operazione su un file gia' aperto il controllo di accesso è solo contro la “modalità di apertura” e non contro i permessi del file
  - **cambiare permessi** dei “propri” file/directory
  - **cambiare proprietario** dei “propri” file/directory
- rete
  - usare socket regolari (tcp, udp, unix, **no raw socket**)
  - usare porte >=1024

diritti dei processi privilegiati

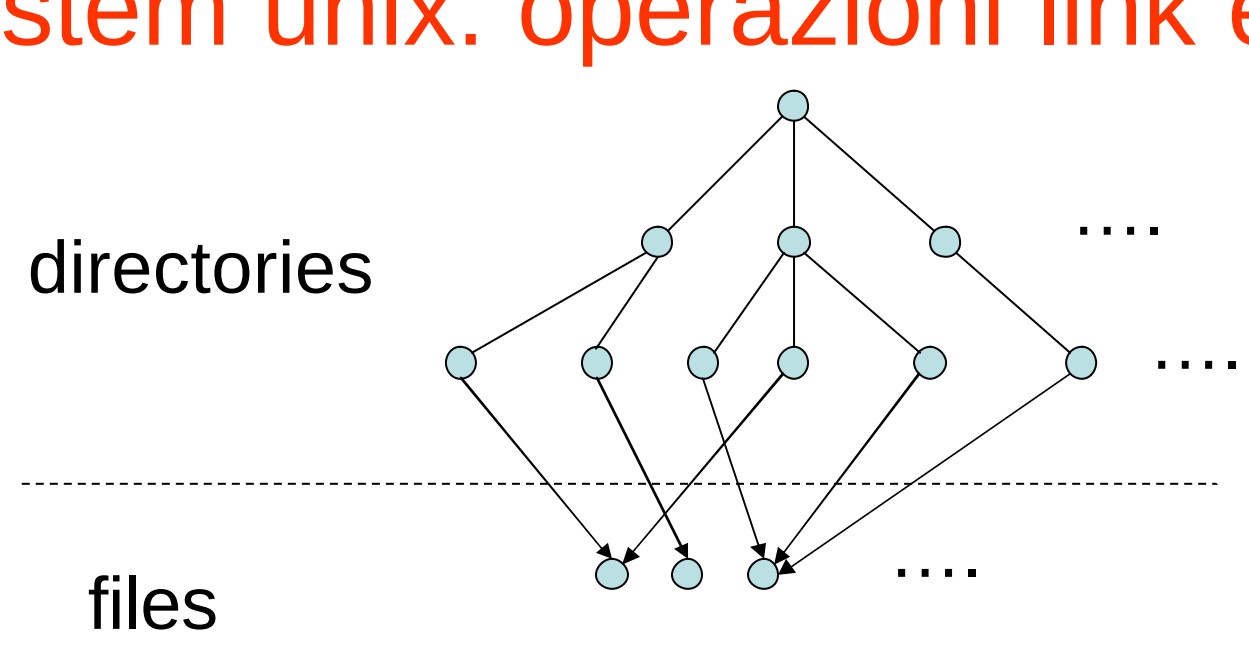
- i processi privilegiati (EUID=0) possono fare “tutto”, ecco alcuni esempi…
- processi
  - qualsiasi segnale su qualsiasi processo
  - renice
  - cambio dei propri UID, EUID, GID, EGID (necessario per login)
- rete
  - rete in modalita' promiscua e generazione di pacchetti qualisasi
  - qualisiasi socket (anche raw)
  - amministrazione: interfacce, tabella di routing
  - binding di well-known ports <1024
- **filesystem**
  - come per EUID?0 ma...
  - **cambiare permessi** e **proprietario** di tutti i file
- varie
  - (u)mount, quota, swap
  - reboot, shutdown, chroot, kernel modules, system clock

