

Mandatory Access Control Systems

Introduzione ai sistemi MAC su Linux con GRSecurity

Marco Squarcina
lavish@gmail.com

Dipartimento di Scienze Ambientali, Informatica e Statistica
SECGROUP @unive
Università Ca' Foscari, Venezia

28 aprile 2011

- ① **Sommario**
- ② **Cosa si intende per “access control”**
- ③ **Problemi del access control system standard**
- ④ **Mandatory Access Control models**
- ⑤ **Role-Based Access Control**
- ⑥ **GRSecurity**
- ⑦ **Altre implementazioni MAC in Linux**
- ⑧ **Bibliografia**

ACCESS CONTROL

di cosa si tratta

Un sistema di *controllo degli accessi* riguarda:

- il modo in cui gli utenti accedono alle risorse di un sistema
- “*Who can do what*”

Ha un ruolo centrale nel mantenimento dei requisiti di:

- **Riservatezza** (*Confidentiality*)
 - **Integrità** (*Integrity*)
 - **Disponibilità** (*Availability*)
- } CIA

ACCESS CONTROL

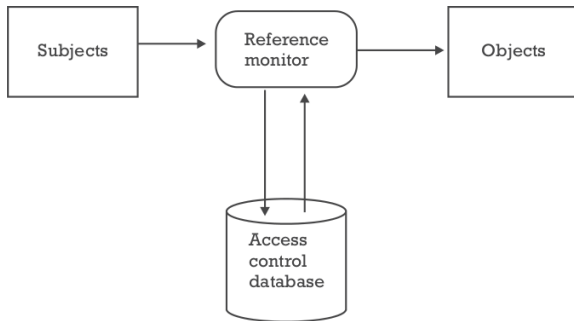
terminologia

Generalmente, un modello di controllo degli accessi può essere definito usando le seguenti entità:

- **utenti**
persone che si interfacciano al sistema informativo
- **soggetti**
processi che eseguono azioni da parte dall'utente (entità attive)
- **oggetti**
risorse del sistema (entità passive)
- **operazioni**
singole azioni effettuate dai soggetti
- **permessi**
autorizzazione ad effettuare qualche azione su degli oggetti

ACCESS CONTROL

- Il **modello** di controllo degli accessi definisce uno schema per specificare ed imporre una policy.
- Il **meccanismo** determina *come* il controllo degli accessi viene implementato.
- La **policy** stabilisce quali soggetti possono accedere a quali oggetti.



ACCESS CONTROL

in linux

Il *controllo degli accessi* in Linux riguarda il modo in cui il **kernel** controlla **processi** (programmi in esecuzione) che accedono a **risorse** (file, directory, socket, ...)

Esempio: web server

Un web server

- può accedere ai file nella directory di webroot
- non può accedere a `/etc/shadow`

Come vengono prese queste decisioni?

STANDARD ACCESS CONTROL

in linux

Il kernel ha un **meccanismo** *hardcoded*, mentre la **policy** è definita dalle proprietà di sicurezza attribuite a file e processi:

- processi:
 - utente e gruppo
- risorse:
 - utente e gruppo
 - bit di accesso (read, write, execute per utente, gruppo, altri)

Esempio: proprietà di sicurezza di processi e file

```
USER      GROUP      TIME CMD
lavish    lavish     00:00:14 /opt/firefox/firefox-bin

-rw----- 1 lavish lavish 1458 ... /home/lavish/.gnupg/secring.gpg
-rw----- 1 lavish lavish 1675 ... /home/lavish/.ssh/id_rsa
```

Quando viene effettuata un'operazione su una risorsa, il processo attivo esegue una *system call*. L'OS esamina *eid* e *egid* del processo oltre agli attributi della risorsa e decide se consentire l'operazione.

STANDARD ACCESS CONTROL

problemi

L'accesso alle risorse è basato sull'utente a cui appartiene tale risorsa:

- se un processo viene compromesso, l'attaccante può accedere alle risorse dell'utente con il quale è stato eseguito tale processo
- non c'è modo di differenziare l'accesso alle risorse in base a processi eseguiti dallo stesso utente

Firefox può accedere alle chiavi private dell'utente che lo esegue, anche se generalmente non c'è ragione per farlo!

Se il browser viene compromesso gli effetti possono essere disastrosi. . .

