Hardening Sistema Scada

# SCADA

## Definizione

Scada (dall'inglese "*Supervisory Control And Data Acquisition*", cioè "controllo di supervisione e acquisizione dati") è definito come un sistema automatico distribuito di monitoraggio elettronico dei sistemi fisici.

Le tipiche funzionalità di un sistema SCADA sono:

* memorizzare eventuali dati ricavabili dai vari componenti fisici;
* segnalare eventuali malfunzionamenti o dati rilevanti ad un operatore umano mediante interfaccia utente (HMI);
* consentire un accesso remoto ai controllori fisici distribuiti, che oltre a raccogliere dati di manutenzione sono anche in grado di modificare l'eventuale processo produttivo, mediante la modifica di eventuali parametri.

Per questo un sistema SCADA è un sistema distribuito complesso composto da questi elementi:

* Un terminale remoto si occupa di convertire i segnali in dati digitali. (Remote Terminal Units: RTU)
* I controlli elettronici per monitorare il sistema fisico svolgono funzioni simili agli RTU tuttavia sono più versatili e programmabili(Programmable Logic Controllers: PLC)
* Una interfaccia utente dove un operatore umano può monitorare o interagire con i componenti fisici del sistema(Human Machine Interface: HMI)
* Una infrastruttura di telecomunicazione in grado di far dialogare i componenti.
* Un data warhouse dove conservare i dati acquisiti dai rilevamenti.
* Un sistema supervisore che oltre a interagire con il database può anche interagire con in componenti fisici, mediante dei comandi.

## Sicurezza sistemi SCADA

SCADA è un sistema che si occupa quindi sia di sicurezza legata a possibili danni che possono subire le strutture o il personale(safety), sia alla sicurezza dei dati (security). Sono stati progettati attacchi a SCADA che avevano lo scopo di sovraccaricare i macchinari monitorati.

Il problema di una sicurezza (intesa come security) nei sistemi SCADA si è andata a rafforzare con l'avvento di stuxnet che nel 2010 attaccò i PLC di controllo delle centrifughe nucleari iraniane(le centrifughe sono impiegate per separare l'uranio pesante da quello impoverito), e distruggendone una su cinque.

Stuxnet attaccava sistemi basati su piattaforma windows, il codice sorgente di stuxnet non è stato reso pubblico in quanto si pensa che a partire da quello sia possibile sviluppare nuovi attacchi anche a versioni SCADA più avanzate.

## Struttura SCADA

Di seguito viene analizzata la struttura di OpenSCADA tale struttura si riferisce al componente supervisore che interagisce con il database. In altre implementazioni di SCADA si tendono a conservare i nomi dei package principali.

I package sono definiti dal team di OpenSCADA come progetti a sé stanti questa definizione mostra anche l'intrinseca complessità della implementazione di SCADA.

**Atlantis**

Il package principale contiene le funzionalità principali di SCADA si appoggia per il suo funzionamento ad altri pacchetti

**Utgard**

Consente l'interfacciamento con gli altri componenti di acquisizione dati tramite protocollo OPC.

**Orilla**

Ospita tutte le UI dei componenti che necessitano della connetività offerta da SCADA.

**Othala**

Consente l'interfacciamento con il sistema SCADA per l'amministratore di sistema e gli sviluppatori.

Sono presenti anche molti altri package che però sono esclusivi di OpenSCADA.

OpenSCADA si dimostra versatile multi-piattaforma e configurabile, le quali configurazioni possono essere modificate dall'amministratore anche a servizio attivo. Sfrutta protocolli standard come OPC per interfacciarsi con i dispositivi.

## Vulnerabilità

Recentemente i sistemi PLC si stanno aprendo a standard pubblici piuttosto che utilizzare i propri standard proprietari. Questa esigenza frutto della necessità di permettere la collaborazioni di diverse tecnologie, ha avuto un grande impatto su SCADA che al contrario di alti sistemi distribuiti si occupa del monitoraggio di componenti ben distribuiti sul territorio e quindi può sfruttare costosi ponti radio, quanto linee telefoniche, telefonia mobile GSM/GPRS, reti WAN, etc. Fondamentalmente è venuto a mancare quello che era il cosiddetto “air gap” ovvero quella forma di sicurezza informatica che consiste nell'assicurarsi che il sistema sia isolato da reti accessibili dall'esterno.