unix – sicurezza nel filesystem

filesystem unix



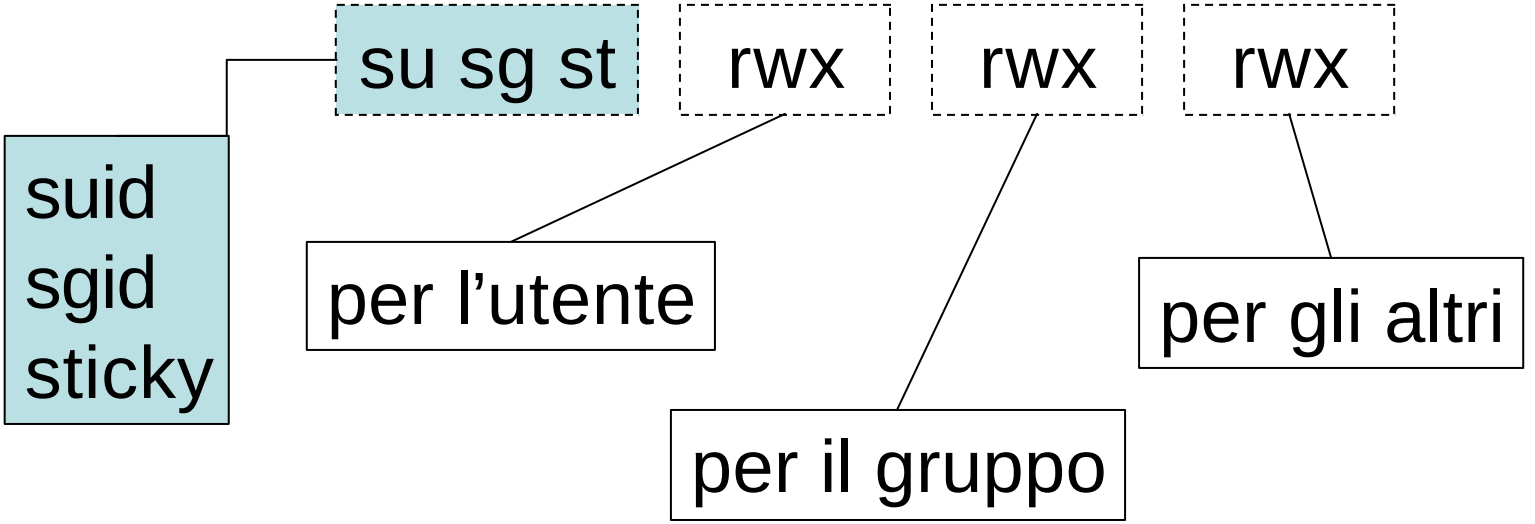
filesystem unix: operazioni link e unlink



- link(srg,dest): creazione di un hardlink dest che punta allo stesso inode di srg
- unlink(x): rimozione di un hardlink x
  - un inode viene rimosso quando non ha hardlink

filesystem unix: permessi

- permessi sugli inode regolari (file)
  - read (r), write (w), execute (x)
- permessi sulle inode directory
  - read (r), write (w), **search** (x)
- gli inode softlink non hanno permessi in ext3 (linux)
- a ciascun inode sono associati dei flag
  - leggermente diversi tra le varie versioni di unix filesystem
  - in Linux (ext2 o ext3):



filesystem unix: chmod

- chmod <*chi*> +/-/= <*cosa*> pippo.txt
  - chi: u=user, g=gruppo, o=other
  - chmod ug+rw pippo.txt (set read e write per user e group
  - chmod o-x pippo.txt (unset execution per gli altri)
  - chmod ug=r (set r per user e group e unset il resto)
- sinstassi “ottale” ancora usata
  - “chmod 660 pippo.txt” configura rw- rw- ---

filesystem unix: operazioni e permessi

- tutte le operazioni (syscall) sui file e sulle directory hanno come parametri dei pathname…
  - richiedono **premesso search** su tutte le directory nominate in tutti i pathname passati come parametro (compresa la directory corrente per pathname relativi)…
  - …e in più richiedono permessi specifici
- permessi specifici richiesti da operazioni su files
  - open create: w sulla directory (permessi del nuovo file stabiliti da parametro e umask)
  - open read: r sul file
  - open write append truncate: w sul file
  - execute: x sul file
  - link: w sulla directory destinazione
  - unlink: w sulla directory
- permessi specifici richiesti da operazioni su files e directory
  - chmod: uid del processo uguale a uid del inode
  - chown: uid del processo uguale a uid del inode
  - stat: -
- permessi specifici richiesti da operazioni su directory
  - mkdir: w sulla directory contenitore
  - rmdir: w sulla directory contenitore
  - readdir: r sulla directory da leggere

set-user-id bit (suid)

- molti comandi che gli utenti eseguono normalmente hanno bisogno dei privilegi di “root” per funzionare correttamente
  - ad esempio: ping, login, su, mount, umount, ping, chsh, passwd, at, gpg, kppp, sudo, cdrecord, exim4, ppp, e molti altri!
- tali comandi hanno il bit “set user id” settato

```
pizzonia@pisolo$ cd /bin
pizzonia@pisolo$ ls -l ping
-rwsr-xr-x  1 root root 30764 Dec 22  2003 ping
```

