Documentation CLIP

1304

Socle et cages CLIP

Version	Date	Auteur	Commentaires
1.1.0	04/09/2008	Vincent Strubel	Ajout de *fuser* avec CLSM_PRIV_PROCFD dans $USER_{clip}$.
1.0.4	06/08/2008	Vincent Strubel	Ajout de *nano*.
1.0.3	30/07/2008	Vincent Strubel	Convention plus lisible pour les références.
1.0.2	17/07/2008	Vincent Strubel	Ajout Figure 2.
1.0.1	16/07/2008	Vincent Strubel	Correction de coquille dans les références. Ajout des montages de */home/admin.rmX*
1.0	11/07/2008	Vincent Strubel	Version initiale, à jour pour CLIP v03.00.04.

Table des matières

Introduction

Le système CLIP met en oeuvre deux mécanismes de cloisonnement logiciel local : un mécanisme lourd, reposant sur *vserver* ([CLIP_1202]) et permettant de partitionner le système en "**cages**", et un mécanisme plus léger, reposant sur l'appel *chroot()* durci par *grsecurity* ([CLIP_1203]) et permettant de sous-partitionner les cages en "**vues**". Les différents types de systèmes CLIP intègrent tous un ensemble commun de cages, dites cages CLIP, au nombre de cinq, ainsi qu'un compartiment logiciel par défaut, dit "**socle**", qui correspond à l'ensemble des fichiers et processus non rattachés à une cage spécifique. Ce substrat commun peut ou non être complété par des cages supplémentaires, en fonction du type de système. Les cages RM d'un système CLIP-RM sont un exemple de telles cages supplémentaires. Le présent document présente ce substrat commun, à l'exclusion des cages supplémentaires, qui sont décrites dans des documents distincts ([CLIP_1401]).

1 Socle CLIP

Le socle CLIP est composé de l'ensemble des composants logiciels qu'y ne s'exécutent dans aucune cage du système (c'est-à-dire dans le contexte *vserver* dit ADMIN, cf. [CLIP_1202]), et des fichiers correspondants. Son arborescence de fichiers est l'arborescence principale du système, qui englobe l'ensemble des arborescences de cages.

1.1 Système de fichiers

1.1.1 Partitionnement et montages

Le système CLIP met en oeuvre un plan de partitionnement très détaillé dans le but de décorréler les possibilités d'accès en écriture[1] à différentes parties du système. Au sein du socle du système, on distingue ainsi un coeur insécable, constitué de la racine du système, de son noyau, des fichiers de configuration essentiels dans */etc*, et des bibliothèques et exécutables fondamentaux installés dans */bin, /sbin, /lib, /usr/bin, /usr/sbin, /usr/lib* et */usr/libexec*. Ce coeur est entièrement contenu dans la partition racine du système, qui n'est donc jamais montée autrement qu'en lecture seule, à l'exception des phases de mise à jour (cf. [CLIP_DCS_13006]). Cette protection en intégrité a deux justifications. D'une part, sur le plan de la sécurité, elle conditionne l'intégrité des fonctions de sécurité principales du système. D'autre part, sur le plan fonctionnel, ce coeur constitue un tout auto-suffisant nécessaire au bon fonctionnement de la procédure de mise à jour, qui permettrait de rattraper tout problème résultant d'une mise à jour d'une couche supérieure du système.

Un accès en écriture est en revanche normal et souhaitable pour plusieurs répertoires de l'arborescence de ce socle. Le bon fonctionnement du système nécessite en effet un accès en écriture à certains répertoires « canoniques » : */var*, */tmp*, */dev*, */proc*… S'y ajoute, dans le cadre de CLIP, la sous-arborescence dédiée aux paquetages « utilisateur » (par exemple X11), lesquels peuvent être mis à jour, de manière classique, à tout moment. Cette sous-arborescence est principalement montée sous */usr/local*. Le grand nombre de points de montages différents nécessite la mise en commun de plusieurs montages (*/var, /dev, /usr/local*) sur une seule partition physique, initialement montée sur le répertoire */mounts*, dont les différents sous-répertoires sont ensuite projetés par montages *bind* au sein de l'arborescence du système.