Infatti i primi attacchi ricordati sui sistemi SCADA sono a opera di ex dipendenti o comunque di persone che riuscivano a introdursi e interagire con gli HMI. Gli attacchi venivano perpetuati tramite eseguibile lanciati da pennetta USB.

Questi esempi e l'esempio di stuxnet sono testimonianze di quanto la sicurezza del sistema sia legato a debolezze della piattaforma in cui gira. Spesso quando si parla di sistemi distribuiti, si deve considerare che si sta trattando di sistemi monolitici, che hanno necessità di una manutenzione in aggiornamenti e verifiche di qualità che spesso viene trascurata. La grande difficoltà della messa in sicurezza di sistemi di questo tipo sta nel fatto di attuare la sicurezza come un processo, che coinvolge in primis gli operatori umani, e a un periodico aggiornamento delle norme di sicurezza

Una vulnerabilità individuata in SCADA è legata alla non granularità al controllore di operazioni inviate ai PLC che essendo molte non vengono controllate adeguatamente e non si filtrano quelle operazioni inviate tramite pacchetti che possono risultare dannose.

Sul filo di questo discorso un possibile hardening di un sistema SCADA sfrutta proprio quegli strumenti che una piattaforma come linux offre in termini di sicurezza. Ovvero un approccio tramite politiche di accessibilità DAC MAC. Le vulnerabilità individuate sono:

Il file di configurazione che il sistema SCADA legge in fase di avviamento.

Le varie connessioni con i PLC a cui possono essere inviati comandi senza passare tramite l'operatore generale SCADA.

Tutti le eventuali operazioni in grado di inviare segnali ai PLC

Inoltre è opportuno proteggere non solo il database ma anche le operazioni che interagiscono con questo.

Una politica MAC che può comportarsi sia da firewall rispetto ad una packet inspection, sia offrire politiche di accessibilità ben si presta per la messa in sicurezza di una installazione SCADA.

# HARDENING

## Sicurezza Fisica

Configurare il BIOS in modo da disabilitare il booting da CD/DVD, Device Esterni, Floppy.

Impostare una password al BIOS

Proteggere il GRUB con una password in modo da ridurre accessi fisici al sistema

## Partizioni dei dischi

Fare in modo che il disco non sia costituito da un’unica partizione. Avere più partizioni consente una maggiore sicurezza nel caso succeda un qualunque disastro. Creando diverse partizioni, i dati possono essere separati e raggruppati. Quando avviene un incidente, solo i dati della partizione sono compromessi, mentre i dati delle altre partizioni rimangono inalterati.

Inoltre alcune delle directories tipiche dei sistemi linux sono esposte ad attacchi di tipo noto, e questo a causa del ruolo che svolgono all’interno del sistema operativo, e quindi è plausibile che siano montate su filesystem separati dal resto del sistema.

In particolare le directories:

**/var** è la directory che contiene i logs del sistema, ed una generazione fuori controllo può in poco tempo esaurire tutto lo spazio della partizione e quindi del disco in una soluzione con un’unica partizione, e questo impedisce al sistema operativo di funzionare correttamente.

**/tmp** è la directory adibita a contenere file temporanei; gode spesso di privilegi alti e quindi rootkit ed exploit vengono spesso lanciati proprio da questa directory, proprio perché quella più vulnerabile.

Analizzando il file **/etc/fstab** è possibile verificare quali siano le partizioni presenti ed i relativi punti di mount al boot, ed è quindi necessario far si che /var e /tmp siano montati su due filesystem separati. Conviene creare partizioni separate anche per le directories usr ed home, ed inoltre abilitare le quote in modo da evitare un riempimento totale delle partizioni in caso di malfunzionamento.

## Minimizzare pacchetti per minimizzare le vulnerabilità

Evitare l’installazione di pacchetti inutili per evitare le vulnerabilità indotte dagli stessi. Questo porta a minimizzare il rischio che la compromissione di un servizio possa portare alla compromissione di altri servizi.