STANDARD ACCESS CONTROL

problemi

I processi e gli utenti possono sempre cambiare le proprietà di sicurezza delle risorse che gli appartengono:

- un utente malevolo o avventato può modificare le proprietà di sicurezza di risorse sensibili rendendole disponibili ad altri
- un'applicazione può dar luogo a perdite di dati sensibili
- il controllo degli accessi standard è **discrezionale**

Un processo malevolo o l'utente stesso può modificare le proprietà di sicurezza dei file contenenti le chiavi private ssh e pgp rendendole disponibili a tutti gli utenti del sistema! Non c'è generalmente motivo di modificare i permessi di questi file...

STANDARD ACCESS CONTROL

problemi

Esistono solo 2 livelli di sicurezza: utenti e root

- policy semplicistica
- un utente malevolo che riesce ad ottenere una shell di root ha accesso a tutte le risorse del sistema

DAC e MAC

a confronto

Le tecniche di controllo degli accessi possono essere **discretionary**, **mandatory** o una combinazione di esse.

Discretionary Security Policy

“any security policy where ordinary users may be involved in the definition of the policy functions and/or the assignment” [1]

Mandatory Security Policy

“any security policy where the definition of the policy logic and the assignment of security attributes is tightly controlled by a system security policy administrator” [1]

Access Control Systems

modelli mandatory

Sono stati sviluppati diversi modelli di tipo mandatory:

- **BLP**, *Bell-La Padula* [2]
 - *Security levels* assegnati a soggetti e oggetti
 - 2 proprietà:
 - *Simple security property*: un soggetto può leggere un oggetto se il *security level* del soggetto domina quello dell'oggetto.
 - *Star property*: un soggetto può scrivere un oggetto se il *security level* dell'oggetto domina quello del soggetto.
 - Garanzia di riservatezza ma non di integrità
- **Biba** [3]
 - Duale del BLP
 - Garanzia di integrità ma non di riservatezza
- **Clark-Wilson** [4]
 - Scopi commerciali piuttosto che militari
 - Centralizzato sull'integrità:
 - *SoD*
 - transazioni

Access Control Systems

modelli mandatory

- **Chinese wall** [5]
 - Oggetti raggruppati in categorie mutualmente disgiunte di “conflitto-di-interessi”
 - Un soggetto può leggere oggetti appartenenti a diverse categorie, ma non alla stessa
- **DTE, Domain-type enforcement** [6]
 - *Domini* associati ai soggetti
 - *Tipi* associati agli oggetti
 - 2 gruppi di permessi:
 - dominio-dominio
 - dominio-tipo

Questi modelli sono generalmente poco flessibili, in quanto sviluppati per specifici contesti.

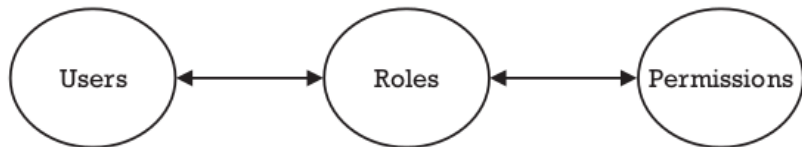
Access Control Systems

RBAC

RBAC (*role-based access control*) nasce come approccio alternativo ai sistemi MAC e DAC in ambito civile.

Molto flessibile, si adatta perfettamente alla struttura di un'organizzazione.

L'accesso alle risorse non è definito in base all'utente, ma al ruolo a cui l'utente è associato.



È stato sviluppato uno standard da parte del NIST [7] poi sfociato nello standard ANSI INCITS 359-2004

RBAC

nel quotidiano

In un qualsiasi ente o azienda, le decisioni sugli accessi da parte delle persone a informazioni o risorse, sono basate sui ruoli che i singoli individui hanno all'interno dell'organizzazione:

- gli individui assumono un ruolo
- processo di definizione dei ruoli basato su un'attenta analisi delle caratteristiche dell'organizzazione

Esempio: ruoli in un ospedale

In un ospedale sono definiti dei ruoli:

- Dottore
- Ricercatore
- Manager
- Infermiere
- ...

RBAC

nel quotidiano

I diritti d'accesso sono definiti in base al ruolo e l'uso delle risorse è riservato agli individui autorizzati ad assumere il ruolo associato.

Esempio: permessi associati ai ruoli in un ospedale

Dottore:

- effettuare diagnosi
- prescrivere farmaci
- ordinare test di laboratorio

Ricercatore:

- raccogliere dati anonimi per studio

...

