User Mode Linux

Groupe : Philippe Audéoud, Samir Bellabes, Arnaud Berthomier, Gregory Kirijean, Pierre Pronchery

Sommaire

I- Définition.....3

II – La préparation.....3

III – Le 1er test.....5

IV- Mettre le réseau.....5

I- Définition

User Mode Linux est un dispositif permettant de lançer plusieurs noyaux linux dans l'espace utilisateur. Il permet d'avoir plusieurs machines virtuelles sur une seule machine physique hôte exécutant Linux. UML utilise sa propre architecture, sa gestion de la mémoire, la machine uml dispose d'ailleurs de son espace mémoire sur la machine hôte.

Par convention, nous appellerons *neptunium* la machine hôte et *neptunium-uml* l'UML.

II – La préparation

Il faut une image d'un OS. Pour cela, le noyau de notre machine hôte doit contenir 2 options : CONFIG_BLK_DEV_LOOP=m et CONFIG_ETHERTAP=m. Le premier sert à pouvoir monter l'iso et le second à utiliser un tunnel TAP pour communiquer avec l'UML.

La création d'une image se fait comme suit :

```
[jadawin@neptunium]$ dd if=/dev/zero of=debian.iso bs=1M count=800
800+0 enregistrements lus.
800+0 enregistrements écrits.
838860800 bytes transferred in 96,881592 seconds (7988577 bytes/sec)
[jadawin@neptunium]$ mkfs.ext3 debian.iso
mke2fs 1.30-WIP (30-Sep-2002)
debian.iso is not a block special device.
Proceed anyway? (y,n) y
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
89664 inodes, 179200 blocks
8960 blocks (5.00%) reserved for the super user
First data block=0
6 block groups
32768 blocks per group, 32768 fragments per group
14944 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840

Writing inode tables: done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 20 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
[jadawin@neptunium]$ mkdir /mnt/uml; sudo mount -o loop debian.iso /mnt/uml


[jadawin@neptunium]$ debootstrap woody /mnt/uml http://ftp.fr.debian.org/debian
I: Retrieving debootstrap.invalid_dists_woody_Release
I: Validating debootstrap.invalid_dists_woody_Release
I: Retrieving debootstrap.invalid_dists_woody_main_binary-i386_Packages
etc ...
I: Base system installed successfully.
```

```

```

Ensuite, il faut créer les périphérique ubd:

```
neptunium:/mnt/uml/dev# mknod ubd0 b 98 0
neptunium:/mnt/uml/dev# ./MAKEDEV ubd
neptunium:/mnt/uml/dev# echo "/dev/ubd0    /    ext3    defaults   0   1" > ../etc/fstab
neptunium:/mnt/uml/dev# echo "proc        /proc   proc    defaults     0    0" >> ../etc/fstab
```

Pour poursuivre, notre installation d'UML, il est très important d'appliquer le correctif **skas** (Separated Kernel Adress Space) sur le noyau de la machine hôte. L'interception des appels systèmes à l'extérieur du noyau de l'UML pose des problèmes de sécurité quant aux corruptions mémoires possibles. La parade du correctif **skas** permet de résoudre des problèmes de sécurité et d'empreinte pour les “honeypots” en rendant le noyau de l'UML complétement inacessible aux processus de l'UML.

