hepia
Haute école du paysage, d'ingénierie et d'architecture de Genève

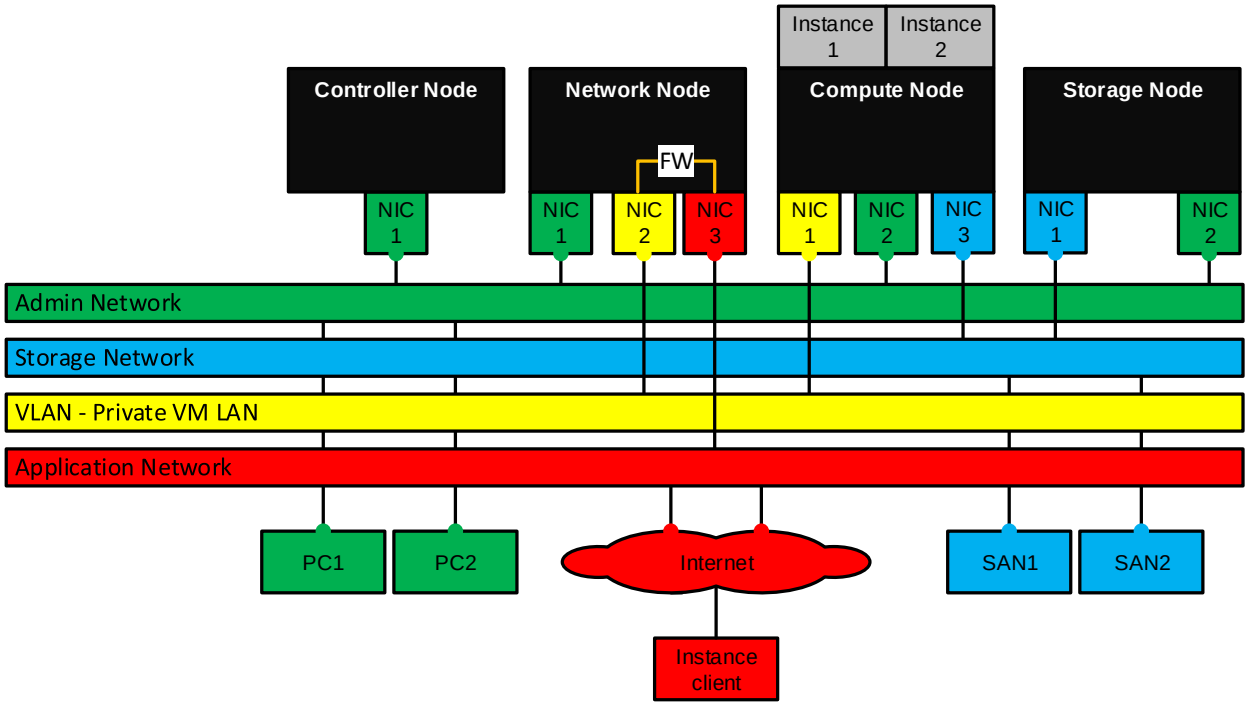
Hes·so
Haute Ecole Spécialisée de Suisse occidentale

SELinux – Analyse sécuritaire d'OpenStack Mai 2014

Table des matières

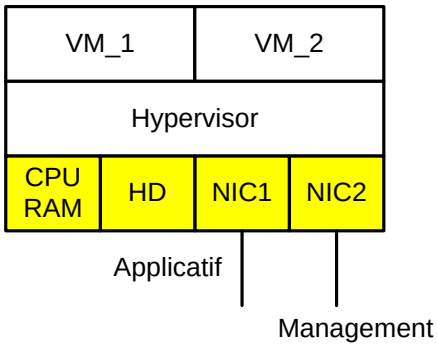
0 Objectifs et cadre de l'étude

La figure ci-dessous extraite d'une analyse de sécurité antérieure servira de cadre :



Principaux objectifs :

- Analyse des risques
- Valider le cloisonnement entre VMs
- Quels sont les mécanismes de sécurité garantissant la séparation des réseaux (applicatif – administration – storage) ?
- Conserver une démarche classique basée sur la décomposition suivante qui présente les 5 principaux domaines d'un système virtualisé



Réseau
séparation, défense périmétrique

Système
comptes actifs, services, logs

Management
accès distant, backup, …

Virtualisation
cloisonnement des VMs, …

Applicatif
flux, …

Nous étudions le cas d'un Cloud privé constitué d'une infrastructure déployée au profit d'une seule entreprise.

Nous considérons que ce Cloud est administré en interne, donc il n'y a aucun accès possible au réseau de management pour les personnes externes à cette entreprise.

Les APIs, les nœuds compute et les autres ressources seront disponibles uniquement à une partie des systèmes et des collaborateurs IT installés dans l'entreprise qui les gèrent ; une autre partie des collaborateurs IT utilisent le service IaaS offert par ce Cloud pour mettre en place des applications serveurs ou autres.

Certaines applications serveurs sont accédées par des clients de l'entreprise à travers Internet.

1 OpenStack & RDO

1.1 RDO[1]

RDO est une communauté de personnes qui utilise et déploie OpenStack sur Red Hat (RDO). Sur le site de la RDO, nous pouvons trouver de la documentation qui aide à prendre rapidement en main OpenStack. Le déploiement d'OpenStack était réputé casse-tête et prenait plus d'une semaine de travail pour le mettre en place. RDO promet un déploiement rapide d'OpenStack, en 15 minutes. Packstack est le logiciel chargé de configurer OpenStack automatiquement, à l'aide de modules puppet, sans intervention manuelle pour son installation. De plus, nous pouvons trouver une aide considérable sur les questions les plus fréquemment postée. Une page[2] est dédiée pour les résolutions de problèmes en lien avec SELinux. Voici un résumé du contenu de la page :

- Localisation des logs SELinux dans le système
- Modification mode SELinux
- Lien vers la page Fedora lié à la résolution de problème en général lié à SELinux
- Information pour activer SELinux et réinitialiser les labels des ressources

Le repository (dépôt de paquetages) RDO doit être installé sur tous les nœuds. Entre autresle paquetage openstack-selinux est téléchargé depuis le repository RDO.

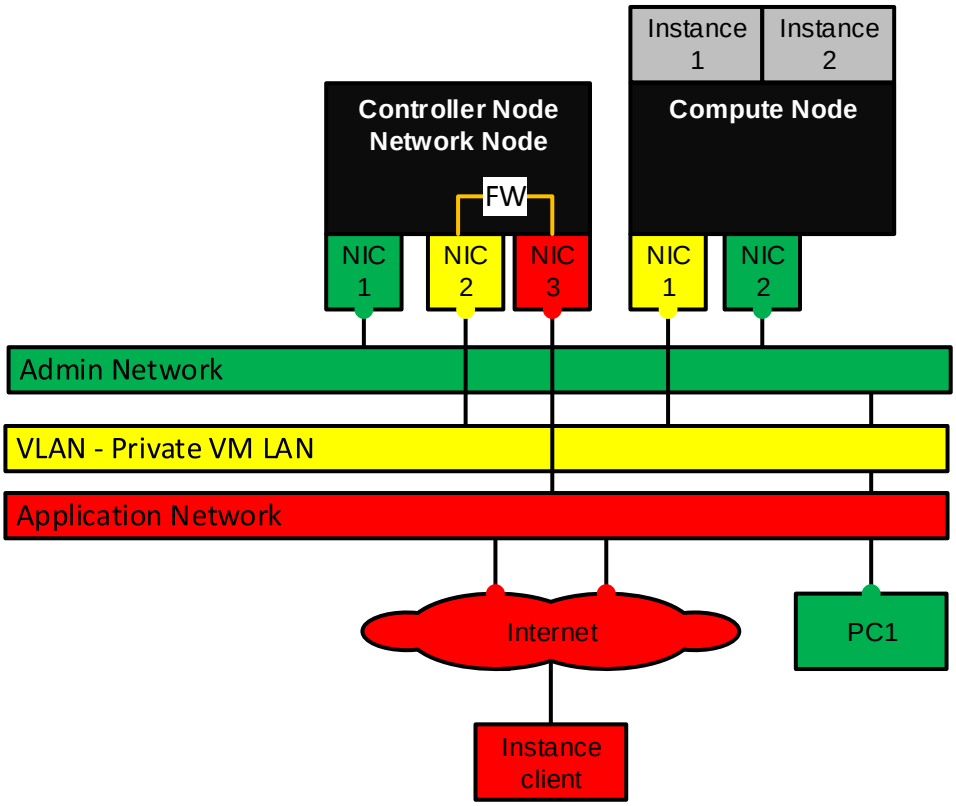
1.2 Choix technologique

Dans ce projet j'utilise Fedora19. Ce choix a été motivé d'abord par la disponibilité de nouveaux outils d'analyse des règles SELinux, voir le blog de Daniel Walsh[3] pour une liste des nouveautés.

