Virtualizzazione e Sicurezza

Giuseppe Paternò
Marzo 2007

Stiamo assistendo alla evoluzione ed implementazione delle tecnologie di virtualizzazione come metodologia di consolidamento dei server dei datacenter. Possiamo considerare però la virtualizzazione anche come un'opportunità per aumentare la sicurezza dei nostri ambienti.

I sistemi di difesa sono sempre stati considerati un'arte sin dagli antichi Romani. Nel medioevo, i manoscritti più preziosi erano protetti da diversi livelli (o in inglese layers) di difesa: erano conservati in castelli che erano sempre posizionati nella collina più alta, circondati da fossati pieni di coccodrilli, nella torre più difficile da raggiungere, e nascosti in una stanza segreta protetta da trappole mortali. Anche se un intruso riesce a penetrare nel primo livello di difesa, più prosegue nel suo attacco, più incontrerà resistenza: in questo modo l'attaccante si stanca e potrebbe non riuscire a superare i livelli di protezione successivi. Anche se i mezzi, gli attaccanti e le cose da proteggere sono cambiate nei secoli, le tecniche di difesa rimangono valide ancora oggi.

Anche nell'informatica, quindi, andrebbere applicata una difesa basata su differenti livelli di protezione (*security layers*). Questa filosofia è chiamata **defense in depth**. Questo pensiero, in breve, rappresenta l'uso di differenti tecniche di sicurezza per mitigare i rischi che un eventuale livello venga compromesso o scavalcato. Un classico esempio è quello di un antivirus su ogni workstation e uno sui mail/file servers. **Questa filosofia dovrebbe essere applicata in ogni sistema, specialmente in quei sistemi contenenti dati critici o in ambienti ad alta sensibilità (militare/governi/finanziari/...)**. Tutti i sistemi, configurazioni, applicazioni e sviluppo software devono essere pensati fin dall'inizio in un ottica di sicurezza, ad esempio dando il minor privilegio possibile ad un utente su una applicazione o restringendo il raggio d'azione di una rete wireless.

Possiamo applicare la filosofia nei seguenti componenti di sicurezza di un sistema (questo si applica in special modo a Linux):

- Installazione di un sistema minimizzato
- Hardening del sistema
- Configurazione di un sistema con Mandatory Access Control (SELinux)
- Utilizzo della virtualizzazione per confinare un servizio/applicazione
- Configurazione sicura dell'applicazione
- Autenticazione degli utenti (strong authentication)
- Aggiornameto periodico delle patch di sicurezza (Red Hat Network, yum, Windows update,...)

I componenti descritti non sono esclusivi tra loro, nè devono essere necessariamente utilizzati tutti. Ovviamente maggiori saranno i componenti attivati, maggiore sarà la sicurezza del sistema. Non esiste una “ricetta magica” per capire quanti layers andranno attivati, dipende dalla tipologia dell'ambiente in cui è calato il sistema. Non mi soffermerò su ciascun componente, ma mi focalzzerò sull’aspetto della virtualizzazione e su quelli ad esso collegati.

Si parla molto di virtualizzazione in questo periodo, ma vorrei fare un pò di chiarezza sull'argomento. La virtualizzazione è una tecnica per ricreare -via software- un ambiente che appare come un hardware al sistema operativo ospite. Tradizionalmente la virtualizzazione viene utilizzata per consolidare molti server fisici in pochi sistemi che ospitano macchine virtuali, mentre alcune volte può venire utilizzata in ambienti di disaster-recovery. Possiamo distinguere tre tipi di virtualizzazione:

Full virtualization (o native virtualization)

Emula completamente l'hardware (Bios, processori, ...) ed è possibile ospitare sistemi operativi senza che questi vengano modificati.
Il classico esempio è VMware.

Operating system-level virtualization (o Single Kernel Image, SKI)

Un esecuzione leggera del sistema operativo: il sistema operativo "master" si duplica in memoria. Il sistema operativo ospite esegue esattamente lo stesso sistema operativo del suo sistema principale: la differenza è che il kernel non viene rimandato in esecuzione.
Esempio: le zone di Solaris 10, Virtuozzo, BSD Jails.