A ces deux partitions, racine et */mounts*, s'ajoutent dans le cas des systèmes CLIP-RM une partition dédiée par cage RM. Cette séparation des systèmes de fichiers est nécessaire au cloisonnement en confidentialité de ces différentes cages entre elles. En effet, deux cages partageant un système de fichiers auquel elles ont accès en écriture peuvent toujours échanger de l'information entre elles, même si leurs arborescences sont confinées à deux répertoires distincts du système de fichiers. Par exemple, le taux d'occupation du système de fichiers constitue dans ce cas une information accessible en lecture / écriture depuis chacune des deux cages, qui peuvent donc établir un canal de communication par ce biais.

Enfin, deux systèmes CLIP complet sont normalement installés sur le disque dur de chaque poste, afin

[1] Ainsi que les autres droits qui peuvent être associés à un montage : exécution de fichiers, utilisation de bits *suid/sgid* ou de *devices*

de permettre la mise en oeuvre des mécanismes de mise à jour du socle ([CLIP_DCS_13006]) et de sauvegarde / restauration ([CLIP_DCS_15094]). Ces deux installations disposent chacune de partitions propres pour leurs exécutables et autres fichiers système, mais partagent en revanche plusieurs partitions communes de données. Plus spécifiquement :

- Chaque installation dispose en propre d'une partition racine et d'une partition */mounts*
- Dans le cas d'un système CLIP-RM uniquement, chaque installation dispose en propre d'un couple de partitions pour RM_H et RM_B.
- Les deux installations partagent une partition de *swap*[2]
- La partition montée sur */home* est partagée entre les deux installations. On notera que le */home* du socle n'est jamais exposé directement aux utilisateurs, mais contient en revanche l'ensemble des données utilisateurs du poste, en particulier :
  - La liste des comptes utilisateurs du système, et les empreintes des mots de passe associés
  - Les répertoires *$HOME* des utilisateurs, sous forme chiffrée ou non (cf. [CLIP_1302]).
  - Certaines clés cryptographiques liées aux utilisateurs du poste, en particulier les clés d'export des clés de chiffrement et signature de supports USB (cf. [CLIP_DCS_15088]).
- La partition dédiée au stockage des journaux système, montée sur */var/log* dans le socle, est commune aux deux installation.
- Les deux partitions réservées aux sauvegardes (cf. [CLIP_DCS_15094]), respectivement de données (montée sur */backup/data*) et du système (montée sur */backup/clip*) sont aussi partagées par les deux installations.
- La partition de démarrage (contenant le noyau et le chargeur de démarrage, et qui n'est montée que pendant la mise à jour du coeur du socle) est aussi commune aux deux installations.

Le plan de partitionnement est résumé dans la Figure 1. Les différents montages du socle CLIP, et les droits associés, sont récapitulés dans les Tableau 1 et Tableau 2. Ces montages sont réalisés à partir du fichier */etc/fstab* par le script de démarrage */etc/init.d/localmount*, à l'exception des montages de */proc, /sys*, */lib/splash/cache*[3] et */var/lib/init.d*[4], qui sont montés directement par */sbin/rc* lors du traitement du *runlevel sysinit* (cf. [CLIP_1301])

On notera plus particulièrement les points suivants :

- L'arborescence */var* est montée avec les droits d'exécution (pas d'option *noexec*). Ceci est nécessaire à l'installeur de paquetages, qui exécute des scripts de pré-/post-installation depuis */var/pkg/lib*. Un raffinement de ces montages (faisant de */var/pkg/lib* un montage distinct, seul à disposer des droits en exécution) pourrait être mis en oeuvre. A ce stade, la menace d'exécution d'un binaire créé dans */var/tmp* par exemple est couverte uniquement par le mécanisme TPE de *Grsecurity* ([CLIP_1203]) et par le fait que ce montage n'est visible que dans le socle et la cage $\text{UPDATE}_{\text{clip}}$.

[2] Ce partage vise uniquement à économiser de la place sur le disque, dans la mesure où le *swap* est chiffré avec une clé aléatoire à chaque démarrage, ce qui interdit le partage d'informations par ce biais entre les deux installations.
[3] Utilisé par le gestionnaire d'écran graphique de démarrage, *media-gfx/splashutils.*
[4] Utilisé par le gestionnaire de scripts de démarrage.