- **un eseguibile con tale bit settato viene eseguito con EUID pari al proprietario del file indipendentemente da chi ne ha richiesto l’esecuzione**

set-group-id bit (sgid)

- il bit set-group-id si comporta in maniera analoga per l'EGID

```
-rwxr-sr-x  1 root tty     /usr/bin/wall
-rwxr-sr-x  1 root crontab   /usr/bin/crontab
```

- **usando set-uid e set-gid non necessariamente si “diventa root”, può bastare un utente o un gruppo che è proprietario di certi file necessari all'applicazione in questione**
- **questo aumenta la sicurezza del sistema**

unix - autenticazione e login

utenze UNIX

- utenti in /etc/passwd
  - world readable
- gruppi in /etc/group
  - contiene anche la lista degli utenti appartenenti a ciascun gruppo
  - world readable
- passwords e altro in /etc/shadow e /etc/gshadow
  - leggibili solo a root

contenuto di /etc/passwd

- Login name
- Optional encrypted password
  - insicuro!!!!
  - vedi /etc/shadow
- Numerical user ID (root ha UID=0)
- Numerical group ID
- User name or comment field
- User home directory (es. /home/pizzonia)
- User command interpreter (es. /bin/bash)

contenuto di /etc/group

- group_name
- password (per il group managment)
  - the (encrypted) group password. If this field is empty, no password is needed.
- the numerical group ID
- user_list
  - all the group member's user names, separated by commas.

contenuto di /etc/shadow

- Login name (foreign key /etc/passwd)
- Encrypted password
- Days since Jan 1, 1970 that password was last changed
- Days before password may be changed
- Days after which password must be changed
- Days before password is to expire that user is warned
- Days after password expires that account is disabled
- Days since Jan 1, 1970 that account is disabled
- A reserved field

comandi

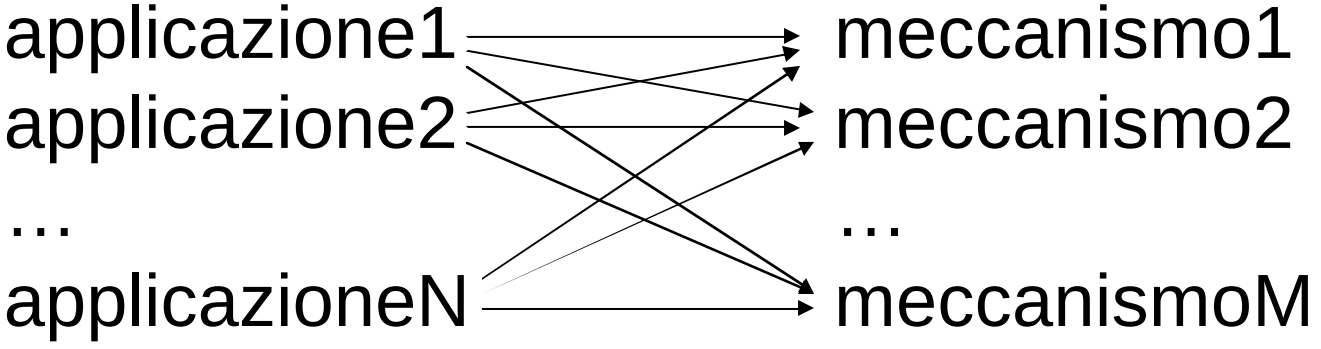
- l’editing a mano dei file passwd groups ecc. è possibile
  - …ma sconsigliato
  - bisogna rispettare rigidamente il formato dei file
    - gli amministratori smaliziati lo fanno
- comandi previsti per la gestione degli utenti e dei gruppi
  - useradd o adduser
  - userdel o deluser
  - usermod
  - groupadd o addgroup
  - groupdel o delgroup
  - groupmod
  - passwd
  - gpasswd

flessibilità nell'autenticazione

- molte sono le politiche e i meccanismi possibili
  - es. autenticazione locale (/etc/passwd)
  - es. autenticazione centralizzata (radius, active directory, ldap, nis)
  - es. diverse politiche: qualcosa che ho/so/sono, dove sono
- molti sono i programmi che richiedono funzioni di autenticazione
  - es. programmi di sistema: atd, chfn, chsh, cron, cupsys, cvs, kcheckpass, kdm, kdm-np, kscreensaver, libcupsys2, login, passwd, ppp, samba, ssh, su, sudo, telnetd, xdm, xscreensaver
  - ma anche applicativi: mysql, apache, ecc.