Paravirtualizzazione

Fondato da Xen Source, è una sintesi delle due precedenti. Offre un ambiente isolato (ovvero il kernel viene eseguito) ed è meno pesante in termini di risorse di una full virtualization. La macchina virtuale non simula l'hardware in toto, ma offre specifiche API che però richiedono la modifica del sistema operativo ospite.
Questo è il caso di Xen.

Nell'ottica del defense in depth, la virtualizzazione può introdurre un livello di sicurezza in più. I programmi girano normalmente in un sistema operativo completo, e se un intruso ne viola i servizi, questo avrà a disposizione l'intero sistema operativo dove possibilmente verranno eseguite altre applicazioni. Fino ad adesso le tecniche per mettere in sicurezza un servizio era di confinarli in gabbie chroot e nelle BSD jails. Queste due metodologie sono efficaci per aumentare il livello di sicurezza di un server perchè offrono una separazione tra l'ambiente chroot/jailed e il resto del sistema, creando una sandbox. **E' stato però scoperto che in rare circostanze è possibile raggirare l'ambiente chroot ed accedere all'intero sistema.** Utilizzare Xen è la perfetta sintesi per ambienti particolarmente sensibili:

- in ambienti Linux enterprise, Xen è una soluzione certificata e supportata rispetto a chroot (es: Red Hat non supporta ufficialmente i programmi in chroot con eccezione di bind, di cui ne esiste un pacchetto ad-hoc);
- è molto più sicuro delle zone di Solaris, perchè il kernel viene caricato in memoria (e un programma non può uscire dalla memoria del kernel)
- è più leggero di una macchina emulata in toto.

A titolo di esempio possiamo costruire una nuova macchina virtuale che serva solamente al nostro scopo (es: un web server), anche senza ssh abilitato.

La figura 1 rappresenta la visione d'insieme di un'architettura sicura basata su Xen sotto Linux. Il Domain 0 (o Xen Hypervisor) è il sistema minimizzato ed hardenizzato che contiene al suo interno esclusivamente le macchine virtuali ed il sistema di gestione, cui accesso è limitato alla sola interfaccia di amministrazione (eth0 o bond0). Al suo interno viene abilitato SELinux, che fornisce un'alta protezione dei processi/servizi, all'interno del quale gireranno le macchine virtuali. Tali macchine virtuali sono sistemi Linux (o altri sistemi che supportano Xen), a sua volta minimizzati, hardenizzati per lo scopo a cui è/sono destinato il/i servizio/i. I servizi verranno erogati protetti a loro volta da SELinux e confinati alla interfaccia di erogazione del servizio. Per definire su quali interfacce verranno abilitati il bridging, si userà il comando di Xen *network-bridge*. Gli esempi seguenti sono stati provati con Red Hat Enteprise Linux (RHEL) 5, ma possono essere eseguiti con Fedora o con una qualsiasi distribuzione Linux che suppoti Xen. Il comando successivo associa l'interfaccia di rete del sistema guest ad una determinata interfaccia (eth1) del sistema host:

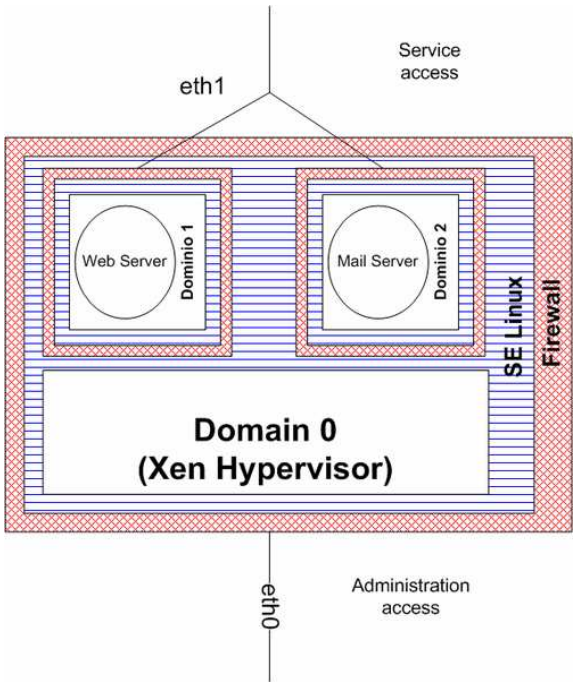

Figura 1 – Visione di insieme di un'architettura sicura con Xen

```
# /etc/xen/scripts/network-bridge netdev=eth1
```

