Introduzione a SELinux

Corso di formazione sulla Sicurezza Informatica

CCR INFN 4-5 nov 2015, Arcetri

Stefano Antonelli- CNAF Bologna

SELinux bene o male?

From: Theodore Tso <tytso@mit.edu>
Newsgroups: fa.linux.kernel
Subject: Re: [PATCH] Version 3 (2.6.23-rc8) Smack: Simplified Mandatory Access
Date: Mon, 01 Oct 2007 19:01:50 UTC
...
SELINUX is so horrible to use, that after wasting a large amount of time enabling it and then watching all of my applications die a horrible death since they didn't have the appropriate hand-crafted security policy, caused me to swear off of it. ...

From: Linus Torvalds <torvalds@linux-foundation.org>
Newsgroups: fa.linux.kernel
Subject: Re: [PATCH] Version 3 (2.6.23-rc8) Smack: Simplified Mandatory Access
Date: Tue, 02 Oct 2007 23:28:01 UTC
...For example, I find SELinux to be so irrelevant to my usage that I don't use it at all. I just don't have any other users on my machine, so the security I care about is in firewalls etc. And that really *is* fundamentally different from a system that has shell access to its users...The problem with SELinux isn't the theory. It's the practice. IOW, it's too hard to use.

...this action by the NSA was the crypto-equivalent of the Pope coming down off the balcony in Rome, working the crowd with a few loaves of bread and some fish, and then inviting everyone to come over to his place to watch the soccer game and have a few beers. There are some things that one just never expects to see, and the NSA handing out source code along with details of the security mechanism behind it was right up there on that list...Larry Loeb, “Uncovering the secret of SELinux (2001)”

SELinux - cenni storici

SELinux nasce da un progetto della NSA con la collaborazione di Secure Compute Corporation (SCC) e MITRE

Il codice viene rilasciato nel dic 2000 con licenza GPL con un comunicato stampa speciale (https://www.nsa.gov/public_info/press_room/2001/se-linux.shtml)

Il progetto è la fusione di progetti in corso da una decina di anni: DTMach, DTOS, Fluke, Flusk, Flask

da FLASK, eredita il Mandatory Access Control (MAC) basato sulla tecnologia Type Enforcement (TE) già testato ed utilizzato in LOCK (installato in molti centri militari)

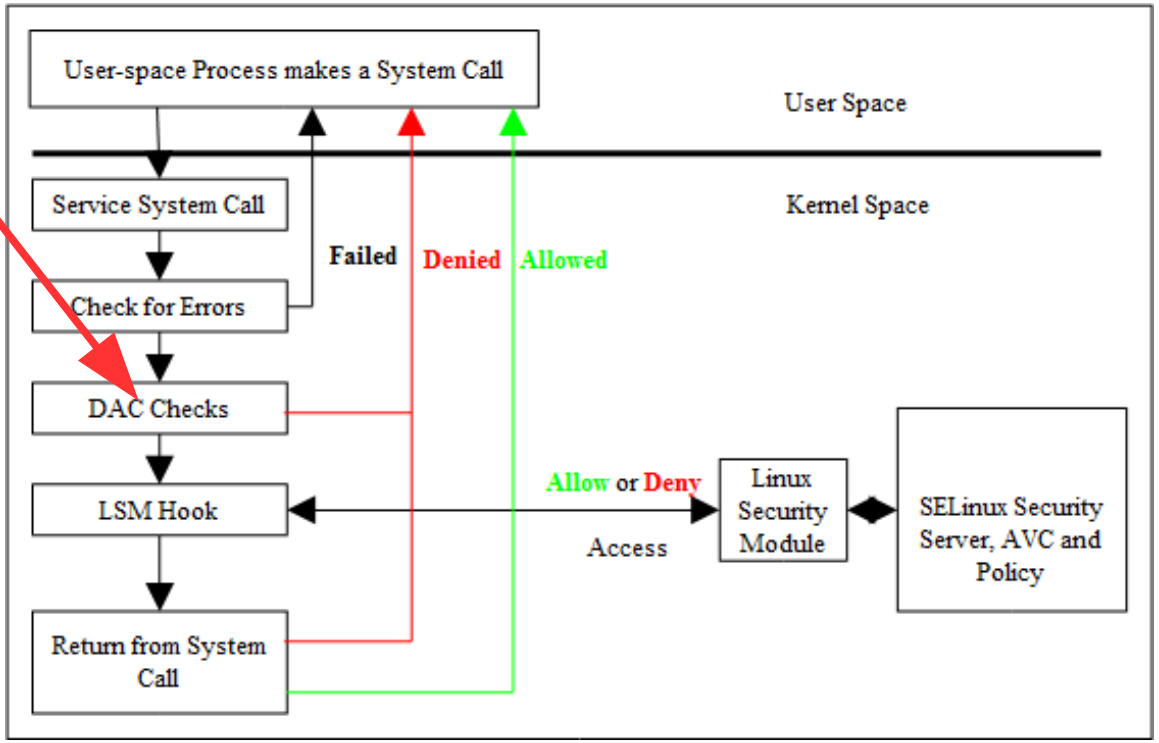
Distribuito come update del kernel 2.2 e 2.4; alla richiesta di L.Torvalds di modificare il prog per “modulizzare” il sistema di sicurezza Linux, viene creato Il sottosistema Linux Security Modules (LSM). Questa soluzione consente di poter scegliere tra diversi MAC (AppArmor, SELinux, TOMOYO ancora in sviluppo, SMACK ???)

