Hardening base e controllo dell'accesso

Marco Prandini
DISI
Università di Bologna

Introduzione

- Qualche richiamo di concetti base
  - La sicurezza è un processo
    - Esamineremo gli aspetti tecnici, senza dimenticare che contromisure appropriate esistono solo se integrate nel quadro generale
    - Nei sistemi complessi l'installazione dei singoli apparati è apparentemente sempre più semplice e virtualizzata, ma non si deve dimenticare che l'interazione logica e gli aspetti fisici possono generare problemi
  - Default deny / minimum privilege
    - Uno dei pochi punti fermi: la sicurezza si gestisce efficacemente solo vietando qualsiasi comportamento non esplicitamente consentito
    - Ciò che è consentito deve essere svolto con i diritti più restrittivi compatibili con l'esecuzione del compito
  - Sicurezza = riservatezza, integrità, autenticità, disponibilità
    - Proprietà diverse, a volte in conflitto
    - Individuare e difendere quelle necessarie caso per caso
      - Per le informazioni, ma anche per i sistemi che le trattano
      - Per ogni copia/istanza delle informazioni/sistemi

Messa in sicurezza fisica

- Un server è prima di tutto un sistema di calcolo, collocato in un ambiente e connesso a una varietà di dispositivi
  - Normalmente si concentrano le difese sul fronte degli attacchi via rete, a componenti software come applicazioni e sistema operativo
  - Le corrispondenti contromisure possono facilmente essere scavalcate da un attaccante con accesso fisico al sistema!
  - Le minacce principali sono:
    - Furto dello storage o dell'intero calcolatore
    - Connessione di sistemi di raccolta dati alle interfacce
    - Avvio del sistema con un sistema operativo arbitrario
  - La gravità di queste minacce dipende fortemante dallo specifico ambiente
- Molti di questi problemi sono cambiati nello scenario sempre più comune di virtualizzazione sul Cloud, ma altri concettualmente simili sono apparsi, e la logica delle stesse contromisure si può adattare

Mettere insieme i pezzi / da vicino

- Riponiamo normalmente fiducia nei computer locali
  - Quanti operatori hanno una vista diretta delle connessioni di mouse e tastiera?
  - E delle connessioni interne tra mainboard, dischi, unità ottiche?

Source: https://www.keelog.com/

Mettere insieme i pezzi / in remoto

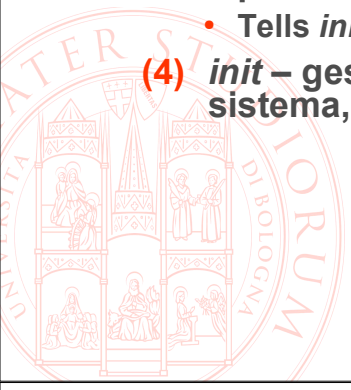
- Se la collocazione è fuori dalla possibilità di controllo diretto, si può considerare di:
  - Scegliere un case che possa essere chiuso e fissato al rack
  - Installare dispositivi di rilevazione delle intrusioni
  - Adottare misure di protezione dei dati che rendano inutile il furto
    - L'accesso ai dati va però abilitato manualmente
  - Disabilitare le periferiche non utilizzate
    - Salvo poi averne bisogno per esigenze nuove

Pianificare l'installazione

- Cosa serve?
  - Le distribuzioni hanno procedure di installazione "amichevoli"
  - È facile finire per installare molti pacchetti la cui esistenza (figuriamoci l'utilità) è ignota al sistemista… ma non agli attaccanti!
  - *Default deny* inizia con la scelta del software strettamente necessario
- Predisponete i dischi con integrità, disponibilità e scalabilità
  - Separando le partizioni che sono facilmente saturabili da attività di sistema (`/var`, `/tmp`) o di utenti (`/home`) da quelle essenziali per il sistema operativo system (`/`)
  - Considerando con attenzione l'uso di sistemi come LVM che permettono di gestire lo spazio (+) ma introducono un layer di complessità (-)
  - Ricorrendo a soluzioni poco pratiche ma molto robuste in casi estremi (es. mount di `/usr` in sola lettura)

Reboot

- Per andare a regime il sistema attraversa un processo di boot, che può essere diviso in queste fasi:
  - (1) BIOS – Individua i dispositivi di possibile caricamento del boot loader e l'ordine per esaminarli
    - Molti BIOS prevedono la possibilità di proteggere con password l'avvio o la modifica della configurazione
  - (2) Boot Loader – Sceglie il sistema operativo e gli passa eventuali parametri
    - Gestione della "maintenance mode"
    - Stesso tipo di protezione con password come descritto per BIOS
  - (3) Sistema opeativo – carica i device driver (da non sottostimare) e avvia il processo *init*
    - Tells *init* the initial runlevel (if overriding the default is needed)
  - (4) *init* – gestisce i *runlevel* o i *target* per coordinare l'inizializzazione del sistema, cioè avviare i servizi nell'ordine corretto



Boot

- Password pro e contro:
  - Se serve una password per l'avvio del sistema, il funzionamento non supervisionato può essere problematico: ad esempio può non ripartire per ore dopo una semplice mancanza di alimentazione
    - In sistemi con gravi exigenze di sicurezza, non sarà comunque l'unica password da fornire, quindi meglio proteggere tutti gli strati
  - Almeno la protezione contro i cambi di configurazione è sempre consigliabile
- MAI affidarsi a un unico strato di protezione
  - Le password del BIOS hanno meccanismi semplici di reset
  - Possono essere indovinate...