E’ buona norma trovare e disabilitare e/o eliminare servizi non desiderati dalla macchina. Per far ciò si utilizza il comando

**/sbin/chkconfig --list |grep '3:on'**

che permette di ritrovare servizi che sono in esecuzione sul runlevel 3, che rappresenta la configurazione di sistema in cui tutti i servizi sono attivi.p

## Kernel e pacchetti aggiornati

È di fondamentale importanza **aggiornare i pacchetti alle ultime versioni disponibili**, in modo che bug delle versioni precedenti, e corretti con le ultime, non possano essere un veicolo di attacco. Stesso discorso vale per il kernel.

Per questa operazione, occorre affidarsi **al package manager del proprio sistema**; generalmente tramite interfaccia grafica è possibile vedere in modo più semplice le versioni di riferimento. È inoltre utile tenere sott’occhio i forum, e le newsletter che notificano gli aggiornamenti di sicurezza del kernel per avere una panoramica più completa.

## Confinamento Cronjobs

Cron specifica chi può e non può lanciare job. Questi controlli sono effettuati tramite in file

***/etc/cron.allow*** che contiene la lista degli utenti che possono lanciare cron

***/etc/cron.deny*** che contiene, invece, la lista degli utenti che non possono usare cron

Per evitare che gli utenti possano utilizzare cron basta aggiungere ALL al file cron.deny

**echo ALL >> /etc/cron.deny**

## Disabilitare il rilevamento delle USB stick

Per disabilitare le usb stick bisogna invece creare innanzitutto il file ***/etc/modprobe.d/no-usb***

e scrivere all’interno

**install usb-storage /bin/true**

## Proteggere le comunicazioni

L’utilizzo di servizi come **telnet**, **ftp**, **rsh**, **comporta che username e password siano trasmessi in chiaro**, il che espone questi dati alla possibile cattura di un packet sniffer in ascolto sulla rete. Onde evitare problemi legati alla possibilità di intercettare username e password che vengono trasmessi in chiaro sulla rete è necessario utilizzare qualche forma di cifratura dei dati, per esempio con l’utilizzo di **OpenSSH**, **scp**, **SFTP**, che forniscono servizi di protezione dei dati trasmessi sulla rete.

In particolare ssh, utilizzato per l’accesso remoto al sistema operativo, viene installato con un configurazione di default che va modificata (file **/etc/ssh/sshd\_config**), **in particolare l’accesso con permessi di root non deve essere consentito ad alcun utente**, ed allo stesso tempo deve essere disabilitata l’**autenticazione senza password**; normalmente ssh è un servizio in ascolto sulla porta 22, e come tale è esposto ad attacchi di forza bruta che in automatico operano su quella stessa porta, per questo motivo è conveniente **modificare la porta di questo servizio, facendo attenzione a non andare in conflitto con la porta di qualche altro demone**.

Nel file **/etc/ssh/sshd\_config**, le modifiche da fare sono:

**PermitRootLogin no**

**PermitEmptyPasswords no**

**Port 41567**

**Protocol 2**

## Chiudere le porte aperte

Tutti i servizi che non sono di stretta necessità per il sistema e che sono in ascolto di connessioni di ingresso, **possono rappresentare dei possibili punti di attacco, per tanto occorre chiudere o disabilitare questi servizi**. Per individuare quelli che sono in ascolto su delle porte, è sufficiente utilizzare netstat, che con la seguente sintassi consente di avere l’elenco delle porte aperte, lo stato della connessione ed il pid del processo che controlla quella connessione.

**netstat -tulpn**

In questo modo è possibile identificare le connessioni attive e fare in modo di chiudere le  porte aperte (stoppando i servizi, o in casi estremi uccidendo i processi).

## Controllo d’accesso

Implementare una politica di tipo Mandatorio (MAC). Eseguire programmi con il kernel MAC, garantisce protezione contro software malevoli che possono danneggiare il sistema, o fornire porte di accesso non autorizzate.

# Implementare un mandatory access control

# La sicurezza introdotta da un MAC consente di monitorare o impedire alcune operazioni rispetto a determinati processi o oggetti. Viene utilizzata per la messa in sicurezza di applicativi o di moduli del kernel.