Per minimizzare i tempi di installazione, nel caso di Fedora o RedHat Enterprise Linux è consigliabile utilizzare un file di kickstart che prepara un'ambiente minimizzato e già hardenizzato. Ad esempio, si installerà una macchina virtuale con:

```
# /usr/sbin/virt-install -f /srv/xen/test2.img -s 4 -n test2 -
r 256 -l nfs:10.10.10.10:/var/netinst/RHEL-5-Server/ -x
ks=http://10.10.10.10/ks/min-rhel5.ks
```

Nell'esempio precedente, il server 10.10.10.10 esporta via NFS una immagine della distribuzione RHEL5 Server e pubblica via web il file di kickstart. Vediamo brevemente la linea di comando. L'opzione "*-f*" specifica il file di immagine del disco virtuale, "*-s*" indica la grandezza del disco, "*-n*" il nome della macchina virtuale, "*-l*" indica da dove prendere la distribuzione e infine "*-x*" passa dei parametri al kernel. In

questo caso, passeremo al kernel la URL da cui prendere il file di kickstart. Il file di kickstart specificato nel comando precedente è disponibile sul mio sito Internet (e in calce a questo whitepaper):

http://www.gpaterno.com/files/min-rhel5.ks
(mirror disponibile su: http://gpaterno.free.fr/files/min-rhel5.ks)

Il file di kickstart installerà un web server minimizzato e hardenizzato con PHP. Questo file può essere usato anche con Fedora commentando le opzioni “key --skip” e il comando “rhnreg_ks” eseguito nella sezione *%post*. Il primo è utilizzato da RHEL5 per identificare che tipo di distribuzione è stata sottoscritta, mentre il secondo registra automaticamente il sistema nel Red Hat Network Satellite presente in azienda e aggiorna il sistema all’ultimo livello di patch disponibile. Ovviamente questo può essere modificato per qualsiasi tool di system management o per altre azioni.

Nella figura 2 vuole dare una visione d'insieme su come verrà erogato un servizio tramite macchine virtuali Xen.

La virtualizzazione può essere utile in caso di restore dopo un eventuale *break-in* da parte di un intruso. Fare il backup di un sistema equivale a copiare il file di immagine disco, nell'esempio precedente il file *test.img*. In caso di break-in possiamo isolare il file di immagine e farne un'analisi forense, mentre per ripristinare velocemente il servizio sarà sufficiente copiare il file originale (il cui backup è stato fatto dopo l'installazione completa) applicandone le relative patch di sicurezza o modificando il proprio programma per evitare ulteriori intrusioni. Immaginiamo quindi quanto **Xen possa aiutarci integrandolo in un processo di sicurezza o di disaster recovery**.

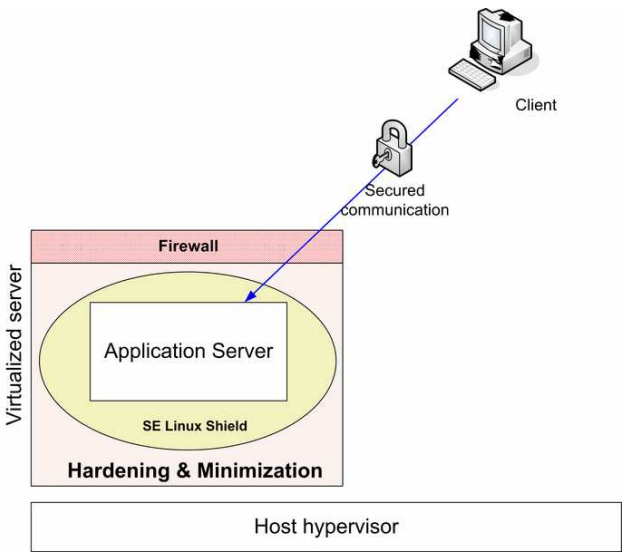

Figura 2 – Esempio di erogazione di un servizio sicuro

Ritengo che la virtualizzazione, preziosa fino ad adesso sulla server consolidation e sulla flessibilità dell'organizzazione dei sistemi, possa fornirci un valido aiuto ad aumentare la sicurezza della nostra infrastruttura, eseguendo macchine virtuali sicure, con applicativi configurati in maniera sicura, vigilati da sistemi di management, protetti da firewalls (perimetrali e applicativi) e sistemi mandatori (SElinux).

