SìLinux – Abilitiamo il Security Enhanced

Introduzione al modulo dell'NSA e consigli pratici

Saverio Fruncillo

Chi sono

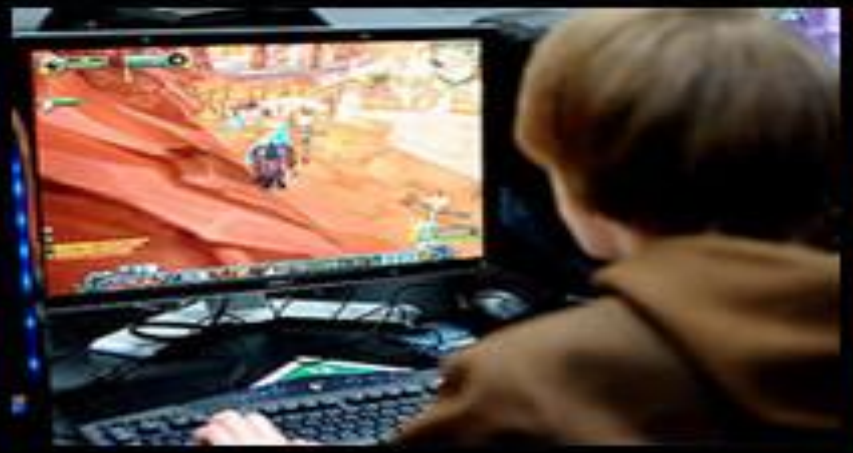
Saverio Fruncillo

- ICT Specialist & IT System Administrator per Millennium SPA
- LPIC1, LFCS, Linux+, CCNA, etc.