- L'arborescence */dev* doit être accessible en écriture, dans la mesure où l'accès à certains des *devices* nécessite les droits en écriture sur le montage correspondant. Une solution plus adaptée serait de modifier les structures *file_operations* associées à ces *devices* dans le noyau Linux, de manière à rendre leur accès compatible d'un système de fichier monté en lecture seule. Pour l'heure, un montage *bind* avec des droits en écriture est réalisé de */mounts/dev* sur */dev*. Certains périphériques sont toutefois nécessaires (sans que cet accès ne requière un montage en lecture-écriture) avant que le montage de */mounts/dev* ne soit possible, en particulier les périphériques en mode bloc correspondant aux partitions du système. Le paquetage *clip-layout/baselayout-clip* crée donc deux copies de chaque *device* statique : une dans */dev*, utilisée uniquement au début de la phase de démarrage, et une dans */mounts/dev*, qui se superpose à la précédente pour autoriser les accès en écriture.

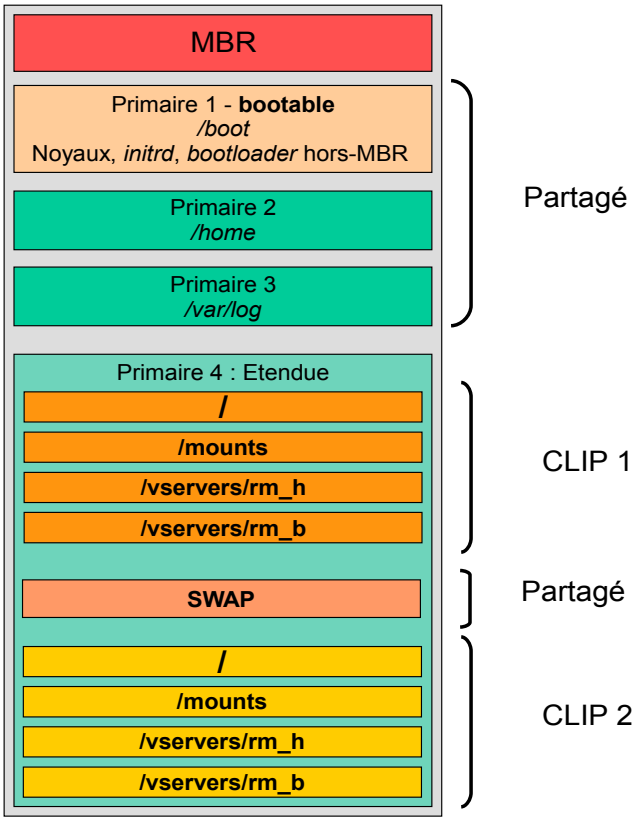

Figure 1: Organisation sur le disque d'une double installation CLIP-RM.

- Le système de fichier */tmp* est en mémoire uniquement, et ne laisse aucun élément rémanent entre deux démarrages.
- Le *swap* est chiffré (de manière à ne pas stocker en clair sur les disques des informations en mémoire potentiellement sensibles). Ce chiffrement est réalisé à l'aide du *device-mapper* Linux,

et plus particulièrement du mode *dm-crypt*. L'utilitaire *cryptsetup* est appelé par le script de démarrage */etc/init.d/localmount* pour réaliser une projection de la partition associé au *swap*[5] vers */dev/mapper/swap0*, sur lequel est ensuite créé le swap. La projection réalise un chiffrement à la volée, en AES-256 dans le mode LRW-BENBI, avant l'écriture sur le disque. La clé de chiffrement de 384 bits[6] est tirée aléatoirement à chaque démarrage. Ce chiffrement est complété par un masque de permissions *devctl* (cf. [CLIP_1201]) nul sur */dev/mapper/swap0*, interdisant tout accès, même en lecture, à l'image claire du *swap*.

- Le montage de */etc*/*core* et */etc/tcb* depuis */home* permet à la fois de maintenir un accès en écriture à ces répertoires, qui contiennent la définition des comptes utilisateur du système, et de les partager avec l'installation CLIP alternative.