flessibilità nell'autenticazione

- cosa succede se vogliamo cambiare la politica
  - tutti i programmi che autenticano devono essere ricompilati con adeguato supporto
  - improponibile
- per un piena flessibilità N·M implementazioni



soluzione ingegneristica: PAM

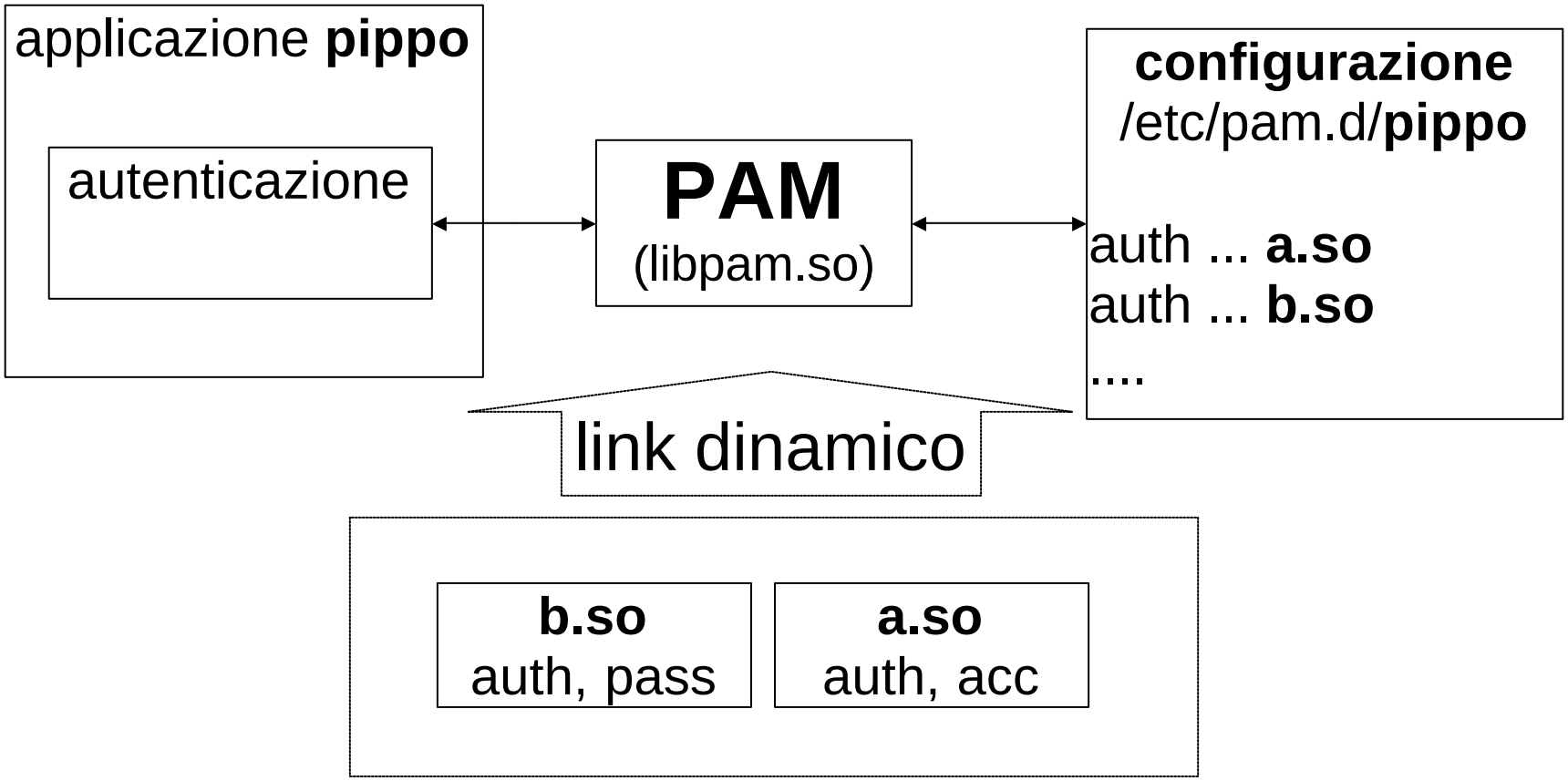
- fattorizzazione della funzionalità dalle applicazioni
  - libreria condivisa
- configurabilità della libreria
- solo M implementazioni necessarie