Bootloader – configurazione (runtime)

- LILO, the Linux Loader
  - Usato fin dagli albori di Linux
- GRUB, the Grand Unified Bootloader
  - GRUB è più potente e flessibile di LILO, è dotato di una shell che permette di eseguire vari comandi per modificare al volo la procedura di avvio: naturalmente questo permette molti abusi
- Entrambi permettono di passare parametri al kernel, i più importanti ai fini della sicurezza sono
  - single
  - Init=…
- Alcune distribuzioni hanno default rischiosi, ad esempio se si riesce a innescare un *maintenance mode* aprono una shell di root senza chiedere password

Boot Loader passwords

- LILO

  `password=YourPasswordHere`

  Imposta una password richiesta al boot, a meno che

  `restricted`

  non sia is specificato, in tal caso la password è richoesta solo per modificare l parametri durante il boot.
- Global vs. Single-entry
  - `password` e `restricted` nella "global section": chiede la password prima di consentire l'aggiunta di parametri – attenzione alle entry non sicure (cd)
  - `password` e `restricted` in una "image section": chiede la password prima di consentire l'aggiunta di parametri, solo per l'immagine specificata
  - `password` nella "global section" e `restricted` in una "image section": chiede la password prima di consentire l'aggiunta di parametri, solo per l'immagine specificata, mentre chiede sempre la password per avviare altre immagini

Boot Loader passwords

- GRUB

`password [--md5] passwd [new-config-file]`

Se specificato nella "global section", imposta una password che sarà richiesta per attivare l'*interactive operation* del bootloader. Opzionalmente può innescare il caricamento di un file di configurazione alternativo

Se specificata per un item specifico del menu, imposta una password che sarà richiesta per avviare quell'item

`lock`

Se specificata per un item specifico del menu, subito dopo `title`, contrassegna quell'item come password-protected.

Funziona solo se esiste una direttiva password nella "global section"

`md5crypt`

Comando utilizzabile al grub prompt per calcolare il password hash da usare con `--md5`

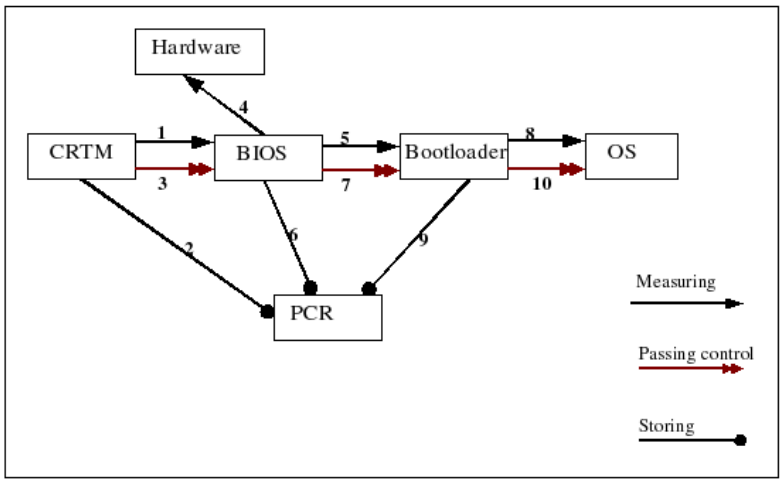
Secure / Trusted Boot

- Problema: come assicurarsi che ogni componente software eseguito da un computer sia autentico, integro e benevolo?
  - Anti-malware verificano le applicazioni
  - Chi verifica gli anti-malware?? Il S.O. (idealmente rendendo AM inutile)
  - Chi verifica il S.O.? Il boot loader potrebbe
  - Chi verifica il boot loader? Il BIOS potrebbe, specialmente se assistito da HW speciale, che non possa essere modificato dal S.O., e quindi sia immune da infezioni

    → *root of (a chain of) trust*
- Due modi
  - *Trusted boot* usa il TPM (Trusted Platform Module)
    - Special hardware chip con funzionalità crittografiche
  - *Secure boot* usa UEFI (Unified Extensible Firmware Interface)
    - Implementazione Software + chiavi in firmware
    - Serve un BIOS standard per la fase di POST
    - Può avvalersi del TPM per velocizzare e migliorare i controlli di integrità