Le faite d'utiliser RDO de Red Hat a aussi fortement motivé ce choix. Red Hat est une entreprise très impliquée dans le processus de développement de la distribution Fedora car elle leurs sert de version test à grande échelle pour les mises à jours et l'intégration de nouveaux logiciels dans RHEL.

J'ai mis en place une infrastructure OpenStack composé de deux nœuds.

[1] http://openstack.redhat.com/Main_Page
[2] http://openstack.redhat.com/SELinux_issues
[3] http://danwalsh.livejournal.com/62711.html



Le premier nœud sert de **contrôleur** et de **gestionnaire réseau** avec les services suivant :

- Dashboard Horizon
- Base de données MySQL
- Queues Qpid
- Keystone
- Glance
- Nova
- Cinder
- vSwitch
- Quantum Agent
- DHCP Server
- Virtual Routing

Le deuxième nœud sert d'hyperviseur (compute node) avec les services essentiels suivant :

- Libvirt
- Nova-compute
- QEMU-KVM

La séparation physique des logs et l'isolation du service nova-compute permet de facilité l'analyse des logs et réduit la pile logicielle pour ne garder que le minimum requis.

2 Analyse des risques

Un Cloud qu'il soit privé ou public est exposé à une certaine forme d'attaque et de menace. Le Cloud computing fait intervenir **quatre domaines de sécurité** avec des niveaux de confiances variables.

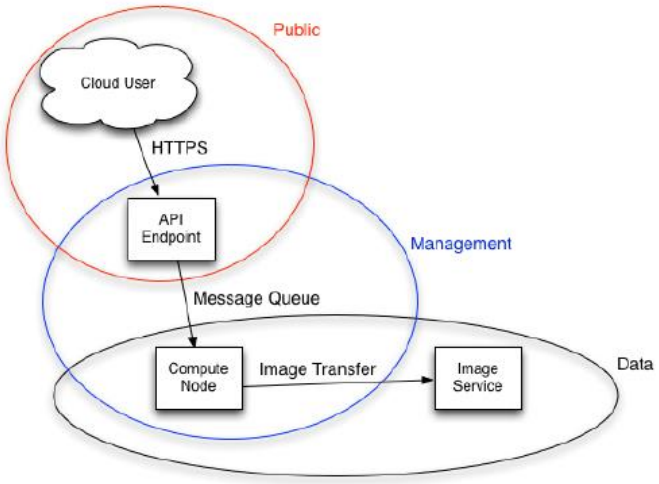
Le tableau ci-dessous est une synthèse du « Security Guide » d'OpenStack

Domaine	Description	Niveau de confiance	
		Cloud privé	**Cloud public**
Public	Représente le trafic provenant d'Internet dans son ensemble ou tout trafic provenant d'un réseau sur lesquelles nous n'avons aucun pouvoir.	Aucune confiance	Aucune confiance
Guest	Représente le trafic inter-VM	Confiance faible à moyenne selon les mécanismes de contrôle mis en place dans les VMs pour empêcher les attaques	Aucune confiance
Management	Représente les appels aux API pour la gestion et au contrôle du Cloud. Des données sensibles sont souvent transportées dans les réseaux qui appartiennent à ce domaine tel que des noms d'utilisateurs, des mots de passes, des configurations, etc.	Confiance élevée	Confiance élevée
Data	Représente le trafic des informations relatives aux services de stockage et les données applicatives.	Dépend du contexte d'utilisation	Dépend du contexte d'utilisation

Du tableau ci-dessus, nous pouvons nous faire une idée des domaines de sécurités auxquels il faut prêter le plus d'attention dans l'objectif de le sécuriser.

Les transitions entre les domaines sont les points les plus sensibles : les attaques proviennent surtout des domaines avec un degré de confiance bas et tentent de transiter vers des domaines avec un niveau de confiance haut pour obtenir plus de privilège.

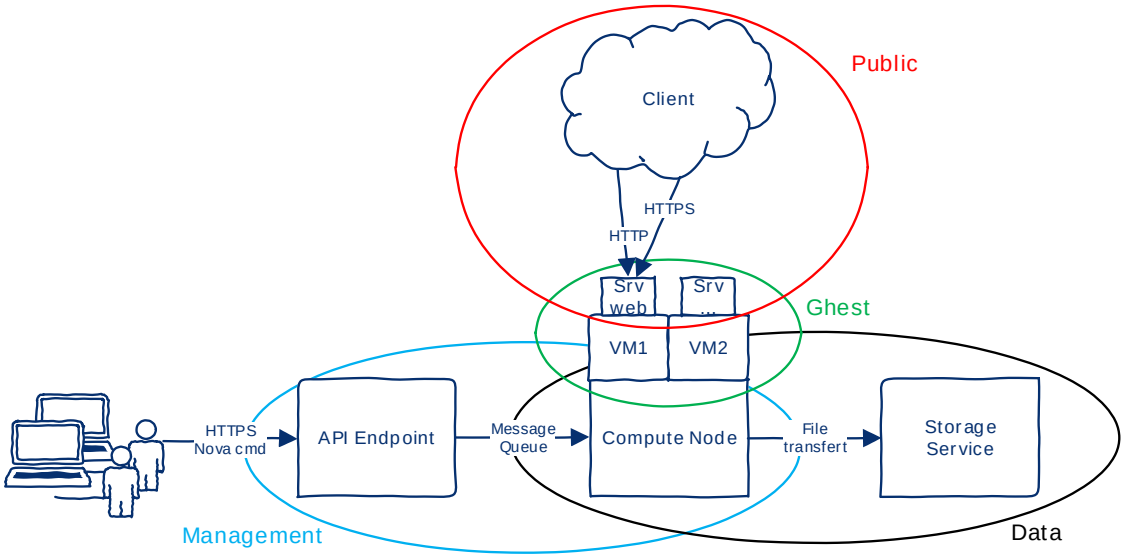
Dans la figure ci-dessous, reprise du Security Guide d'OpenStack (section 4), nous pouvons observer deux points de transition inter-domaine sensibles.



A partir de ce point, nous pouvons déjà nous orienter vers la sécurisation du domaine de sécurité à qui nous faisons le moins confiance, c'est-à-dire le domaine Public.

En effet, comme nous étudions le cas d'un Cloud privé, notre réseau de management et notre réseau stockage peuvent être considérés comme isolés.

Dans la figure ci-dessous, nous allons détailler le nœud compute pour représenter le domaine de sécurité Public. Nous faisons abstraction du nœud network pour éviter de surcharger la figure.



Une étude, datant de mai 2013 de l'université de Princeton, présente une revue de toutes les vulnérabilités répertoriées dans les bases de données CVE pour les hyperviseurs KVM et XEN.

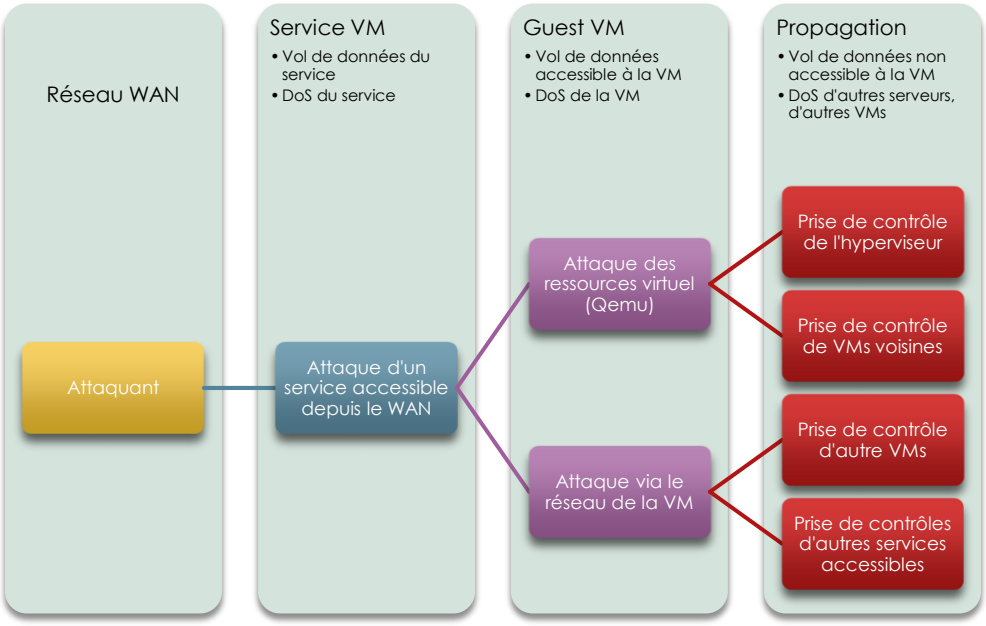
L'objectif de cette étude est d'extraire des cheminements d'attaques potentiels, de comprendre les attaques existantes et d'aider dans la prise de décision sur les positionnements des défenses. Le dernier point nous intéresse tout particulièrement dans cette étude.

