Virtualisation, micro-noyaux

1 Virtualisation

La virtualisation est à la mode en ce moment : typiquement, c'est faire tourner un OS (invité) depuis un autre OS (hôte). On peut distinguer 3 techniques principales : émulation (avec éventuelle assistance matérielle), para-virtualisation, et conteneurs. Ces techniques ont été souvent poussées individuellement par certains outils, même si ensuite la plupart des outils ont commencé à utiliser presque l'ensemble des techniques. On peut essayer de classifier ainsi :

Émulation : bochs, xmame, ... avec éventuellement une recompilation JIT : qemu, valgrind, la calculatrice HP50, ...

Émulation avec assistance matérielle : kvm, virtualbox, DOSBox, ...

Para-virtualisation : Xen, uml

Conteneurs : processus Unix, VServer, OpenVZ, ...

On peut y ajouter les machines virtuelles de haut niveau telles que la Java Virtual Machine, le runtime ocaml, .NET, ...

Le choix de quelle solution utiliser dépend de plusieurs facteurs :

Modification de l'invité : est-ce que l'on peut se permettre de modifier l'invité pour pouvoir le faire tourner dans la solution de virtualisation ?

L'isolation : dans quelle mesure les moyens d'action de l'OS invité doivent être limités ?

La vitesse d'exécution : est-ce qu'on peut se permettre que l'OS invité tourne (beaucoup ?) plus lentement que s'il était directement exécuté par la machine ? Comment le temps processeur est-il partagé ?

Bien sûr, il est très difficile d'exceller dans ces trois facteurs à la fois, il est plus facile de choisir ceux que l'on préfère. On va observer cela sur quelques exemples.

1.1 Processus Unix

L'exemple le plus connu, c'est encore un processus Unix : vos programmes tournent en fait dans une machine virtuelle : un espace d'adressage propre, le programme y fait ce qu'il veut, mais n'a pas un accès direct à toute la machine.

Dans quelle mesure un programme tournant dans un processus est "modifié" par rapport à tourner directement sur une machine, sans OS ?

Dans quelle mesure est-il isolé des autres processus ?

Dans quelle mesure tourne-t-il aussi vite que s'il était tout seul sur la machine ?

1.2 Para-virtualisation, conteneurs

Lancez `~sthibault/uml/one` pour lancer un seul exemplaire de la machine virtuelle que nous avions utilisée pour travailler avec iptables (il faut attendre quelques minutes pour qu'elle se lance). Dedans, lancez la commande `dmesg`. Constatez qu'il n'y est fait nulle part mention de périphériques matériels. UML (User-Mode Linux) est effectivement une version de Linux qui est modifiée pour tourner en tant que simple processus Linux. Les programmes qui tournent dedans n'ont cependant pas été modifiés, et ont l'impression de tourner en tant que root (ce qui est vrai, du point de vue du noyau UML).

Dans l'hôte (donc le Linux "normal"), lancez

```
ps x | grep "\[linux" | wc -l
```

pour compter le nombre de processus "linux". Dans l'invité (donc le Linux virtualisé), lancez

```
ps ax | grep -v "\[" | wc -l
```

pour compter le nombre de vrais processus y tournant. Constatez qu'à un processus près on obtient le même nombre : les processus de l'invité sont en fait visibles au niveau hôte, ce sont des processus comme d'autres !

Dans l'invité, lancez

```
sleep 1h &
```

Relancez les commandes `ps`, constatez qu'il y a effectivement un processus en plus des deux côtés.

Posez-vous de nouveaux les question de modification, d'isolation, et de vitesse. D'autre solutions (telles que chroot, OpenVZ, Vservers) fournissent d'autres niveaux d'isolation.

1.3 Émulation

Lancez
```
qemu -net nic,model=rtl8139 -net user \
                -snapshot -hda ~sthibault/debian-hurd.img
```
Note : si à un moment vous n'avez plus de souris, remarquez le titre de la fenêtre de qemu : il faut appuyer sur ctrl+alt pour sortir du mode "grab".

Dans l'hôte, observez dans `ps x` qu'il n'y a ici qu'un seul processus qemu, bien que dans l'invité il y ait bien de nombreux processus. Il s'agit ici vraiment d'émulation : `qemu` simule complètement un processeur virtuel. Vous aurez également constaté que pendant le boot le noyau mentionne toute une série de matériels : l'OS invité "voit" du matériel virtuel, et ne se rend pas compte qu'il ne tourne pas sur une vraie machine, il n'a pas été modifié du tout.

Une grosse différence entre `bochs` et `qemu` (qui sont tous deux des émulateurs) est que `qemu` ne fait pas qu'interpréter les instructions de processeur une par une : il prend des morceaux entier du code de l'invité, et les recompile à la volée (JIT, Just In Time) en des versions "simulées", qui tournent bien plus vite que la version interprétée, mais tout de même pas aussi vite qu'une exécution directe sur le vrai processeur.

Posez-vous de nouveau le trio de questions, remarquez les options qui ont été passées à `qemu`.

1.4 Émulation avec assistance matérielle

Les processeurs récents fournissent une aide pour l'émulation : plutôt que ce soit `qemu` qui simule un processeur (et lui fait croire ce qu'il veut), c'est le vrai processeur qui fait tourner l'OS invité, et ce n'est que lorsque l'OS invité essaie d'accéder au matériel que le processeur redonne la main à `qemu`, pour simuler le matériel virtuel. Il s'agit de l'option `kvm` de `qemu`. Le code de l'OS invité tourne ainsi à pleine vitesse sur le processeur, sauf quand il commence à toucher au matériel.