- Pluggable Authentication Modules (PAM)
  - diffuso nel mondo unix
  - windows non ha una soluzione altrettanto flessibile

PAM: 4 servizi

- **auth**
  - autentica l'utente (normalmente tramite password ma non necessariamente!)
  - può assegnare delle credenziali al processo
    - non assegna UID e GID, usato per esempio per i tiket kerberos
- **account.** garantisce l'accesso per tutto ciò che non riguarda l'autenticazione. Esempio: accessi basati sul'ora, risorse di sistema, utente locale o remoto, ecc.
- **session.** logging o altre attività in apertura o chiusura di sessione.
- **password.** gestisce l'aggiornamento delle password

PAM: architettura



login con password

il login con password prevede 4 fasi

1. richiesta all'utente delle credenziali
   - es. username e password
2. verifica nel db degli utenti della correttezza delle credenziali
3. cambio di utenza
   - setuid, setgid, seteuid, setegid, setreuid,...
4. execve della shell

PAM

login con password

esecuzione

testuale		grafico
getty o telnetd **`login:`** exec “login <user>”	**ssh** conosce già l’utente (passato tramite protocollo) pam_auth **`password:`**	**xdm o kdm** pam_auth **`username` ……** **`password` …….**
login pam_auth **`password:`**		
login (pam ha verificato le credenziali fornite dall’utente) setuid, setgid, seteuid, setegid, ecc	**ssh** (pam ha verificato le credenziali fornite dall’utente) setuid, setgid, seteuid, setegid, ecc	**xdm o kdm** (pam ha verificato le credenziali fornite dall’utente) setuid, setgid, seteuid, setegid, ecc
login exec shell	**ssh** exec shell	**xdm o kdm** exec window manager
shell **`pippo$ _`**	**shell** **`pippo$ _`**	**ambiente grafico** finestre e menu

PAM: esempio di configurazione

```
/etc/pam.d/$ cat login
# PAM configuration for login
auth        requisite  pam_securetty.so
auth        required   pam_nologin.so
auth        required   pam_env.so
auth        required   pam_unix.so nullok
#auth       required        pam_permit.so
account     required   pam_unix.so
session     required   pam_unix.so
session     optional   pam_lastlog.so
password    required   pam_unix.so nullok
  obscure min=4 max=8
```

PAM: successo e/o fallimento

- ciascun modulo ritorna “successo” o “fallimento”
- **required, requisite**. se il modulo ritorna fallimento il login non ha successo.
  - se un “requisite” fallisce si termina la procedure immediatamente
- **sufficient**. il successo di un tale modulo è sufficiente per l'autenticazione (anche se c'è stato un fallimento precedente)
- **optional.** il valore di ritorno è ignorato.

esercizio

- abilitare l'accesso senza password usando pam_permit.so

riferimenti

- Peter Hernberg - User Authentication HOWTO
  - installato sul sistema o in rete (Google)
- The Linux-PAM System Administrators' Guide
  - installato sul sistema o in rete (Google)

unix - hardening

unix hardening: ispezione servizi

- attivi
  - netstat --inet -a
    - opzioni utili: -n –p
    - /etc/services: lista delle well know port
  - ps aux
    - lista dei processi
- installati e/o configurati
  - runlevel
    - risponde con il runlevel corrente tipicamente 2 o 3
  - /etc/init.d/*
    - uno script per ciascun servizio installato
  - link agli init script /etc/rcN.d/*
    - N è il runlevel
  - superserver /etc/inetd.conf
    - telnet, finger, ecc.

unix hardening: comandi privilegiati

- Verifica tutti i comandi/servizi suid e sgid
  - è veramente necessario che siano suid/sgid
  - considera l'uso di sudo

sudo

- permette l’esecuzione controllata di comandi privilegiati da parte di utenti
- configurazione in /etc/sudoers
- esempio di configurazione

```
pete pro = /usr/bin/passwd [A-z]*, !/usr/bin/passwd root

pizzonia ALL = (ALL) NOPASSWD: ALL

User_Alias     WEBMASTERS = will, wendy, wim
WEBMASTERS     www = (www) ALL, (root) /usr/bin/su www
```