Trigger Source	KVM
Network	2 (5.3%)
Guest VM User-Space	13 (34.2%)
Guest VM Kernel-Space	12 (31.6%)
Dom0/Host OS	11 (28.9%)
Hypervisor	0 (0.0%)
Total	**38**

La synthèse des attaques effectuées dans ce rapport est alarmante : la plupart des attaques existantes, utilise comme source un code dans l'espace utilisateur (User-Space) (Ring 3) d'une VM invitée.

Ce qui est alarmant, c'est qu'une VM invitée non privilégiée a assez de privilège pour représenter une menace à l'hyperviseur sous-jacent. Pour donner un chiffre, 65.8% des vulnérabilités connus sous KVM sont initié depuis une VM invitée. **La sécurisation des VMs est un point non négligeable dans la démarche de sécurisation d'OpenStack**.

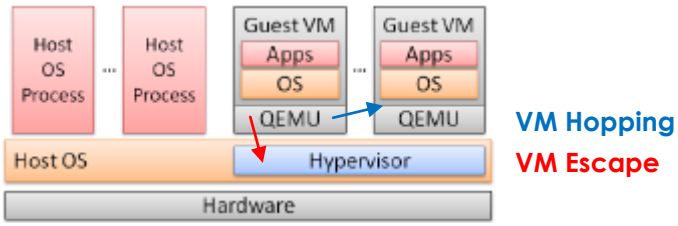
Le tableau ci-dessus extrait du rapport de l'université de Princeton permet de voir les éléments à sécuriser. Le Host OS est aussi un élément critique à protégé surtout si des services hébergés sur celui-ci sont exposés vers un réseau étranger/externe.



3 Sécurisation du nœud compute

La démarche pour sécuriser un service reste standard ; les best-practices pour configurer sont toujours de mise. Par exemple, le service doit être configuré pour n'offrir que ce qui est nécessaire pour l'entreprise et il doit être régulièrement patché avec les mises à jours sécuritaire dans les plus brefs délais, etc.

Concernant la sécurisation d'une VM : comment empêcher ou rendre très difficile la propagation d'une attaque vers l'hyperviseur (VM Escape) ou vers des VMs voisines (VM Hopping) ?



Dans l'architecture KVM, **chaque VM est exécutée dans un processus Qemu séparé dans l'espace mémoire utilisateur**. Le processus Qemu émule un certain nombre de ressources pour faire fonctionner la VM.

Qemu est une cible importante visée par l'attaquant afin de réaliser un VM Escape ou un VM Hopping. Il est essentiel que l'exécutable soit protégé par des mécanismes qui rendent le plus difficile possible la corruption de la mémoire. Ces protections sont souvent à activer à la compilation de l'exécutable Qemu.

3.1 Mécanisme de protection de la mémoire

Les mécanismes les plus utilisés sont ASLR, Stack Canaries, NX et PIE. Pour plus de détails sur ces techniques, je conseille de jeter un coup d'œil à l'annexe C de l'excellent livre intitulé « A Bug Hunters Diary[4] » ou à cet article wikipedia[5].

L'utilitaire checksec.sh[6] développé par Tobias Klein, auteur du livre « A Bug Hunters Diary », permet de lister ces mécanismes :

```
[root@controller1 Téléchargements]# ./checksec.sh --file `which qemu-system-x86_64`
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      FILE
Full RELRO      Canary found      NX enabled    PIE enabled     No RPATH   No RUNPATH   /usr/bin/qemu-system-x86_64
```

Conclusion, **Red-Hat a bien activé les mécanismes de protection les plus utilisés à la compilation**.

Le mécanisme de protection sVirt (SELinux) est activé par défaut dans le paquetage libvirt sur les nœuds compute.

[4] "A Bug Hunter's Diary" de Tobias Klein, ISBN-13: 978-1-59327-385-9 : http://mirror7.meh.or.id/Network%20n%20Security/Bug%20Hunter%20Diary.pdf
[5] http://en.wikipedia.org/wiki/Stack_buffer_overflow
[6] http://www.trapkit.de/tools/checksec.html

```
[root@F19 labotd]# systemctl status libvirtd.service
libvirtd.service - Virtualization daemon
   Loaded: loaded (/usr/lib/systemd/system/libvirtd.service; enabled)
   Active: active (running) since lun. 2013-12-16 15:30:08 CET; 4 months 19
days ago
 Main PID: 1023 (libvirtd)
   CGroup: name=systemd:/system/libvirtd.service
           ├─ 1023 /usr/sbin/libvirtd
           ├─ 1122 /sbin/dnsmasq --conf-
file=/var/lib/libvirt/dnsmasq/default...
```

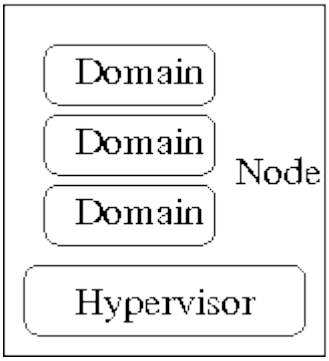
Chaque processus QEMU est exécuté dans un domaine de sécurité SELinux différent.

Le disque d'une VM est labélisé avec un contexte SELinux accessible uniquement depuis le processus Qemu correspondant (cf. Annexe A4).

Je vais réaliser une **étude du code source de ce module sVirt**, pour savoir **quels mécanismes sont réellement implémentés**.

3.2 Concept et architecture de LibVirt

Le code de **sVirt a été intégré au code Libvirt** version 0.6.1 en 2009 cf. http://libvirt.org/news.html. Libvirt est une API qui permet de standardiser les appels faits vers un hyperviseur. Que l'hyperviseur soit du Qemu/KVM ou du VMware ou du VirtualBox ou encore d'autres, les appels pour par exemple la création de machines virtuelle, de réseau ou de pool de stockage restent les mêmes.



Libvirt contient pour **chaque type d'hyperviseur un driver spécifique**.

3.2.1 Connexion

Les appels libvirt peuvent être faits sur un hyperviseur local ou distant.

La librairie est écrite en C mais il existe des bindings pour les langages de programmation les plus utilisés.

Pour utiliser l'API et pouvoir lancer des appels vers un hyperviseur, il faut créer un pointeur nommé virConnectPtr

Exemple :

Types de connections :

QEMU with TLS (qemu+tls://)

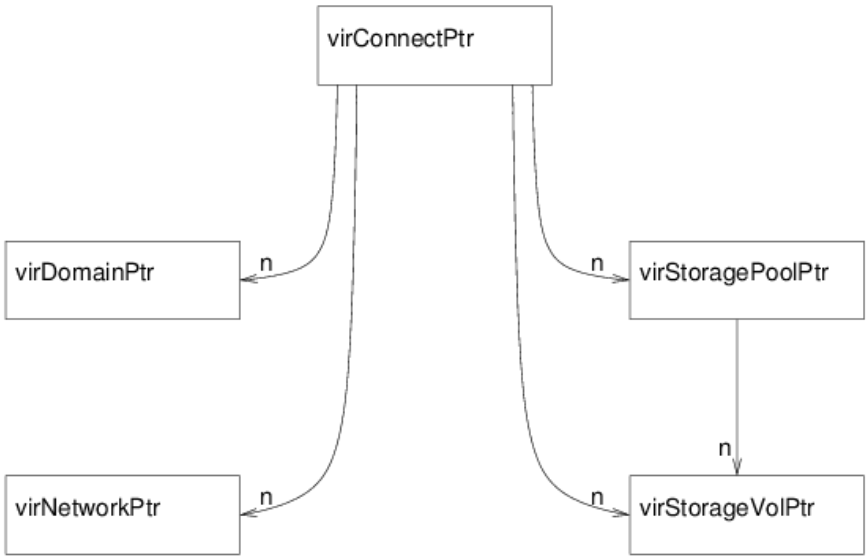
QEMU with direct TCP (qemu+tcp://)

VMware ESX (esx://)

VMware VPX (vpx://)

3.2.2 Utilisation : Deux manières de procéder

- Soit utiliser les **appels standards pour créer une VM à l'aide de pointeur**. Voir la figure suivante reprise de la documentation LibVirt.



