By Tony2

Published: 2008-06-16 19:55

Some Tips On OpenVZ Deployment

I rely heavily on **OpenVZ**. In this article I would like to share some of my personal experiences in OpenVZ deployment. I assume that the readers already know **how to install OpenVZ** and the basics of OpenVZ. This article describes some tips on OpenVZ usage via the command line. If you prefer GUI to command line, please turn to **how to install WebVZ**.

The setup described here follows these guidelines:

- the real server has minimum software installed (I use debian Etch withminimal installation) as the starting point. Additional applicationsare installed as needed along the deployment.
- the real servershould be as secure as possible. On the other hand, I want to keep itsimple and easy to setup/maintain. So I chose a compromise: I rely only on what can be easily deployed with debian and don't go for extrasecurity stuff like openwall, selinux, grsecurity, etc.
- each needed service is deployed in a separate container, so that they interfere each other as little as possible
- Intrusion Detection for the real server as well as the containers is deyployed on the real server using **OSSEC**
- firewalling (iptables) is done on the real server; the containers run only the services
- I rely on ssh as the only mean to access and maintain the real server and the containers.

Basic Security

Before deploying any ovz containers, I make some changes to the configuration of the real server to make it more secure:

- disable root password
- add a user *admin-user* who can sudo everything; this user has a simple password
- add a user *ssh-user* who can ssh to the real server; after uploading ssh key for this user, I change the line in *.ssh/authorized_keys* to read:

```
command="/bin/su - admin-user" ssh-rsa AAAA...
```

and change */etc/pam.d/su* to allow only this user to *su*:

```
auth       required   pam_wheel.so group=ssh-user
```

- to copy files from/to the real server, I create another user *sftp-user* and **install MySecureShell**
- change *sshd_config* so that:

- only *ssh-user* and *sftp-user* are allowed to connect
- clear password authentication is disable (after uploading ssh keys for *ssh-user* and *sftp-user*)
- sshd runs on a non-standard port

The above scheme works as follows: to connect to the real server, we connect as *ssh-user*. Then we must type in password for *admin-user*. If someone gains ssh key for *ssh-user*, he still must know the password for *admin-user* in order to gain access to the server (failure of */bin/su - admin* will immediately generate an email alert by OSSEC).

To copy files from/to the server, we use sftp with the account *sftp-user*. If someone gains ssh key for this user, this is not such a big problem, since he can access only files under his *$HOME*.

Creating OpenVZ Containers

I find it more comfortable to create a template for all containers, so that when I need a new container, I simply make make a clone from the template. I use only debian stable for the real server as well as for the containers. So, the first step is to create a template and tune it to my taste:

- create a new container:

```
vzctl create 2002 --ostemplate debian-4.0-amd64-minimal
```

- set some basic parameters:

```
vzctl set 2002 --ipadd 192.168.100.2 --nameserver 1.2.3.4 --hostname host2 --save
```

- start the container:

```
vzctl start 2002
```

- enter the container:

```
vzctl enter 2002
```

- I prefer to keep everything at the bare minimum, and add stuff asneeded. Since installing a package with debian is so easy, it takesvery little effort to install any package we need. So I make thefollowing changes for the template:

- run aptitude and uncheck *Options/Dependency handling/Install Recommended packages automatically*
- remove some packages that I don't want in the template:

```
bsdmainutils
ed
groff-base
info
iptables
libconsole
libgdbm3
man-db
manpages
nano
netcat
openssh-client
openssh-server
quota
ssh
traceroute
```

- edit */etc/apt/sources.list* to tune it to my preferences
- then I stop the template:

```
vzctl stop 2002
```

Then anytime I need a new container, I use a script `vz-clone` as follows:

```
#!/bin/bash

# script to clone an openvz VE

set -e

if [ -z "$2" ]; then
    echo "Usage: $0 <veid> <new-id>"
    exit 1
fi

cfg="/etc/vz/conf/$1.conf"
newcfg="/etc/vz/conf/$2.conf"

if [ ! -e $cfg ]; then
    echo $cfg not found!
 exit 1
fi

VEID=$1
. $cfg
veprivate="$VE_PRIVATE"

VEID=$2
. $cfg
newveprivate="$VE_PRIVATE"

if [ -e $newcfg ]; then
    echo $newcfg already exists!
 exit 1
fi
```

```
if [ -e $newveprivate ]; then
   echo $newveprivate already exists!
 exit 1
fi

if vzlist | fgrep -w -q $1
then
   vzctl stop $1
fi

echo "Cloning $cfg to $newcfg"
cp -a $cfg $newcfg

echo "Cloning $veprivate to $newveprivate"
mkdir -p $newveprivate
cd $veprivate
tar cf - . | (cd $newveprivate && tar xf -)

echo "Do not forget to edit $newcfg (you need to edit at least HOSTNAME and IP_ADDRESS)"
echo "Also do not forget to make an alias"
```

Usage:

```
sudo sh vz-clone 2002 2010
```

```
Cloning /etc/vz/conf/2002.conf to /etc/vz/conf/2010.conf
  Cloning /vz/private/2002 to /vz/private/2010
  Do not forget to edit /etc/vz/conf/2010.conf (you need to edit at least HOSTNAME and IP_ADDRESS)
Also do not forget to make an alias
```

Depending on your */etc/vz/vz.conf*, the paths in the above might be different. I use the below settings:

```
VE_ROOT=/vz/root/$VEID
VE_PRIVATE=/vz/private/$VEID
```

Then we need to edit */etc/vz/conf/2010.conf*, change e.g. HOSTNAME to **host10**, *IP_ADDRESS* to **192.168.100.10** and we are ready to go with the new container. We will also make an alias for the new container, which will be described in the nextsection.

