Gestion de parc *Windows* depuis *Unix*

Pascal Cabaud
UFR EILA, Université Paris Diderot
Case 7002, 75205 Paris cedex 13
Laurent Joly
CRL, Université Paris Diderot
Case 7016, 75205 Paris cedex 13

Résumé

Le gestionnaire de parc Windows *est systématiquement confronté aux problématiques suivantes :*

- *le matériel hétérogène,*
- *le déploiement de nouveaux systèmes,*
- *l'incompatibilité entre deux logiciels sur un même système,*
- *la lutte anti-virale.*

La virtualisation est plus qu'une solution d'informatique verte : elle facilite grandement la vie du gestionnaire de parc. En effet, une machine virtuelle n'est rien d'autre qu'un fichier de configuration (processeur, mémoire, disque, interface réseau, etc.) et un « support de stockage » qui peut se résumer à un gros fichier. Dès lors, installer un nouveau système se réduit à déployer ces données sur le parc. Elle affranchit des problématiques de parc hétérogènes et d'incompatibilité entre logiciels. Elle permet aussi d'améliorer la lutte anti-virale.

Aujourd'hui, dans un environnement Unix *on peut facilement déployer et gérer des systèmes* Windows*, par des commandes facilement automatisables, en n'utilisant que des logiciels libres ou gratuits.*

Mots clefs

ClamAV, gestion de parc, *GNU/Linux*, *Samba*, virtualisation, *Windows*.

1 Introduction

Notre parc étudiant est utilisé par deux profils d'utilisateurs distincts : les étudiants de master et les autres. Ce parc représente au total 230 postes dont certains ont plus de cinq ans. Certains logiciels utilisés sont incompatibles entre eux et ne cohabitent pas sur un même système *Windows*. Des enseignants demandent un environnement *Unix* depuis plusieurs années déjà.

Plutôt que de continuer à gérer un amorçage multiple, à la rentrée 2007 nous avons commencé à ne plus installer les postes que sous *GNU/Linux*. Les systèmes *Windows* sont lancés sur une couche de virtualisation. Après un an avec *VMware Player* (*cf*. [1], gratuit dans notre cadre Éducation-Recherche), nous utilisons maintenant, tant au CRL qu'à l'UFR EILA, *Sun xVM VirtualBox* (*cf*. [2], gratuit aussi dans notre environnement). Ce dernier a pour principal avantage de se gérer intégralement en ligne de commandes (et donc à distance et/ou de manière automatisable) même si une interface graphique est fournie.

2 La virtualisation

Le pré-requis pour toute la suite est d'avoir un parc sous *GNU/Linux* et un démon *OpenSSH* sur chacun des postes. Pour commencer, installer *Sun xVM VirtualBox*, créer un disque virtuel puis créer la machine virtuelle et la configurer. À la fin de cette dernière opération, on installe le système *Windows* de la même manière que sur un poste classique, sur le disque virtuel.

Le disque virtuel est un gros fichier. La machine virtuelle se matérialise par un fichier XML contenant diverses informations, à commencer par la quantité de RAM, l'emplacement et le type des disques virtuels, les interfaces réseau, les partages, les périphériques audio, vidéo, *etc*. Ce fichier XML est créé et modifié à l'aide de la commande `VBoxManage`. Nous copions ensuite le disque virtuel sur les postes par `ftp`, `scp` ou `rsync`. Nous recréons la machine virtuelle sur chaque poste par un script (commandes identiques à celle de la création initiale sur le master) ; on pourrait éventuellement la