SELinux - architettura

SELinux è uno dei sistemi di sicurezza, di controllo d'accesso, che “poggia” sul Linux Security Module (LSM)

E' un MAC, a differenza del DAC (Discretionary AC) nativo di Linux, ed agisce in kernel space dopo il DAC.
I controlli d'accesso sono estesi con il Role Based Access Control (RBAC)

L'idea su cui si basa è quella del “minimo privilegio” cioè quella di fornire al processo o all'utente I permessi minimi per eseguire l'azione richiesta; può essere utile nei casi di 0-day exploit



Il TE è uno dei modi di sicurezza; ci sono anche il Multi-Level Security (MLS) ed il Multi-Category Security (MCS) (modello BLP “*no write down, no read up*” in contrasto al modello BIBA “*no write up, no read down*”)

SELinux – jargon

- user: esistono utenti di default in SELinux. Ogni utente Linux è mappato in uno o più utenti SELinux (# `semanage login -l`). L'utente SELinux (# `semanage user -l`) è diverso da quello Linux e può essere visto come un gruppo o classe di utenti; non cambia durante una sessione. E' il primo campo del “security context”, _u
- object: è qualsiasi risorsa del tipo file, directory, porta, socket, servizio, etc. a cui si accede tramite un processo o subject. E' classificato secondo la risorsa che fornisce insieme ai permessi che ne descrivono le caratteristiche
- subject: è una entità attiva; utente, processo o device che può agire su un object; ha associato un security context e può essere trusted o untrusted
- role: per aumentare il controllo di accesso al dominio TE, SELinux ricorre al RBAC. Il significato del role è dato dalla policy. Il suffisso è _r
- type: è il modo principale per definire un accesso, suffisso _t
- security context: detto anche label, è una stringa composta da tre argomenti obbligatori `user:role:type` ed un quarto facoltativo `:range` (MLS). E' assegnato ad ogni oggetto, processo. E' un attributo usato per dire se è consentito l'accesso ad un oggetto da parte di un processo
- policy: insieme di regole che definiscono i diritti di accesso; posono essere scritte in linguaggio CIL e compilate
- domain: consiste di uno o più processi associati ad un type di un security context. La policy descrive come il domain interagisce con l'oggetto

SELinux - architettura

Gli elementi del TE sono il “domain” o “context” di sicurezza che dà le regole con cui un “subject” (utente, processo software, flusso) accede ad un “object” (file, device, porta I/O, sockets...). Ogni “subject” viene eseguito in un “domain” o contesto di sicurezza definito e tutti gli “objects” hanno un certo tipo (livello di secretezza) associato.

