RSBAC

Sicurezza Attraverso il Kernel

Andrea Pasquinucci

AIPSI
Associazione Italiana Professionisti
Sicurezza Informatica

Cosa ???

- Modello di sicurezza Unix: *Discretionary Access Control (DAC)*
- Perché non è sufficiente
- *Mandatory Access Control (MAC)*
- Il Kernel, l'hardware ed i controlli di sicurezza
- RSBAC: idea ed implementazione
- RSBAC: un esempio

Unix & Sicurezza

- UNIX
  - Logica modulare
  - Bottom Up
  - Multi Utente, Multi Tasking, Multi ...
- Sicurezza:
  - Semplice da capire e gestire
  - Mondo diviso in due:
    - Controllore
    - Controllati

Unix & Sicurezza 2

- SCOPO: **Integrità della esecuzione del codice** (accesso alle periferiche, HW, multi-... ecc.)

- ASSUME: *utenti e programmi NON sono maligni*

Unix & Utenti

- Root
  - Amministratore del sistema => "*Deus in Machina*"
  - Accede a qualunque risorsa:
    - Hardware & software
    - Qualunque dato di qualunque utente
    - Qualunque applicazione
    - Qualunque ...
- Utenti
  - Totale controllo dei propri dati
  - Limitato accesso a qualunque altra cosa

Modello sicurezza

- Modello **DAC**: *Discretionary Access Control*
- Non è possibile imporre politiche di sicurezza avanzate
  - Gli utenti possono fare quello che vogliono dei propri dati, in barba a qualunque regola
  - Usando i Gruppi è possibile organizzare gli utenti in modo da gestire le risorse e gli accessi in modo ragionevole, ma non sufficiente
  - Root ha accesso a tutto, anche a cose che non lo riguardano, ad es. i dati degli utenti
  - Troppe applicazioni *girano* come root

Unix e Sicurezza Kernel

- Come garantire l'integrità dell'esecuzione del codice e dei dati in un sistema multi-utente/multi-task?
- Kernel-mode <==> User-mode
- Kernel-mode:
  - Unico che ha accesso all'HW
  - Ridotto (?) in dimensioni, comune a ogni applicazione
- User-mode:
  - Ove *girano* le applicazioni e *vivono* gli utenti

System call

- Kernel-space e user-space **separati dall'HW**
- Un programma che gira in user-space per accedere ad una risorsa HW (disco, mouse, video, rete ...) deve:
  - Preparare i parametri per l'accesso di cui ha bisogno e scriverli in registri della CPU
  - Flippare un bit di un registro per chiamare una **system-call**
  - L'HW interrompe l'esecuzione del programma ed attiva il kernel che esegue l'accesso all'HW e ritorna i dati e l'esecuzione al programma