L'uso di ruoli per il controllo degli accessi si presta molto bene per sviluppare policy di sicurezza.

RBAC non è un sistema di controllo degli accessi di tipo DAC:

- gli utenti non possono passare permessi ad altri utenti a loro discrezione

Esempio: passaggio di permessi in un ospedale

Un dottore può ordinare dei farmaci, ma non può passare il permesso di ordinare dei farmaci ad un infermiere

RBAC è un sistema di controllo di sicurezza con una forte accezione *mandatory*:

- possibilità di controllare a livello centralizzato la policy
- *security administrator* responsabile di fare l'*enforcing* della policy

In un sistema informatico i ruoli possono essere definiti come:

- `mail_admin`
amministratore del server di posta
- `web_admin`
amministratore del server web
- `svn_user`
utente remoto a cui è concesso esclusivamente l'utilizzo di un repository SVN
- ...

Di cosa si tratta?

- Patch per i kernel della serie 2.4 e 2.6
- Layer di sicurezza aggiuntivo
- Insieme di features che concorrono alla creazione di un sistema che rispetta il principio di minimo privilegio

Sviluppo

- Nasce inizialmente come port di Openwall per Linux 2.4 nel 2001 [▶ Openwall](#)
- Ultima versione 2.2.2 per il kernel Linux 2.6.32.39 e 2.6.38.4 scaricabile sotto licenza GPL dal sito del progetto [▶ GRSecurity](#)

Documentazione disponibile sul wiki del progetto [8]

PaX

- ASLR, varie protezioni sulla memoria
- Disponibile indipendentemente da GRSecurity

Role-based Access Control

- Indipendente dal filesystem o dall'architettura usata
- Learning sull'intero sistema, su specifici ruoli o specifici soggetti
- Generazione automatica della policy
- Basso overhead

Altro

- Trusted Path Execution
- Avanzato sistema di auditing
- Restrizioni su Chroot, /proc e dmesg

Entità fondamentali

- **Ruoli**

- di tipo utente, corrispondenti ai tradizionali utenti in Linux
- di gruppo, corrispondenti al gruppo primario degli utenti Linux
- speciali, specifici di GRSecurity

- **Soggetti**

- file eseguibili

- **Oggetti**

- file
- capability
- risorse
- flag di PaX
- IP ACLs

GRSECURITY

gerarchia e transizione di ruoli in RBAC

In GRSecurity i ruoli vengono assegnati secondo la seguente **gerarchia**:

Special role → User role → Group role → Default role

Nel caso in cui nessun ruolo sia applicabile, verrà assegnato il ruolo di default

L'amministratore della policy può inoltre specificare le **transizioni** di ruolo consentite

GRSECURITY

policy di RBAC

La **policy** di RBAC è specificata in un file contenente l'insieme delle regole del sistema. Per ogni ruolo sono specificati dei soggetti, ognuno dei quali ha determinati permessi sugli oggetti assegnati.

Struttura della policy

```
role <role1> <rolemode>
<role attributes>

subject / <subject mode>
<subject attributes>
  / <object mode>
  <extra objects>
  <capability rules>
  <IP ACLs>
  <resource restrictions>

subject <extra subject> <subject mode>
<subject attributes>
  / <object mode>
  <extra objects>
  ...

role <role2> <rolemode>
...
```

GRSECURITY

policy di RBAC

Esempio di policy

```
role admin sA
subject / rvka
        / rwcamlxi

role default G
role_transitions admin
subject /
        /                r
        /opt             rx
        /home            rwxcd
        /mnt             rw
        /dev
        /dev/grsec      h
...

```

Elenco dei principali *mode* assegnabili ai ruoli

Mode	Significato
u	ruolo utente
g	ruolo di gruppo
s	ruolo speciale
A	ruolo amministrativo
G	il ruolo può usare l'utility di grsecurity per autenticarsi al kernel
I	learning abilitato per il ruolo
N	ruolo che non richiede autorizzazione

Elenco dei principali *attributi* assegnabili ai ruoli:

- `role_transitions <specialRole1> .. <specialRoleN>`
specifica a quali ruoli speciali il determinato ruolo può autenticarsi
- `role_allow_ip <IP>/<optional netmask>`
restringe l'utilizzo di un ruolo ad una lista di IP

