Proteggere i files sensibili con SE-Linux

Giuseppe Paternò

Agosto 2008

Questa pubblicazione è dedicata a mio nonno Francesco Stella,
la prima persona che ha creduto in me
e nelle mie capacità.

Sommario

Scopo del documento

Un sistema ad accesso mandatorio, quale SE-Linux, permette di proteggere in maniera esaustiva un computer contenente files e dati sensibili. La sua gestione è però molto onerosa in quanto è necessario possedere delle solide conoscenze non solo dell'ambiente Linux, ma anche di sicurezza, sistemi mandatori e come funziona al suo interno SE-Linux. Spesso però da un lato non si necessita di tanta sicurezza, dall'altro non si dispone di sufficiente personale altamente specializzato per riuscire a gestire sistemi con dati sensibili. Questo whitepaper, basato su un caso reale, vuole illustrare una metodologia che coniuga alta sicurezza con una gestione di un ambiente unix non differente da quella tradizionale. Il documento si rivolge agli amministratori di sistema che abbiano solide basi di Linux e di sicurezza informatica, con una conoscenza di base di SE-Linux.

Biografia dell'autore

Giuseppe Paternò è un esperto di sicurezza informatica e di architetture dei datacenters. Ha conseguito le certificazioni CCNP, RHDSS, RHCA e RHCDS ed è membro della Italian Linux Society, oltre ad essere un membro attivo della comunità di *sikurezza.org*. La sua passione ha spinto Giuseppe ad esplorare fin da giovanissimo tutti i settori dell'informatica, con particolare riguardo al settore della sicurezza e delle reti, senza tralasciare le nuove sfide tecnologiche.

Attualmente lavora come Solution Architect ed EMEA Security Expert per Red Hat, partecipando ai progetti di sicurezza, ma nel suo passato spiccano esperienze di lavoro interessanti tra cui Sun Microsystems, IBM e Wind/Infostrada.

Maggiori informazioni sono disponibili sul sito *http://www.gpaterno.com/* e su LinkedIn *http://www.linkedin.com/in/gpaterno.*

Nota di copyright

Commenti e suggerimenti

Qualsiasi commento o suggerimento é benvenuto e può essere inviato via e-mail a *info@gpaterno.com*. Successive edizioni ed eventuali correzioni di questa pubblicazione saranno disponibili sul sito Internet *http://www.gpaterno.com/.*

Introduzione

Mi chiedono spesso se sia possibile utilizzare SE-Linux per proteggere certe aree del filesystem o per proteggere determinate applicazioni, assimilando SE-Linux ad una specie di “firewall applicativo”. Sento spesso usare il termine “firewall applicativo” parlando di SE-Linux, ed alcune volte ammetto di averlo fatto anch'io, anche se si tratta di un concetto sbagliato. Faccio un passo indietro, facendo un breve riassunto di SE-Linux[1]. Security Enhanced Linux, o per brevità SE-Linux, è un set di patches del kernel di Linux sviluppate dalla US National Security Agency (NSA) per portare un sistema ad accesso mandatorio (Mandatory Access Control, MAC) in ambiente Linux, similmente ad altre piattaforme unix quali Trusted Solaris. SE-Linux si appoggia alla modularità di sicurezza del kernel Linux chiamata Linux Security Modules (LSM), così come si appoggiano altri sistemi di sicurezza quale ad esempio AppArmor[2].

Non mi soffermerò a parlare della storia di SE-Linux, ma vorrei presentarvi un caso concreto di un cliente che possa essere facilmente riadattato per quei sistemi che debbano ospitare files sensibili. Nel caso in questione, l'obiettivo è di proteggere alcuni files in formato PDF estremamente sensibili a cui un'applicazione web-based deve accedere, ma a cui i sistemisti di gestione non possono in alcun modo accedere. L'applicazione risiede in un cluster in bilanciamento a tre nodi e i files acceduti tramite GFS[3], il filesystem clustered presente nella distribuzione Red Hat Enterprise Linux (vedere l'architettura in figura 1). In molti dei casi basta applicare una buona protezione usando le metodologie DAC[4], ovvero permessi a filesystems e Posix ACL, ma in questo caso era necessario essere sicuri delle protezioni ed era necessario effettuare log di accesso.