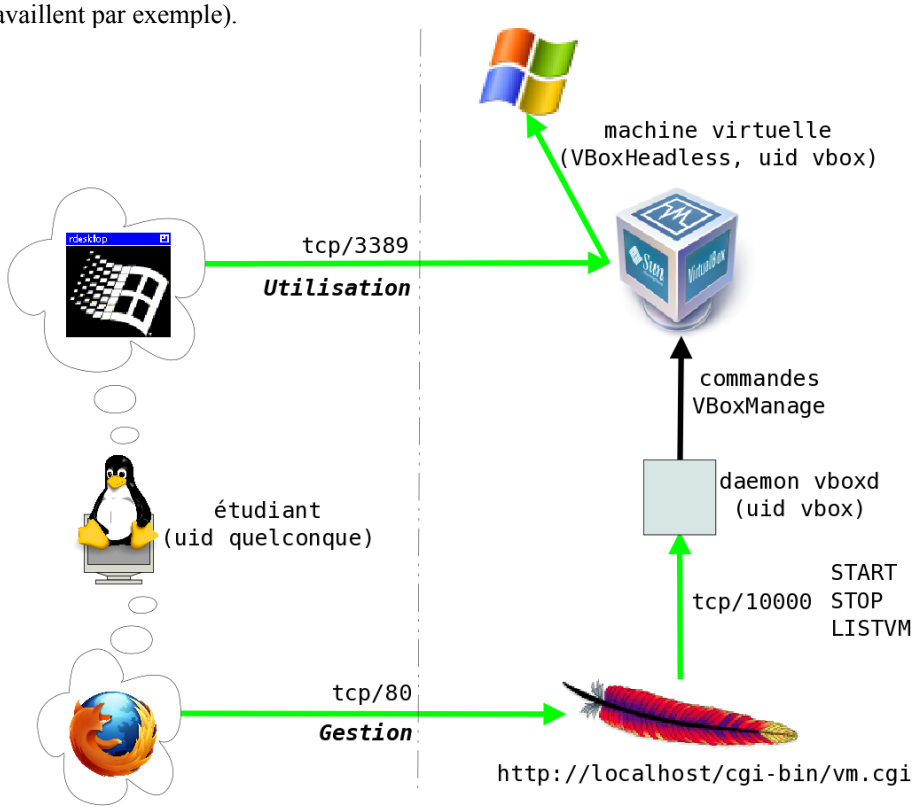
copier comme le disque virtuel. La gestion peut intégralement se faire en ligne de commandes et s'automatise donc facilement.

Pour automatiser la gestion des postes *GNU/Linux* (lancement des commandes ci-dessus sur N postes), nous utilisons *ClusterIt* (*cf.* [3]) qui envoie une même commande à un ensemble de machines par SSH (utiliser une authentification par clefs).

VirtualBox permet de lancer des machines virtuelles « sans écran » *i. e.* sans interface graphique. Les systèmes invités sont alors accessibles localement par RDP (`localhost:3389`). L'étudiant se connecte à la machine virtuelle de son poste avec *rdesktop*, un client RDP (*cf.* [4]) : l'avantage majeur est que l'utilisateur n'a plus aucune interaction directe avec le logiciel de virtualisation.

Cette installation sécurise l'accès aux ressources. En effet, *VirtualBox* utilise un module noyau pour gérer l'accès au matériel. Sans ce verrouillage, un utilisateur malveillant peut lancer une machine virtuelle avec sa propre identité et potentiellement venir avec sa propre machine virtuelle, système sur lequel il est *root* ; il peut alors par exemple injecter des trames Ethernet à l'envi. Au contraire, notre solution nous assure que seuls les systèmes que nous maîtrisons sont exécutés.

Un démon `vboxd` développé par nos soins écoute sur `localhost:10000` et tourne sous l'identité `vbox`. Il accepte quelques commandes dont en particulier `START vmname` et `STOP vmname`. Les machines sont aussi lancées sous l'identité d'un compte `vbox` (par `vboxd`). Enfin, au démarrage du système *GNU/Linux*, un script `/etc/init.d/default-vm` lit dans un fichier le nom de la machine virtuelle à lancer par défaut et se connecte à `vboxd` pour démarrer ce système. Certains cours nécessitent un serveur *Apache* sur chaque poste. Pour rendre les utilisateurs autonomes, nous l'avons réutilisé en développant un petit CGI (qui communique avec notre démon `vboxd`) pour permettre aux étudiants uniquement d'arrêter une machine puis d'en lancer une autre. Notre installation est prévue pour qu'une seule machine virtuelle fonctionne à la fois mais nous pouvons en lancer une seconde (pour la configurer pendant que les étudiants travaillent par exemple).



Figue 1 - *Communications sur le poste étudiant*

VirtualBox offre de figer les disques virtuels par un mécanisme de clichés (*snapshots*). Ces derniers sont de deux types : classiques et différentiels. Nous n'avons pas testé le déploiement de clichés différentiels mais cela permettrait d'économiser la bande passante lors de l'ajout d'un logiciel. En revanche, cela interdirait les réglages post-déploiement (changement de nom de machine, jonction à un domaine *Windows*…). Ces clichés rendent inutile *Windows SteadyState* (*cf.* [5]) qui ralentit considérablement les ouvertures et fermetures de session.