Info datata 2012: un buon sistema di controllo SELinux per tutto il sistema contiene, in media, circa 100k regole (# `sesearch -A`)

SELinux è un MAC basato su controllo delle “labels”. La “label” è il contesto di sicurezza assegnato ad ogni oggetto/soggetto del sistema protetto da SELinux

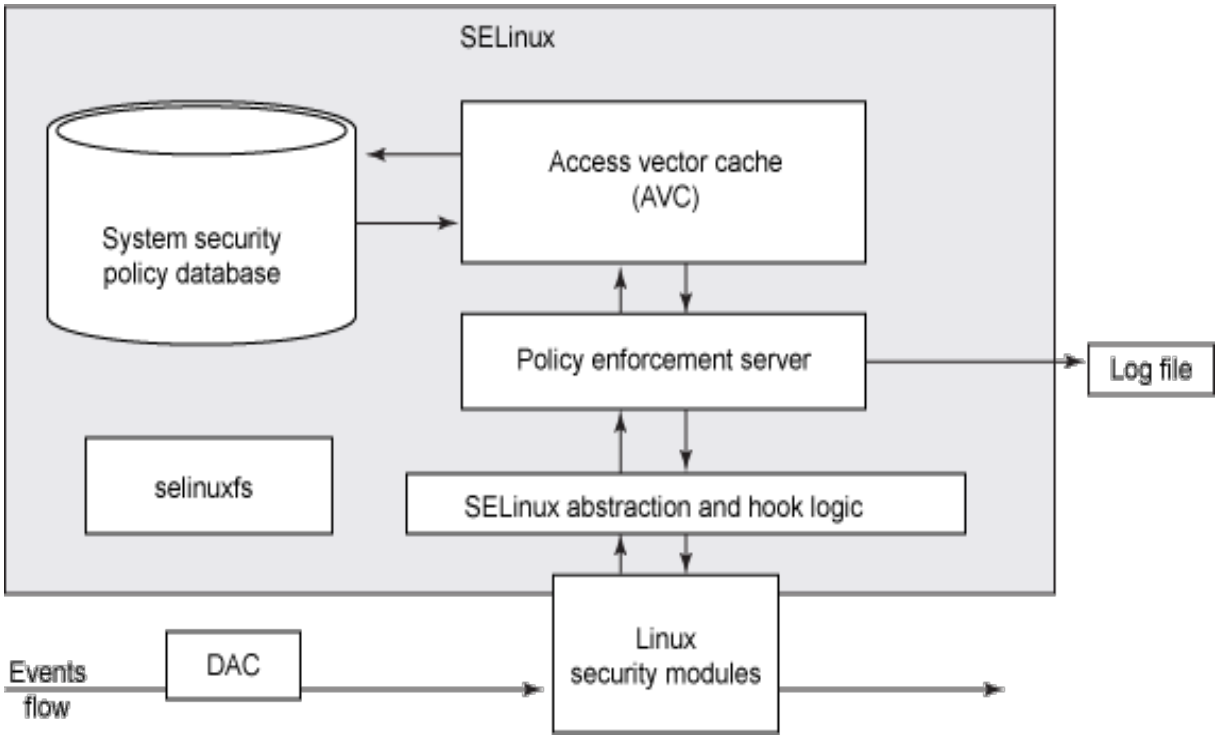
Con il TE, SELinux va oltre il DAC nativo di Linux; ad es. indicare quali files possono essere creati, modificati ma non eliminati, etc. e confinare, in caso di compromissione di un servizio controllato da SELinux, l'attaccante

Riferimento utile ed interessante che descrive lo sviluppo di SELinux ed LSM: *http://www.ibm.com/developerworks/library/l-secure-linux-ru/index.html*