Trusted boot

- Parte dal TPM
  - Core Root of Trust for Measurement (CRTM)
  - Registers (PCR)
- Raccoglie prove di (violazioni della) integrità
- Pospone i controlli fintanto che non dispone
  - Delle crypto keys
  - Di abbastanza memoria per fare i calcoli necessari



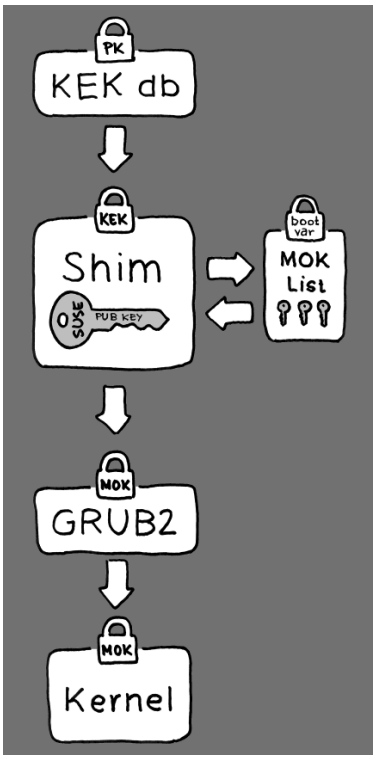
UEFI e secure boot

- EFI (Intel) nasce come interfaccia più flessibile del BIOS tra S.O. e firmware
- UEFI forum standardizza e aggiorna la specifica
  - http://www.uefi.org/
- UEFI è un "mini OS"
  - BIOS boot su MBR:
    - 400 bytes di assembler nel boot sector
    - 4 partizioni primarie o 3 primarie + 11 unità logiche
  - EFI con GPT
    - filesystem dedicato (100-250MB) per i boot loader
    - Partizioni fino a 9ZB in numero illimitato
- UEFI verifica ogni componente software prima di passare il controllo a BL/SO
  - Richiede la disponibilità di un database di chiavi
  - Blocca il boot appena rileva una difformità

UEFI e secure boot in Linux

1) La Platform Key ufficiale verifica un piccolo pre-boot-loader, **shim**
   - La chiave key usata per "firmare" shim deve essere fornita dal costruttore HW
   - È una chiave Microsoft!
2) Shim può usare o trasferire MOKs (Machine Owner Keys)
   - Per validare il bootloader
   - Per validare moduli custom del kernel

- Componenti aggiuntivi del kernel devono essere firmati per poterli caricare
  - L'utente genera le MOKs
  - L'utente deposita le MOKs in shim
  - Al boot successivo, shim trova le chiavi nella fase di setup, e chiede conferma per salvarle in firmware → **consenso esplicito e basato su password sempre richiesto!**



https://www.suse.com/communities/blog/uefi-secure-boot-details/

UEFI links

- https://www.linux.com/publications/making-uefi-secure-boot-work-open-platforms
- http://www.rodsbooks.com/linux-uefi/
- http://www.linux-magazine.com/Online/Features/Coping-with-the-UEFI-Boot-Process
- https://help.ubuntu.com/community/UEFI
- http://askubuntu.com/questions/760671/could-not-load-vboxdrv-after-upgrade-to-ubuntu-16-04-and-i-want-to-keep-secur
- https://knowledge.windriver.com/en-us/000_Products/000/010/040/060/020/000_Wind_River_Linux_Security_Profile_Developer's_Guide,_8.0/070/000/010
- https://www.suse.com/communities/blog/uefi-secure-boot-details/
- https://lwn.net/Articles/519618/

Accesso alle risorse

- La catena che permette di gestire l'accesso alle risorse è composta di quattro fasi:

- Identificazione
  - Determinare qual è il soggetto che richiede accesso
- Autenticazione
  - Verificare se il soggetto ha credenziali valide per identificarlo
- Autorizzazione
  - Decidere se il soggetto ha titolo per eseguire una data operazione
- Auditing
  - Tracciare le operazioni sopra elencate per attribuire la responsabilità degli eventi sul sistema e diagnosticare problemi nella definizione delle politiche o nella loro implementazione

Autenticazione – prova dell'identità

- Tre prove:

- "qualcosa che sei" - conferma dell'identità per confronto di una caratteristica fisiologica o comportamentale con un dato "biometrico" di riferimento.
- "qualcosa che hai" - conferma dell'identità attraverso il possesso di un oggetto riconoscibile da parte della macchina (una scheda a banda magnetica, una smart card, un token RFID, …)
- "qualcosa che sai" - conferma dell'identità dimostrando la conoscenza di un dato segreto concordato in precedenza (password o personal identification number o chiave)