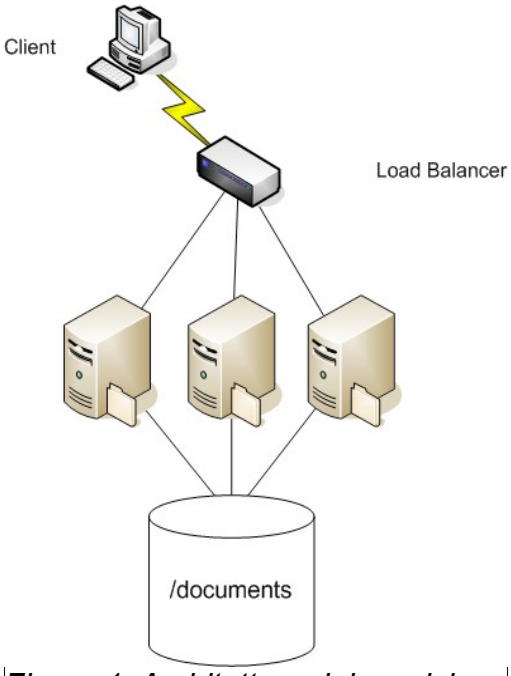

Figura 1: Architettura del servizio

Nota Bene. Non siamo entrati nel merito della protezione del PDF in se stesso. In realtà in una buona filosofia “Defense in Depth” sarebbe opportuno proteggere ogni singolo layer e non solo il sistema operativo. Se per caso riescono a trovare una falla applicativa e riescono a scaricare il PDF, esso risulterebbe in chiaro. Le ipotesi sarebbero o criptare con una random password il PDF (e comunicarlo al destinatario in maniera sicura) oppure utilizzare l'infrastruttura di protezione con certificati digitali.

1 SE-Linux è ospitato dall'NSA sul sito *http://www.nsa.gov/selinux/*
2 AppArmor è disponibile sul sito *http://forge.novell.com/modules/xfmod/project/?apparmor*
3 GFS supporta gli extended attributes (xattr), che di solito permettono di immagazinare le informazioni relative alle ACL. SE-Linux usa proprio gli extended attributes per scrivere le informazioni relative ai file-contexts. Se si vuole utilizzare un altro filesystem, è necessario capire se questo supporti gli extended attributes.
4 Discretionary Access Control (DAC) è quella normalmente presente nel sistema operativo unix, che si contrappone al Mandatory Access Control

I requisiti che il cliente mi ha dato e alcuni aggiunti da me che io stesso mi sono voluto imporre erano:

- Univoca identificazione degli utenti (accesso LDAP)
- I log relativi a audit/sicurezza vanno inviati ad un syslog centralizzato
- Le utenze di gestione non possono diventare root, ma devono eseguire i soltanto alcuni comandi tramite il programma “sudo”
- Le utenze di gestione devono avere diversi privilegi di accesso, da operatore a system administrator autorizzato
- Gli utenti non possono fare “su -” per accedere all'utenza root anche se conoscono la password, tranne quelli esplicitamente autorizzati
- Nessun utente, tranne *root* e l'utente applicativo *appserv* possono accedere alla directory “*/documents/*” e relativi documenti/subdirectories
  - L'utente *root* ha diritto di accedere alla directory, ma le operazioni di lettura devono essere sottoposte ad audit
  - L'utente applicativo non deve essere sottoposto ad audit per la salvaguardia delle performance dell'applicazione.
  - Con l'utente applicativo *appserv* verrà eseguito l'application server e verranno eseguiti alcuni batch che hanno diritto all'accesso alla directory

La scelta implementativa

Ho voluto realizzare un compromesso tra sicurezza e manutenibilità del sistema. Nei sistemi di derivazione Red Hat, ovvero Fedora, CentOS e Red Hat Enterprise Linux, esistono due tipologie di policy preconfezionate disponibili con la distribuzione: la *targeted* e la *strict*. La modalità targeted ha il compito di proteggere e confinare i daemons, lasciando aperto tutto quello che non è esplicitamente confinato (quello che non è esplicitamente confinato è chiamato anche *unconfined_t*). Nella modalità strict ogni singolo programma e ogni singolo “ambiente” è confinato, dovendo impostare delle regole di abilitazione (si chiamano *domain transition*).