SELinux - architettura

Come opera SELinux

- Il processo/subject esegue un'azione permessa dal DAC su un oggetto;
- la richiesta è intercettata dal LSM ed è trasferita al SELinux abstraction...insieme al contesto di sicurezza del subject/object

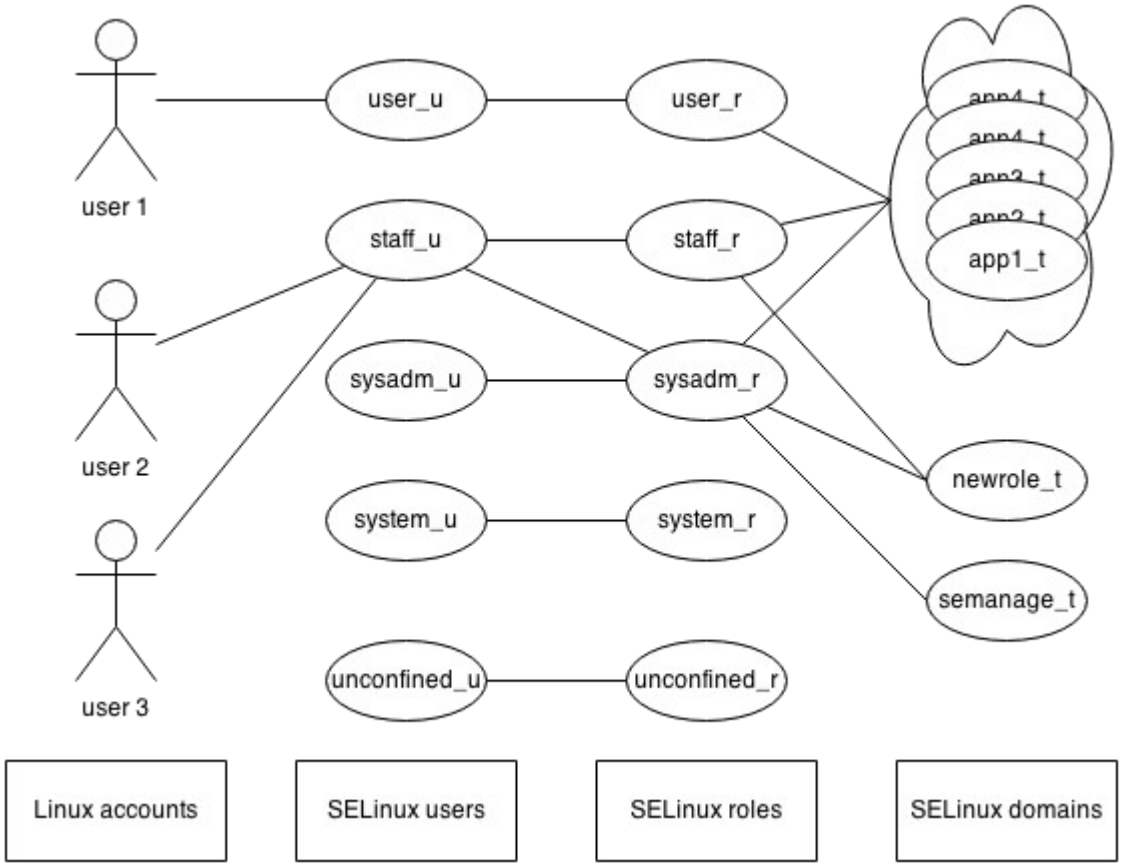


- l'informazione passa al modulo PES responsabile della decisione
- per decidere, il PES contatta il AVC che ha in cache le regole (performance)
- se il AVC non ha la regola, la richiesta è inoltrata al SPS database
- se l'azione richiesta è in accordo con la regola l'operazione è permessa altrimenti no e tutta l'informazione è scritta nel log file