# A livello enterprise il mercato offre due possibili implementazioni di un MAC:

# SELinux

# AppArmor

# SELinux è un sistema complesso, completo, con una bassa curva di apprendimento.

# Il sistema è nato per garantire la sicurezza in ambito militare, è stato progettato infatti dalla NSA, l'agenzia di sicurezza nazionale americana. I suoi punti di forza sono la flessibilità delle varie politiche MAC su cui si basa.

# AppArmor è invece un sistema che vanta una maggiore usabilità ed è più adatto per chi si sta avvicinando per la prima volta a configurazioni di un sistema MAC.

# Successivamente analizzerò alcuni aspetti di SELinux con l'obbiettivo di esplicitare per quali motivi la nostra scelta è ricaduta su AppArmor.

## SELinux

## SELinux è un modulo per la sicurezza del kernel di linux, in quanto tale è piuttosto efficiente, e permette un controllo anche a basso livello sui processi dell'utenza e sul servizio di accesso ai file del sistema. Essendo una implementazione per un MAC, si basa su alcuni concept:

# “Type Enforcement”- assegna a ciascun soggetto(generalmente un processo) un contesto di accessibilità relativamente a un oggetto(file, processo ecc);

# “Role-based access control”- assegna a ciascun soggetto uno o più ruoli. Ciascun ruolo corrisponde generalmente a un ruolo aziendale, e ha una tabella di accessibilità corrispondente. Il ruolo viene selezionato o assegnato all'utente che deve poi validarsi per renderlo attivo e quindi poter esercitare il proprio diritto di accesso.

# “Multilevel security” - Fondamentalmente suddivide le informazioni in livelli di sicurezza, fa in modo che utenti, che possiedono un determinato livello di accessibilità, non siano in grado di visualizzare informazioni che non gli competono. Nel caso di SELinux questo concept concorre a definirlo formalmente come sistema tacciabile di fiducia. Il multilevel security è di difficile applicazione in quanto soffre, di alcune rigidità quando si tenta di diminuire la classificazione di sicurezza di alcuni file.

# Come in AppArmor anche in SELinux è presente un permissive mode, in cui le violazioni di sicurezza o accessi negati non vengono impediti, ma semplicemente annotati in un log, cosa che veniva comunque eseguita anche nell'enforcement mode. Questa modalità è consigliata durante l'implementazione di nuove policy, ma anche impiegata per il learning. In questo modo tutti gli accessi negati possono essere triggerati da una sola esecuzione risparmiando al programmatore l'onere di correggerli singolarmente.

# La parola chiave che ritorna in ogni articolo trattante SELinux è “ben dettagliato”: come abbiamo detto ogni file user ha un proprio ruolo, dominio o (type of enforcement), nonché utente di appartenenza. Questo è valido anche per ogni file, processo, porta di rete, e altre interfacce di I/O.

## Contesto o label

# Le informazioni assegnate a ciascun file che ne definisce utente, ruolo e dominio è detta label o contesto, dove il termine label si riferisce al significante della tripletta, mentre il termine contesto si riferisce al significato.

# Il label è appunto composto dei tre campi user, ruolo e dominio. Prima di analizzare puntualmente i tre componenti del label, vorrei sottolineare come tutti e tre concorrono a implementare di fatto i 3 concept su cui si basa SElinux.

# Lo SELinux user consente di assegnare i ruoli a un utente reale che può corrispondere a uno e uno solo utente SELinux. Di default tuttavia esistono SELinux user che non corrispondono a alcun utente reale.

|  |  |  |
| --- | --- | --- |
| SELinux user | Descrizione | Usati come |
| unconfined\_u | SELinux users che non hanno restrizioni. Generalmente gli Unconfined users non sono soggetti a nessuna restrizione sono pensati per sistemi dove solo solo i servizi di connettività di rete dovrebbero essere confinati | Qualsiasi utente di uno specifico sistema |
| root | Lo SELinux user che indica il root account | il Linux root account |
| sysadm\_u | SELinux user con un diretto ruolo di amministrazione | Linux accounts che funge esclusivamente da amministratore di sistema |
| staff\_u | SELinux user che può eseguire delle operazioni come amministratore e non. | Linux accounts usato sia come amministratore che come utente |
| user\_u | SELinux user per un account senza privilegi amministrativi | Unprivileged Linux accounts |
| system\_u | Speciale SELinux user indicato per operazioni di sistema | Non usato direttamente |

# SELinux tiene traccia di un mapping tra account utente e utente SELinux. Dare un'occhiata alla login mapping ci permette quindi di avere una idea dei privilegi assegnati a ciascun utente, questo perché ciascun account è assegnato a un solo SELinux user.

