Linux-VServer
Una macchina, tante macchine.

Micky Del Favero - Dino Del Favero
micky@delfavero.it - dino@delfavero.it

BLUG - Belluno Linux User Group
Linux Day 2005 - Feltre 26 novembre 2005

Cos'è VServer

Un sistema di partizionamento basato sui *Security Contexts* che permette di creare più *Virtual Private Servers (VPS)* simili ad un server normale che condividono le stesse risorse hardware e lo stesso kernel.

Un VPS è un server virtuale che, dal punto di vista dei processi che ospita, è in tutto e per tutto uguale ad un server reale.

Caratteristiche

- Una sola macchina fisica e un solo kernel condivisi fra tutti i VServer.
- Su ogni VServer può venir installata una qualsiasi distribuzione.
- Ogni processo di ogni VServer gira alla massima velocità possibile come girasse sul server host.
- Qualsiasi servizio può venir lanciato su un VServer senza necessità di modifica alcuna rispetto ad un server reale.
- Ogni VServer ha il proprio database di utenti distinti dagli altri (incluso root).

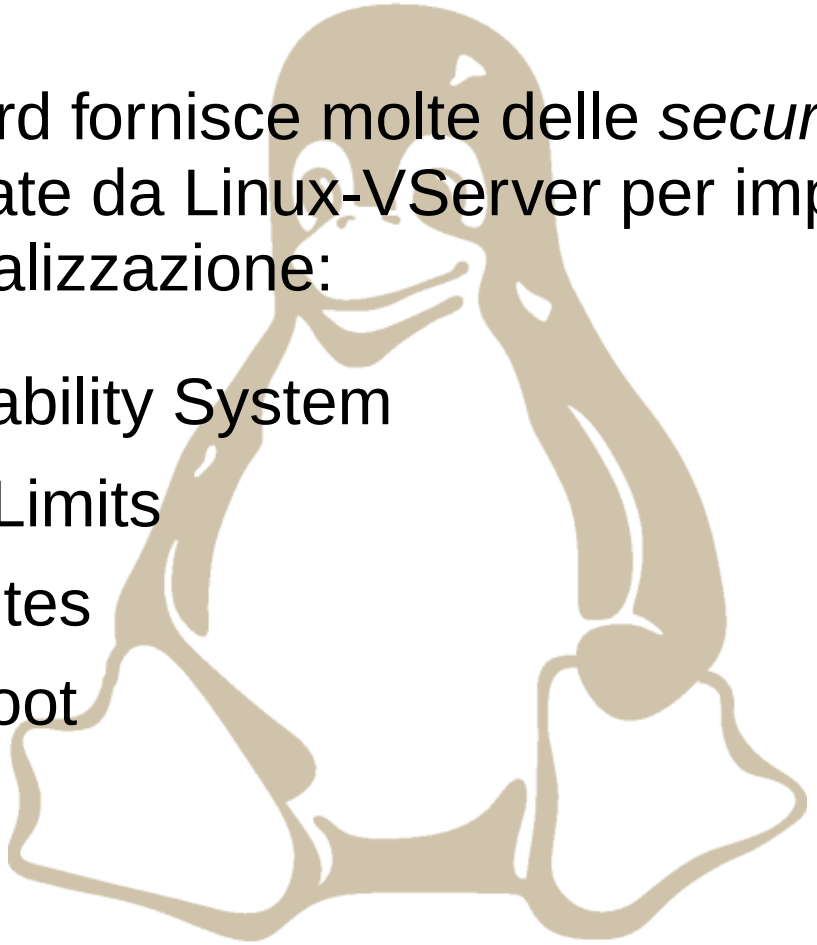
Vantaggi

- Performace identiche ad un server reale, no overhead!
- Minima occupazione di spazio su disco grazie alla condivisione dei binari fra i vari VServer.
- Indipendenza dei VServer dall'hardware degli host su cui girano purché l'architettura sia compatibile.
- Gli strumenti di amministrazione si comportano come su un server reale.
- Sicurezza: un VServer anche se compromesso rimane isolato dagli altri.

Tecnologia di Linux-VServer

Il kernel standard fornisce molte delle *security features* che vengono utilizzate da Linux-VServer per implementare il sistema di virtualizzazione:

- Linux Capability System
- Resource Limits
- File Attributes
- Change Root



Modifiche al kernel

Al fine di garantire l'isolamento fra i vari VServer che contemporaneamente girano nell'host è necessario apportare alcune modifiche al kernel, esse sono:

- Context Separation
- Network Separation
- The Chroot Barrier
- Upper Bound for Caps
- Filesystem XID Tagging

Context Separation

Lo scopo è quello di nascondere al contesto preso in esame tutti i processi al di fuori di esso e proibire qualsiasi interazione fra processi appartenenti a contesti diversi.

Si è resa necessaria l'estensione di alcune strutture dati per renderle sensibili ai contesti e per permettere la differenziazione di identici uid usati in contesti differenti.

Un contesto di default, identico a tutti gli altri, è usato per permettere il boot dell'host.