Travaillons à présent sur le noyau de l'UML. Il faut télécharger les sources du noyau voulu et le correctif UML (spécifique à Debian dans notre cas). Nous avons utilisé le *noyau 2.6.11-7* et le patch *skas 2.6.11-v8*

```
jadawin@neptunium:~/linux-2.6.11.7$ patch -p1 < ../skas-2.6.11-v8.patch
patching file arch/i386/kernel/entry.S
patching file arch/i386/kernel/ptrace.c
patching file include/asm-i386/thread_info.h
patching file include/linux/ptrace.h
patching file kernel/fork.c
patching file include/linux/mm.h
patching file include/linux/proc_mm.h
patching file mm/Makefile
patching file mm/mmap.c
patching file mm/mprotect.c
patching file mm/proc_mm.c
patching file arch/um/include/skas_ptrace.h
patching file arch/i386/Kconfig
patching file arch/i386/kernel/ldt.c
patching file arch/i386/kernel/sys_i386.c
patching file include/asm-i386/desc.h
patching file include/asm-i386/ptrace.h
patching file include/asm-i386/mmu_context.h
Hunk #2 succeeded at 45 with fuzz 2 (offset 4 lines).
Hunk #3 succeeded at 70 (offset 4 lines).
```

Maintenant, il nous reste à configurer le noyau et à le compiler. Le patch uml rajoute en fait, comme on le voit ci-dessus, une architecture, pour la suite, on doit donc préciser que l'on veut un noyau pour ladite architecture. La commande make peut prendre comme argument la variable ARCH, ainsi pour configurer et compiler notre noyau, on tape les commandes suivantes :

```
jadawin@neptunium:~/linux-2.6.11.7$ make menuconfig ARCH=um
jadawin@neptunium:~/linux-2.6.11.7$ make linux ARCH=um
```

Il est essentiel de mettre les modules suivants : CONFIG_UML_NET = y pour l'activation des fonctionnalités réseau de UML et CONFIG_UML_NET_TUNTAP = y pour communiquer avec l'hôte grâce à un tunnel TUN/TAP.

III – Le 1er test

Aprés la surchauffe du processeur lors de la compilation vous entendrez distinctement un ***ding*.** Cela signifie que c'est cuit.

```
./linux con0=null,fd:3 con1=fd:0,fd:1 umid=debian
Checking PROT_EXEC mmap in /tmp...OK
tracing thread pid = 21107

Debian GNU/Linux 3.1 neptunium-uml tty1

neptunium-uml login:
```

Par défaut, linux prends l'iso *root_fs*, mais on peut lui spécifier *ubd0=/home/jadawin/debian.iso*.

Si vous arrivez au prompt, c'est que votre UML tourne.

IV- Mettre le réseau

Le plus important est de communiquer entre l'hôte et l'UML dans un premier temps. Pour cela:

```
jadawin@neptunium$ vi /etc/network/interfaces
[...]
auto tap1
iface tap1 inet static
	address 192.168.5.254
	netmask 255.255.255.0
	up route add -host 192.168.5.27 dev tap1
	up echo "1" > /proc/sys/net/ipv4/ip_forward
[...]
```

Une fois cette étape de réalisée, il faut activer le tunnel TAP avec tunctl.

```
jadawin@neptunium$ tunctl -u 1000 -t tap1 (1000 étant l'id de la personne qui executera ./linux)
Set 'tap1' persistent and owned by uid 1000
jadawin@neptunium$ chmod 666 /dev/net/tun
jadawin@neptunium$ ifup tap1
```

Lancons l'UML et configurons son réseau.

```
neptunium$ ./linux  con0=null,fd:3 con1=fd:0,fd:1 eth0=tuntap,tap1,192.168.5.254 umid=debian
neptunium-uml login:

neptunium-uml$ ifconfig 192.168.5.27 up
neptunium-uml$ route add default gw 192.168.5.254
neptunium-uml$ ping 192.168.5.254
4 packets transmitted, 4 packets received, 0% packet loss
neptinium-uml$ ping 10.43.43.155
4 packets transmitted, 4 packets received, 0% packet loss
```

A l'heure actuelle, l'UML ne peut pas communiquer avec le reste du monde. Il faut configurer iptables pour faire du routage.

```
neptunium$ sudo iptables -t nat -A POSTROUTING -s 192.168.5.27 -o eth0 -d 0.0.0.0/0 -j SNAT --to
10.43.43.155
neptunium-uml$ ping puce
1 packets transmitted, 1 packets received, 0% packet loss
```