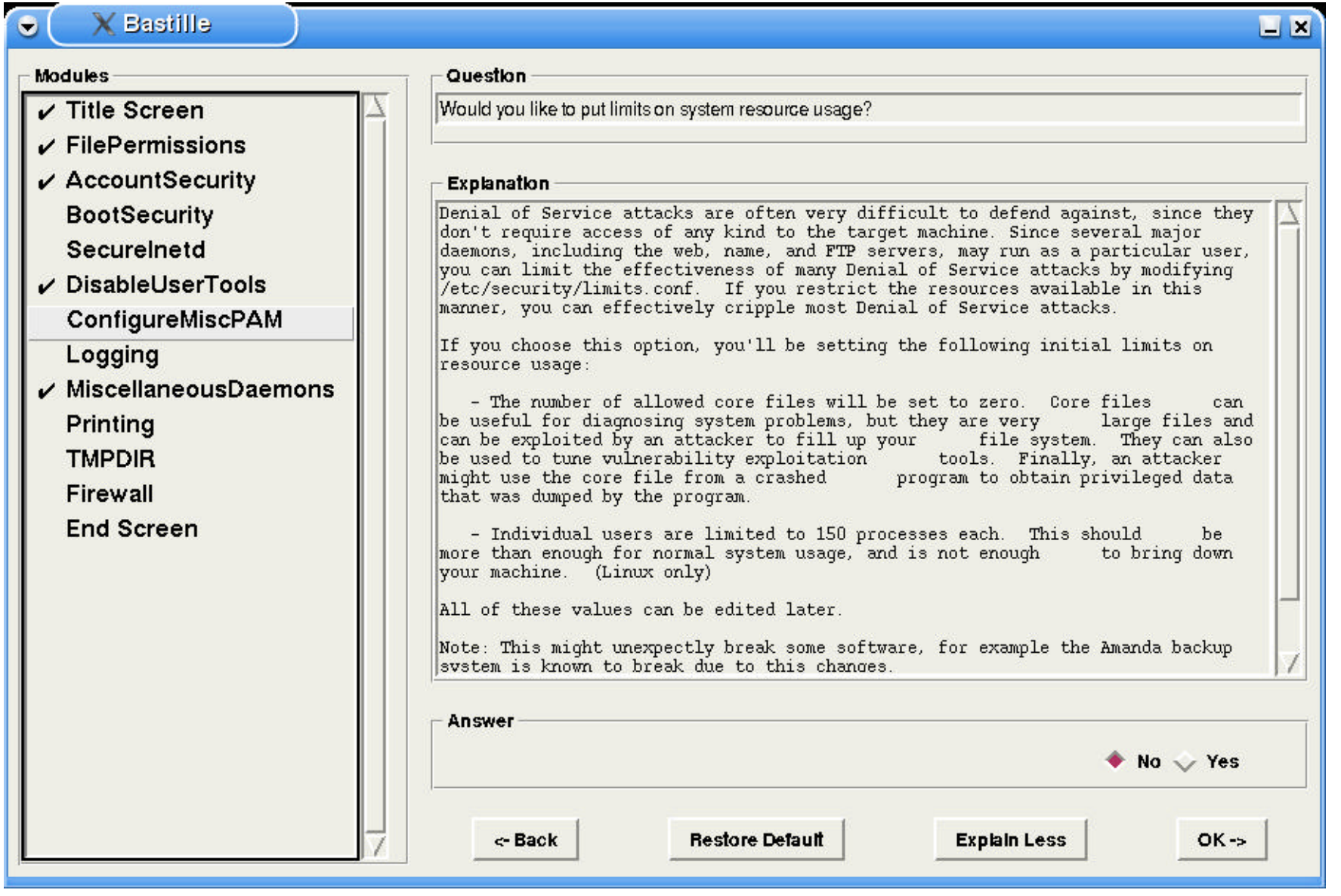
esercizio

- dai una occhiata al tuo sistema unix e commenta lo stato attuale della macchina.
  - probabilmente troverai servizi/processi attivi che non conosci, fanne un elenco e cerca la documentazione
  - fai un elenco degli eseguibili suid, cerca la documentazione relativa
- suggerisci attività di hardening

bastille

- strumento per l'hardening automatico
- basato su domande
  - molto flessibile
- permette alla fine del questionario di applicare le configurazioni e di fare rollback
- permette di salvare e riutilizzare configurazioni generate con bastille