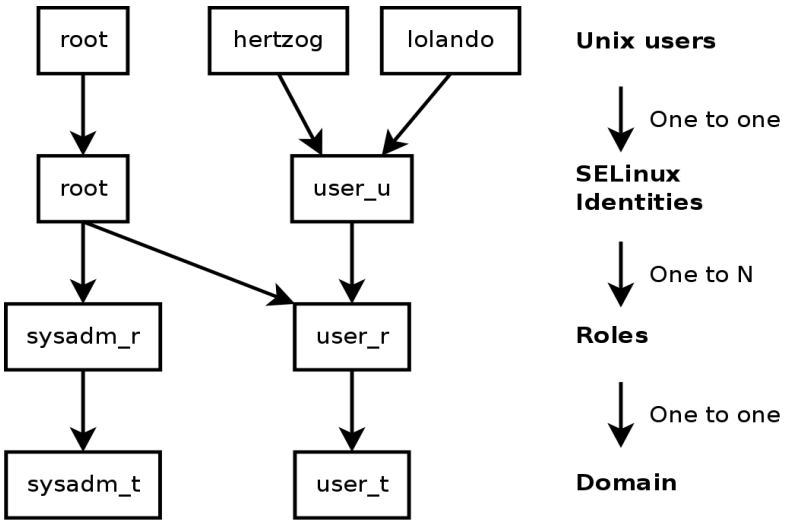
SELinux - user

Utenti e ruoli nel MAC SELinux hanno un significato diverso dal DAC Linux. L'utente SELinux non cambia durante una sessione (`su` per uno user Linux) ma può assumere più ruoli. Ad ogni utente/subject è assegnata una label con 4 attributi: role _r, user _u, type/context _t. Il quarto valore si riferisce al MLS o MCS (`# semanage login -l`)

L'opzione "-Z" data a molti comandi Linux, consente di vedere gli attributi MAC. `$ id -Z = unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023`



La maggiorparte degli utenti hanno lo stesso user, _u, ed il controllo è gestito dal tag type, _t



SELinux - configurazione

I file di configurazione, per distro RH-like, si trovano in */etc/selinux/*. Le modifiche vanno fatte utilizzando i "tool" opportuni (ad es. `semanage`) oppure modificando alcune variabili di valore booleano. Cambiare i valori di queste variabili è più comodo, e consigliato, che avere a che fare con le regole di policy (`getsebool`, `setsebool`, `man -k _selinux`; `man -k _selinux | nl` => 808 output)

Il file noto è il *config* (la famosa opzione SELINUX = disabled ! oppure la policy da usare); nella dir targeted ci sono

- `seusers` il file che mappa utenti Linux con utenti SELinux
- `setrans.conf` la tabella che traduce le categorie MCS in formato "umano"
- `policy` la dir che contiene la policy attuale nella forma di un singolo file con estensione la versione della policy
- `context` la dir che contiene i "context" di default del sistema
  - `customizable_types` lista di tipi che, se applicati, ad un oggetto non vengono modificati da un successivo `restorecon`
  - `files` dir con le info di default di tutto il sistema, delle homedir etc.
- `modules` la dir che contiene le policy attive (le versioni recenti di SELinux sono modulari e la policy globale è fatta da sotto-policy chiamate moduli)

SELinux – policy & module

Una “policy” contiene le definizioni per identità degli utenti, tipo di oggetto, ruoli e le regole con cui i tipi interagiscono. Poter cambiare la policy dinamicamente è un aspetto importante del SELinux “moderno” (modular policy; comando `semodule` o `semanage module, audit2allow`).

Le “policy” di SELinux possono essere

- strict negare tutte le richieste che non hanno regole di “allow”, le uniche. Utile per server con pochi servizi
- targeted “allegerisce” la durezza di SELinux; i servizi girano protetti dal DAC standard di Linux con tipo unconfined_t tranne per alcuni, ad es. httpd ed altri, che, con la loro policy, all'avvio, passano in un loro “domain”

Un modulo di SELinux facilita l'utilizzo perché consente modifiche delle policy “al volo” e permanenti, o meno a seconda del comando o dell'opzione usata, dopo successivi aggiornamenti. La scrittura di un modulo è facilitata combinando l'output del log file ed alcuni tool da CLI o GUI