Sono stato combattuto tra quale delle due modalità utilizzare: sicuramente la prima è più facile da gestire, ma ha dei problemi a “confinare” i system administrators, mentre la seconda offre una sicurezza senza dubbio maggiore a livello militare, ma necessita di una preparazione sistemistica notevole, che normalmente operatori non hanno.

La scelta è stata un compromesso tra tool di sistema, SE-Linux in modalità targeted e l'introduzione del Multi-Category Security (MCS). MCS è la possibilità di assegnare ad un file una determinata categoria ed assegnare a ciascun utente l'abilitazione alla lettura/scrittura su quella determinata categoria, in conformità con il concetto “*no read up, no write down”* presente nel modello Bell-LaPadula[5]. Ad esempio, si possono mettere i files in categorie come “Confidential”, “Top Secret” o “HR Department” ed assegnare all'utente *pippo* i diritti di accesso a determinate categorie.

5 Maggiori info sul modello Bell-Lapadula su http://en.wikipedia.org/wiki/Bell-LaPadula_model

Nel documento vengono date per assunti i seguenti passi:

- Installazione miminale del sistema operativo
- Hardening di base e configurazione dell'autenticazione
- Attivazione di SE-Linux con le policy targeted: nel file */etc/sysconfig/selinux* assicurarsi di avere le seguenti righe
  - SELINUX=enforcing
  - SELINUXTYPE=targeted

Creazione della categoria

Per proteggere i files usando MCS ho dovuto definire una categoria che per comodità ho chiamato *TopSecret* per etichettare sia le directories che i files sensibili. Questa definizione è stata implementata nel file */etc/selinux/targeted/setrans.conf* aggiungendo la seguente riga:

```
s0:c3=TopSecret
```

Tutti i files e le directories dovranno avere questa label per poter essere protetti in modo corretto. Per dare questa attribuzione bisogna utilizzare il comando *chcat* come dal seguente esempio:

```
chcat +TopSecret <nome del file>
```

Il comando è da eseguire per ogni file da proteggere, il che potrebbe essere svantaggioso se per caso un sistema automatico non lo prevede. Nel mio caso, ho deciso di implementarlo direttamente usando i file-contexts, in modo da usare il comando *restorecon* in maniera ricorsiva, come descriverò nel paragrafo successivo.

Implementare il modulo SE-Linux

Sulle distribuzioni di derivazione RedHat è necessario avere installato il pacchetto *selinux-policy-devel* che installa l'ambiente di sviluppo dei moduli SE-Linux. Prima di procedere a creare e compilare un modulo SE-Linux, è necessario disabilitare temporaneamente la “blindatura” (enforcing) di SE-Linux, mettendolo in modalità permissive, cioè solo in log, con il comando:

```
# setenforce 0
```

Nota bene. Alla fine di questo capitolo, ricordatevi di rimettere in enforcing il sistema con il comando “*setenforce 1”*. Potete vedere lo stato di SE-Linux con il comando “*sestatus*”.