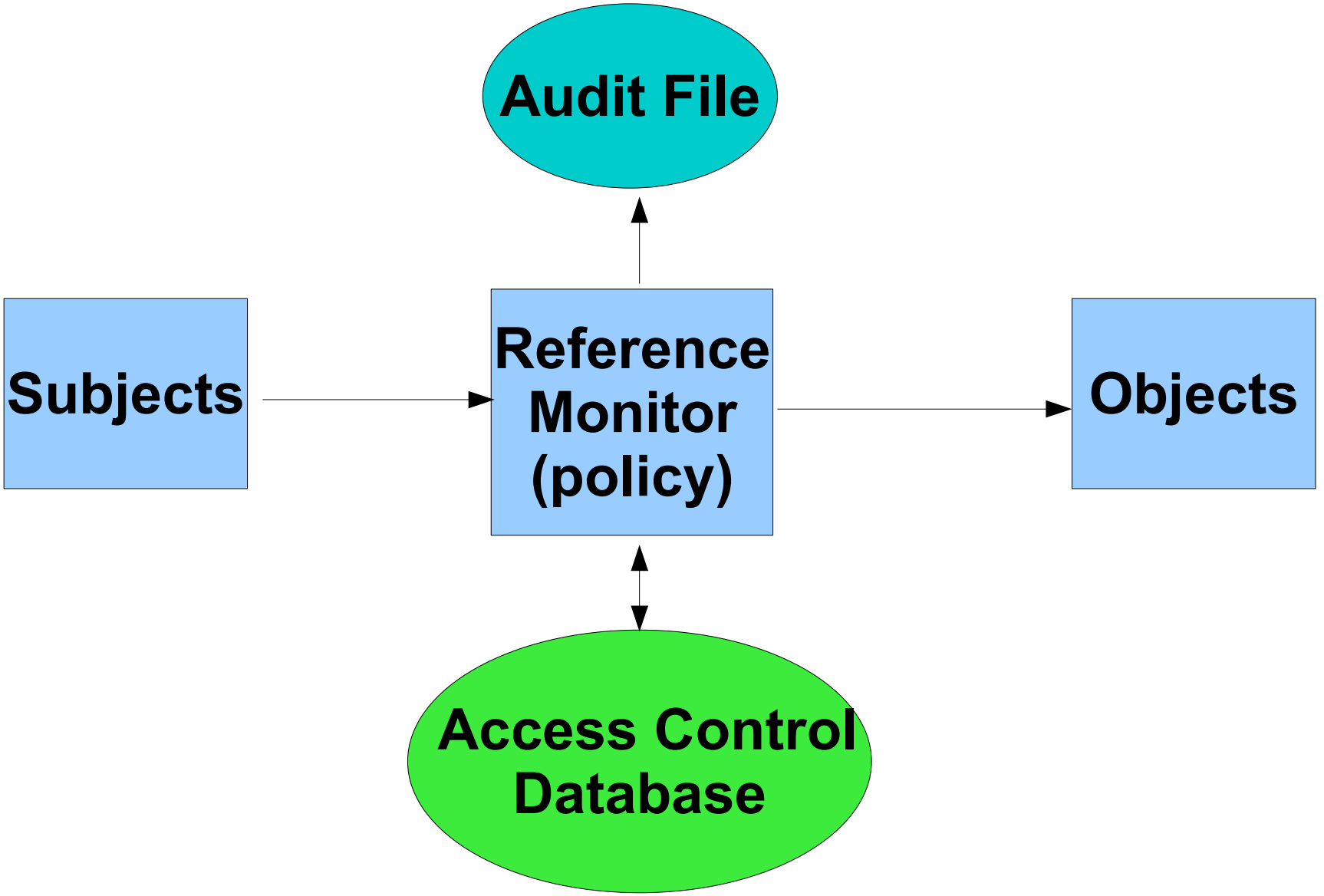
Kernel isolation

- La system-call protetta dall'HW permette al kernel di gestire le risorse in modo indipendente dalle applicazioni