Gestione utenti

- Gli utenti/gruppi possono essere creati con tool grafici o a riga di comando
  - `adduser`, `addgroup`
- Ogni utente DEVE appartenere almeno a un gruppo
  - Normalmente il sistema ne crea uno omonimo, con solo l'utente dentro
- Ogni utente può appartenere a un numero variabile di altri gruppi
- Gli account utente possono essere in uno stato *locked*, che impedisce di usarli per l'accesso interattivo, ma consente ai processi di girare con tale identità
  - Minimo privilegio!
- Il comando *passwd* si usa
  - Per cambiare le password (proprie, salvo root che può cambiarle a tutti)
  - Per settare l'account allo stato lock (`-l`) o unlock (`-u`)
    - Ovviamente solo root può farlo

Gestione utenti, gruppi, ownership

- Comandi adduser e addgroup
  - Leggere le man pages
  - Provare a creare utenti di fantasia
- Verificare gli effetti su questi file
  - /etc/passwd
  - /etc/shadow
  - /etc/group
  - /etc/gshadow
- Comando chown <new_owner:new_group> <file>
  changes the file's owner and/or group

Verificare l'identità e l'accesso degli utenti

whoami	riporta il proprio username
id [username]	dà informazioni sull'identità e sul gruppo di appartenenza di un utente
who	indica chi e' attualmente collegato alla macchina
last	elenca lo storico dei collegamenti

Autenticazione con password

- Tipicamente le credenziali locali sono in
  - */etc/passwd*, world-readable, una riga per utente:

```
prandini:x:500:500:Marco Prandini:/fat/home:/bin/bash
```

  - */etc/shadow*, accessibile solo a root, linee corrispondenti a passwd

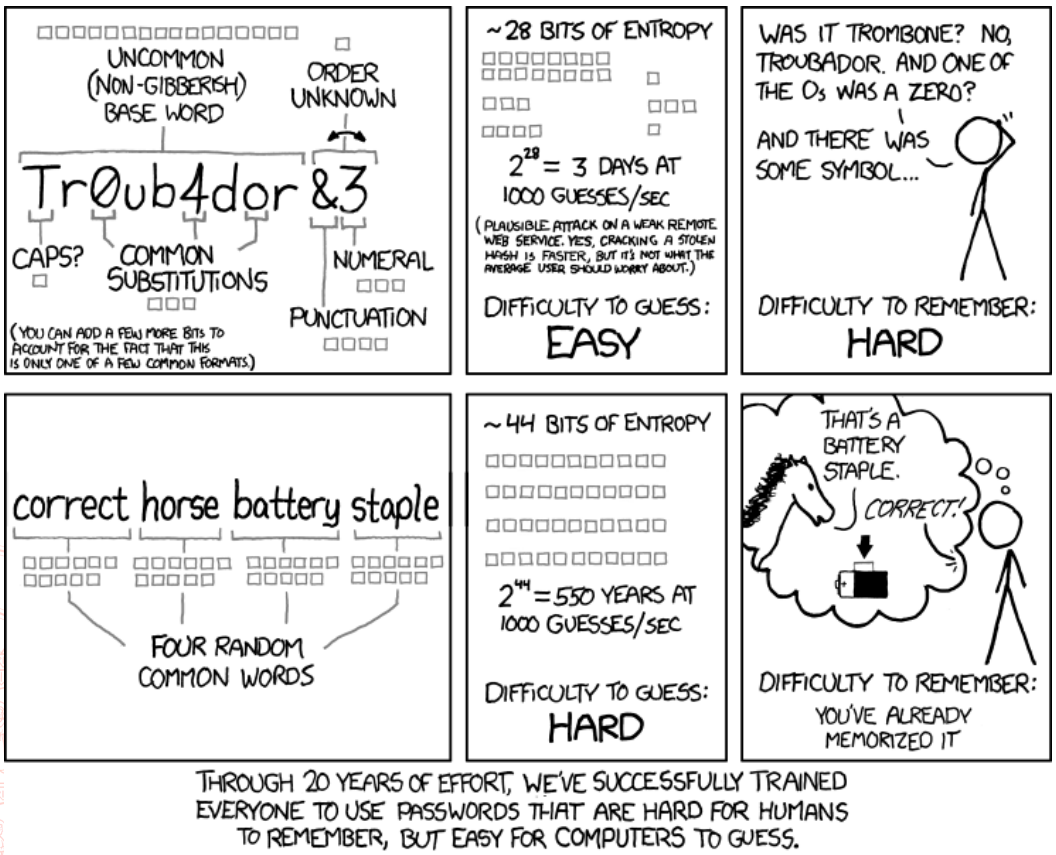
```
prandini:$1$/PBy29Md$kjC1F8dvHxKhnvMTWelnX/:12156:0:99999:7:::
```

  - Nota: non rimuovere il segnaposto 'x' nel secondo campo di passwd, o il sistema non guarderà il file shadow e non riconoscerà la password