Elenco dei principali *mode* assegnabili ai soggetti

Mode	Significato
h	processo nascosto
l	abilita il learning per il processo
o	non eredita le ACL
p	processo protetto
v	il processo può vedere i processi nascosti

Elenco dei principali *attributi* assegnabili ai soggetti:

- `user_transition_allow <user1> .. <user n>`
- `user_transition_deny <user1> .. <user n>`
- `group_transition_allow <user1> .. <user n>`
- `group_transition_deny <user1> .. <user n>`

specificano quali utenti/gruppi il determinato processo può/non può impersonare

Elenco dei principali *mode* assegnabili agli oggetti

Mode	Significato
r	oggetto accessibile in lettura
w	oggetto accessibile in scrittura e in <i>appending</i>
a	oggetto accessibile in <i>appending</i>
c	permette la creazione di file/directory
x	permette l'esecuzione dell'oggetto
d	permette la rimozione del file/directory
h	oggetto nascosto
m	permette la creazione e l'assegnamento di <i>setuid/setgid</i> a file/directory
l	permette di creare <i>hardlink</i>

Esistono altri *mode* utili per *auditing*

GRSECURITY

gerarchia di soggetti/oggetti in RBAC

Il matching delle regole avviene dal pathname più specifico a quello meno specifico

Esempio: gerarchia soggetti/oggetti nella policy

```
role marco u
subject /
  /                r
  /tmp             rwcd
  /bin             rx
  /root            h
  /home/marco      r
  /home/marco/stuff/foo  r
  ...
subject /bin/rm
  /home/marco/stuff      rwd
  ...
```

GRSECURITY

domini in RBAC

È possibile combinare diversi utenti in modo che usino la stessa policy. I **domini** funzionano come i ruoli, solo la sintassi è differente:

```
domain <domainname> u <user1> .. <userN>  
domain <domainname> g <group1> .. <groupN>
```

Esempio: definizione di un dominio

```
domain trusted_users u marco riccardo matteo  
subject /  
...
```

GRSECURITY

altre feature di RBAC

- Restrizioni sulle **capability**

`<+|-><CAP_NAME>`

- Restrizioni sulle **risorse**

`<resource name> <soft limit> <hard limit>`

- Policy sui **socket**

`connect <IP/host>/<netmask>:<portrange> <socketTypes> <protocols>`
`bind <IP/host>/<netmask>:<portrange> <socketTypes> <protocols>`

- Flag di **PaX**

`<-|+>PAX_<feature>`

GRSECURITY

utility di amministrazione

Il tool per amministrare grsecurity è **gradm**. Di seguito le principali opzioni:

-E	Abilita RBAC
-D	Disabilita RBAC
-C	Controlla la correttezza della policy
-S	Controlla lo stato di RBAC
-F	Abilita il learning system-wide
-P [rolename]	Crea la password per RBAC o per un ruolo speciale
-R	Riavvia RBAC
-L <filename>	Specifica dove salvare i learning logs
-a <rolename>	Autentica ad un ruolo speciale che richiede una password
-u	Rimuove l'utente dal corrente ruolo speciale
-n <rolename>	Effettua la transizione verso un ruolo speciale che non richiede autenticazione

ALTRE IMPLEMENTAZIONI

- **RSBAC** [9]
Avviato nel '97, utilizzato in produzione dal 2000. Patch per il kernel. Estremamente flessibile. Learning mode. Basato sul framework GFAC. Poco diffuso.
- **SELinux** [10]
Rilasciato da NSA come software opensource nel 2000. Basato sul framework FLASK. Usa LSM. Learning mode tramite sw di terze parti. Incluso nel kernel Linux [11]
- **AppArmor** [12]
Nato nel '98 con il nome *SubDomain*. Usa LSM. Learning mode. Incluso nel kernel Linux dalla versione 2.6.36.
- **TOMOYO Linux** [13]
Nato nel 2006, nel kernel Linux dal 2.6.30. Usa LSM. Learning mode. Sviluppato inizialmente per sistemi embedded.
- **SMACK** [14]
Nato nel 2007, nel vanilla dal 2.6.25. Usa LSM. Poco diffuso.