# A ciascun SELinux sono assegnati diversi ruoli. Un ruolo è definito mediante SELinux Policy, ne parleremo più avanti.

# Il tipo o dominio, dipendentemente ci si riferisca a un file o un processo, rappresenta un identificativo dell'oggetto rispetto alla azione che può svolgere o che può essere svolta su di esso.

# Ad esempio un eseguibile di apache che consente di iniziare un dialogo http con il server di log avrà il tipo http\_log\_init. I privilegi relativi al tipo sono elencati mediante booleani generalmente delle assegnazioni di verità a ciascun tipo, che ne consentono la lettura, l'esecuzione ecc.

# Queste regole definite Boolean sono impostate mediante comandi, ma sono esposte in un file generato ogni volta all'occorrenza (dal comando “getsebool -a” ).

# Una SELinux Policy è appunto un insieme di regole Boolean assegnato a un determinato ruolo.

# Durante l'implementazione delle politiche se si usano i comandi SELinux consente di ripristinare le impostazioni di partenza. Inoltre è possibile analizzare il log e ricevere dei suggerimenti per individuare il comando in grado di consentire eventuali accessi negati riscontrati, in modo da poter costruire in maniera guidata una policy. Questo approccio “guidato” è stato presentato nel summit red Hat del 2013, in contrapposizione all'opinione comune per cui SELinux è estremamente complicato da implementare, e contro alla diffusa soluzione di spegnerlo, in caso qualcosa non funzioni.

# APPARMOR

## Introduzione

AppArmor è un'implementazione del «Linux Security Module» per il controllo degli accessi vincolante basato sul nome. AppArmor racchiude individualmente i programmi in un insieme di file e capacitàà posix 1003.1e draft.

AppArmor è installato e caricato in modo predefinito e utilizza i profili di un'applicazione per determinare quali file e permessi siano necessari all'applicazione. Alcuni pacchetti installano i propri profili e ulteriori profili possono essere trovati nel pacchetto *apparmor-profiles*.

sudo apt-get install apparmor-profiles

I profili di AppArmor dispongono di due modalità di esecuzione:

1. Apprendimento (complaining/learning): le violazioni del profilo sono consentite e vengono registrate. Utile per verificare e sviluppare nuovi profili.
2. Esecutiva (enforced/confined): obbliga a rispettare la politica del profilo e registra le violazioni.

## Profili

I profili di apparmor sono dei file di testo situati nella cartella /etc/apparmor.d/ .

Il nome dei file è rappresentativo del path dell’applicativo di cui rappresentano il profilo. Avremo cioè che, volendo profilare il comando ping, situato in /bin/ping il nome del file profilo sarà bin.ping.

Di seguito è rappresentata la struttura di un profilo di apparmor :

# profile to configure foo

#include<tunables/global>

@{HOME} = /home/\*/ /root/ # variabile

/usr/bin/foo {

#include <abstractions/base>

network inet tcp,

capability setgid,

/bin/mount ux,

/dev/{,u}random r,

/etc/ld.so.cache r,

/etc/foo/\* r,

/lib/ld-\*.so\* mr,

/lib/lib\*.so\* mr,

/proc/[0-9]\*\* r,

/usr/lib/\*\* mr,

/tmp/ r,

/tmp/foo.pid wr,

/tmp/foo.\* lrw,

/@{HOME}/.foo\_file rw,

/@{HOME}/.foo\_lock kw,

link /etc/sysconfig/foo -> /etc/foo.conf,

deny /etc/shadow w,

owner /home/\*/\*\* rw,

/usr/bin/foobar cx,

/bin/\*\* px -> bin\_generic

# comment on foo's local profile, foobar.

foobar {

/bin/bash rmix,

/bin/cat rmix,

/bin/more rmix,

/var/log/foobar\* rwl,

/etc/foobar r,

}

}

In un profilo, i commenti sono sempre preceduti dal simbolo # .