Source	Point de montage	Type	Options
<proc>	*/proc*	*proc*	*rw,nosuid,nodev,noexec*
<sys>	*/sys*	*sysfs*	*rw,nosuid,nodev,noexec*
/dev/ROOT_PART	*/*	*ext3*	*ro,noatime*
/dev/mapper/swap0	*<none>*	*swap*	*defaults* (chiffrement AES-256, en mode LRW-BENBI).
<cachedir>	*/lib/splash/cache*	*tmpfs*	*rw,nosuid,nodev,mode=0644*
<svcdir>	*/var/lib/init.d*	*tmpfs*	*rw,nosuid,nodev,mode=0755*
/dev/HOME_PART	*/home*	*ext3*	*rw,nosuid,nodev,noexec,noatime*
/dev/MOUNTS_PART	*/mounts*	*ext3*	*rw,noatime*
/home/etc.users/tcb	*/etc/tcb*	*bind*	*rw,nosuid,nodev,noexec,noatime*
/home/etc.users/core	*/etc/core*	*bind*	*rw,nosuid,nodev,noexec,noatime*
/mounts/usr	*/usr/local*	*bind*	*rw,nodev,noatime*
/mounts/var	*/var*	*bind*	*rw,nosuid,nodev,noatime*
/mounts/dev	*/dev*	*bind*	*rw,nosuid,noexec,noatime*
/mounts/admin_priv/etc.admin	*/etc/admin*	*bind*	*rw,nosuid,noexec,nodev,noatime*
/mounts/admin_priv/etc.ike2	*/etc/admin/ike2*	*bind*	*rw,nosuid,noexec,nodev,noatime*
<tmp>	*/tmp*	*tmpfs*	*rw,noexec,nosuid,nodev,noatime, mode=1777*
<shm>	*/dev/shm*	*tmpfs*	*rw,noexec,nosuid,nodev,noatime*
/dev/LOG_PART	*/var/log*	*ext3*	*rw,nosuid,nodev,noexec,noatime*
/dev/DATA_BACKUP_PART	*/backup/data*	*ext3*	*ro,nosuid,nodev,noexec,noatime*
/dev/BACKUP_PART	*/backup/clip*	*ext3*	*ro,nosuid,nodev,noexec,noatime*

Tableau 1: Montages du socle CLIP.

[5] Cette partition est identifiée par la ligne *swap* du fichier *fstab*, qui prend par exemple la forme :

```
/dev/sda9   none   swap   defaults,noauto
```

On notera bien que la présence d'une telle ligne ne signifie pas qu'un *swap* non chiffré est créé sur */dev/sda9*, mais que le script *localmount* utilise la partition */dev/sda9* comme support de *swap* chiffré.

[6] 256 bits pour le chiffrement, plus 128 bits pour le *"tweak"* LRW.

- Au sein d'un poste CLIP-RM, un montage de type *loop* est utilisé pour projeter une image de système de fichiers sur */home/adminrmX*, pour chaque cage RM_X. Ce montage est ensuite remonté dans la cage RM, comme répertoire *$HOME* de l'utilisateur *_admin* de la vue ADMIN (cf. [CLIP_1401]). Il est ainsi possible de stocker le contenu de ces répertoires *$HOME* dans la partition */home* commune (et ainsi de partager ce contenu entre les deux installations CLIP du poste), sans pour autant partager entre les différentes cages RM un accès en écriture au même système de fichiers sous-jacent[7]. Ces montages *loop* sont réalisés dans le socle, plutôt que directement dans les cages, ce qui permet de ne pas inclure de support pour ce type de montages dans *vsctl* ([CLIP_1202]). Ces montages ne sont par ailleurs pas définis dans le *fstab* du poste, mais créés au démarrage par le script */etc/init.d/clip_viewers* (*app-clip/clip-vserver*) ([CLIP_1301]).

Source	Point de montage	Type	Options
/dev/RMH_PART	*/vservers/rm_h*	*ext3*	*rw,nodev*
/dev/RMB_PART	*/vservers/rm_b*	*ext3*	*rw,nodev*
/home/admin.rmh	*/home/adminrmh*	*ext2*	*loop,rw,nosuid,nodev,noexec,noatime*
/home/admin.rmb	*/home/adminrmb*	*ext2*	*loop,rw,nosuid,nodev,noexec,noatime*

Tableau 2: Montages supplémentaires du socle CLIP au sein d'un système CLIP-RM.