- Soit utiliser **un fichier XML qui décrit les actions à entreprendre pour la création d'une VM**
  - Pour plus d'information sur le format de ce fichier XML pour la manipulation de domaine (machine virtuelle) → http://libvirt.org/formatdomain.html

- Pour plus d'information sur le format de ce fichier XML pour la manipulation d'éléments du réseau →http://libvirt.org/formatnetwork.html
- …

Dans Libvirt, la **sécurité des ressources est assurée au sein du driver KVM par un autre driver générique nommé « security driver ».** La sécurité a été implémentée sous forme de driver pour **supporter plusieurs solutions alternatives telles que SELinux, Apparmor** et encore d'autres.

Nom	Modifié le	Type	Taille
security_apparmor.c	03.02.2014 12:16	Fichier C	27 Ko
security_apparmor.h	21.09.2012 04:02	Fichier H	2 Ko
security_dac.c	03.02.2014 12:16	Fichier C	36 Ko
security_dac.h	03.12.2013 17:44	Fichier H	2 Ko
security_driver.c	12.07.2013 14:04	Fichier C	3 Ko
security_driver.h	03.12.2013 17:44	Fichier H	9 Ko
security_manager.c	06.01.2014 14:43	Fichier C	22 Ko
security_manager.h	03.12.2013 17:44	Fichier H	7 Ko
security_nop.c	03.12.2013 17:44	Fichier C	9 Ko
security_nop.h	21.09.2012 04:02	Fichier H	1 Ko
security_selinux.c	21.02.2014 12:24	Fichier C	76 Ko
security_selinux.h	21.09.2012 04:02	Fichier H	1 Ko
security_stack.c	03.12.2013 17:44	Fichier C	18 Ko
security_stack.h	21.12.2012 09:26	Fichier H	2 Ko
virt-aa-helper.c	12.07.2013 14:04	Fichier C	38 Ko

Figure : L'essentiel du code sVirt contenu dans Libvirt 1.2.2

Dans le fichier security_selinux.c, une fonction redirige les appels faits vers les fonctions de sVirt implémentées dans le driver SELinux.

C'est exactement, l'information que je recherche.

Voici les appels qui fixent un label à l'image et au processus d'une VM :

```
virSecurityDriver virSecurityDriverSELinux = {
    …
    .domainSetSecurityImageLabel        =
virSecuritySELinuxSetSecurityImageLabel,
    …
    .domainSetSecurityProcessLabel      =
virSecuritySELinuxSetSecurityProcessLabel,
…
};
```

Listing de toutes les méthodes qui permettent de fixer un label aux ressources :

- domainSetSecurityAllLabel
- domainSetSecurityImageLabel
- domainSetSecurityDaemonSocketLabel
- domainSetSecuritySocketLabel
- domainSetSecurityProcessLabel
- domainSetSecurityChildProcessLabel
- domainSetSecurityHostdevLabel
- domainSetSavedStateLabel
- domainSetSecurityImageFDLabel
- domainSetSecurityTapFDLabel

3.3 Comment les labels sont fixés dans les nœuds computes ?

Pour répondre à cette question, j'ai choisi d'analyser l'architecture logique d'OpenStack[7].

Cette analyse va me guider vers le module responsable du dialogue avec le système hyperviseur.

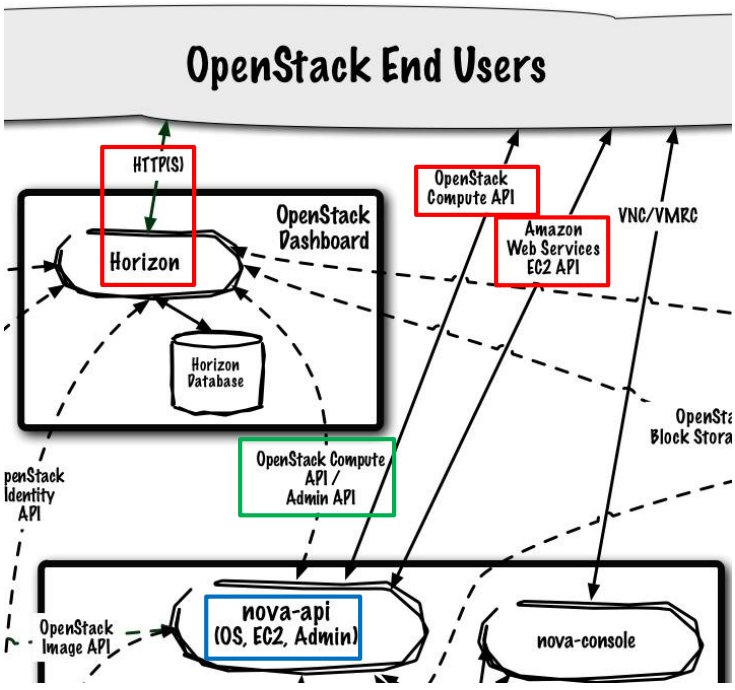


Figure : Architecture logique d'OpenStack, centrée sur la création de VM par un end-user

[7] http://docs.openstack.org/grizzly/openstack-object-storage/admin/content/logical-architecture.html

Dans la figure précédente, je peux relever que pour créer une VM, il faut soit utiliser directement le Compute API ou l'EC2 API d'Amazon ou bien passer par l'interface web Horizon qui elle fera les appels Compute API en arrière-plan.

Ces appels sont reçus par nova-api.

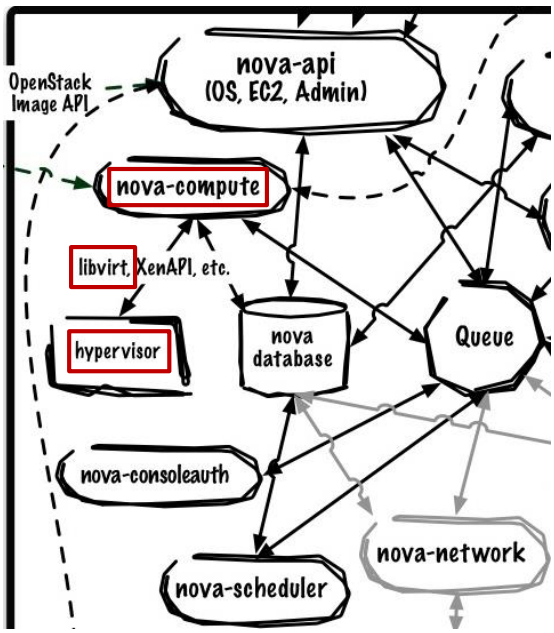


Figure : Le module nova-compute utilise libvirt pour communiquer avec l'hyperviseur

Nova-API a un lien avec la base de données Nova et avec le serveur de queue.

Le serveur de queue est utilisé pour soumettre la création de VM.

Nova-scheduler reçoit la requête et décide de quel nœud compute devra héberger la nouvelle VM.

Le rapport de thèse de Master de Benoît Chalut présente un exemple de message de queue au format AMQP → http://www.tdeig.ch/kvm/Chalut_RTM.pdf (point 2 du paragraphe 4.1.6)

Le module nova-compute qui reçoit la requête de création de VM, la soumet à son hyperviseur à l'aide de la librairie Libvirt.

Pour information, le module nova-compute utilise le format XML du paragraphe pour soumettre sa requête de création à l'hyperviseur. Voir le rapport précédent point 3 du paragraphe 4.1.6

3.4 Analyses des VMs créés via OpenStack

Comme le code de nova-compute fait appel à la librairie Libvirt pour créer ces instances, **sVirt fixe les labels en arrière-plan sans que nova-compute doive faire une demande explicite**.