Robustezza delle password

- Troppi utenti continuano a scegliere password facili da indovinare
- Educarli è importante ma non sempre sufficiente
  - Buon saggio di Bruce Schneier che suggerisce come scegliere bene: http://www.wired.com/politics/security/commentary/securitymatters/2007/01/72458?currentPage=all
- Due contromisure:
  - proattiva (impedire password deboli, effetto collaterale post-it)
    - A seguire, esempio di configurazione con PAM
  - reattiva (scoprire le password deboli e fare un discorsetto all'utente)
    - John the Ripper http://www.openwall.com/john/

Password strength - http://xkcd.com/936/



Età delle password

- Il file shadow contiene dati sulla validità temporale della password, esaminabili e modificabili con *chage*:

```
<name>:<pw>:<date>:PASS_MIN_DAYS:PASS_MAX_DAYS:PASS_WARN_AGE:INACTIVE:EXPIRE:
```

- Significato e nome del file da cui viene preso il valore di default::

/etc/login.defs	PASS_MAX_DAYS	Maximum number of days a password is valid.
/etc/login.defs	PASS_MIN_DAYS	Minimum number of days before a user can change the password since the last change.
/etc/login.defs	PASS_WARN_AGE	Number of days when the password change reminder starts.
/etc/default/useradd	INACTIVE	Number of days after password expiration that account is disabled.
/etc/default/useradd	EXPIRE	Account expiration date in the format YYYY-MM-DD.

Imporre le caratteristiche della password

- pam_cracklib.so è la libreria per il controllo della robustezza delle password (all'atto della scelta)
- In */etc/pam.d/system-auth* o */etc/pam.d/common-password* la riga che inizia con `password requisite` può accettare i seguenti parametri (dopo `pam_cracklib.so`):
  - minlen = (Minimum length of password)
  - lcredit = (Length credit for lower case letters)
  - ucredit = (Length credit for upper case letters)
  - dcredit = (Length credit for digits)
  - ocredit = (Length credit for other characters)
- Esempio del meccanismo dei crediti:
  - minlen=8 dcredit=1
    - Qualsiasi password di almeno 8 caratteri va bene
    - Qualsiasi password di (8-*n*) caratteri va bene se *n* sono cifre numeriche

Limitare il riuso delle password

- Gli stessi file di PAM consentono di regolare il riutilizzo delle password. Per esempio, aggiungendo il parametri sottolineati:

```
password    required     pam_cracklib.so ... difok=3
password    sufficient   pam_unix.so ... remember=26
```

1) una nuova password deve avere almeno 3 caratteri diversi dalla precedente
2) le ultime 26 password sono memorizzate e non possono essere riutilizzate

User lockout dopo errori di autenticazione

- ATTENZIONE a non fare il gioco dell'attaccante, che sbagliando di proposito può impedire all'utente di entrare
- Sempre attraverso PAM si può utilizzare il modulo *tally*:

```
auth        required    pam_tally.so onerr=fail no_magic_root
account     required    pam_tally.so deny=5 no_magic_root reset
```

  - La prima riga abilita il conteggio dei tentativi falliti di accesso
  - La seconda riga blocca l'account quando il numero di fallimenti raggiunge la soglia specificata con *deny*
  - Un login corretto resetta il contatore
- Il comando *faillog* permette di esaminare lo stato di un account e di riattivare un account bloccato per eccesso di tentativi errati di autenticazione

Il super-utente

- L'utente *root* ha privilegi illimitati, va difeso contro ingressi abusivi ma va anche minimizzata la probabilità di fare errori
  - Usare un account non-privilegiato, basta per il 99% del tempo
  - Disabilitare l'accesso diretto da GUI e console
  - Ottenere temporaneamente i diritti di *root* solo per eseguire i task di amministrazione
- Due modi di ottenere i diritti di *root*
  - **su** (switch user) facile ma inadatto a sistemi in cui più amministratori condividono responsabilità
    - Richiede che tutti conoscano la password di root
    - Dà a tutti accesso illimitato al sistema
  - **sudo** (do as super-user)
    - Esegue un singolo comando coi privilegi di root
    - Richiede la password dell'utente che lo lancia (per prevenire distrazioni e uso di terminali incustoditi - coffee break attacks)
    - Configurabile (limita quali programmi sono eseguibili da ogni utente)
    - Si usi ***visudo*** per editare il file di configurazione `/etc/sudoers` (controlla la sintassi e installa il file evitando errori che potrebbero chiudere fuori un utente) – si veda la man page *sudoers* per la sintassi