  - **INTEGRITA' è OK**

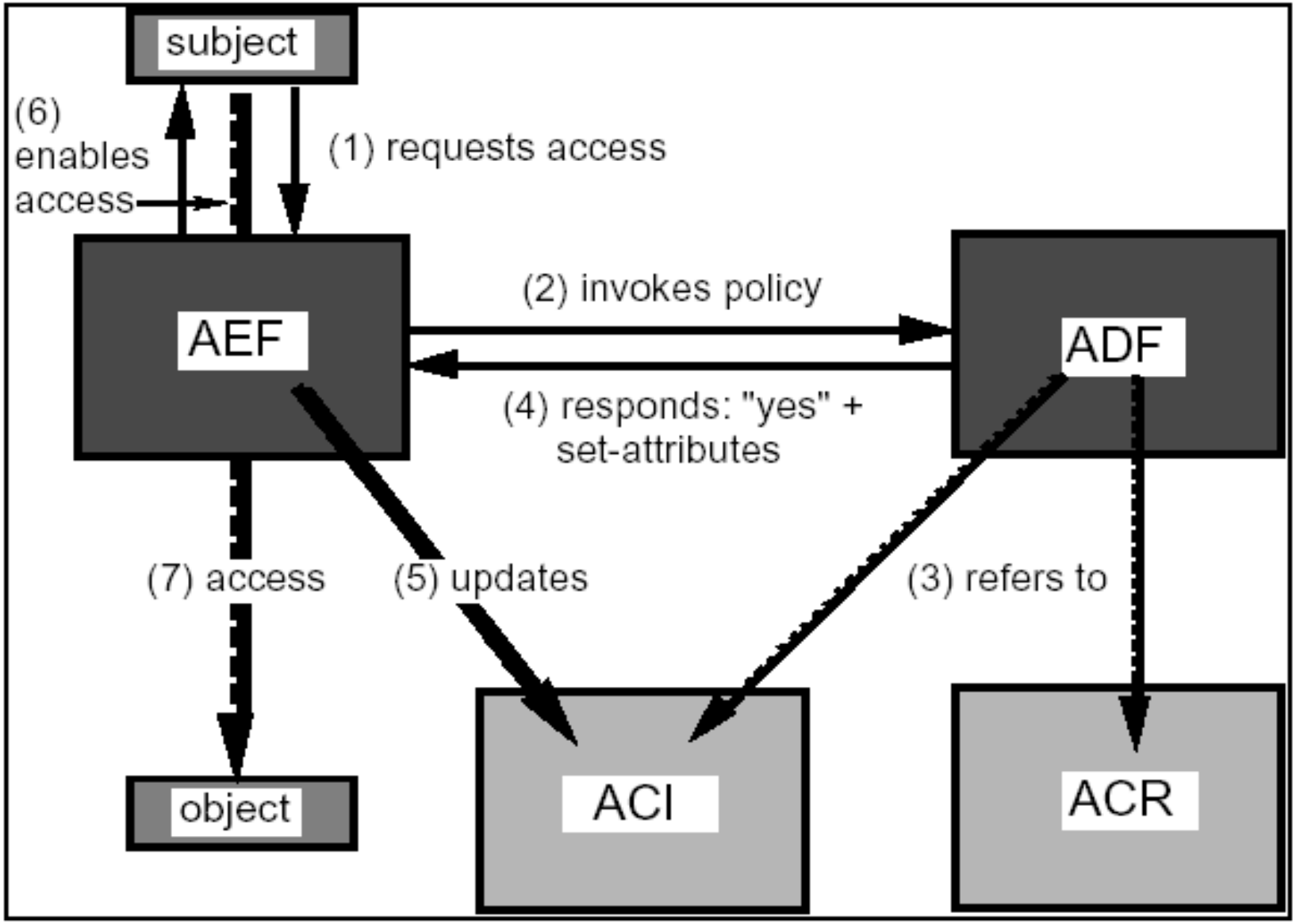
- Troppo poco per politiche di controllo accessi e sicurezza avanzate