Preuve :

1) Xml format libvirt de création instances absence demande label
2) Labels instances
3) Service libvirtd
4) Ps -efZ

```
 [root@F19 labotd]# systemctl status libvirtd.service
libvirtd.service - Virtualization daemon

   Loaded: loaded (/usr/lib/systemd/system/libvirtd.service; enabled)

   Active: active (running) since lun. 2013-12-16 15:30:08 CET; 4 months 20
days ago

 Main PID: 1023 (libvirtd)

   CGroup: name=systemd:/system/libvirtd.service

           ├─  345 /usr/bin/qemu-system-x86_64 -machine accel=kvm -name
insta...

           ├─ 1023 /usr/sbin/libvirtd

           ├─ 1122 /sbin/dnsmasq --conf-
file=/var/lib/libvirt/dnsmasq/default...

           └─32545 /usr/bin/qemu-system-x86_64 -machine accel=kvm -name
insta...


[root@F19 labotd]# ps -eZ | grep qemu

system_u:system_r:svirt_t:s0:c5,c789 345 ?     00:00:07 qemu-system-x86

system_u:system_r:svirt_t:s0:c346,c402 32545 ? 00:00:06 qemu-system-x86

[root@F19 labotd]# ps -efZ | grep qemu

system_u:system_r:svirt_t:s0:c5,c789 qemu  345     1  2 10:50 ?
00:00:07 /usr/bin/qemu-system-x86_64 -machine accel=kvm -name instance-
0000001a -S -machine pc-i440fx-1.4,accel=kvm,usb=off -cpu
SandyBridge,+erms,+smep,+fsgsbase,+rdrand,+f16c,+osxsave,+pcid,+pdcm,+xtpr,+t
m2,+est,+vmx,+ds_cpl,+monitor,+dtes64,+pbe,+tm,+ht,+ss,+acpi,+ds,+vme -m 512
```

```
-smp 1,sockets=1,cores=1,threads=1 -uuid 76ba9b04-2769-4127-96b1-eef01075b2bc
-smbios type=1,manufacturer=Fedora Project,product=OpenStack
Nova,version=2013.2-1.fc20,serial=20832181-dad7-dd11-81ce-
60a44c51de13,uuid=76ba9b04-2769-4127-96b1-eef01075b2bc -no-user-config -
nodefaults -chardev
socket,id=charmonitor,path=/var/lib/libvirt/qemu/instance-
0000001a.monitor,server,nowait -mon
chardev=charmonitor,id=monitor,mode=control -rtc base=utc,driftfix=slew -no-
kvm-pit-reinjection -no-shutdown -device piix3-usb-
uhci,id=usb,bus=pci.0,addr=0x1.0x2 -drive
file=/var/lib/nova/instances/76ba9b04-2769-4127-96b1-
eef01075b2bc/disk,if=none,id=drive-virtio-disk0,format=qcow2,cache=none -
device virtio-blk-pci,scsi=off,bus=pci.0,addr=0x4,drive=drive-virtio-
disk0,id=virtio-disk0,bootindex=1 -netdev
tap,fd=26,id=hostnet0,vhost=on,vhostfd=29 -device virtio-net-
pci,netdev=hostnet0,id=net0,mac=fa:16:3e:cf:3c:87,bus=pci.0,addr=0x3 -chardev
file,id=charserial0,path=/var/lib/nova/instances/76ba9b04-2769-4127-96b1-
eef01075b2bc/console.log -device isa-serial,chardev=charserial0,id=serial0 -
chardev pty,id=charserial1 -device isa-serial,chardev=charserial1,id=serial1
-device usb-tablet,id=input0 -vnc 10.2.4.81:1 -k en-us -vga cirrus -device
virtio-balloon-pci,id=balloon0,bus=pci.0,addr=0x5

system_u:system_r:svirt_t:s0:c346,c402 qemu 32545  1  2 10:50 ?
00:00:06 /usr/bin/qemu-system-x86_64 -machine accel=kvm -name instance-
00000019 -S -machine pc-i440fx-1.4,accel=kvm,usb=off -cpu
SandyBridge,+erms,+smep,+fsgsbase,+rdrand,+f16c,+osxsave,+pcid,+pdcm,+xtpr,+t
m2,+est,+vmx,+ds_cpl,+monitor,+dtes64,+pbe,+tm,+ht,+ss,+acpi,+ds,+vme -m 512
-smp 1,sockets=1,cores=1,threads=1 -uuid 8c898aa7-d8de-4fb9-8ef0-36a4a3485a80
-smbios type=1,manufacturer=Fedora Project,product=OpenStack
Nova,version=2013.2-1.fc20,serial=20832181-dad7-dd11-81ce-
60a44c51de13,uuid=8c898aa7-d8de-4fb9-8ef0-36a4a3485a80 -no-user-config -
nodefaults -chardev
socket,id=charmonitor,path=/var/lib/libvirt/qemu/instance-
00000019.monitor,server,nowait -mon
chardev=charmonitor,id=monitor,mode=control -rtc base=utc,driftfix=slew -no-
kvm-pit-reinjection -no-shutdown -device piix3-usb-
uhci,id=usb,bus=pci.0,addr=0x1.0x2 -drive
file=/var/lib/nova/instances/8c898aa7-d8de-4fb9-8ef0-
36a4a3485a80/disk,if=none,id=drive-virtio-disk0,format=qcow2,cache=none -
device virtio-blk-pci,scsi=off,bus=pci.0,addr=0x4,drive=drive-virtio-
disk0,id=virtio-disk0,bootindex=1 -netdev
tap,fd=25,id=hostnet0,vhost=on,vhostfd=26 -device virtio-net-
pci,netdev=hostnet0,id=net0,mac=fa:16:3e:c1:7f:19,bus=pci.0,addr=0x3 -chardev
file,id=charserial0,path=/var/lib/nova/instances/8c898aa7-d8de-4fb9-8ef0-
```

```
36a4a3485a80/console.log -device isa-serial,chardev=charserial0,id=serial0 -
chardev pty,id=charserial1 -device isa-serial,chardev=charserial1,id=serial1
-device usb-tablet,id=input0 -vnc 10.2.4.81:0 -k en-us -vga cirrus -device
virtio-balloon-pci,id=balloon0,bus=pci.0,addr=0x5
```

A. Annexe

A.1 Installation d'OpenStack sur Fedora desktop 19 x86_64

```
## Mise à jour du système

yum –y update

## Redémarrage système

reboot

## Installation du dépôt logiciel Red Hat d'OpenStack

yum install -y http://rdo.fedorapeople.org/openstack-havana/rdo-release-
havana.rpm

## Installation de l'installeur Red Hat d'OpenStack

yum install -y openstack-packstack

## Exécution de l'installeur d'OpenStack

## Installation de tous les composants d'OpenStack sur seul nœud

## Temps d'exécution ~15 minutes

packstack –allinone

## Configuration générée automatiquement

## Pour afficher la configuration

cat /root/packstack-answers-*

## Localisation log de l'installation

cd /var/tmp/packstack/
```

A.2 Configuration réseau

```
[root@localhost ~]#

systemctl disable NetworkManager.service

rm '/etc/systemd/system/dbus-org.freedesktop.NetworkManager.service'

rm '/etc/systemd/system/multi-user.target.wants/NetworkManager.service'

rm '/etc/systemd/system/network.target.wants/NetworkManager-wait-
online.service'

[root@localhost ~]# chkconfig network on

systemctl restart network.service

[root@localhost ~]# vi /etc/udev/rules.d/70-persistent-net.rules

SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*",
ATTR{address}=="b8:ac:6f:65:31:e5", ATTR{dev_id}=="0x0", ATTR{type}=="1",
KERNEL=="eth*", NAME="eth0"

[root@localhost ~]# cd  /etc/sysconfig/network-script/

[root@localhost ~]# mv ifcfg-eno1 ifcfg-eth0

[root@localhost ~]# vi /etc/sysconfig/network-script/ifcfg-eth0

...

[root@localhost ~]# reboot
```