## #Include<tunables/global>

Attraverso l’istruzione #include, si può dare al programma l’accesso a una serie di cartelle o file che sono richiesti anche da altri programmi. Usando gli include è possibile ridurre la dimensione di un profilo. Esistono tre tipi di inclusioni:

- **Abstractions**, che rappresentano un insieme di task base per le applicazioni (accesso ai meccanismi di autenticazione, system accounting, accesso alle routine di name service,…)

- **Program chunks** contengono i controlli d’accesso a programmi specifici che l’amministratore vuole regolare in base alle politiche del sistema. Ogni chunk è usato da un singolo programma.

- **Tunables** sono regole di inclusione da altri file. Consente di usare un file comune con regole di inclusione per molteplici applicazioni.

## Variabili locali

Le variabili locali sono definite all’inizio del profilo. Vengono spesso utilizzate per create shortcuts, come nell’esempio:

@{CHROOT\_BASE}=/tmp/foo

/sbin/syslog-ng {

...

# chrooted applications

@{CHROOT\_BASE}/var/lib/\*/dev/log w,

@{CHROOT\_BASE}/var/log/\*\* w,

...

}

## Alias

possono essere usati al posto delle variabili locali per riscrivere percorsi

alias /home/ -> /mnt/users/

## Rules

Nella restante parte del profilo sono invece contenute delle regole di vario genere, in relazione all’aspetto da controllare. Vi sono:

1. **Path Entries**: contengono informazioni circa i file a cui l’applicazione può accedere.
2. **Capability Entries**: determinano i privilegi di cui gode il processo. Queste capability si costruiscono anteponendo la parola “capability” alle capacità descritte nella POSIX.1e
3. **Network Entries**: determinano il tipo di connessione (tcp, datagram, raw) .

Per quello che concerne le Path Entries, le opzioni utilizzabili sono:

| Options | File | Descrizione |
| --- | --- | --- |
| read | r | permesso di lettura |
| write | w | permesso di scrittura |
| link | l | regola la creazione di link verso altri file |
| file locking | k | permesso di bloccare un file |
| file append | a | possibilità di aggiungere dati ad un file |
| deny | deny | nega l’accesso a file o cartelle |
| owner | owner | espressione condizionale per indicare la proprietà di un file (euid e fsuid uguali per file e processo) |
| memory map | m | permesso di mappare in memoria una file eseguibile |

per file eseguibili invece si hanno le seguenti opzioni:

| Option | File | Description |
| --- | --- | --- |
| inherit | ix | deve stare nella stessa cartella del genitore |
| profile | px | richiede che esista un profilo per il programma eseguito |
| Local Profile | cx | richiede che esista un profilo locale per il programma eseguito |
| Unconstrained | ux | esegue il programma senza profile. Modalità da evitare per ragioni di sicurezza |

e infine vi sono anche delle espressioni utilizzate per effettuare l’aggregazione dei percorsi

| Glob | Description |
| --- | --- |
| \* | sostituibile da qualsiasi lettera o numero, eccetto /  (matching con tutti file di una directory) |
| \*\* | sostituibile da qualsiasi sequenza di lettere o numeri, anche / (matching con tutti i file e directory presenti in una cartella ) |
| ? | sostituibile da una lettera o numero, eccetto / |
| [ abc ] | sostituibile con il singolo carattere a, b o c |
| [ a-c ] | sostituibile con il singolo carattere a, b o c |
| {ab, cd} | alternativa: divisibile in due regole, una che effettua un matching con ab e l’altra con cd |
| [ ^a ] | sostituibile con qualunque carattere eccetto che con a |

## Creazione Profilo

I test sono stati effettuati su una macchina in cui è installata la distribuzione Linux OpenSuse 13.01.

Lanciando come root user il comando

apparmor\_status

visualizziamo lo stato dell’applicazione e quindi tutti i profili attivi sulla macchina, sia in modalità complain che in modalità enforce.