Uno speciale constesto, *Spectator*, permette una vista globale dei processi presenti nell'host.

Network Separation

Un' ulteriore separazione è necessaria per confinare i processi in un sottoinsieme dello spazio degli indirizzi di rete disponibili all'host.

Per garantire l'isolamento bisogna tener conto di parecchi problemi che potrebbero sorgere, come per esempio il poter, da parte di un processo, fare il binding dell'indirizzo speciale IPADDR_ANY, questo è garantito essere possibile senza modifica alcuna al processo che gira nel VServer.

Al fine di ridurre l'overhead risultante non si fa uso di network device virtuali, si è invece intervenuti sul sistema di binding dei socket e del packet transmission all'interno del kernel.

The Chroot Barrier

Uno dei problemi che affliggono la chiamata a sistema chroot() di Linux è che l'informazione associata a chroot() è volatile e può venir persa alla successiva chiamata, è sufficiente che un processo chiami open(), chroot() e fchdir() perché questo avvenga.

Le prime versioni di Linux-VServer hanno cercato di risolvere il problema in vari modi, attualmente si usa un modo molto semplice ed efficace: un mark, chiamato Chroot Barrier, messo nella directory padre del VServer che impedisce lo sconfinamento.

Upper Bound for Caps

Dato che le Linux Capability non implementano la parte relativa al filesystem descritta da POSIX Capabilities che renderebbe gli eseguibili setuid e setgid sicuri e che è più sicuro avere un upper bound sicuro per tutti i processi in un contesto si è aggiunta una ulteriore maschera di capability che limita tutti i processi appartenenti al contesto della maschera.

Filesystem XID Tagging

Modifica necessaria per avere un maggior isolamento dei contesti e per poter abilitare il *Contest Disk Limit* e il *Per Context Quota Support* su una partizione condivisa fra più VServer.

Aggiungere un conte**X**t **ID** ad ogni file non è banale perché richiede o la modifica alla rappresentazione su disco del filesystem oppure necessita di sfruttare qualche bit delle strutture dati esistenti.

Una soluzione non invasiva consiste nello sfruttare i bit più significativi di UID e GID per memorizzare lo XID.

Filesystem XID Tagging

Una volta che l'informazione sul contesto è disponibile in ogni inode la successiva modifica comprende estendere il controllo di accesso in modo tale che consideri anche il contesto.

Attualmente tutti gli accessi agli inode considerano le restrizioni riguardanti il contesto ad eccezione di quelli fatti nei contesti Host e Spectator.

Un file che non ha lo XID viene considerato appartenente al contesto Host. Alla prima modifica al file lo XID cambia e diventa quello del contesto in cui è avvenuta la modifica.

Tagging Methods

- **UID32/GID32 o EXTERNAL** Utilizza lo spazio inutilizzato all'interno degli inode per memorizzare l'informazione, ma richiede modifica al filesystem (ext2/3 ok, presto gli altri). Vantaggi: UID e GID a 32 bit.
- **UID32/GID16** I 16 bit più significativi del GID portano l'informazione relativa allo XID trasparentemente. Vantaggi: funziona con tutti i filesystem. Svantaggi: GID ridotto a 16 bit.
- **UID24/GID24** Gli 8 bit più significativi di UID e GID portano l'informazione relativa allo XID trasparentemente. Vantaggi: funziona con tutti i filesystem. Svantaggi: UID e GID ridotti a 24 bit.

Ulteriori features

Linux-VServer oltre a garantire l'isolamento dei VServer sull'host ha in sè ulteriori caratteristiche importanti:

- Unification
- Private Namespaces
- Token Bucket Extensions
- Context Disk Limits e Per-Context Quota

Unification

File comuni a più contesti che realisticamente non cambiano molto spesso (librerie o binari) possono essere hard-linkati riducendo l'occupazione di disco, il caching degli inode e la mappa in memoria delle librerie condivise.

Problema: codice maligno in un contesto può distruggere o modificare i file condivisi minando la sicurezza.

I file condivisi vengono resi immutabili usando l'Immutable File Attribute e rimuovendo dai contesti la capability necessaria a modificare questo attributo. Un attributo addizionale permette la rimozione contesto-dipendente dell'attributo Immutable per permettere aggiornamento di librerie o binari all'interno di un contesto.

Token Bucket Extensions

Usato per controllare la distribuzione delle risorse quali: Hard CPU Limit, Scheduler Priorities e Network Bandwidth Limitation, fra i vari contesti.

Per ogni quanto di tempo T un “bucket” di dimensione S è riempito con una quantità R di “token”. Ad ogni tick del clock, un processo in esecuzione consuma un token del bucket, quando il bucket è vuoto il processo viene posto in una coda *hold* finché una quantità minima M di token non viene ripristinata nel bucket, il processo poi sarà rischedulato.

Context Disk Limits e Per-Context Quota

Context Disk Limits:
Attraverso l'uso del tag XID sui file viene tenuto conto del numero degli inode e dei blocchi usati per ogni filesystem rendendo possibile implementare i Disk Limit per ogni contesto.