En fait, ce que nous avons appelé jusqu'ici "OS hôte", est en réalité un OS invité : lorsqu'au démarrage de votre poste vous choisissez entre Windows ou Linux, cela démarre l'outil virtualbox, qui utilise cette assistance matérielle pour qu'on ait une impression de vitesse quasiment aussi grande qu'en éxécution directe sur le matériel. Regardez le fichier `/sys/block/sda/device/model` : le disque que vous utilisez est en fait virtuel !

Appuyez sur ctrl-alt-f2, vous vous retrouvez avec une bannière de login particulière : le nom de la machine qui y est indiqué est différent de celui que vous avez dans votre bureau : il s'agit du **vrai** OS hôte s'exécutant directement sur la machine. On ne peut malheureusement pas s'y connecter ; on constaterait avec `ps` qu'on ne voit pas les processus lancés dans votre session graphique, seulement quelques processus virtualbox. Utilisez alt-f3 pour revenir à votre bureau.

1.5 Et la sécurité ?

On a vu des failles au sein d'un processus. Est-ce que cela permet d'attaquer d'autres processus ? Est-ce que cela permet d'attaquer des processus d'autres utilisateurs ?

Il existe également des failles au niveau du noyau : un bête programme lancé en tant qu'utilisateur simple peut alors contrôler le noyau. Dans le cas d'UML, cela permet-il de contrôler le noyau de l'OS hôte ? Et dans le cas de `chroot` (où il n'y a pas de noyau intermédiaire, c'est simplement le système de fichier accessible qui change) ?

Lorsque l'on met en place des services (web, mail, ldap, ...), on peut utiliser des machines physiques différentes pour chaque service, pour bien les isoler. Mais cela consomme de la place et de l'énergie. Selon le niveau d'isolation et de sécurité que l'on veut conserver, on pourra utiliser l'une ou l'autre des solutions de virtualisation disponibles. Remarquez comment la plupart des solutions permettent de déléguer la gestion des services : on peut donner l'accès `root` à quelqu'un d'autre, tant que c'est dans une machine virtuelle !

L'assistance matérielle pour l'émulation est souvent désactivée par défaut sur les machines du commerce. Une raison est simplement l'existence des blue pills : une fois qu'un attaquant a pris le contrôle d'un système, rien ne l'empêche de faire tourner l'ensemble de ce système dans une machine virtuelle, et lui faire croire ainsi ce qu'il veut, restant ainsi caché à volonté du côté hôte, et pouvant agir sans que ni l'utilisateur ni ses antivirus n'aient de moyen de constater son existence (le système ne s'exécute même pas plus lentement !)

2 Machines Virtuelles Java (JVM)

Les JVM utilisent en fait le même principe : simplement, le programme invité est écrit en Java, et il tourne dans une JVM, qu'on appelle souvent *sandbox* (bac à sable, rappelant ainsi les barrières autour du bac, empêchant un enfant turbulent d'en faire sortir le sable). La question reste alors : quelle isolation veut-on fournir à la machine virtuelle Java ? Posez-vous la question dans le cas d'une applet java ou un plugin flash tournant au sein d'un navigateur web.

3 Système à micro-noyau

Reprenez la machine virtuelle `debian-hurd.img`.

Lancez `ps -fe | less` pour voir l'ensemble des processus. Remarquez la présence notamment des processus `ext2fs`, `netdde` et `pfinet` . Il s'agit ici d'un système à micro-noyau : le noyau est réduit à la gestion de processus et mémoire, et l'on fait tourner dans de simples processus tout le reste : le processus `ext2fs` interprète le système de fichier `/`, le processus `netdde` pilote la carte réseau, et le processus `pfinet` implémente la pile TCP/IP.

Déplacez-vous dans le répertoire `/ftp:/ftp.fr.debian.org/` et constatez que les commandes shell usuelles y fonctionnent, essayez éventuellement avec d'autres sites `ftp`. Regardez `ps -fe` de nouveau, constatez qu'un processus `ftpfs` y est apparu. Pourquoi tourne-t-il en tant que `nobody` plutôt que `root` ?

Lancez `showtrans /ftp`: cela veut dire qu'il s'agit d'un répertoire rendu spécial, où lorsqu'on ouvre un répertoire, cela lance en fait un processus `ftpfs`, qui fournit le contenu du répertoire.

Logguez-vous en tant qu'utilisateur `demo` . Lancez

```
settrans -c montest /hurd/ftpfs / ftp.kernel.org
```

Regardez dans le répertoire `montest` . Constatez dans `ps -fe` qu'un processus `ftpfs` est apparu sous l'identité `demo`.

Lancez

```
dd < /dev/zero > image bs=1M count=10
```

pour créer un fichier vide, puis formatez-le :

```
/sbin/mke2fs image
```

Et enfin montez-le :

```
settrans -c montest2 /hurd/ext2fs image
```

Rentrez dedans, tapez `df .` pour constater la place disponible, et observez la présence d'un nouvel `ext2fs` dans `ps -fe`, sous l'identité `demo`. En sortant la fonction de fourniture de système de fichier du noyau, on a permis ainsi aux utilisateurs d'avoir leurs propres systèmes de fichier, en toute sécurité puisqu'ils tournent sous leur identité.