https://www.credly.com/users/saverio-fruncillo/badges#credly

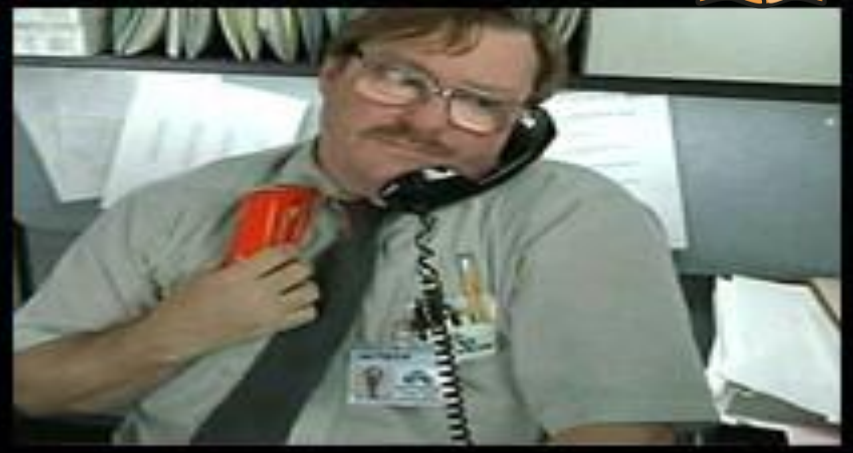
LINUX DAY

SYSADMIN



What my friends think I do

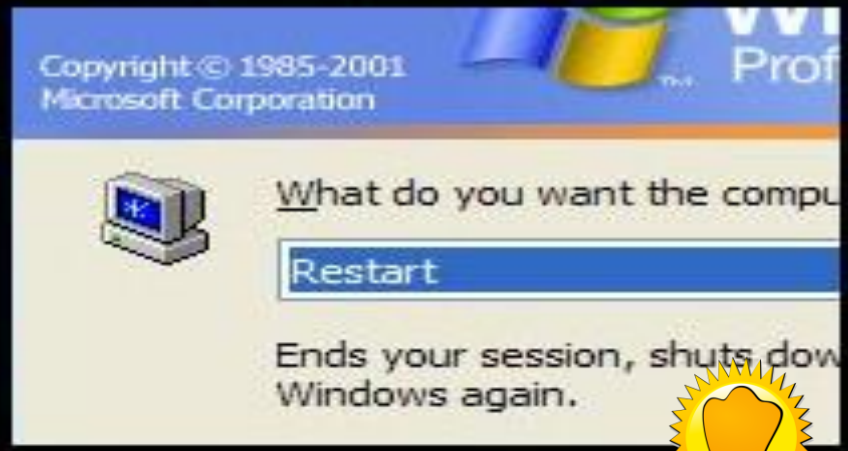
What my mom thinks I do



What society thinks I do

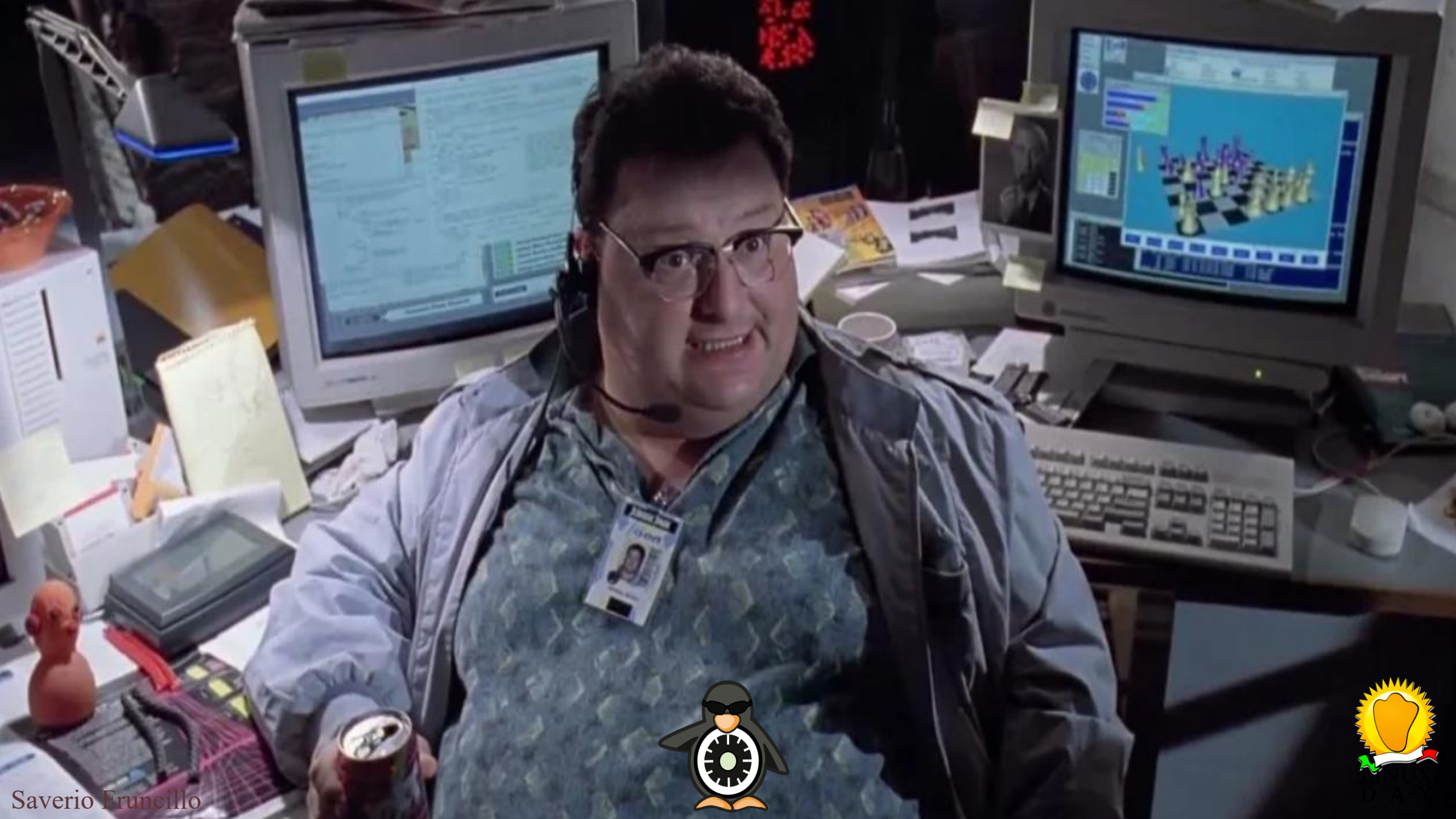
What my boss thinks I do

What I think I do



What I really do

Saverio Fruncillo



Saverio Fruncillo

Pillole di storia

1998 – SELinux nasce da un progetto della NSA (National Security Agency) con l'obiettivo di implementare il MAC (Mandatory Access Control) per proteggere i sistemi da compromissioni
2000 – Il progetto viene reso pubblico coinvolgendo numerosi attori tra cui RedHat
2003 – SELinux è integrato nel kernel linux grazie al framework LSM (Linux Security Modules), creato appunto per permettere l'integrazione modulare di sistemi di sicurezza

Architettura

SELinux si basa sull'architettura Flask (progetto MAC dell'NSA), consentendo di definire policy di sicurezza granulari e flessibili.