Evitare l'eccessivo uso di risorse coi *limits*

- Linux permette di settare i limiti di uso di varie risorse da parte di ogni processo
- Efficace per prevenire *denial-of-service* sia accidentali che malevoli
- I limits possono essere
  - hard (imposti da root come valore massimo, non possono essere superati dall'utente)
  - soft (valori di default, modificabili dall'utente entro il massimo permesso dal S.O. e dal limite hard)
- Limits can be configured
  - Localmente a ogni shell
    - Comando **ulimit**
  - Centralmente col modulo pam_limit (applicati a ogni login)
    - File `/etc/limits.conf`
    - Linee nella forma: <**domain**> <**type**> <**item**> <**value**>
      - **Domain: user, @group, ***
      - **Type: hard, soft, - (both)**
      - **Item: (vedi prossima slide)**

Evitare l'eccessivo uso di risorse coi *limits*

limit description	type for limits.conf	option for ulimit
socket buffer size		-b
maximum size of core files created	core	-c
maximum size of a process's data segment	data	-d
maximum scheduling priority (`nice')	nice	-e
maximum size of files written by the shell and its children	fsize	-f
maximum number of pending signals	sigpending	-i
maximum size a process may lock into memory	memlock	-l
maximum resident set size	rss	-m
maximum number of open file descriptors	nofile	-n
pipe buffer size		-p
maximum number of bytes in POSIX message queues	msqqueue	-q
maximum real-time scheduling priority	rtprio	-r
maximum stack size	stack	-s
maximum amount of cpu time	cpu (minutes)	-t (seconds)
maximum number of user processes	nproc	-u
size of virtual memory		-v
maximum number of file locks	locks	-x
address space limit	as	
maximum number of logins for this user	maxlogins	
maximum number of logins on the system	maxsyslogins	
priority to run processes with	priority	
directory to chroot the user to	chroot	

Generalità sul controllo dell'accesso a risorse

- In principio, il controllo dell'accesso è decidere se un ***soggetto*** può eseguire una specifica ***operazione*** su di un ***oggetto***
- Il modo più banale per esprimere i *permessi* sarebbe una matrice completa

Subject / Object	User1	User2	Group3	...	...
File1	read	read write	--	...	...
Dir2	list	modify	--	...	...
Socket3	write	--	read	...	...
...	...	...	...	...	...
...	...	...	...	...	...

La matrice distribuita

- Migliaia di soggetti, milioni di oggetti!
- La maggior parte delle "celle" è sempre al valore di default → potrebbe essere omessa
- Due modi di risparmiare spazio e decentralizzare:
  - Partizionare la matrice per soggetto: *capability lists*
    - Una lista associata a ogni soggetto del sistema
    - Contiene solo gli oggetti su cui il soggetto ha permessi ≠ default
  - Partizionare la matrice per soggetto: *access control lists* (ACL)
    - Una lista associata a ogni soggetto del sistema
    - Contiene solo i soggetti che hanno permessi ≠ default sull'oggetto
    - Esplicitamente implementata da POSIX e MS Windows
    - Il filesystem Unix tradizionale ha negli inode una ACL "rigida", che elenca sempre e solo tre soggetti:
      - L'utente proprietario (U)
      - Il gruppo proprietario (G)
      - Il gruppo implicito che contiene tutti gli utenti ≠ U e ∉ G

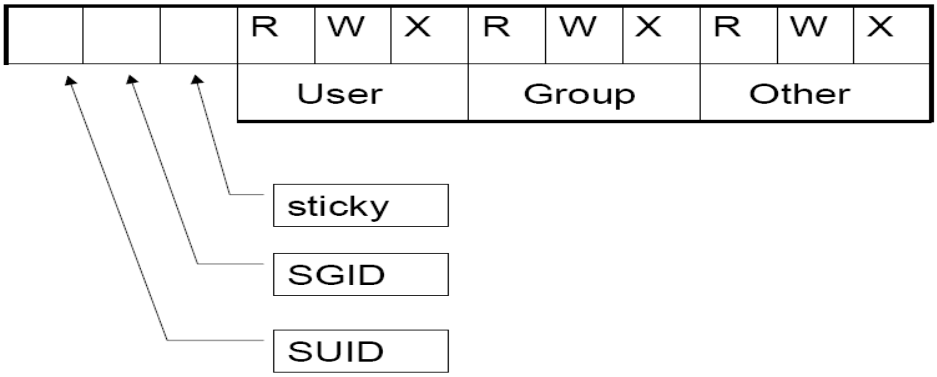
      e i relativi permessi

Modelli di controllo dell'accesso

- I due paradigmi fondamentali sono
  - DAC (Discretionary access control):
    - Ogni oggetto ha un proprietario
    - Il proprietario decide i permessi
    - Le ACL sono l'esempio tipico
  - MAC (Mandatory access control):
    - La proprietà di un oggetto non consente di modificarne i permessi
    - C'è una policy centralizzata decisa da un *security manager*
    - Tipicamente espressa come qualche genere di capability lists
- Ci sono modelli più complessi
  - RBAC (Role-based access control):
    - I permessi sono assegnati ai *ruoli*
    - Utile se i soggetti possono assumere dinamicamente ruoli differenti a seconda del contesto (cosa devono fare, dove si trovano, in che tempi operano…)
  - e varianti...