I valori di uso corrente, massimo e riservato sono mostrati ad ogni richiesta sul filesystem come ogni contesto fosse su un filesystem riservato.

Per-Context Quota:
Usa hash separati per i differenti contesti su un filesystem condiviso, non è necessaria per abilitare la quota su VServer che risiedono su differenti partizioni.

Un esempio pratico

Si compila un kernel patchato con Linux-VServer e si installa con i pacchetti necessari:

```
micky@teodora:~$ cd /usr/src
micky@teodora:~$ wget http://www.kernel.org/pub/linux/\
kernel/v2.6/linux-2.6.12.4.tar.bz2
micky@teodora:~$ wget http://www.13thfloor.at/vserver/\
s_rel26/v2.0/patch-2.6.12.4-vs2.0.diff
micky@teodora:~$ tar -jxvf linux-2.6.12.4.tar.bz2
micky@teodora:~$ cd linux-2.6.12.4
micky@teodora:~$ bzcat ../patch-2.6.12.4-vs2.0.diff.bz2 | patch -p1
micky@teodora:~$ make -j menuconfig
micky@teodora:~$ fakeroot make-kpkg kernel_image
micky@teodora:~$ cd ..
micky@teodora:~$ su -
root@teodora:~# dpkg -i kernel-image-2.6.12.4-vs2.0_i386.deb
root@teodora:~# apt-get install util-vserver vserver-debiantools
```

Un esempio pratico

Si crea il VServer:

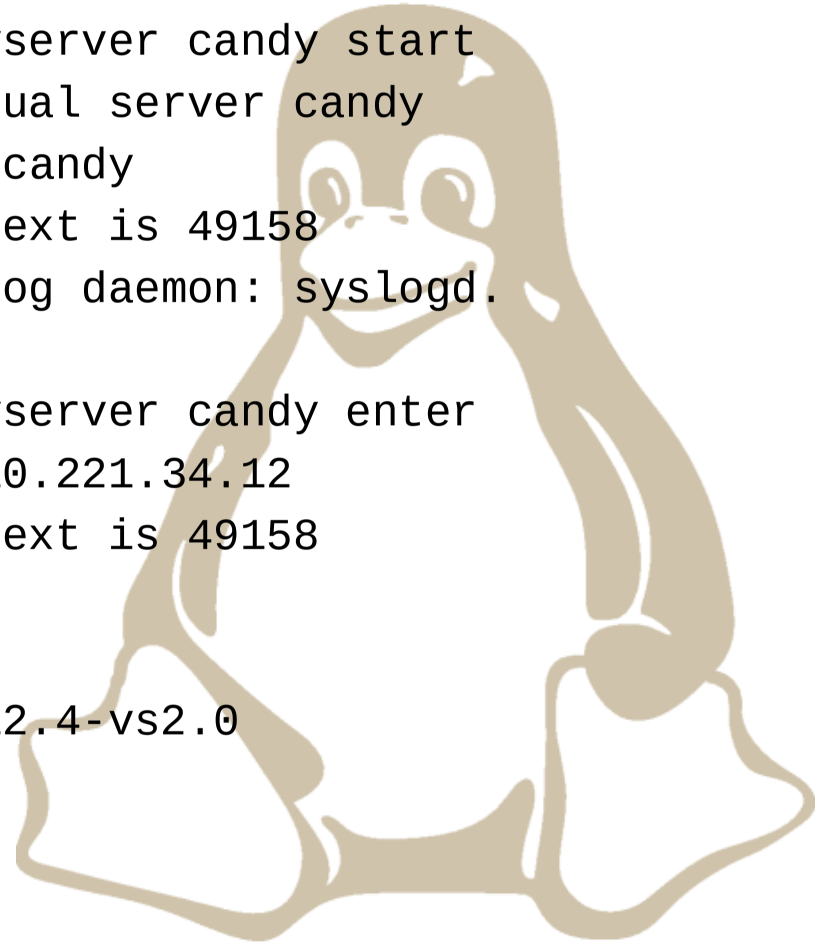
```
root@teodora:~# newvserver --hostname candy --domain mesina.net \
--ip 10.221.34.12
I: Retrieving debootstrap.invalid_dists_sarge_Release
...
Server candy is not running
Restarting rebootmgr.
Stopping the reboot manager
Starting the reboot manager

You should now adjust /etc/vservers/candy.conf to suit your needs,
or else just go ahead and type ‘vserver candy start’ to start
your new virtual server.  debian/rules!
root@teodora:~#
```

Un esempio pratico

E si fa fare il boot al VServer:

```
root@teodora:~# vserver candy start
Starting the virtual server candy
Host name is now candy
New security context is 49158
Starting system log daemon: syslogd.
...
root@teodora:~# vserver candy enter
ipv4root is now 10.221.34.12
New security context is 49158
candy:/#
uname -a
Linux candy 2.6.12.4-vs2.0
candy:/# exit
logout
root@teodora:~#
```



Domande? Grazie per l'attenzione.