VirtualBox offre essentiellement deux modes de connexion au réseau : en mode *bridge* (pont) et en mode NAT. Nous utilisons essentiellement le *bridge* mais le NAT s'avère parfois utile avec certains logiciels qui testent l'adresse MAC de l'interface réseau. Dans les deux cas, un répertoire d'échanges entre l'hôte et le système virtualisé est possible avec *VirtualBox* ; *Samba* est une autre alternative pour partager ce répertoire.

3 Anti-virus

Nous voudrions maintenant supprimer des systèmes *Windows* notre actuel logiciel anti-virus. Nous utilisons encore le logiciel fourni dans le cadre de l'accord ministériel ainsi que sa console d'administration mais nous le trouvons trop gourmand en ressources (CPU et mémoire).

Les codes malicieux ont deux vecteurs : réseau et supports amovibles. Pour le réseau, notre *proxy-cache* bloque `.exe`, `.com`, `.dll`, *etc*. et depuis une dizaine d'années, ceci s'avère redoutablement efficace. Pourtant, cela ne protège pas de tout, loin s'en faut (contrôles *ActiveX* par exemple), il nous reste donc, au niveau du *proxy-cache*, à configurer un « redirecteur » *Squid* vers un anti-virus. Pour les supports amovibles, nous utilisons *ClamAV* (anti-virus libre, *cf.* [7]) avec *ClamFS* (pseudo-système de fichiers *FUSE*, *cf.* [8]). Les systèmes de fichiers sont montés automatiquement dans `/media/foo` ; `/media` est par ailleurs monté en *ClamFS* (`/clamfs/media`). Pour bloquer l'accès à certains fichiers, comme avec le *proxy-cache*, il suffit de partager ce répertoire par SMB avec *Samba* en utilisant la directive `veto files`. Évidemment, un fichier dont l'accès est interdit par ce mécanisme est lisible dès modification de son extension. Dans cette configuration, seuls les fichiers sur les supports amovibles sont analysés sur le poste et non plus l'ensemble du système d'exploitation.

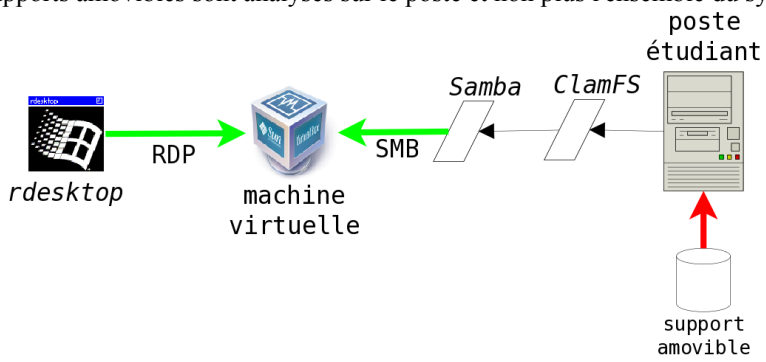


Figure 2 - *Sécurisation de l'accès aux supports amovibles*

Nous testons actuellement le nouvel anti-virus proposé par Microsoft (*Security Essentials*, *cf.* [6]). D'après la documentation, l'une des principales contraintes lors de son développement était qu'il ne monopolise jamais plus de 50 % du CPU, ce qui se vérifie dans la pratique — hormis lors d'une analyse complète. Nous envisageons aussi de monter les disques virtuels sur le système hôte pour une analyse régulière avec *ClamAV*.