A.3 Configuration de l'installation des composants OpenStack sur un seul nœud

```
[general]

# Path to a Public key to install on servers. If a usable key has not
# been installed on the remote servers the user will be prompted for a
# password and this key will be installed so the password will not be
# required again
CONFIG_SSH_KEY=

# Set to 'y' if you would like Packstack to install MySQL
CONFIG_MYSQL_INSTALL=y

# Set to 'y' if you would like Packstack to install OpenStack Image
# Service (Glance)
CONFIG_GLANCE_INSTALL=y

# Set to 'y' if you would like Packstack to install OpenStack Block
# Storage (Cinder)
CONFIG_CINDER_INSTALL=y

# Set to 'y' if you would like Packstack to install OpenStack Compute
# (Nova)
CONFIG_NOVA_INSTALL=y

# Set to 'y' if you would like Packstack to install OpenStack
# Networking (Neutron)
CONFIG_NEUTRON_INSTALL=y

# Set to 'y' if you would like Packstack to install OpenStack
# Dashboard (Horizon)
CONFIG_HORIZON_INSTALL=y
```

```
# Set to 'y' if you would like Packstack to install OpenStack Object
# Storage (Swift)
CONFIG_SWIFT_INSTALL=y

# Set to 'y' if you would like Packstack to install OpenStack
# Metering (Ceilometer)
CONFIG_CEILOMETER_INSTALL=y

# Set to 'y' if you would like Packstack to install Heat
CONFIG_HEAT_INSTALL=n

# Set to 'y' if you would like Packstack to install the OpenStack
# Client packages. An admin "rc" file will also be installed
CONFIG_CLIENT_INSTALL=y

# Comma separated list of NTP servers. Leave plain if Packstack
# should not install ntpd on instances.
CONFIG_NTP_SERVERS=

# Set to 'y' if you would like Packstack to install Nagios to monitor
# openstack hosts
CONFIG_NAGIOS_INSTALL=y

# Comma separated list of servers to be excluded from installation in
# case you are running Packstack the second time with the same answer
# file and don't want Packstack to touch these servers. Leave plain if
# you don't need to exclude any server.
EXCLUDE_SERVERS=

# The IP address of the server on which to install MySQL
CONFIG_MYSQL_HOST=10.2.2.2
```

```
# Username for the MySQL admin user
CONFIG_MYSQL_USER=root

# Password for the MySQL admin user
CONFIG_MYSQL_PW=4e2cbae521df4a80

# The IP address of the server on which to install the QPID service
CONFIG_QPID_HOST=10.2.2.2

# The IP address of the server on which to install Keystone
CONFIG_KEYSTONE_HOST=10.2.2.2

# The password to use for the Keystone to access DB
CONFIG_KEYSTONE_DB_PW=e7800f65c67e45c3

# The token to use for the Keystone service api
CONFIG_KEYSTONE_ADMIN_TOKEN=2c4817399c4f41e2b3f613d8ef87a1f0

# The password to use for the Keystone admin user
CONFIG_KEYSTONE_ADMIN_PW=c83b0265cda449a7

# The password to use for the Keystone demo user
CONFIG_KEYSTONE_DEMO_PW=fab8eb30abf24eb0

# Kestone token format. Use either UUID or PKI
CONFIG_KEYSTONE_TOKEN_FORMAT=PKI

# The IP address of the server on which to install Glance
CONFIG_GLANCE_HOST=10.2.2.2

# The password to use for the Glance to access DB
CONFIG_GLANCE_DB_PW=51310ff2d2114cd5
```

```
# The password to use for the Glance to authenticate with Keystone
CONFIG_GLANCE_KS_PW=67ebb19f1b7242c6

# The IP address of the server on which to install Cinder
CONFIG_CINDER_HOST=10.2.2.2

# The password to use for the Cinder to access DB
CONFIG_CINDER_DB_PW=37c29c4e7f584f49

# The password to use for the Cinder to authenticate with Keystone
CONFIG_CINDER_KS_PW=8cf735b07b0b4d81

# The Cinder backend to use, valid options are: lvm, gluster, nfs
CONFIG_CINDER_BACKEND=lvm

# Create Cinder's volumes group. This should only be done for testing
# on a proof-of-concept installation of Cinder.  This will create a
# file-backed volume group and is not suitable for production usage.
CONFIG_CINDER_VOLUMES_CREATE=y

# Cinder's volumes group size. Note that actual volume size will be
# extended with 3% more space for VG metadata.
CONFIG_CINDER_VOLUMES_SIZE=20G

# A single or comma separated list of gluster volume shares to mount,
# eg: ip-address:/vol-name
CONFIG_CINDER_GLUSTER_MOUNTS=

# A single or comma seprated list of NFS exports to mount, eg: ip-
# address:/export-name
CONFIG_CINDER_NFS_MOUNTS=
```

```
# The IP address of the server on which to install the Nova API
# service
CONFIG_NOVA_API_HOST=10.2.2.2

# The IP address of the server on which to install the Nova Cert
# service
CONFIG_NOVA_CERT_HOST=10.2.2.2

# The IP address of the server on which to install the Nova VNC proxy
CONFIG_NOVA_VNCPROXY_HOST=10.2.2.2

# A comma separated list of IP addresses on which to install the Nova
# Compute services
CONFIG_NOVA_COMPUTE_HOSTS=10.2.2.2

# The IP address of the server on which to install the Nova Conductor
# service
CONFIG_NOVA_CONDUCTOR_HOST=10.2.2.2

# The password to use for the Nova to access DB
CONFIG_NOVA_DB_PW=cebd2e4795314c82

# The password to use for the Nova to authenticate with Keystone
CONFIG_NOVA_KS_PW=5cc93889132941f8

# The IP address of the server on which to install the Nova Scheduler
# service
CONFIG_NOVA_SCHED_HOST=10.2.2.2

# The overcommitment ratio for virtual to physical CPUs. Set to 1.0
# to disable CPU overcommitment
```

```
CONFIG_NOVA_SCHED_CPU_ALLOC_RATIO=16.0

# The overcommitment ratio for virtual to physical RAM. Set to 1.0 to
# disable RAM overcommitment
CONFIG_NOVA_SCHED_RAM_ALLOC_RATIO=1.5

# Private interface for Flat DHCP on the Nova compute servers
CONFIG_NOVA_COMPUTE_PRIVIF=lo

# The list of IP addresses of the server on which to install the Nova
# Network service
CONFIG_NOVA_NETWORK_HOSTS=10.2.2.2

# Nova network manager
CONFIG_NOVA_NETWORK_MANAGER=nova.network.manager.FlatDHCPManager

# Public interface on the Nova network server
CONFIG_NOVA_NETWORK_PUBIF=eth0

# Private interface for network manager on the Nova network server
CONFIG_NOVA_NETWORK_PRIVIF=lo

# IP Range for network manager
CONFIG_NOVA_NETWORK_FIXEDRANGE=192.168.32.0/22

# IP Range for Floating IP's
CONFIG_NOVA_NETWORK_FLOATRANGE=10.3.4.0/22

# Name of the default floating pool to which the specified floating
# ranges are added to
CONFIG_NOVA_NETWORK_DEFAULTFLOATINGPOOL=nova
```

```
# Automatically assign a floating IP to new instances
CONFIG_NOVA_NETWORK_AUTOASSIGNFLOATINGIP=n

# First VLAN for private networks
CONFIG_NOVA_NETWORK_VLAN_START=100

# Number of networks to support
CONFIG_NOVA_NETWORK_NUMBER=1

# Number of addresses in each private subnet
CONFIG_NOVA_NETWORK_SIZE=255

# The IP addresses of the server on which to install the Neutron
# server
CONFIG_NEUTRON_SERVER_HOST=10.2.2.2

# The password to use for Neutron to authenticate with Keystone
CONFIG_NEUTRON_KS_PW=3a023af1f29e466e

# The password to use for Neutron to access DB
CONFIG_NEUTRON_DB_PW=3261ec6b280b4792

# A comma separated list of IP addresses on which to install Neutron
# L3 agent
CONFIG_NEUTRON_L3_HOSTS=10.2.2.2

# The name of the bridge that the Neutron L3 agent will use for
# external traffic, or 'provider' if using provider networks
CONFIG_NEUTRON_L3_EXT_BRIDGE=br-ex

# A comma separated list of IP addresses on which to install Neutron
# DHCP agent
```

```
CONFIG_NEUTRON_DHCP_HOSTS=10.2.2.2

# The name of the L2 plugin to be used with Neutron
CONFIG_NEUTRON_L2_PLUGIN=openvswitch

# A comma separated list of IP addresses on which to install Neutron
# metadata agent
CONFIG_NEUTRON_METADATA_HOSTS=10.2.2.2

# A comma separated list of IP addresses on which to install Neutron
# metadata agent
CONFIG_NEUTRON_METADATA_PW=d95b9ce4010640e8

# The type of network to allocate for tenant networks
CONFIG_NEUTRON_LB_TENANT_NETWORK_TYPE=local

# A comma separated list of VLAN ranges for the Neutron linuxbridge
# plugin
CONFIG_NEUTRON_LB_VLAN_RANGES=

# A comma separated list of interface mappings for the Neutron
# linuxbridge plugin
CONFIG_NEUTRON_LB_INTERFACE_MAPPINGS=

# Type of network to allocate for tenant networks
CONFIG_NEUTRON_OVS_TENANT_NETWORK_TYPE=local

# A comma separated list of VLAN ranges for the Neutron openvswitch
# plugin
CONFIG_NEUTRON_OVS_VLAN_RANGES=

# A comma separated list of bridge mappings for the Neutron
```

```
# openvswitch plugin
CONFIG_NEUTRON_OVS_BRIDGE_MAPPINGS=

# A comma separated list of colon-separated OVS bridge:interface
# pairs. The interface will be added to the associated bridge.
CONFIG_NEUTRON_OVS_BRIDGE_IFACES=

# A comma separated list of tunnel ranges for the Neutron openvswitch
# plugin
CONFIG_NEUTRON_OVS_TUNNEL_RANGES=

# Override the IP used for GRE tunnels on this hypervisor to the IP
# found on the specified interface (defaults to the HOST IP)
CONFIG_NEUTRON_OVS_TUNNEL_IF=

# The IP address of the server on which to install the OpenStack
# client packages. An admin "rc" file will also be installed
CONFIG_OSCLIENT_HOST=10.2.2.2

# The IP address of the server on which to install Horizon
CONFIG_HORIZON_HOST=10.2.2.2

# To set up Horizon communication over https set this to "y"
CONFIG_HORIZON_SSL=n

# PEM encoded certificate to be used for ssl on the https server,
# leave blank if one should be generated, this certificate should not
# require a passphrase
CONFIG_SSL_CERT=

# Keyfile corresponding to the certificate if one was entered
CONFIG_SSL_KEY=
```

```
# The IP address on which to install the Swift proxy service
# (currently only single proxy is supported)
CONFIG_SWIFT_PROXY_HOSTS=10.2.2.2

# The password to use for the Swift to authenticate with Keystone
CONFIG_SWIFT_KS_PW=6c677e9255fc48be

# A comma separated list of IP addresses on which to install the
# Swift Storage services, each entry should take the format
# <ipaddress>[/dev], for example 127.0.0.1/vdb will install /dev/vdb
# on 127.0.0.1 as a swift storage device(packstack does not create the
# filesystem, you must do this first), if /dev is omitted Packstack
# will create a loopback device for a test setup
CONFIG_SWIFT_STORAGE_HOSTS=10.2.2.2

# Number of swift storage zones, this number MUST be no bigger than
# the number of storage devices configured
CONFIG_SWIFT_STORAGE_ZONES=1

# Number of swift storage replicas, this number MUST be no bigger
# than the number of storage zones configured
CONFIG_SWIFT_STORAGE_REPLICAS=1

# FileSystem type for storage nodes
CONFIG_SWIFT_STORAGE_FSTYPE=ext4

# Whether to provision for demo usage and testing
CONFIG_PROVISION_DEMO=y

# The CIDR network address for the floating IP subnet
CONFIG_PROVISION_DEMO_FLOATRANGE=172.24.4.224/28
```

```
# Whether to configure tempest for testing
CONFIG_PROVISION_TEMPEST=n

# The uri of the tempest git repository to use
CONFIG_PROVISION_TEMPEST_REPO_URI=https://github.com/openstack/tempest.git

# The revision of the tempest git repository to use
CONFIG_PROVISION_TEMPEST_REPO_REVISION=master

# Whether to configure the ovs external bridge in an all-in-one
# deployment
CONFIG_PROVISION_ALL_IN_ONE_OVS_BRIDGE=y

# The IP address of the server on which to install Heat service
CONFIG_HEAT_HOST=10.2.2.2

# The password used by Heat user to authenticate against MySQL
CONFIG_HEAT_DB_PW=cd455189f9fd4955

# The password to use for the Heat to authenticate with Keystone
CONFIG_HEAT_KS_PW=d90ff126bd184858

# Set to 'y' if you would like Packstack to install Heat CloudWatch
# API
CONFIG_HEAT_CLOUDWATCH_INSTALL=n

# Set to 'y' if you would like Packstack to install Heat
# CloudFormation API
CONFIG_HEAT_CFN_INSTALL=n

# The IP address of the server on which to install Heat CloudWatch
```

```
# API service
CONFIG_HEAT_CLOUDWATCH_HOST=10.2.2.2

# The IP address of the server on which to install Heat
# CloudFormation API service
CONFIG_HEAT_CFN_HOST=10.2.2.2

# The IP address of the server on which to install Ceilometer
CONFIG_CEILOMETER_HOST=10.2.2.2

# Secret key for signing metering messages.
CONFIG_CEILOMETER_SECRET=49daa7cafb1e4bbf

# The password to use for Ceilometer to authenticate with Keystone
CONFIG_CEILOMETER_KS_PW=ca947f58e65d40af

# To subscribe each server to EPEL enter "y"
CONFIG_USE_EPEL=y

# A comma separated list of URLs to any additional yum repositories
# to install
CONFIG_REPO=

# The IP address of the server on which to install the Nagios server
CONFIG_NAGIOS_HOST=10.2.2.2

# The password of the nagiosadmin user on the Nagios server
CONFIG_NAGIOS_PW=30e80322ee824a15
```