- Ma cosa vogliamo ???

Framework di controllo MAC



Esempio: RSBAC



Dove ?

- Ovviamente **nelle system call**
  - Prima che il kernel esegua l'istruzione richiesta, viene aggiunto del codice (in kernel space) che controlla che la richiesta sia ammissibile secondo le politiche di sicurezza
    - MODIFICA DEL KERNEL
  - Controllo dettagliato
  - Controllo in un ambito ben definito
  - Controllo in un solo ambito
  - RSBAC: AEF = modifica del codice kernel delle system-call

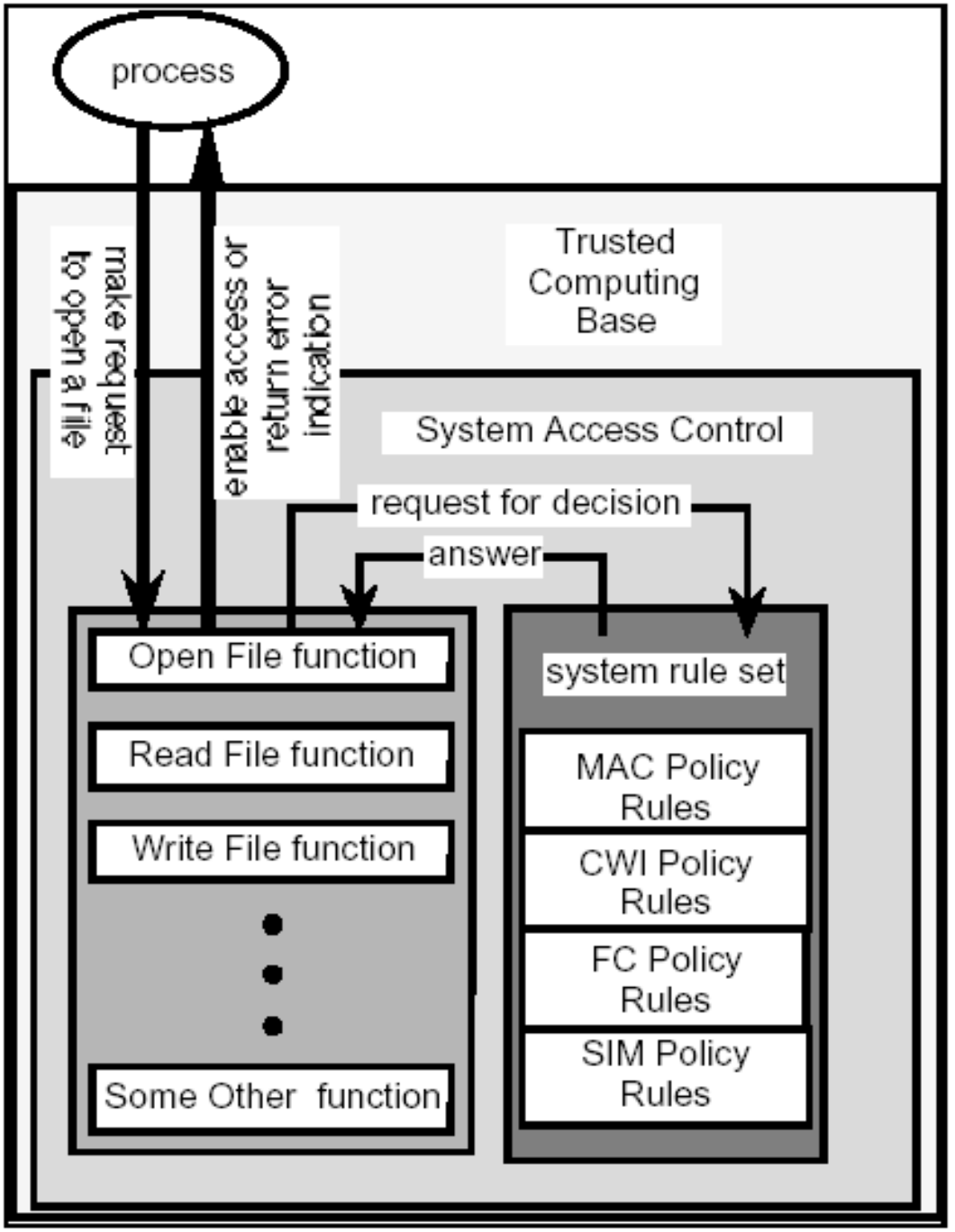
Chi ?

- 3 ruoli:
  - **Security Officer**
  - Root
  - Utenti
- Root non può:
  - Diventare Security Officer se non ne conosce la password
  - Cambiare la password del Security Officer
  - Cambiare le regole di sicurezza
- Root è un utente come un altro per le regole di sicurezza

RSBAC in a nutshell

- Struttura modulare
  - AEF: Access-control **Enforcement** Facility
  - ADF: Access-control **Decision** Facility
  - AEF dipende da piattaforma (modifica kernel)
  - ADF uguale per tutte piattaforme
  - A sua volta ADF è modulare, contiene diversi moduli decisionali
  - Il security officer può attivare i moduli ADF ritenuti utili o necessari
- Adotta GFAC (*Generalized Framework for Access Control)* e Rule Set (La Padula, http://www.acsac.org/secshelf/09.pdf)

RSBAC - 2


process
make request to open a file
enable access or return error indication
Trusted Computing Base
System Access Control
request for decision
answer
Open File function
Read File function
Write File function
Some Other function
system rule set
MAC Policy Rules
CWI Policy Rules
FC Policy Rules
SIM Policy Rules

Logica di ADF

- risposta positiva solo se almeno un modulo lo permette e nessuno lo vieta esplicitamente
- Le decisioni sono prese sulla base
  - della richiesta
  - delle regole presenti
  - dello stato del sistema (ora, quali file aperti, cosa sta facendo il processo ecc.)
- Una risposta positiva aggiorna il ACI (stato del sistema di AC)
- Una risposta negativa viene segnalata come errore

Moduli Principali

- AUTH gestione alcune capabilities per processi (modulo sempre richiesto)
  - auth_may_setuid: setuid verso any UID
  - auth_capabilities: può setuid verso {lista di UID}
- FF File Flags: assegna proprietà ad interi file-system, ad esempio si può marcare */home* come no_exec
- ACL permette di specificare le ACL per l'accesso ad un file da parte di un qualunque utente, la lista di permessi è lunga...