1.1.2 Répertoires partagés

Plusieurs répertoires de l'arborescence du socle sont exposés à dessein dans les cages, en particulier :

- Le répertoire */etc/core* est exposé dans toutes les cages (CLIP, mais aussi RM si le système incorpore de telles cages) à l'exception de la cage X11. Il contient les fichiers commun à l'ensemble du système, en particulier la composante non sensible de la définition des comptes utilisateurs (fichiers *passwd* et *group*). Des liens symboliques */etc/passwd* et */etc/group* pointant vers les fichiers de */etc/core* sont créés indépendamment dans les arborescences du socle et de chaque cage, afin de permettre l'utilisation de ces fichiers par des applications standard. La composante sensible des définitions de comptes utilisateurs (fichiers *shadow* individuels contenant les empreintes *bcryt* de mots de passe utilisateurs - cf. [CLIP_1302]) est en revanche stockée dans un montage distinct, */etc/tcb* qui n'est quant à lui exposé que dans le socle, afin d'interdire toute tentative de recherche de mots de passe depuis les cages. Les fichiers de */etc/core* sont accessibles en lecture-écriture depuis le socle, mais uniquement en lecture depuis les différentes cages. Le répertoire contient aussi le fichier *screen.geom* utilisé par les scripts de post-installation des paquetages (CLIP et RM) pour déterminer la géométrie de l'écran du poste (cf. [CLIP_1101]). Ce fichier n'ayant pas vocation à être modifié, il est protégé par un attribut *ext3* IMMUTABLE.
- Le répertoire */etc/shared* contient des fichiers de configuration communs à tous les paquetages de la distribution CLIP (c'est-à-dire ceux installés dans le socle et les cages CLIP), notamment ceux installés par la bibliothèque *glibc* (*/etc/services* par exemple) ou *ncurses* (définitions de

[7] Un tel partage créerait un canal de communication entre les cages RM, par la modification depuis chaque cage de l'espace disponible sur le système de fichiers partagés. Voir aussi [CLIP_1202].

terminaux dans */etc/shared/terminfo*). Ce répertoire est exposé en lecture seule dans le socle et dans toutes les cages CLIP (cage X11 exceptée), mais pas dans les éventuelles cages RM.

- Le répertoire */etc/admin* contient les fichiers de configuration modifiables par l'administrateur CLIP local. Ce répertoire est partagé en lecture-écriture par le socle et la cage $ADMIN_{clip}$. Il est de plus exposé en lecture seule dans toutes les autres cages CLIP, à l'exception de la cage X11, mais pas en revanche dans les éventuelles cages RM. Ce répertoire contient uniquement des fichiers non sensibles en confidentialité.
- Le répertoire */etc/admin/ike2* constitue le pendant de */etc/admin* pour ce qui concerne les fichiers modifiables par l'administrateur qui sont sensibles en confidentialité, c'est-à-dire les clés ACID utilisées pour l'authentification IKE (cf. [CLIP_1501]). Ce répertoire est partagé en lecture-écriture par le socle et la cage $ADMIN_{clip}$. Il n'est en revanche exposé dans aucune autre cage.
- Le répertoire */mounts/update_priv/pkgs* contient les paquetages CLIP téléchargés par la cage $UPDATE_{clip}$, qui sont exploités aussi bien par cette cage (pour les téléchargement et les mises à jour de paquetages secondaires) que par le socle lui-même (pour les mises à jour de paquetages primaires). Il est donc partagé en lecture-écriture par ces deux compartiments.
- De même, le répertoire */var/pkg* contient les données du système de gestion de paquetages, et est partagé en lecture-écriture par le socle et $UPDATE_{clip}$.
- Le répertoire */tmp/.X11-unix*, qui contient la socket de connexion au serveur X11, est partagé par le socle (lecture-écriture), la cage $USER_{clip}$ et ses vues visionneuses (lecture seule) et la cage X11 (lecture-écriture).
- Le répertoire */var/run/authdir*, qui contient le *cookie Xauthority* d'authentification auprès du serveur X11 initialement généré par *xdm*, est partagé par le socle (lecture-écriture), la cage X11 (lecture-écriture) et la cage $USER_{clip}$ (lecture seule).
- Enfin, le répertoire */var/log* contient les journaux du système (sur une partition dédiée), et est partagé en lecture-écriture par le socle et $AUDIT_{clip}$.