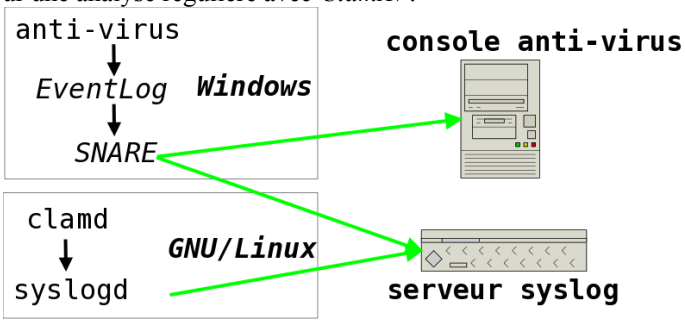


Figure 3 - *Centralisation des alertes virales*

Reste la centralisation des alertes virales, fonctionnalité majeure. Nous utilisions la console de centralisation de l'anti-virus fourni à nos établissements mais aussi *SNARE* (*cf.* [9] ; voir aussi *NTsyslog*, *cf.* [10]). Ce dernier envoie les messages collectés par l'*Event Log* à notre serveur `syslog` ; comme l'anti-virus génère des messages, ces derniers se retrouvent dans nos journaux `syslog`. Le démon `clamd` journalise aussi *via* `syslog`. En consultant nos journaux, nous pouvons ainsi surveiller l'activité de nos anti-virus. Rappelons au passage que les logiciels de surveillance des journaux sont nombreux : citons par exemple *Logsurfer* ou *Swatch* pour la gestion d'alertes et *PHP-Syslog-ng* pour l'interface Web de consultation.

4 Piloter *Windows* à distance

Le pilotage d'un système à distance se résume à copier des fichiers et exécuter quelques commandes. Il s'avère aussi nécessaire d'accéder à distance à la base de registre.

La copie de fichiers est simple : avec *Samba*, vient la commande `smbclient` utilisable de manière interactive ou non. Pourvu que les partages administratifs (`C$`) soient activés, nous avons accès à distance à l'ensemble des systèmes de fichiers. La commande `smbcacls` permet de régler les ACL sur les fichiers et les répertoires à distance. Enfin, la commande `rpcclient` offre quant à elle de nombreuses possibilités, à commencer par l'arrêt d'un système à distance. L'usage de ces commandes est classique pour les administrateurs familiers de *Samba* et il est bien documenté.

Pour lancer des commandes à distance — sans rien installer sur les systèmes *Windows* —, nous utilisons *Samba-TNG* (*cf.* [14]), un projet dérivé de *Samba*. Cette version intègre les commandes `cmdat` et `regedit`[1]. En copiant un script sur un système, on peut ensuite le lancer (dans la minute qui suit) *via* `cmdat`. Ce dernier l'ajoute à la file (avec `at.exe`) ; on dispose donc d'un ersatz de `rexec`, presque immédiat et sans interaction (ce qui complique le traitement des résultats ou nécessite des tests après l'exécution du script).

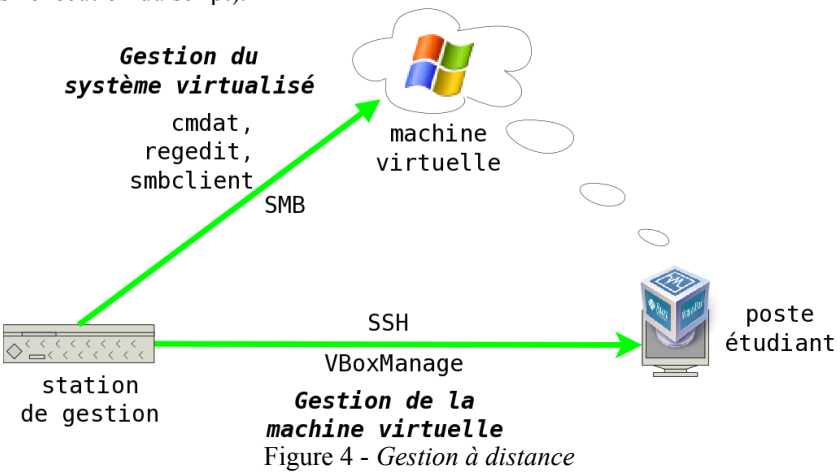


Figure 4 - *Gestion à distance*

Revenons un instant à nos machines virtuelles. Après déploiement d'une nouvelle machine, une fois l'adresse MAC de cette dernière réglée sur chacun des postes, il faut :

- lancer le nouveau système sur les postes ;
- changer le nom d'hôte (avec `regedit`, à ce jour nous n'avons pas trouvé d'autre solution que d'éditer une demi-douzaine de clefs du registre) sur chaque système virtuel et redémarrer (avec `rpcclient`) ;
- rejoindre le domaine sur chaque machine virtuelle (script lancé via `cmdat` et appelant `netdom.exe`, commande des *Support Tools*, *cf.* [15]) et redémarrer.