Ho creato una directory, es: *docsecret*, e ho creato due files al suo interno:

- *docsecret.te*, che contiene le policy e la definizione dei nuovi tipi
- *docsecret.fc*, che contiene i contesti da applicare i files

I sorgenti di entrambi i files sono nelle *Appendici A e B*. Il file *docsecret.te* contiene la definizione del tipo *docsecret_t* che ci servirà per identificare univocamente i files, proteggendoli dai daemons confinati dalla policy targeted di SE-Linux. Inoltre contiene i permessi per attribuire la label di *docsecret_t* da parte dei sistema (nel caso di autorelablel del filesystem o per il restorecon).
Compiliamo il modulo con il comando:

```
# make -f  /usr/share/selinux/devel/Makefile
```

e lo carichiamo nelle policy con:

```
# semodule -i docsecret.pp
```

Il modulo sarà permanete attraverso i vari reboot perchè *semodule*, oltre a caricarlo in memoria, provvede a copiare il file nella directory */etc/selinux/targeted/modules/active/modules*. Con il comando "*semodule -l*" si possono visualizzare i moduli di SE-Linux caricati in memoria. Abbiamo due opzioni adesso per attribuire i corretti attributi SE-Linux alla directory dei documenti, utilizzare l'autorelabel del filesystem o il comando *restorecon*. L'autorelabel ha il compito di "rimettere a posto" le labels SE-Linux a secondo dei file-contexts specificati nella policy SE-Linux attiva in quel momento. L'operazione è semplicissima:

```
# touch /.autorelabel
# reboot
```

Al riavvio di sistema, gli script di start-up delle distribuzioni di derivazione Red Hat provvederanno a rimettere a posto le labels. Gli script comunque si basano sul comando *restorecon* che possiamo invocare direttamente. Il comando prende le impostazioni dei file-context in memoria e li applica sul file o ricorsivamente su una directory, ad esempio:

```
# restorecon -ivvr /documents
```

Vediamo nel dettaglio gli switches usati: *-r* significa applica ricorsivamente, *-v* (o piu' lettere *v*) aumentano la verbosity dell'output, *-i* ignora i files che non esistono. Se avete un utente non root, ad esempio l'utente *admin*, noterete che provando a visualizzare il contenuto della directory root con il comando *"ls -laZ /"* accanto alla dicitura "documents" ci sono dei punti di domanda. Avete appena confinato gli utenti che non hanno accesso alla label *TopSecret*!! Se provate a fare "*cd /documents*", anche se i permessi della directory sono 644, avrete un bel "permission denied"!

Abilitare l'utente applicativo

Abbiamo confinato gli utenti di sistema, ma come abilitare l'utente applicativo *appserv* ad accedere ai documenti? Di default tutte le utenze non hanno accesso a nessun file che abbia una categoria associata, con l'eccezione dell'utente *root*. Per abilitare un utenza ad accedere alla label di *TopSecret*, ad esempio per l'utenza applicativa *appserv*, eseguiamo il seguente comando

```
# chcat -l +TopSecret appserv
```

Per verificare che all'utente sia stato assegnato il livello corretto, si può eseguire il comando *semanage* come dal seguente esempio:

```
# semanage login -l
Login Name                SELinux User              MLS/MCS Range
__default__               user_u                    s0
appserv                   user_u                    -TopSecret
root                      root                      SystemLow-SystemHigh
```

In realtà il comando *chcat* funge da wrapper per i comandi *chcon* e *semanage*; consiglio di approfondire questi due comandi anche solo attraverso le due man pages.

Personalizzazione di sudo

Abbiamo attribuito all'utente *appserv* che eseguirà il nostro application server la label per accedere ai files con il flag *TopSecret*. Ora dobbiamo abilitare gli utenti amministrativi, ad esempio *admin*, ad eseguire i programmi preposti per il loro ruolo, nel mio caso lo shell script di avvio e stop del container Jboss, oltre che ai comandi *reboot* e *poweroff*.
Non mi soffermerò sulla configurazione di sudo, ma c'è un'aspetto di sudo che è molto importante: l'esecuzione come utente *root* non fa perdere la label di sicurezza di SE-Linux. Proviamo ad esempio ad aggiungere in */etc/sudoers* le seguenti linee:

```
admin      ALL= NOPASSWD:/usr/bin/id
admin      ALL= NOPASSWD:/bin/cat
```