SELinux - logging

Studiare I log file di SELinux, con o senza i suoi tools è il modo migliore per capire il comportamento di SELinux

Il file di log e' molto “verboso” ma ci sono gli strumenti (CLI o GUI, ad es. GUI `seaudit`, rpm `setools-gui`) per analizzarlo

Non e' obbligatorio per applicazioni sviluppate “SELinux-aware” controllare gli eventi o loggarli in audit.log

Gli eventi di SELinux vengono “loggati” in

- /var/log/dmesg (al boot): [0.002000] SELinux: Initializing.
  [0.002000] SELinux: Starting in permissive mode
  [0.002000] SELinux: Starting in permissive mode
- /var/log/messages (prima dell'avvio del demone auditd)
- /var/log/audit/audit.log (se e' attivo il demone auditd)

I tipi di eventi loggati sono:

- AVC (Access Vector Cache): eventi di tipo “denied” o “granted”
- Generici: errori di sistema, cambio di conf o di attributi o di stati. Lista di eventi molto lunga (`awk '{print $1}' /var/log/audit/audit.log | grep -v AVC |sort | uniq -c | nl`)

SELinux - logging

Esempio di evento AVC denied “loggato”:

```
type=AVC msg=audit(1446279920.905:175): avc:  denied  { read } for
pid=5064 comm="mysqld" name="config.inc.php" dev="dm-3" ino=805579617
scontext=system_u:system_r:mysqld_t:s0
tcontext=unconfined_u:object_r:httpd_sys_content_t:s0 tclass=file
```

type=AVC: evento di tipo AVC
msg=audit(...): timestamp dell'evento da “epoch”; convertibile `` `date -d @timestamp` ``
avc: tipo di entry
denied: cosa ha fatto SELinux (denied o granted)
{ read }: operazione per cui e' stato richiesto il permesso
for pid: il PID del processo che ha richiesto il “read”
comm: il comando (senza opzioni e limitato a 15 caratteri) che ha eseguito l'operazione
name: il nome dell'oggetto del processo (in questo caso il file; puo' essere path, src, etc.)
dev: il “device” dove si trova: in questo caso, il file
ino: numero di inode del file (con un `'find'` possiamo trovarlo; conosciamo anche il “device”)
scontext: il contesto di sicurezza del processo
tcontext: il contesto di sicurezza del “target”
tclass: classe dell'oggetto ”target”

L'analisi del log consente di capire le eventuali azioni da compiere; ad es. usando il messaggio “type=AVC...” ed i comandi `audit2why` e/o `audit2allow`; uso classico è:

`echo “type=AVC  msg=audit...” | audit2why` oppure `audit2allow`

SELinux – comandi & esempi

L'opzione "*-Z*" data a molti comandi Linux, consente di vedere gli attributi MAC. L'info più utile, la maggior parte delle volte, è il type _t.

```
$ id -Z
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
$ touch test_1.txt
$ ls -Z test_1.txt
-rw-rw-r--. antonelli antonelli
unconfined_u:object_r:user_home_t:s0 test_1.txt
# ps auxZ | grep httpd
…
system_u:system_r:httpd_t:s0    apache    3755  0.0  0.3
284208  6808 ?        S    Nov01   0:00 /usr/sbin/httpd
-DFOREGROUND
```

Comando utile per gestire la "policy" SELinux è "`semanage`". Ad es., la lista degli utenti e ruoli definiti è: `# semanage user -l`

SELinux – comandi & esempi

comando utile per ripristinare i context di default è

```
# restorecon -Rv /srv/www
```

`restorecon` è il comando da usare quando si passa da modalità `permissive` ad `enforcing` per ri-labellare il filesystem