Permessi ACL

- ADD_TO_KERNEL - for kernel modules (drivers)
- ALTER - Change control information for IPC
- APPEND_OPEN
- CHANGE_GROUP
- CHANGE_OWNER
- CHDIR
- CLONE-clone() or fork() call.
- CLOSE
- CREATE
- DELETE
- EXECUTE
- GET_PERMISSIONS_DATA-get UNIX permissions.
- GET_STATUS_DATA- stat() system call.
- LINK_HARD
- MODIFY_ACCESS_DATA-Modify access time.
- MODIFY_ATTRIBUTE-Modify RSBAC attribute.
- MODIFY_PERMISSIONS_DATA- Modify UNIX permissions.
- MODIFY_SYSTEM_DATA- Change system data (ports,...).
- MOUNT
- READ
- READ_ATTRIBUTE- Read RSBAC attribute.
- READ_OPEN
- READ_WRITE_OPEN
- REMOVE_FROM_KERNEL- for kernel modules (drivers)
- RENAME
- SEARCH
- SEND_SIGNAL-send signal to other process
- SHUTDOWN
- SWITCH_LOG-switch RSBAC log levels.
- SWITCH_MODULE-switch RSBAC module.
- TERMINATE- terminate process
- TRACE- trace process (ptrace() call).
- TRUNCATE
- UMOUNT
- WRITE
- WRITE_OPEN
- .....

Role Compatibility

- Role: lista di soggetti
- Target: lista di oggetti [tipi: FILE, DIR, DEV, IPC (pipe,...), SCD (System Control Data: ports, kernel logs...), USER, PROCESS, NONE, FD (file descriptor: FILE ∪ DIR), NETDEV, NETTEMP, NETOBJ, NETTEMP_NT]
- Request: azione (come ACL pagina precedente)
  - Crea un nuovo Target-Type (usando RC_TYPES)
  - Crea un nuovo Role-Type (usando RC_ROLES)
  - Assegna al Role-Type le azioni volute sul Target-Type appena creato
  - Metti i file di interesse nel Target-Type
  - Metti i programi (file) di interesse nel Role-Type

Altri Moduli

- MAC Bell-La Padula
- FC Functional Control (simple separation of duties)
- SIM Security Information Modification(read_only data)
- PM Simone Fischer-Hübner's Privacy Model (protect data according to German and EU directive on privacy)
- DAZ Malware/Antivirus Scan
- CAP Linux Capabilities
- JAIL chroot enhancement
- PAX PaX anti buffer-overflow module
- ...

Esempio: RC Apache AppendOnly

```
# Crea Ruolo Apache
COPYORIGINROLE=0  # general user
ROLE=`rc_get_item list_unused_role_nr | head -n 1`
rc_copy_role $COPYORIGINROLE $ROLE
rc_set_item ROLE $ROLE name 'Role_Apache'

# Crea Tipo Apache File
TYPE=`rc_get_item list_unused_fd_type_nr | head -n 1`
rc_set_item TYPE $TYPE type_fd_name 'Apache_AO_FD'

# Assegna a ROLE i diritti su TYPE
rc_set_item -a ROLE $ROLE type_comp_fd $TYPE
  APPEND_OPEN CHANGE_OWNER CHDIR CLOSE CREATE
  GET_PERMISSIONS_DATA GET_STATUS_DATA
  MODIFY_ACCESS_DATA MODIFY_PERMISSIONS_DATA READ
  READ_ATTRIBUTE READ_OPEN SEARCH LOCK WRITE
```

Esempio: RC Apache - 2

```
# root è readonly, gli altri utenti NULLA
rc_set_item -a ROLE $Role_System_nr type_comp_fd $TYPE
  CHDIR CLOSE GET_PERMISSIONS_DATA GET_STATUS_DATA
  READ READ_ATTRIBUTE READ_OPEN SEARCH

# assegna /usr/sbin/httpd al ruolo ROLE allo startup
attr_set_file_dir RC FILE /usr/sbin/httpd
  rc_force_role $ROLE

# assegna la dir /var/http/ al tipo TYPE
attr_set_file_dir DIR /var/http/ rc_type_fd $TYPE
```

Risultato

Nella directory /var/http/

- httpd può leggere e scrivere creando nuovi file o appendendo a file già esistenti
- httpd non può cancellare o modificare file già esistenti
- Root può leggere i file
- Nessun altro utente ha accesso a /var/http/
- *Nessuno può mai cancellare o modificare alcun file in /var/http/ !*

Grazie !

Altre informazioni:

http://www.aipsi.org/

http://www.ucci.it/

http://www.rsbac.org/

Andrea Pasquinucci

a.pasquinucci-At-aipsi.org www.aipsi.org

pasquinucci-At-ucci.it www.ucci.it

Copyright e Licenza