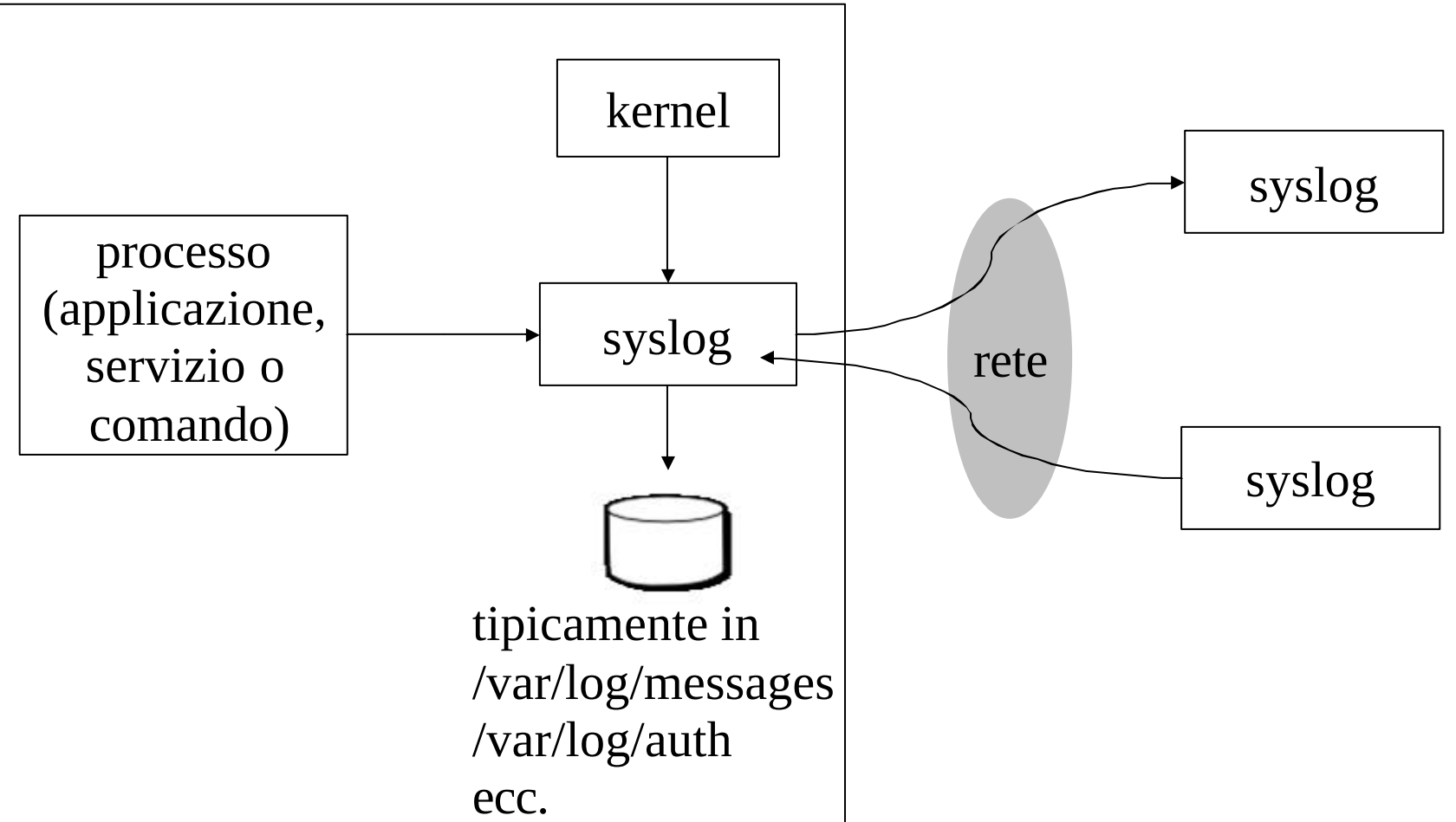
bastille


Bastille
Modules
✔ Title Screen
✔ FilePermissions
✔ AccountSecurity
BootSecurity
SecureInetd
✔ DisableUserTools
ConfigureMiscPAM
Logging
✔ MiscellaneousDaemons
Printing
TMPDIR
Firewall
End Screen
Question
Would you like to put limits on system resource usage?
Explanation
Denial of Service attacks are often very difficult to defend against, since they don't require access of any kind to the target machine. Since several major daemons, including the web, name, and FTP servers, may run as a particular user, you can limit the effectiveness of many Denial of Service attacks by modifying /etc/security/limits.conf. If you restrict the resources available in this manner, you can effectively cripple most Denial of Service attacks.
If you choose this option, you'll be setting the following initial limits on resource usage:
- The number of allowed core files will be set to zero. Core files can be useful for diagnosing system problems, but they are very large files and can be exploited by an attacker to fill up your file system. They can also be used to tune vulnerability exploitation tools. Finally, an attacker might use the core file from a crashed program to obtain privileged data that was dumped by the program.
- Individual users are limited to 150 processes each. This should be more than enough for normal system usage, and is not enough to bring down your machine. (Linux only)
All of these values can be edited later.
Note: This might unexpectly break some software, for example the Amanda backup system is known to break due to this changes.
Answer
No
Yes
<- Back
Restore Default
Explain Less
OK ->