Es. creazione e `cp` o `mv` del file e successivo `restorecon` nella DocumentRoot del web server (es. su VM)

```
$ echo "<h1> TEST SELinux </h1>"> test.html;ls -Z test.html
-rw-rw-r--. antonelli antonelli unconfined_u:object_r:user_home_t:s0
test.html
mv /home/antonelli/test.html /var/www/html/;ls -Z
/var/www/html/test.html
-rw-rw-r--. antonelli antonelli unconfined_u:object_r:user_home_t:s0
/var/www/html/test.html ===> errore http (si risolve col restorecon)
# restorecon -v /var/www/html/test.html
restorecon reset /var/www/html/test.html context
unconfined_u:object_r:user_home_t:s0-
>unconfined_u:object_r:httpd_sys_content_t:s0
#ls -Z /home/antonelli/test.html => user_home_t
# cp /home/antonelli/test.html /var/www/html/; ls -Z
/var/www/html/test.html => httpd_sys_content_t
```

SELinux – comandi & esempi

Oltre `semanage` per la gestione di user, port ed altri objects, SELinux offre una serie di macro con valore booleano che possono essere gestite dai comandi `getsebool` e `setsebool`. E' consigliato, in questi casi, modificare questi valori invece di ricorrere al `semanage`. Le modifiche non sono persistenti a meno di usare l'opzione `-P`.

Un altro tool utile per capire le “policy” di SELinux è `sepolicy` al quale posso passare diversi argomenti.

`# matchpathcon` dà il security context di default per SELinux per il percorso specificato prendendo l'informazione dal file contexts

`# seinfo` tool per interrogare la policy SELinux

SELinux – comandi & esempi

moduli di policy “al volo” in caso la policy di SELinux blocchi, ad es., l'esecuzione di una shell remota che prova a leggere il file */etc/shadow* . Dal file di log:

```
echo "type=AVC msg=audit(1446128957.821:4764): avc:  denied  { getattr }
for  pid=24329 comm="httpd" path="/etc/shadow" dev="dm-1" ino=402989571
scontext=system_u:system_r:httpd_t:s0
tcontext=system_u:object_r:shadow_t:s0 tclass=file" | audit2why
(audit2allow -w)
```

`audit2allow -M test_shadow_mod` dà in output il comando per attivare la policy
`semodule -i test_shadow_mod.pp` ed i due files
test_shadow_mod.te (type enforcement)

```
module test_shadow_mod 1.0;
require {
	type httpd_t;
	type shadow_t;
	class file getattr;
}
#============= httpd_t ==============
allow httpd_t shadow_t:file getattr;
```

ed il file *test_shadow_mod.pp*

`# file test_shadow_mod.pp` *test_shadow_mod.pp: SE Linux modular policy version 1, 1 sections, mod version 17, MLS, module name test_shadow_mod\003*

Alcune info ± utili

http://www.slideshare.net/null0x00/
http://www.slideshare.net/Jhon_Masschelein/se-linux-42322789?next_slideshow=5
http://userspace.selinuxproject.org/
SLIDE integrated development environment for SELinux
http://userspace.selinuxproject.org/archive/slide.php
http://www.crypt.gen.nz/selinux/links.html
https://www.nsa.gov/research/selinux/index.shtml
http://danwalsh.livejournal.com/
http://freecomputerbooks.com/books/The_SELinux_Notebook-4th_Edition.pdf

selinux-policy-devel-3.13.1-23.el7_1.18.noarch
selinux-policy-3.13.1-23.el7_1.18.noarch
selinux-policy-targeted-3.13.1-23.el7_1.18.noarch
selinux-policy-doc-3.13.1-23.el7_1.18.noarch
libselinux-utils-2.2.2-6.el7.x86_64
libselinux-python-2.2.2-6.el7.x86_64
libselinux-2.2.2-6.el7.x86_64
setools-libs-3.3.7-46.el7.x86_64
setools-console-3.3.7-46.el7.x86_64
libsemanage-2.1.10-16.el7.x86_64
libsemanage-python-2.1.10-16.el7.x86_64
setroubleshoot-server-3.2.17-4.1.el7_1.x86_64
setroubleshoot-plugins-3.0.59-1.el7.noarch
setroubleshoot.x86_64
policycoreutils-2.2.5-15.el7.x86_64