A.4 sVirt

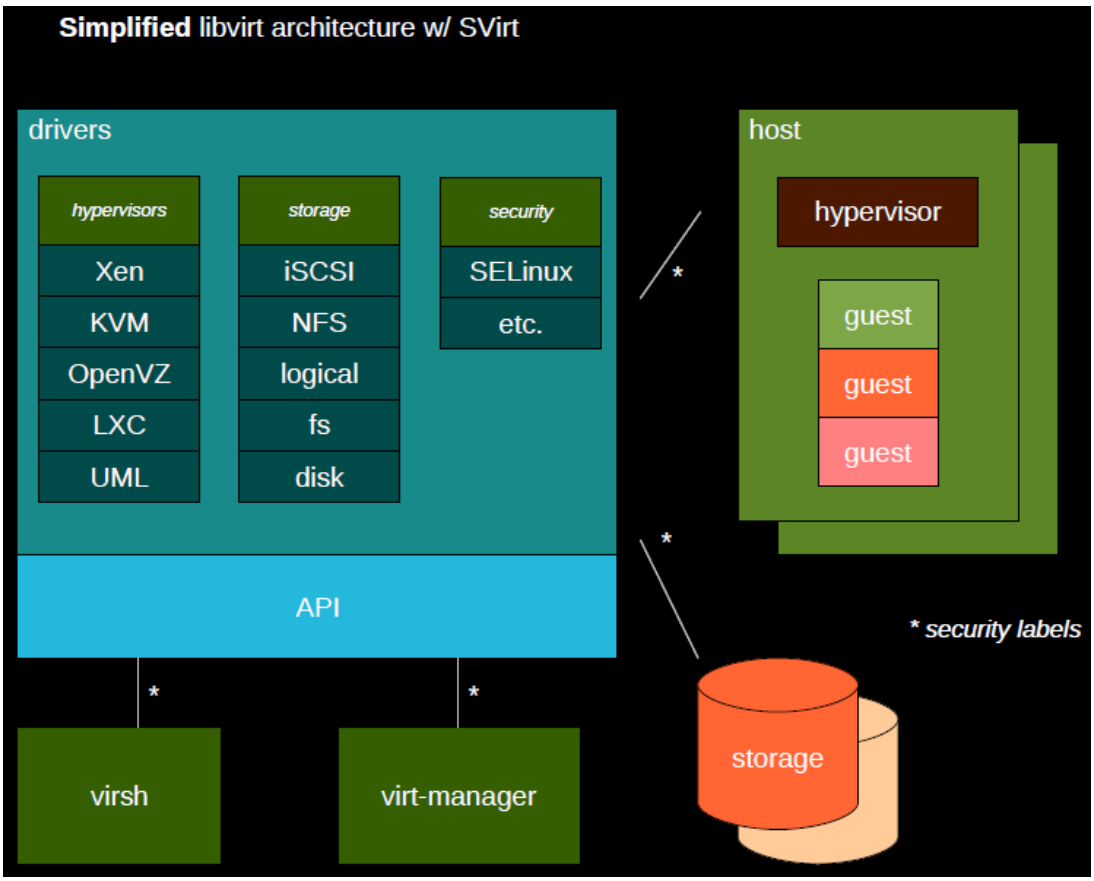
OBJECTIF DE SVIRT :

sVirt est en mesure d'appliquer des **labels** de sécurité **MCS** distincts à chaque VM.

La politique MAC veillera à empêcher toute tentative d'une VM_A à accéder à une VM_B si les labels de sécurité diffèrent ; garantissant ainsi le cloisonnement des VMs

sVirt entre en action dès que nous configurons et gérons les VM via **libvirt**.