On notera a contrario que certains répertoires, contenant des fichiers sensibles en confidentialité, ne sont exposés ni en lecture ni en écriture dans aucune des cages. Il s'agit en particulier :

- De */dev* (et */mounts/dev*), qui contient des périphériques dont la lecture pourrait indirectement donner accès à des informations sensibles, par exemple les */dev/ttyN*, */dev/fbN*, ainsi que les fichiers en mode bloc correspondant au disque racine.
- De */etc/tcb* comme mentionné plus haut.
- De */home*, qui contient en particulier les images chiffrées des partitions utilisateur, et les clés (chiffrées) associées. L'accès au clés chiffrées des partitions utilisateur pourrait en effet, au même titre que l'accès aux fichiers *shadow* de */etc/tcb*, permettre de tenter une recherche exhaustive hors-ligne des mots de passe utilisateur (cf. [CLIP_1302]).
- Des partitions de sauvegarde montées sur */backup/clip* et */backup/data*, qui contiennent des versions antérieures de ces différents éléments sensibles.

1.1.3 Périphériques

Le contrôle des périphériques exposés à la couche utilisateur contribue à réduire l'interface entre le noyau et la couche utilisateur, et participe à ce titre à la défense en profondeur du composant le plus privilégié du système. Ce contrôle repose sur l'utilisation d'une configuration statique de */dev*, créée à l'aide de commandes *mknod*, plutôt que l'emploi d'un gestionnaire dynamique de type *udev*, comme cela est désormais la norme pour les distributions Linux grand public. Cette configuration est créée (en deux exemplaires : */dev* et */mounts/dev*, voir plus haut) par le paquetage *clip-layout/baselayout-clip*, à l'aide d'une configuration *MAKEDEV* spécifique : *generic-clip*. A ce stade, les fichiers *devices* ainsi créés sont les suivants :

- périphériques de base en mode caractère : *null, full, zero, mem* (accès en lecture ou écriture interdit par *Grsecurity*), *kmem* (accès en lecture ou écriture interdit par *Grsecurity*), *port* (idem), *random, urandom, ramX*
- périphériques *loop* : *loopX*
- terminaux virtuels : *ttyX, vsc[a]X, console, tty*
- pseudo terminaux *System V* : *ptmx*
- disque IDE/ATA : *hdaX*
- disque SCSI (USB/SATA) : *sdaX, sdbX* (et *sdc* dans le cas d'une configuration CLIP-GTW)
- disque raid : *mdX*, uniquement dans le cas d'une configuration CLIP-GTW (*clip-layout/baselayout-clip* généré avec le drapeau USE *clip-gtw*)
- framebuffer vidéo : *fbX*
- périphériques d'entrées : *input/mouseX*
- périphériques d'administration de CLIP-LSM : */dev/veriexec* et */dev/devctl* (cf. [CLIP_1201])

1.2 Processus

Le socle est le compartiment logiciel par défaut lors du démarrage du système. L'ensemble des séquences de démarrage et d'arrêt y sont exécutées. Le socle fonctionne de manière non interactive : aucune session utilisateur n'y est possible. En dehors des phases de démarrage et d'arrêt (qui sont décrites plus en détail dans [CLIP_1301]), les processus actifs dans le socle sont limités à un certain nombre de démons :

- ***init*** (*sys-apps/sysvinit*), en attente de commandes d'arrêt ou de redémarrage sur la *socket unix /dev/initctl*. Voir aussi [CLIP_1301].
- ***spmd*** (*net-firewall/racoon2*), démon de mise à jour des politiques de sécurité IPsec, en attente de demandes de mises à jour (émises par *iked*) sur la *socket unix /var/run/racoon/spmif*. Les connexions à cette *socket* sont authentifiées par la connaissance du secret */etc/ike2/spmd.psk* (*root:root*, permissions *0400*). Voir aussi [CLIP_1501].
- ***iked*** (*net-firewall/racoon2*), démon de négociation d'associations de sécurité IPsec, qui dialogue sur le réseau sur les ports 500 et 4500 (UDP), et interagit localement avec *spmd* sur la *socket unix /var/run/racoon/spmif*. Voir aussi [CLIP_1501].