Per creare correttamente il profilo di un’applicazione, dobbiamo utilizzare il programma mentre Apparmor si trova in complain mode. Questo procedimento prevede l’esecuzione di varie operazioni sull’applicazione da profilare, quali:

1. l’avvio
2. l’arresto
3. il riavvio
4. utilizzo di eventuali comandi da riga di comando
5. l’utilizzo di tutte le parti di cui si compone l’applicazione

Per eseguire tutta l’analisi e l’apprendimento del comportamento dell’applicazione, bisogna utilizzare innanzitutto il comando:

aa-genprof /percorso/al/binario/da/profilare

Questo andrà a leggere le attività svolte dall’applicazione sul log.

Quindi una volta lanciato il comando, inizierà una sessione di utilizzo dell’applicazione, che dovrà includere tutte le operazioni di base e quindi che cercherà di delineare il comportamento dell’applicazione stessa.

Ultimata la sessione si ritornerà sul terminale per aggiornare la politica, tramite le direttive di genprof, il quale:

1. (S)can system logs : scanning del log per individuare i record dell’applicazione e quindi andrà a creare sotto il controllo dell’operatore il profilo
2. (F)inish : chisura di genprof

Per un profilo ancora più fine e accurato, bisogna mettere il profilo appena generato in complain mode

aa-complain /etc/apparmor.d/profilo

e quindi mediante il comando

aa-logprof

si andrà ad analizzare il log per verificare se ancora sono presenti delle azioni dell’applicazione ancora non contemplate all’interno del profilo corrente.

Dopo aver effettuato tutti i test, si può infine mettere il profilo in modalità enforce .

aa-enforce /etc/apparmor.d/profilo

## Amministrazione del sistema

Oltre a mettere a disposizione comandi utili alla creazione di vari profili Apparmor ne fornisce altri per la gestione del sistema stesso.

* */etc/init.d/apparmor \{stop|start|reload\}* - che controllano il servizio stesso
* *aa-unconfined* - mostra una lista di applicazioni e servizi con porte di rete aperte che non hanno la protezione di appArmor abilitata
* *aa-autodep* - è usato per generare un profilo minimale per AppArmor per un’applicazione o servizio. Il profilo generato avrà bisogno il più delle volte di una sistemata da parte dell’amministratore.
* *aa-easyprof* - fornisce un semplice interfaccia per la generazione delle politiche di AppAmor. Supporta l’uso di template e politiche di gruppo per la profilazione rapida di un’applicazione. I profili creati sono meno restrittivi rispetto a quelli creati manualmente o con l’uso di aa-genprof e aa-logprof

# ULTERIORI INFORMAZIONI

## Distribuzione linux

Abbiamo scelto di installare sulla macchina da mettere in sicurezza la distribuzione di Linux openSUSE. La nostra scelta è ricaduta su questa distribuzione in quanto AppArmor risulta meglio integrato all’interno del sistema e perfettamente funzionante.

Avevamo inizialmente scelto Ubuntu, su cui il funzionamento era minato da alcuni bug, che dovrebbero essere stati risolti negli ultimi mesi.

## Distribuzione Scada

Come distribuzione SCADA su cui effettuare i test è stata scelta EclipseSCADA. E’ un’implementazione di un sistema SCADA basata su una piattaforma Java. Inizialmente nato sotto il nome di openSCADA, il quale utilizzava parecchie tecnologie sviluppate dalla fondazione Eclipse. Così per aumentarne anche la visibilità, il progetto è stato portato all’interno dell’universo Eclipse. Tra i vari tools messi a disposizione, vi è un client, che simula il funzionamento di un sistema scada.

Abbiamo scelto appunto questo client per effettuare i nostri studi .

# LINK UTILI

* <http://www.eclipse.org/downloads/download.php?file=/eclipsescada/downloads/org.eclipse.scada.hmi/drops/0.2.0/R201501260629/products/adminclient/org.eclipse.scada.hmi.products.adminclient.app-linux.gtk.x86_64.tar.gz&mirror_id=580>
* <https://eclipse.org/eclipsescada/index.html>