Il est donc intégré dans libvirt en tant que drivers dans la catégorie « security », voir **figure** ci-dessous.



Une simple option lors de la création de la VM permettra de demander à ce que la VM soit « isolée », la chaîne d'outils s'occupera de faire l'étiquetage et la configuration des politiques en arrière-plan.

SITUATION ACTUELLE (Fedora 18 ?) AVEC SELINUX ACTIVE SANS SVIRT

SELinux est déjà en mesure de fournir une protection MAC générale à la virtualisation dans Linux, telles que la protection de l'intégrité et confidentialité des images disque en offrant une isolation solide du processus hyperviseur du reste du système.

Il n'y a aucun support explicite pour la virtualisation Linux dans SELinux, et toutes les VM sont actuellement dirigées dans le même contexte de sécurité. Ainsi, il n'y a pas d'isolation MAC appliquée entre VM.

SITUATION ACTUELLE AVEC SELINUX ACTIVE AVEC SVIRT :

Quand on active SELinux, le service sVirt démarre automatiquement puis créé et gère les labels des VM.

Le service sVirt est inclus dans libvirt.

Le **labeling dynamique** est recommandé dans la plupart des cas.

Mais nous avons la possibilité de désactiver le labeling dynamique et de créer nos propres labels statiques. Dans ce cas, c'est à nous de vérifier que les labels statiques sont uniques pour chaque VM.

Concrètement c'est le **contrôle d'accès Multi-Level Security (MLS)** qui est utilisé pour différencier le contexte de sécurité de chaque VM active.

Démonstration, affichage du domaine du processus d'une VM :

$ = shell utilisateur

= shell root

Activer SELinux. /etc/SELinux/config enforcing

Redémarrer ordinateur

```
$ getenforce
```

Constater service libvirt actif :

```
# service --status-all
```

Si pas de VM existante, créer une VM avec l'image UbuntuServer.raw du bureau :

```
# virt-install --import --connect qemu:///system --name VM1 --ram 1024 --vcpus 1 --file /home/labotd/Desktop/UbuntuServer.raw
```

Si une VM est existante mais éteinte alors démarrer la VM :

```
# virsh start NOM_VM
```

Récupérer le label défini dynamiquement à VM1 à l'aide de la commande ps qui affiche tous les processus qemu-kvm démarrés actuellement :

```
# ps -C qemu-kvm -o label,command
```

```
LABEL                                COMMAND
system_u:system_r:svirt_t:s0:c649,c  /usr/bin/qemu-kvm -S -M pc-
851                                  0.14 -enable-kvm -m 1024 -
                                     smp
                                     2,sockets=2,cores=1,thread
                                     s=1    -name    VM1    -uuid
                                     397b09bf-8bbb-8da9-f12f-
                                     2b8884649881 -nodefconfig -
                                     nodefaults -ch
```

Vérifier le label de l'image disk :

```
# ls -Z /home/labotd/Desktop/UbuntuServer.raw
```

```
-rw-r--r--.     qemu     qemu     system_u:object_r:svirt_image_t:s0:c649,c851
/home/labotd/Desktop/UbuntuServer.raw
```

Vérifier dans le fichier xml qui décrit la configuration de la VM :

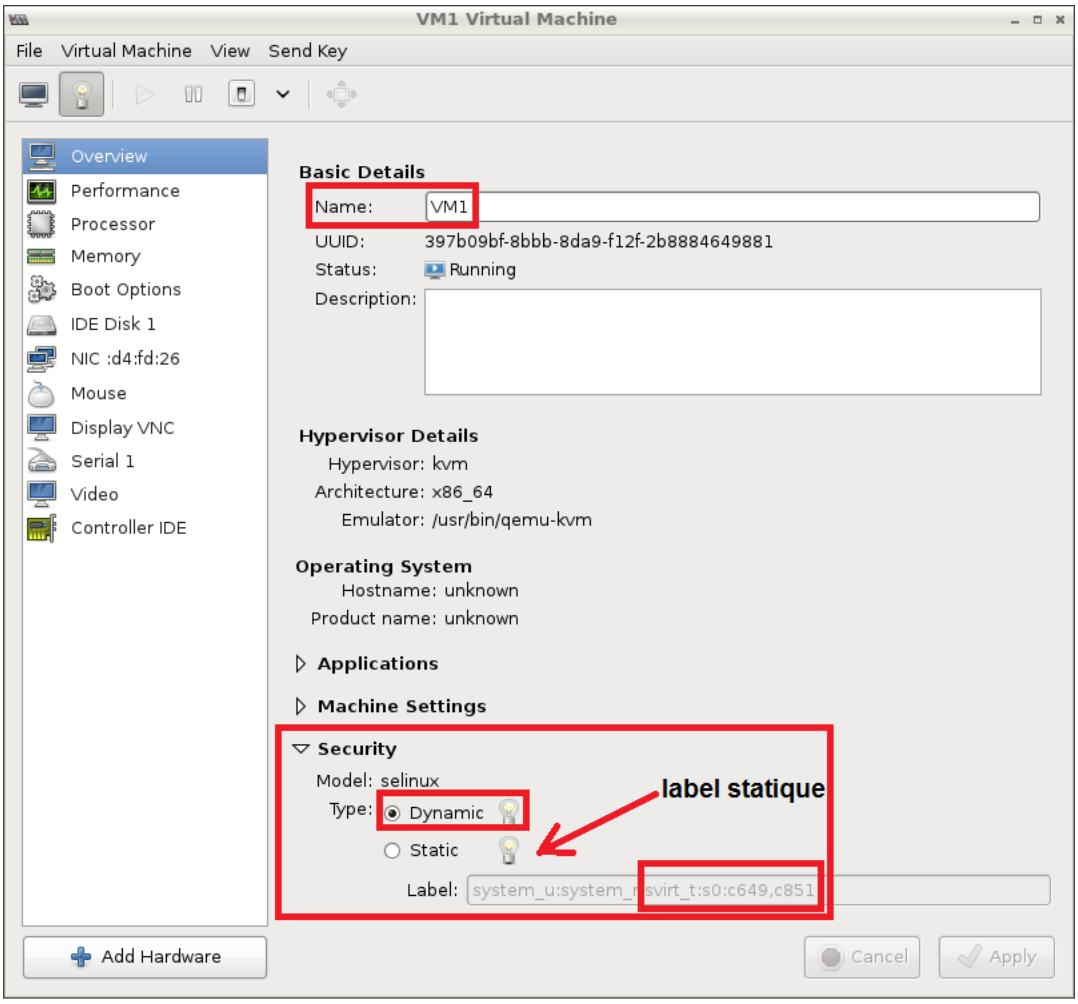
```
# virsh dumpxml VM1 | grep label
```

```
<seclabel type='dynamic' model='selinux' relabel='yes'>

<label>system_u:system_r:svirt_t:s0:c649,c851</label>

<imagelabel>system_u:object_r:svirt_image_t:s0:c649,c851</imagelabel>

</seclabel>
```

Type/Description	SELinux Context
Virtualized guest processes. MCS1 is a random MCS field. Approximately 500,000 labels are supported.	system_u:system_r:svirt_t:MCS1

Type/Description	SELinux Context
Virtualized guest images. Only *svirt_t* processes with the same MCS fields can read/write these images.	system_u:object_r:svirt_image_t:MCS1
Virtualized guest shared read/write content. All *svirt_t* processes can write to the svirt_image_t:s0 files.	system_u:object_r:svirt_image_t:s0
Virtualized guest shared read only content. All svirt_t processes can read these files/devices.	system_u:object_r:svirt_content_t:s0
Virtualized guest images. Default label for when an image exits. No *svirt_t* virtual processes can read files/devices with this label.	system_u:object_r:virt_content_t:s0

Configurer label dans virt-manager :



Références :

sVirt: Hardening Linux Virtualization with Mandatory Access Control

Présentation sur sVirt faite en 2009 par le créateur de sVirt (James Morris). James Morris est employé de Red Hat, collègue de Daniel Walsh : http://namei.org/presentations/svirt-lca-2009.pdf

Cahier de charge de sVirt : http://selinuxproject.org/page/Svirt_requirements_v1.0

Documentation Red Hat Enterprise Linux 6 - Virtualization Administration Guide qui présente la signification des labels, voir §4.2 : http://www.tdeig.ch/kvm/liens.pdf

Document IBM (Virtualization for Linux on IBM x86 servers - KVM security) avec information sur la virtualisation KVM et son fonctionnement (p.9 → p.11): http://www.tdeig.ch/kvm/liens.pdf