Autorizzazioni su Unix Filesystem

- Ogni file (regolare, directory, link, socket, block/char special) è descritto da un i-node
- Un set di informazioni di autorizzazione, tra le altre cose, è memorizzato nell'i-node
  - (esattamente un) utente proprietario del file
  - (esattamente un) gruppo proprietario del file
  - Un set di 12 bit che rappresentano permessi standard e speciali



Significato dei bit di autorizzazione

- Leggermente diverso tra file e directory, ma in gran parte deducibile ricordando che
  - Una directory è semplicemente un file
  - Il contenuto di tale file è un database di coppie (nome, i-node)

R = read (lettura del contenuto)

Lettura di un file

Elenco dei file nella directory

W = write (modifica del contenuto)

Scrittura dentro un file

Aggiunta/cancellazione/rinomina di file in una directory

NOTA che il permesso 'W' in una directory consente a un utente di cancellare file sul contenuto dei quali non ha alcun diritto

X = execute

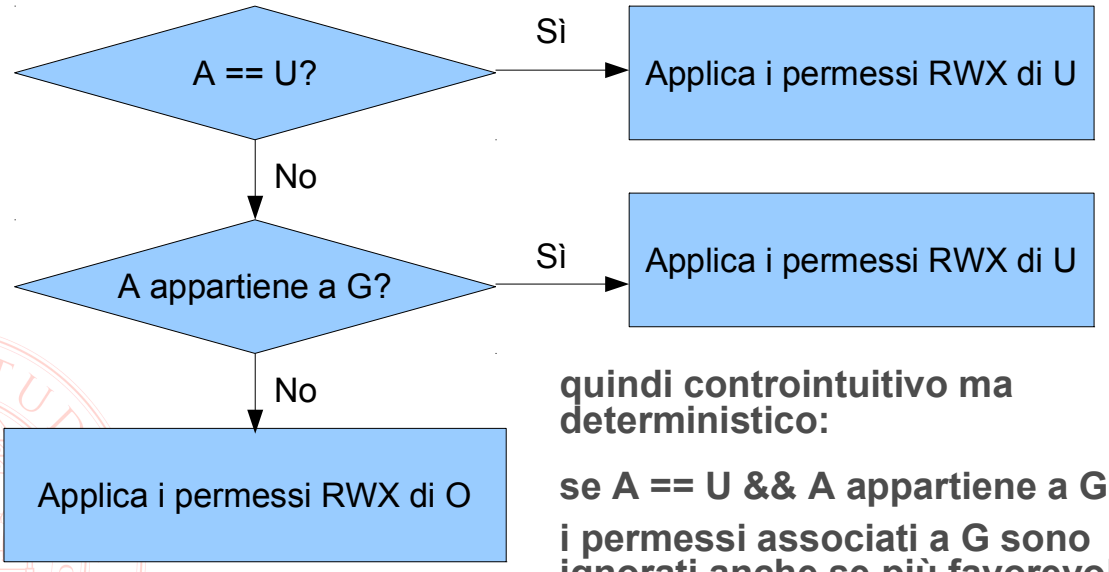
Esegui il file come programma

Esegui il lookup dell'i-node nella

NOTA: l'accesso a un file richiede il lookup di tutti gli i-node corrispondenti ai nomi delle directory nel path → serve il permesso 'X' per ognuna, mentre 'R' non è necessario

Composizione dei permessi

- Quando un utente “A” vuole eseguire un’operazione su di un file, il sistema operativo controlla i permessi secondo questo schema:



quindi controintuitivo ma deterministico:

se A == U && A appartiene a G

i permessi associati a G sono ignorati anche se più favorevoli

Controllo dei permessi predefiniti

- Servono automatismi per assegnare i permessi alla creazione
- Ownership
  - l’utente creatore è assegnato come proprietario del file
  - Il gruppo attivo dell’utente creatore è assegnato come gruppo proprietario
    - Default = gruppo predefinito, da */etc/passwd*
    - L’utente lo può cambiare a mano nella sessione con **newgrp**
    - Può cambiare automaticamente nelle directoy con SGID settato
- Permessi = “tutti quelli sensati” tolta la umask
  - “tutti quelli sensati”
    - `rw-rw-rw-` (666) per i file, l’eseguibilità è un’eccezione
    - `rwxrwxrwx` (777) per le directory, la possibilità di entrarci è la regola
    - la *umask* quindi può essere unica: una maschera che toglie i permessi da non concedere
  - poiché in Linux il gruppo di default group di un utente contiene solo l’utente stesso, una umask sensata è 006 (toglie agli “other” lettura e scrittura)
    - È un settaggio utile per collaborare, crea file manipolabili da tutti i membri del gruppo, a patto che questo sia settato correttamente
  - col comando **umask** si può interrogare e settare interattivamente, per rendere persistente la scelta si usano i file di configurazione della shell