Si les commandes du projet *Samba-TNG* venaient à ne plus être utilisables, on pourrait activer le serveur Telnet intégré à *Windows* ou remplacer par un serveur SSH pour *Windows* (*cf.* [16]) ou encore utiliser d'autres solutions comme *BackOrifice* (pour lequel un client *Unix* existe). Ces solutions alternatives nécessiteraient tout autant de précautions quant au filtrage des accès vers les systèmes virtualisés.

5 Gestion

Nous avions déjà une interface Web pour lancer les postes d'une salle par *Wake On LAN*, accéder à l'inventaire ou aux sauvegardes. Nous avons ajouté :

- le lancement et l'arrêt d'une machine virtuelle dans une salle ou seulement sur un poste ;
- l'inventaire des machines virtuelles avec leurs réglages (mémoire, disque virtuel, adresse MAC, *etc.*) ;

[1] Le manque de documentation sur ces commandes laisse planer un doute sur leur pérennité mais c'est celle du projet *Samba-TNG* lui-même qui paraît compromise…

VirtualBox inclut une interface SOAP mais nous avons développé un petit démon (en Perl avec `Net::Server`, *cf.* [20]) ; il écoute sur un port TCP et répond à quelques requêtes comme `START vmname`, `HRDRST vmname`, `LISTVM`, *etc*. On peut s'y connecter en local, avec une interface pour l'utilisateur, ou à distance, depuis notre console. Une nouvelle console Web, *VirtualBox Web Console* a vu le jour il y a peu (*cf.* [21]) et semble très prometteuse.

6 Conclusion

Les possibilités offertes sont nombreuses. Dans notre environnement, nous avons été très satisfaits de pouvoir changer nos systèmes très rapidement ; nous disposons de nos outils et les utilisateurs ont des systèmes répondant à leurs besoins. Enfin, les ressources sont mieux utilisées (anti-virus, *snapshots*), ce qui était important sur nos vieux PC (1.5 GHz, 1 Go de mémoire pour certains). Sur des machines « modernes », les instructions de virtualisation dans les processeurs *x86*, leurs cœurs multiples et une importante quantité de RAM rendent la solution transparente. Sur des machines plus anciennes, *Windows XP* se contentant de 512 Go de mémoire, avec un bureau léger (*Xfce*), si le processeur s'y prête, cela reste vrai.

La virtualisation offre un avantage, majeur dans le cas d'un parc hétérogène : le matériel vu par le système invité (*Windows* nous concernant) est identique quel que soit le matériel physique réel. Autre avantage : on a alors le choix entre « NATer » le système invité ou au contraire le « bridger ». La première option permet par exemple de ne pas changer l'adresse MAC après déploiement mais impose alors d'installer *Samba-TNG* sur les postes.

Rares sont les applications supportant mal la virtualisation. Le seul exemple que nous ayons rencontré est une application de reconnaissance vocale pour l'apprentissage des langues : la latence induite rend la reconnaissance inutilisable. Pour le reste, les performances vues de l'utilisateur sont au pire comparables, au mieux identiques à un système natif, ceci dans notre usage bureautique.

Pour aller plus loin, on pourrait imaginer déporter les disques virtuels par iSCSI ou, inversement, héberger les machines virtuelles sur un serveur distant et y accéder par RDP…

Bibliographie

[1] http://www.vmware.com

[2] http://www.virtualbox.org

[3] http://www.garbled.net/clusterit.html

[4] http://www.rdesktop.org

[5] http://www.microsoft.com/windows/products/winfamily/sharedaccess/

[6] http://www.microsoft.com/security_essentials/

[7] http://www.clamav.net

[8] http://clamfs.sourceforge.net

[9] http://www.intersectalliance.com/projects/BackLogNT/

[10] http://ntsyslog.sourceforge.net

[11] http://www.crypt.gen.nz/logsurfer/

[12] http://swatch.sourceforge.net

[13] http://code.google.com/p/php-syslog-ng/

[14] http://wiki.samba-tng.org

[15] http://support.microsoft.com/kb/838079

[16] http://www.freesshd.com

[17] http://www.microsoft.com/technet/sysinternals/ProcessesAndThreads/PsExec.mspx

[18] http://rce.sourceforge.net

[19] http://www.bo2k.com

[20] http://search.cpan.org/search?query=net::server

[21] http://code.google.com/p/vboxweb/