- ***pwcheckd*** (*app-clip/pwcheckd*), démon de vérification de mots de passe utilisateurs, permettant le déverrouillage de l'écran X11 depuis $USER_{clip}$. Ce démon est en attente de demandes de vérification sur la *socket unix /mounts/user_priv/var/run/pwcheckd* (*/var/run/pwcheckd* dans $USER_{clip}$). Voir aussi [CLIP_1302].
- ***databackupsrv*** (*app-clip/clip-data-backup-srv*), démon de gestion des requêtes de sauvegarde / restauration des données, en attente de telles requêtes sur la *socket unix /mounts/admin_priv/var/run/databackup* (*/var/run/databackup* dans $ADMIN_{clip}$). Voir aussi [CLIP_DCS_15094].
- ***backupsrv*** (*app-clip/clip-backup-srv*), démon de gestion des requêtes de sauvegarde / restauration des données, en attente de telles requêtes sur la *socket unix /mounts/admin_priv/var/run/backup* (*/var/run/backup* dans $ADMIN_{clip}$). Voir aussi [CLIP_DCS_15094].
- ***usersrv*** (*app-clip/clip-useradmin*), démon de gestion des requêtes de changement de mot de passe d'utilisateurs de la cage $USER_{clip}$, en attente de telles requêtes sur la *socket unix /mounts/user_priv/var/run/useradmin* (*/var/run/useradmin* dans $USER_{clip}$). Voir aussi [CLIP_DCS_15093].
- ***usersrvadmin*** (*app-clip/clip-useradmin*), démon de gestion des requêtes de changement de mot de passe d'utilisateurs de la cage $ADMIN_{clip}$, en attente de telles requêtes sur la *socket unix /mounts/admin_priv/var/run/useradmin* (*/var/run/useradmin* dans $ADMIN_{clip}$). Voir aussi [CLIP_DCS_15093].
- ***usersrvaudit*** (*app-clip/clip-useradmin*), démon de gestion des requêtes de changement de mot de passe d'utilisateurs de la cage $AUDIT_{clip}$, en attente de telles requêtes sur la *socket unix /mounts/audit_priv/var/run/useradmin* (*/var/run/useradmin* dans $AUDIT_{clip}$). Voir aussi [CLIP_DCS_15093].
- ***usbadmin_clip*** (*app-clip/clip-usb-keys*), démon de gestion des supports amovibles USB de niveau CLIP, en attente de connexions sur la *socket unix /mounts/user_priv/var/run/usb_clip* (*/var/run/usb_clip* dans $USER_{clip}$). Dans le cas d'un système CLIP-RM uniquement, ce démon est complété par ***usbadmin_rm_h*** et ***usbadmin_rm_b***, pour la gestion des supports amovibles USB de niveaux RM_H et RM_B respectivement, en attente de connexions sur les *sockets unix /mounts/user_priv/var/run/usb_rm_{h,b}*. Voir aussi [CLIP_DCS_15088].
- ***mdadmd*** (*app-clip/clip-mdadm*), démon de gestion des synchronisations RAID, en attente de connexions sur la *socket unix /mounts/admin_priv/var/run/mdadm* (*/var/run/mdadm* dans $ADMIN_{clip}$). Ce démon n'est présent et lancé que dans le cas d'un système CLIP déployé sur un disque dur en configuration RAID, c'est-à-dire uniquement dans les configurations CLIP-GTW à ce jour.
- ***xdm*** (*x11-apps/xdm*), démon d'ouverture de session graphique, dialoguant avec le serveur X11 de la cage X11 sur la *socket unix /tmp/.X11-unix*, et en attente de connexions utilisateur. Voir aussi [CLIP_1302]. Un processus ***xmessage*** accompagne ce démon, uniquement lorsqu'aucune session utilisateur n'est active. Ce client graphique offre deux boutons, permettant respectivement d'éteindre et de redémarrer le poste. Il est terminé automatiquement lors de

l'ouverture d'une session graphique[8], et relancé par *xdm* à la fin de celle-ci[9].

Outre ces démons, plusieurs scripts peuvent être lancés dans le socle dans le cadre de l'administration des utilisateurs et de la gestion des supports USB sécurisés. Ces scripts sont détaillés dans les documents [CLIP_DCS_15093] et [CLIP_DCS_15088].

2 Cage ADMIN$_{clip}$

La cage ADMIN$_{clip}$ permet l'administration locale des paramètres dynamiques d'un poste CLIP. Elle dispose à ce titre d'un accès en écriture à certains montages contenant de tels paramètres. Elle fonctionne uniquement de manière interactive, et est accessible depuis une session ADMIN dans la cage USER$_{clip}$.