Se proviamo a loggarci sul sistema con l'utente *admin* e digitiamo il comando "*sudo id*" otterremo il seguente risultato:

```
$ sudo /usr/bin/id
uid=0(root) gid=0(root)
groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel)
context=user_u:system_r:unconfined_t
```

Avete notato il context associato al nostro utente? Normalmente l'utente root ha come context *root:system_r:unconfined_t:SystemLow-SystemHigh.* L'utente admin, anche se tramite sudo è diventato root (uid=0), ha mantenuto il contesto SE-Linux da utente (*user_u)*. Una prova interessante è la seguente:

```
$ sudo cat /documents/test
cat: /documents/test: Permission denied
```

Notiamo quindi che, nonostante diventiamo *root*, l'utente *admin* non ha accesso alla directory protetta attraverso SE-Linux.

Abbiamo ottenuto quello che volevamo: fare gestire in maniera semplice il sistema, pur mantenendo la confidenzialità dei files contenuti nella directory */documents/*. L'operazione negata verrà loggata nel sottosistema di audit, ovvero nel file */var/log/audit/audit.log*; discuteremo successivamente del subsystem di audit/log e come configurarlo efficientemente.

Nota bene: E' bene non abilitare tutti i comandi all'utente o al gruppo attraverso sudo! Gli utenti di gestione NON devono accedere al subsystem SE-Linux o al subsystem di audit, in particolare ai comandi *setenforce* e *auditctl*. Date solo i permessi ai comandi o al gruppo di comandi che vi interessano.

Ulteriori restrizioni di accesso per "root"

Ricordiamo che l'utente *root* comunque rimane attivo ed ha la possibilità di eseguire comandi strategici, quali ad esempio mettere SE-Linux in permissive o disabilitare il sottosistema di audit. Dobbiamo quindi salvaguardare l'accesso diretto a root:

- Modificando il comportamento del comando "su"
- Configurando in maniera opportuna il daemon di ssh

Come da requisito, abbiamo deciso di lasciare comunque la possibilità di eseguire il comando *su* ad un gruppo di persone. Cancellare il comando *su* dal sistema o evitarne la sua esecuzione potrebbe essere problematico in quanto alcuni cron jobs eseguono a loro volta il comando *su*. Anche alcuni batches potrebbero eseguire *su,* quindi il comando è da mantenere. Per facilità è possibile sfruttare il gruppo *wheel* già presente nel sistema, che ha anche un modulo pam associato (*pam_wheel*). Ho configurato il file di pam per il comando *su*, ovvero */etc/pam.d/su*, nel seguente modo:

```
#%PAM-1.0
auth       sufficient pam_rootok.so
auth       required   pam_wheel.so use_uid
auth       include         system-auth
account    sufficient pam_succeed_if.so uid = 0 use_uid quiet
account    include         system-auth
password   include         system-auth
session    include         system-auth
session    optional   pam_xauth.so
```

In questo modo solo gli appartenenti al gruppo wheel possono eseguire il comando "su", anche se altri utenti conoscono la password di root.

Il successivo passo è quello di disabilitare l'accesso diretto all'utente root tramite SSH, agendo su */etc/ssh/sshd_config* impostare *PermitRootLogin no.* E' bene anche circoscrivere gli utenti abilitati all'ingresso interattivo tramite ssh con le opzioni AllowUsers e/o AllowGroups.

Volendo potremmo restringere ulteriormente l'accesso a *root*, lasciando nel file */etc/securetty* soltanto la console e le virtual console su cui *root* può collegarsi.

Subsystem di audit

Di default SE-Linux tiene traccia di tutti I tentativi di accesso a risorse non autorizzate (deny logs), ad esempio se il nostro utente amministrativo *admin* tenta di accedere alla directory *"/documents"*. Uno dei miei requisiti era anche poter tracciare l'utente *root* che ha diritto di accedere alla directory per scopi amministrativi: sarebbe molto sospetto se *root* cominciasse ad aprire un certo numero di PDF. Come fare quindi? Il sistema operativo Linux dispone di un sottosistema di audit molto flessibile, capace di intercettare le chiamate al sistema e riportarle in un file di log. Inoltre tramite un modulo chiamato *audispd*[6], è possibile redirezionare i log del sottosistema di audit nel syslog, quindi inviandoli ad un syslog remoto nel caso un utente smaliziato provasse a cancellare I logs di audit.