Bit speciali / per i file

I tre bit più significativi della dozzina (11, 10, 9) configurano comportamenti speciali legati all'utente proprietario, al gruppo proprietario, e ad altri rispettivamente

- BIT 11 – SUID (Set User ID)
  - Se settato a 1 su di un programma (file eseguibile) fa sì che al lancio il sistema operativo generi un processo che esegue con l'identità dell'utente proprietario del file, invece che quella dell'utente che lo lancia
- BIT 10 – SGID (Set Group ID)
  - Come SUID, ma agisce sull'identità di gruppo del processo, prendendo quella del gruppo proprietario del file
- BIT 9 – STICKY
  - OBSOLETO, suggerisce al S.O. di tenere in cache una copia del programma

Permessi delicati da tenere sotto controllo

- SUID e SGID sono un modo efficace di implementare interfacce per utenti standard verso processi privilegiati
  - Cambio password: guardare /usr/bin/passwd
  - Pianificazione di attività: guardare /usr/bin/crontab e /var/spool/cron/
  - etc.
- I programmi con questi permessi vanno sorvegliati, perché chiunque li lanci acquisisce temporaneamente privilegi elevati
  - Pochi programmi e molto vincolati
  - Rischi: bug di questi programmi che porti a eseguire operazioni arbitrarie invece di quelle progettate, programmi diversi a cui sono dati per errore questi privilegi
- Usare find per trovarli
  - `find / -type f -perm +6000`
- Altre ricerche interessanti per la sicurezza
  - file world-writable (`-perm +2`)
  - file senza proprietario, rimasti da account cancellati (`-nouser`)

Bit speciali / per le directory

- Bit 11 per le directory non viene usato
- Bit 10 – SGID
  - Precondizioni
    - un utente appartiene (anche) al gruppo proprietario della directory
    - il bit SGID è impostato sulla directory
  - Effetto:
    - l'utente assume come gruppo attivo il gruppo proprietario della directory
    - I file creati nella directory hanno quello come gruppo proprietario
  - Vantaggi (mantenendo umask 0006)
    - nelle aree collaborative il file sono automaticamente resi leggibili e scrivibili da tutti i membri del gruppo
    - nelle aree personali i file sono comunque privati perché proprietà del gruppo principale dell'utente, che contiene solo l'utente medesimo
- Bit 9 – Temp
  - Le "directory temporanee" cioè quelle world-writable predisposte perché le applicazioni dispongano di luoghi noti dove scrivere, hanno un problema: chiunque può cancellare ogni file
  - Questo bit settato a 1 impone che nella directory i file siano cancellabili solo dai rispettivi proprietari

Gestione dei permessi

chmod è usato per modificare i permessi

Modo simbolico:

```
chmod 'a=r,g-rx,u+rwx' file,
```

- "all" ricevono esattamente il permesso 'r',
- a "group" vengono rimossi se presenti i permessi 'rx'
- a "user" vengono aggiunti i permessi 'rwx'

Modo numerico (base ottale):

```
chmod 2770 miadirectory
```

2770 octal = 010 111 111 000 binary = SGID rwx rwx ---

```
chmod 4555 miocomando
```

4555 octal = 100 101 101 101 binary = SUID r-x r-x r-x

Attributi

- Gli attributi sono primariamente utili per il fs tuning
  - compressed (c), no dump (d), extent format (e), data journalling (j), no tail-merg-ing (t), undeletable (u), no atime updates (A), synchronous directory updates (D), synchronous updates (S), and top of directory hierarchy (T)
- Alcuni sono rilevanti per la sicurezza
  - append only (a) – utile per impedire il taglio dei logfile
  - immutable (I) – vieta cancellazione, creazione di link verso il file, rinomina e scrittura, utile per i file di sistema
  - secure deletion (s) – sovrascrive con zeri i blocchi dei file cancellati (sicurezza molto limitata ma valida contro strumenti in linea)
- Tools
  - **chattr** per modificarli
  - **lsattr** per visualizzarli

POSIX Access Control Lists

- Le ACL estendono la flessibilità di autorizzazione
- Vantaggi:
  - Specificare una lista arbitraria di utenti e gruppi coi relativi permessi (comunque scelti tra rwx) in aggiunta agli owner
  - Ereditare la maschera di creazione dalla directory
  - Limitare tutti i permessi simultaneamente (esempio mask sotto)
- Esempio:

```
user::rw-
user:lisa:rw-          #effective:r--
group::r--
group:toolies:rw-      #effective:r--
mask::r--
other::r--
```

- Strumenti:
  - `setfacl` per impostare, `getfacl` per visualizzare le ACL (ls -l mostr un '+' dopo i permessi se ACL è presente per un file)
  - `man acl`