Flask è un'architettura di sicurezza che separa:
Decisione: chi può fare cosa (policy server)
Applicazione: enforcement nel kernel (security server)

Introduce il concetto di security context dove ogni processo e file ha una etichetta che definisce cosa può fare e con chi può interagire

APPROCCIO PROATTIVO – LIMITA I DANNI ANCHE IN CASO DI EXPLOIT RIUSCITO

MAC vs DAC

DAC – Discretionary Access Control, il controllo è gestito dal proprietario dell'oggetto:
I classici permessi chmod e chown
Se l'utente crea un file può decidere chi può leggerlo o modificarlo

chmod 644 documento.txt
chown pippo:pippo documento.txt

Curiosità: il filesystem è DAC

MAC vs DAC

MAC – Mandatory Access Control, il controllo degli accessi è obbligatorio e definita da una policy centralizzata. Gli utenti non possono modificarla.
Il sistema decide, e nemmeno root può aggirare la policy

Anche se Pippo ha permessi DAC sul file, SELinux può impedirgli l'accesso:

ls -Z documento.txt # mostra il security context
getenforce # verifica se SELinux è attivo

Curiosità: SELinux usa attributi estesi (xattr), di conseguenza il fs deve supportare xattr

SELinux non sostituisce DAC ma lo affianca

Security Hardening

Saverio Fruncillo

LINUX DAY

Kernel-Level MAC

https://www.coker.com.au/selinux/play.html

Confronto con AppArmor

Caratteristiche	SELinux	AppArmor
Tipo di controllo	Mandatory Access Control (MAC)	Mandatory Access Control (MAC)
Modello di policy	Basato su etichette (label)	Basato su percorsi (path)
Granularità	Estrema	Semplice, meno granulare
Flessibilità	Altissima e complessa	Semplice
Distribuzioni default	RHEL based	Debian based
Gestioni profili	Richiede policy complesse e compilazione	Profili leggibili e facilmente modificabili
Logging e audit	Completo e dettagliato	Semplice ma meno esaustivo

In sintesi SELinux è più potente e adatto a sistemi enterprise, mentre AppArmor è flessibile e rapido.

Le 3 modalità operative (getenforce)

Enforcing –SELinux applica le policy e blocca le violazioni
setenforce 1

Permissive – SELinux non blocca ma logga ogni violazione (troubleshooting)
setenforce 0

Disable – SELinux spento
/etc/selinux/config

Nota importante: touch /.autorelabel dopo il reset della password o riattivazione del SELinux

Vediamo come funziona

Due importanti concetti da conoscere sono:

- Labeling
- Type Enforcement

Vediamo come funziona

Labeling:

- Files, processi, porte, etc. sono “etichettati” con un SELinux context
- Per file e directory, queste etichette sono salvate come attributi estesi nel filesystem
- Per processi, porte, etc., vengono gestiti dal kernel

Etichette

Ogni oggetto (file, processo, socket...) ha una etichetta detta security context

ls -Z
system_u:system_r:httpd_t:s0

- system_u: utente SELinux
- system_r: ruolo
- httpd_t: tipo (dominio)
- s0: livello di sensibilità (usato in MLS/MCS)(livello di riservatezza)(opzionale)

Httpd_t (processo di Apache)

Gestire le etichette

– ls -Z
– id -Z
– ps -Z
– netstat -Z

Vediamo come funziona

Apache web server, pur essendo sicuro, ha accesso molto esteso nel sistema e potrebbe essere un problema
Apache web server ha un eseguibile in /usr/bin.

Osservando il suo SELinux context si osserva httpd_exec_t:

```
[root@redhat-based rocky]# ls -lZ /usr/sbin/httpd
-rwxr-xr-x. 1 root root system_u:object_r:httpd_exec_t:s0 581320 15 lug 02.00 /usr/sbin/httpd
[root@redhat-based rocky]# ls -dZ /etc/httpd/
system_u:object_r:httpd_config_t:s0 /etc/httpd/
[root@redhat-based rocky]# ps axZ | grep httpd
system_u:system_r:httpd_t:s0     134463 ?        Ss     0:43 /usr/sbin/httpd -DFOREGROUND
system_u:system_r:httpd_t:s0     728857 ?        S      0:00 /usr/sbin/httpd -DFOREGROUND
system_u:system_r:httpd_t:s0     728858 ?        Sl     1:08 /usr/sbin/httpd -DFOREGROUND
system_u:system_r:httpd_t:s0     728867 ?        Sl     1:03 /usr/sbin/httpd -DFOREGROUND
system_u:system_r:httpd_t:s0     728915 ?        Sl     1:03 /usr/sbin/httpd -DFOREGROUND
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 1111915 pts/4 S+   0:00 grep --color=auto httpd
[root@redhat-based rocky]# netstat -tunlpZ | grep httpd
tcp6       0      0 :::80                   :::*                    LISTEN      134463/httpd         system_u:system_r:httpd_t:s0
[root@redhat-based rocky]#
```

Tutto ciò che è relativo al processo del web server è etichettato con httpd_t:

Vediamo come funziona

Type enforcement

- Ha senso avviare un processo con etichetta httpd_t in grado di interagire con un file con etichetta shadow_t?