File di kickstart

```
#
# Minimized RHEL5 for web server, to be used under Xen
# Proudly done on 2006 by Giuseppe Paterno' <gpaterno@redhat.com>
#

#platform=x86, AMD64, or Intel EM64T
# System authorization information
auth  --useshadow  --enablemd5
# System bootloader configuration
bootloader --location=mbr --driveorder=xvda
# Partition clearing information
clearpart --all --initlabel
# Use text mode install
text
# Firewall configuration
firewall --enabled --http --ssh
# Run the Setup Agent on first boot
firstboot --disable
# System keyboard
keyboard it
# System language
lang en_US
# Use NFS installation media
nfs --server=10.10.10.10 --dir=/var/netinst/RHEL-5-Server/
# Network information
network --bootproto=dhcp --device=eth0 --onboot=on --hostname webserver
# Reboot after installation
reboot

#Root password
rootpw --iscrypted $1$HtCu0VE4$zEJ3j0B3UzCQhUTsO4k8M1
# SELinux configuration
selinux --enforcing
# Do not configure the X Window System
skipx
# System timezone
timezone  Europe/Rome
# Install OS instead of upgrade
install
# Clear the Master Boot Record
zerombr
# Disk partitioning information
part /boot --asprimary --bytes-per-inode=4096 --fstype="ext3" --ondisk=xvda
--size=100
part pv.2 --size=0 --grow --ondisk=xvda
volgroup VolGroup00 --pesize=32768 pv.2
logvol swap --fstype swap --name=LogVol01 --vgname=VolGroup00 --size=272 --
grow --maxsize=544
logvol / --fstype ext3 --name=LogVol00 --vgname=VolGroup00 --size=1024 --
grow
## Skip key
key --skip

%packages
```

```
@core
@web-server
-squid
-NetworkManager
-caching-nameserver
-bind
-bluez-utils
-bluez-pin
-bluez-libs
-wpa_supplicant
-autofs
-nfs-utils
-ypbind
-yp-tools
-portmap
-pinfo
-htmlview
-redhat-menus
-desktop-file-utils
-GConf2
-libglade2
-gtk2
-pango
-cairo
-finger
-webalizer
-jwhois
-libXinerama
-irda-utils
-tux
-parted
-pcmciautils
-dump
-rmt
-rsh
-stunnel
-tcpdump
-vconfig
-diskdumputils
-nc
-sysreport
-acpid
-rsync
-httpd-manual
-unix2dos
-words
-nfs-utils-lib
-ipsec-tools
-telnet
-ftp
-eject
-syslinux
-mkbootdisk
-mtools
-dosfstools
-nano
-lftp
-traceroute
```

```
-mgetty
-talk
-rdist
-mlocate
-apmd
-ppp
-rp-pppoe
-gpm
-crash
-man-pages
-utempter
-dos2unix
ntp

%post
/sbin/chkconfig sshd off
/sbin/chkconfig httpd on

## Crea un’utenza di servizio
/usr/sbin/groupadd -g 300 admin
/usr/sbin/useradd -u 300 -g admin -d /home/admin -m admin
echo "redhat" | /usr/bin/passwd --stdin admin

## Registra su RedHat Network
echo "Registering on RHN ..."
/usr/sbin/rhnreg_ks --username="username" --password="password" --
activationkey=89c30f97f244fc957c5d6f7c49ae3a1e --
serverUrl=https://mysatellite.lan/XMLRPC

## Aggiorna il sistema con le ultime patch
echo "Updating…"
/usr/bin/yum –y update
```