CONFRONTO IMPLEMENTAZIONI

SELinux

Meccanismi di controllo degli accessi in **SELinux**:

- *Type Enforcement* (TE)
- *RoleBased Access Control* (RBAC)
- *Identiy-Based Access Control* (IBAC)

Processi a file hanno un *security context* associato, definito come:

```
user:role:type:level
```

Esempio: security context

```
marco:staff_r:firefox_t:s0
```

```
marco:object_r:user_home_t:s0
```

CONFRONTO IMPLEMENTAZIONI

SELinux

Tipo e ruolo non sono statici (*type transitions* e *role transitions*)

Esempio: TE

```
allow system_t etc_t:file { getattr read execute };
```

assicura ai processi di tipo `system_t` l'accesso a file di tipo `etc_t` con i permessi `getattr read execute`

Esempio: RBAC

```
role system_r types { user_t staff_t system_t };
```

permette la transizione di tipo per il ruolo `system_r` in `user_t staff_t system_t`

Esempio: IBAC

```
user system_u roles { user_r staff_r system_r };
```

permette la transizione di ruolo per l'utente `system_u` in `user_r staff_r system_r`

CONFRONTO IMPLEMENTAZIONI

GRSecurity vs SELinux

SELinux

- Estremamente flessibile e potente
- Documentazione completa ed aggiornata
- Installazione e configurazione molto complesse
- Userland dev'essere patchato per supportare i security context
- No learning automatico (esistono tool esterni)

GRSecurity

- Meno flessibile di SELinux, implementa un solo modello di sicurezza (RBAC)
- Documentazione non sempre affidabile (wiki)
- Installazione e configurazione molto semplici
- Supporta qualsiasi distribuzione
- Learning mode system-wide built-in

IT'S DEMO TIME!

- [1] [Loscocco et al., 1998] P. A. Loscocco, S. D. Smalley, P. A. Muckelbauer, R. C. Taylor, S. J. Turner, and J. F. Farrell. *The Inevitability of Failure: The Flawed Assumption of Security in Modern Computing Environments*. In Proceedings of the 21st National Information Systems Security Conference, pages 303–314, Oct. 1998
- [2] [Bell, 2005] Bell, David Elliott. *Looking Back at the Bell-La Padula Model* In Proceedings of the 21st Annual Computer Security Applications Conference. Tucson, Arizona, USA. pages 337–351, Dec. 2005
- [3] [Biba, 1977] Biba, K. J. *Integrity Considerations for Secure Computer Systems* Bedford, MA: The MITRE Corporation, 1977.
- [4] [Clark et al., 1987] Clark, D. D., and D. R. Wilson. *A Comparison of Commercial and Military Computer Security Policies* IEEE Symposium of Security and Privacy, 1987, pp. 184–194.

- [5] [Brewer et al., 1989] D. F.C. Brewer, M. J. Nash *The Chinese Wall Security Policy* In Proc. IEEE Computer Society Symposium on Research in Security and Privacy, April 1989, pp. 215–228.
- [6] [Badger et al., 1996] Badger, L. et al. *A Domain and Type Enforcement Prototype* Usenix Computing Systems Volume 9, Cambridge, MA, 1996.
- [7] [NIST 2000] R. Sandhu, D.F. Ferraiolo, D, R. Kuhn *The NIST Model for Role Based Access Control: Toward a Unified Standard* Proceedings, 5th ACM Workshop on Role Based Access Control, July 26-27, 2000, Berlin, pp.47-63 <http://csrc.nist.gov/rbac/sandhu-ferraiolo-kuhn-00.pdf>
- [8] [GRSec] *Grsecurity Wiki*
<http://en.wikibooks.org/wiki/Grsecurity>
- [9] [RSBAC] *RSBAC handbook, Security Modules*
http://www.rsbac.org/documentation/rsbac_handbook/security_models

- [10] [SELinux] *Security-Enhanced Linux*
<http://www.nsa.gov/research/selinux/index.shtml>
- [11] [FLASK] *Flask: Flux Advanced Security Kernel*
<http://www.cs.utah.edu/flux/fluke/html/flask.html>
- [12] [AppArmor] newblock *AppArmor security project wiki*
http://wiki.apparmor.net/index.php/Main_Page
- [13] [TOMOYO] *TOMOYO Linux : Behavior oriented system analyzer and protector*
<http://tomoyo.sourceforge.jp/index.html.en>
- [14] [SMACK] *Home of Smack: the Simplified Mandatory Access Control Kernel for Linux* <http://schaufler-ca.com/>