Type enforcement è parte della policy che dice “un processo con etichetta httpd_t può avere accesso di lettura a file etichettati con httpd_config_t”

Gestire le etichette

Si possono usare i tools SELinux come chcon, semanage o restorecon per modificare le etichette
Le etichette sono settate nel momento della creazione del file
Gli RPM possono settare le etichette come parte della installazione
Il processo di login setta il default context (unconfined)

```
semanage fcontext -a -t httpd_sys_content_t "/srv/webdata(/.*)?"
restorecon -Rv /srv/webdata
```

La stramaggioranza delle volte se non và, è perchè l'etichetta è errata,
o una policy che va modificata

Esempio

- Hai un server Apache in esecuzione su CentOS/RHEL con SELinux attivo
- Crei un file `/var/www/html/test.html` con permessi DAC che consentono la lettura a tutti:
  - `touch /var/www/html/test.html`
  - `chmod 644 /var/www/html/test.html`
- Quando provi ad accedere a `http://localhost/test.html`, ricevi un errore 403 Forbidden
- Controlli il contesto SELinux del file:
  - `ls -Z /var/www/html/test.html`
  - `unconfined_u:object_r:default_t:s0 test.html`
- Il tipo `default_t` **non è consentito** per l'accesso da parte di Apache secondo la policy SELinux.
- Cambi il contesto SELinux del file al tipo corretto:
  - `chcon -t httpd_sys_content_t /var/www/html/test.html`
- Ora Apache può accedere al file

Protezione contro Zero-Day Attack

un'applicazione web vulnerabile a un RCE (Remote Code Execution).
Un attaccante riesce a eseguire `/bin/bash` tramite un payload.

Senza SELinux: il processo compromesso può accedere a file, socket, eseguire comandi, leggere chiavi SSH… è libero di agire secondo i permessi DAC.

Con SELinux: anche se l'attaccante esegue `/bin/bash`, il processo è confinato nel suo contesto (es. `httpd_t`) e **non può accedere a** `/home`**,** `/etc/ssh`**, o aprire socket non autorizzati**, perché la policy MAC lo vieta.

Tips and Tricks (SELinux facile facile)

- Cambiamo il fcontext con un solo SUPER comando se non sappiamo dove prenderlo:
- Chcon —reference

```
[rocky@localhost ~]$ ls -dZ site
system_u:object_r:home_root_t:s0 site
[rocky@localhost ~]$ ls -dZ /var/www
system_u:object_r:httpd_sys_content_t:s0 /var/www
[rocky@localhost ~]$ chcon --reference /var/www/ site
[rocky@localhost ~]$ ls -dZ site
system_u:object_r:httpd_sys_content_t:s0 site
```

- Voglio riportarlo come era prima (default label):

```
[rocky@localhost ~]$ restorecon -v -R site/
Relabeled /home/rocky/site from system_u:object_r:httpd_sys_content_t:s0 to system_u:object_r:user_home_t:s0
Relabeled /home/rocky/site/index.html from unconfined_u:object_r:httpd_sys_content_t:s0 to unconfined_u:object_r:user_home_t:s0
[rocky@localhost ~]$ ls -dZ site/
system_u:object_r:user_home_t:s0 site/
```

- Restorecon è il tuo amico

Tips and Tricks

- Usare audit2allow per compilare un file di policy per far funzionare tutte le policy fallite e loggate
- In /var/log/audit/audit.log troviamo, oltre aa altri eventi di auditing, i log SELinux, con event type AVC (Access Vector Cache).

```
ausearch -c "httpd_t" --raw | audit2allow -M httpd_t
semodule -i httpd_t.pp
```

Audit2allow è un tool molto potente che va usato con cautela

- Installare “setroubleshoot” – installa una serie di tools per aiutare a diagnosticare e fixare problemi in SELinux

Considerazioni finali

Non disabilitarlo
SELinux è sempre più semplice
SELinux può salvarti da un evento di breach o perfino 0-day attack
Non è un antimalware, ma protegge dalle epidemie
Un modulo di sicurezza del NSA senza costi aggiuntivi

SELinux non è un nemico, ma un alleato:
se ti blocca, è perché sta facendo il suo lavoro. Capirlo è il primo passo per usarlo con fiducia.

Quindi ABILITIAMOLO

Saverio Fruncillo

LINUX DAY

GRAZIE

Saverio Fruncillo

LINUX
D A Y