Il sistema è stato personalizzato agendo sul file */etc/audit/audit.rules* inserendo la riga seguente:

```
-a exit,always -S open -S truncate -F dir=/documents -F uid!=300
```

In breve, questo comando tiene traccia degli eventuali accessi attraverso le chiamate (syscall) *open* e *truncate* alla directory */documents* se la userid non è equivalente alla UID 300, ovvero la UID che abbiamo dato all'utente applicativo *appserv*.

In questo modo possiamo loggare qualsiasi tentativo di accesso dagli utenti alla directory contenente i files sensibili. Anche se non è possibile accedervi da utenti di gestione/ operatori perchè "blindati" dalle policy SE-Linux, questa regola permette anche di loggare gli accessi da utenti autorizzati (es: root): in questo modo si ha il totale controllo di chi sta facendo cosa.

Per configurare Il plugin di log a syslog (audispd) dobbiamo agire sul file */etc/audisp/plugins.d/syslog.conf* e specificando "*active = yes*".

6 Il plugin audispd per syslog è disponibile nel pacchetto *audispd-plugins*

Uso delle ACL

Come dicono quelli che si occupano di sicurezza, essere paranoici non è mai abbastanza. Non sarebbe uno “spreco” lasciare i permessi di lettura a tutti sulla directory */documents* anche se siamo coperti da SE-Linux? Cosa succede se per sbaglio viene disabilitato SE-Linux, es: digitando per troubleshooting *setenforce 0*?
Nel caso di studio ho comunque deciso di applicare delle ACL di default sulla directory */documents,* in modo che comunque i files vengano anche protetti da un normale sistema DAC. Nel dettaglio ho eseguito i seguenti comandi:

```
# chmod 0750 appserv:appserv /documents/
# setfacl -m appserv:rwx /documents/
# setfacl -m root:rwx /documents/
# getfacl --access /documents/ | setfacl -d -M- /documents/
```

Conclusione

SE-Linux è sicuramente un sottosistema interessante, ma è molto difficile da gestire anche per gli utenti esperti. E' importante sapere utilizzare le appropriate tecnologie nell'appropriato contesto: usare SE-Linux potrebbe essere superfluo su un sistema di database cui unico accesso è effettuato tramite il protocollo del database e l'accesso SSH è limitato alla sola rete di management e ai soli amministratori. Tuttavia potrebbe essere una buona idea poterlo usare sui sistemi di “frontiera” su cui transitano gli accessi degli utenti o su cui ci siano files sensibili da proteggere, come nel caso descritto in questo whitepaper.

Ci tengo a sottolineare quanto sia importante adottare l'approccio “Defense in Depth” nel pensare alla sicurezza, e che la sicurezza è più importante quanto è più vicino al dato da proteggere.

Appendice A: docsecret.te

```
policy_module(docsecret,1.0.20)

require {
        type unconfined_t;
        type restorecon_t;
        type setroubleshootd_t;
        type fs_t;
        type default_t;
}

type docsecret_t;
files_type(docsecret_t)

## for allowing restorecon
allow docsecret_t fs_t:filesystem associate;

allow restorecon_t docsecret_t:dir { read relabelfrom relabelto
getattr };
allow restorecon_t docsecret_t:file { read relabelfrom relabelto
getattr };
allow unconfined_t docsecret_t:dir rw_dir_perms;
allow unconfined_t docsecret_t:file rw_file_perms;
```

Appendice B: docsecret.fc

```
##
## Directory principale dei documenti
##

/documents(/.*)?      gen_context(system_u:object_r:docsecret_t,s0,c3)
```