Working With OpenVZ Containers

The ovz containers are identified by number. I find it easier to refer to them by name/alias, so that I don't have to remember for example *2010* is the id of the container running dns service. Apart from that, I also want tofree myself from remembering the different commands *vzctl*, *vzlist*, *vzquota*, etc. and their parameters. So I create some simple scripts to help myself.

- First I create a list of aliases */etc/vz-aliases*:

```
# aliases for openvz VE's
2001  test
2002  template
2010  dns
2020  ldap
2030  mail
2040  web
...
```

- To translate between ID's and aliases, I create a script */usr/local/bin/vz-get-alias* as below and make *vz-get-veid* as symlink to *vz-get-alias*:

```
#!/bin/sh
vz_alias_file="/etc/vz-aliases"

case $0 in
*vz-get-alias)
```

```
    cat $vz_alias_file | egrep "^[[:space:]]*$1[[:space:]]" | awk '{print $2}'
    ;;
*vz-get-veid)
    cat $vz_alias_file | egrep "[[:space:]]$1[[:space:]]*$" | awk '{print $1}'
 ;;
esac
```

- Then I put frequent commands to manipulate ovz containers to a script called `/usr/local/bin/vz-cmd-generic`:

```
#!/bin/sh
set -e

## handle vz-list first, since it requires no ID/alias
case $0 in
*vz-list)
    sedfile=`mktemp`
 cat /etc/vz-aliases | egrep '^[0-9]' | \
 sed 's/\([0-9]*\) *\([a-zA-Z0-9-]*\)/s,\1 .*,\&\2,/' > $sedfile
 sudo vzlist "$@" | sed 's/                    $//' | \
 sed -f $sedfile | \
 sed '1s/$/ALIAS/'
 exit
 ;;
esac

## the other commands require an ID or alias
if [ -z "$1" ]; then
    echo "Usage: $0 <veid>|<alias> [<args>]"
 exit 1
fi

veid=`/root/bin/vz-get-veid $1`
```

```
if [ -z "$veid" ]; then
    veid=$1
fi

shift

case $0 in
*vz-start)
    sudo vzctl start $veid
 ;;
*vz-restart)
    sudo vzctl restart $veid
 ;;
*vz-stop)
    sudo vzctl stop $veid
 ;;
*vz-enter)
    sudo vzctl enter $veid
 ;;
*vz-exec)
    sudo vzctl exec $veid "$@"
 ;;
*vz-edit)
    sudo vi /etc/vz/conf/$veid.conf
 ;;
*vz-quota-ls)
    sudo vzquota stat $veid
 ;;
*vz-ubc)
    sudo head -2 /proc/user_beancounters
 sudo cat /proc/user_beancounters | egrep -A23 "^[[:space:]]+${veid}:"
 ;;
```

```
esac
```

And make all the command `vz-start`, `vz-stop`, `vz-exec`, etc. as symlinks to this script `vz-cmd-generic`.

Usage is then simple:

- to list all running containers:

```
vz-list
```

```
 VEID    NPROC STATUS  IP_ADDR        HOSTNAME    ALIAS
 2010       15 running 192.168.100.10  host10      dns
 2020        8 running 192.168.100.20  host20      ldap
 2030       23 running 192.168.100.30  host30      mail
 2040       11 running 192.168.100.40  host40      web
```

- to list all containers (including those which are not running):

```
vz-list -a
```

```
 VEID    NPROC STATUS  IP_ADDR        HOSTNAME    ALIAS
 2002        - stopped 192.168.100.2   host2       template
 2010       15 running 192.168.100.10  host10      dns
 2020        8 running 192.168.100.20  host20      ldap
 2030       23 running 192.168.100.30  host30      mail
 2040       11 running 192.168.100.40  host40      web
```

- to start/stop/restart a container:

-

```
vz-start dns
```

-

```
vz-stop dns
```

-

```
vz-restart dns
```

- to exec a command inside a container:

```
vz-exec dns aptitude update
```

- to check UBC of a container:

```
vz-ubc dns
```

- to check quota of a container:

```
vz-quota-ls dns
```

- it is also possible to use an ID instead of an alias:

```
vz-ubc 2010
```

It is also a good thing to keep the alias unique across different real servers, so that we can share `/etc/vz-aliases` between them without conflicts.

This article is already quite long, so let's stop here. We will continue in the next part, where we discuss issues like how to deploy Instrusion Detection with OSSEC, how to monitor and set UBC parameters for containers, etc.