unix - log management e auditing

syslog

- syslog è il servizio di logging normalmente attivo su ogni macchina Unix



priorità

severità o livello

- debug
- info: messaggio informativo
- notice: situazione normale ma significativa
- warning, *warn*
- err, *error*
- crit
- alert: una azione deve essere presa immediatamente
- emerg, *panic*

categorie di messaggi

(facilities o selettori)

- auth, authpriv, cron, daemon, ftp, kern, lpr, mail, mark, news, security (same as auth), syslog, user, uucp, local0...local7

il servizio in unix

- /etc/init.d/sysklogd start
- /etc/init.d/sysklogd stop
- alcuni file importanti
  - /etc/syslog.conf
  - /var/log/messages
  - /var/log/auth.log
  - /var/log/daemon.log
  - /var/log/kern.log
  - /var/log/mail.log
  - /var/log/user.log
  - ecc

configurazione di syslog

- /etc/syslog.conf
- ciascuna linea segue il formato
  <selettore> <azione>
- esempio: log di tutti i messaggi della categoria “auth” e importanza maggiore uguale a “notice” dentro /var/log/auth

```
auth.notice  /var/log/auth

auth.!err    #variante: importanza minore di err
auth.=warn    #variante: importanza uguale a warn
*.err  #variante: qualsiasi categoria purché >err
```

configurazione di syslog: esempi

```
cron.*                          -/var/log/cron.log
daemon.*                        /var/log/daemon.log
kern.*                          /var/log/kern.log
auth,authpriv.*                     /var/log/auth.log
*.*;auth,authpriv.!=none                /var/log/syslog


*.=info;*.=notice;*.=warn;\
  auth,authpriv.none;\
  cron,daemon.none;\
  mail,news.none
  /var/log/messages
```

remote logging

- syslogd può comportarsi da server
  - opzione “-r” da configurare nello script di startup /etc/init.d/sysklogd (debian)
- syslogd può comportarsi da client
  - è un particolare tipo di azione, es.:
    *.warn @193.204.161.48
    auth.crit @syslogserver.dia.uniroma3.it
- vantaggi
  - fare il log di molte macchine su una sola può snellire moltissimo i successivi auditing
  - un hacker per modificare i log su un syslog server deve penetrare anche in tale macchina
  - il syslog server può essere ben protetto

rotazione dei file di log

- i log tendono ad essere estremamente voluminosi
- per velocizzare elaborazioni (es. auditing) sui dati recenti si preferisce spezzarli periodicamente

- la soluzione dipende dalla distribuzione
  - soluzione basata sul comando logrotate e configurata in /etc/logrotate.d
  - soluzione “ad hoc” tramite uso di cron
    - usata da debian per i file generati da syslog

dimensione dei log

- durante attacchi DoS i log possono raggiungere dimensioni enormi
- è buona regola...
  - comprimere i vecchi spezzoni di log creati con logrotate
  - montare /var/log su un filesystem separato per evitare di saturare gli altri filesystem e bloccare utenti e applicazioni
    - oppure usare “disk quota”

acct

- meccanismo di logging dei file eseguiti
  - system call execve
  - /var/account/pacct
- il log viene generato dal kernel
- log in formato binario (efficiente)
- comandi
  - accton <nomefilelog> #: accende e spegne il logging
  - lastcomm <utente> # ultimi processi eseguiti
  - sa # report sull'uso dei comandi

log auditing

- logwatch
  - estrae le linee interessanti dai file di log e le invia via mail all'amministratore
  - supporta molti servizi, modulare, configurabile
- Lire
  - comparabile a logwatch ma più sofisticato
- swatch
  - analisi "al volo" dei log
  - esecuzione di comandi reattiva
  - guarda un solo file per volta, una linea per volta
- log surfer
  - comparabile a swatch ma più sofisticato