2.1 Configuration

2.1.1 Configuration *Vserver*

La configuration de la cage est fournie par l'arborescence */etc/jails/admin*, installée par le paquetage *app-clip/core-services*. Les principaux éléments de cette configuration sont les suivants (cf. [CLIP_1202]) :

- numéro de contexte *vserver* : 503.
- racine : */admin*.
- commande initiale */sbin/sshd*.
- pas d'adresse IP fixe, l'adresse (unique) de la cage est définie à la valeur du paramètre dynamique USER_ADDR (adresse commune de ADMIN$_{clip}$, AUDIT$_{clip}$ et USER$_{clip}$) lors du lancement de la cage.
- Les capacités maximales autorisées dans la cage sont celles généralement autorisées dans les cages CLIP (cf. [CLIP_1202]), complétées de *CAP_NET_BIND_SERVICE* et *CAP_SYS_CHROOT*, qui sont nécessaires au fonctionnement du démon *sshd*, et de *CAP_SYS_TIME*, qui permet de modifier la date et l'heure du système, soit en résumé :
  - *CAP_CHOWN*
  - *CAP_DAC_OVERRIDE*
  - *CAP_DAC_READ_SEARCH*
  - *CAP_FOWNER*

[8] Par le script */usr/local/etc/X11/xdm/GiveConsole*, qui est invoqué automatiquement à chaque ouverture de session.
[9] Par le script */usr/local/etc/X11/xdm/Xsetup_0*, invoqué automatiquement chaque fois que *xdm* relance son interface d'authentification.

- *CAP_FSETID*
- *CAP_KILL*
- *CAP_SETGID*
- *CAP_SETUID*
- *CAP_NET_BIND_SERVICE*
- *CAP_SYS_CHROOT*
- *CAP_SYS_TIME*

- De même, les drapeaux de contexte *vserver* sont les drapeaux par défaut listés dans [CLIP_1202], soit :
  - *INFO_INIT*
  - *HIDE_VINFO*
  - *HIDE_MOUNT*
  - *HIDE_NETIF*
  - *VIRT_CPU*
  - *VIRT_MEM*
  - *VIRT_LOAD*
  - *VIRT_UPTIME*

2.1.2 Montages

Source	Point de montage	Type	Options
/mounts/admin_root	*/*	*bind*	*ro,nodev,noatime*
/lib	*/lib*	*bind*	*ro,nosuid,nodev,noatime*
/usr/lib	*/usr/lib*	*bind*	*ro,nosuid,nodev,noatime*
/usr/local/lib	*/usr/local/lib*	*bind*	*ro,nosuid,nodev,noatime*
/etc/core	*/etc/core*	*bind*	*ro,nosuid,nodev,noexec,noatime*
/etc/shared	*/etc/shared*	*bind*	*ro,nosuid,nodev,noexec,noatime*
/mounts/admin_priv/etc.admin	*/etc/admin*	*bind*	*rw,nosuid,nodev,noexec,noatime*
/mounts/admin_priv/etc.ike2	*/etc/admin/ike2*	*bind*	*rw,nosuid,nodev,noexec,noatime*
/mounts/admin_priv/home	*/home*	*bind*	*rw,nosuid,nodev,noexec,noatime*
/home/adminclip	*/home/admin*	*bind*	*rw,nosuid,nodev,noexec,noatime*
/mounts/admin_priv/var	*/var*	*bind*	*rw,nosuid,nodev,noexec,noatime*
/mounts/admin_priv/dev	*/dev*	*bind*	*ro,nosuid,noexec,noatime*
<admtmp>	*/tmp*	*tmpfs*	*rw,nosuid,nodev,noexec,noatime, mode=1777,size=16m*
/proc	*/proc*	*bind*	*ro,nosuid,nodev,noexec,noatime,nolock*
<none>	*/dev/pts*	*devpts*	*rw,nosuid,noexec,noatime,nolock, gid=5,mode=620*
/dev/mapper/stockage_amovible _<user>_clip	*/mnt/usb* (à la demande)	*vfat*	*rw,nosuid,nodev,noexec,uid=<user>, gid=<user>*

Tableau 3: Montages de la cage ADMIN$_{clip}$.

Les sources sont relatives à la racine du système, les points de montages à la racine de la cage (*/admin*).

Les montages constituant l'arborescence de la cage sont répertoriés dans le Tableau 3. La cage possède sa racine propre montée depuis */mount/admin_root,* qui fournit l'ensemble des exécutables disponibles dans la cage. En revanche, l'ensemble des bibliothèques du socle sont rendues accessibles en lecture seule par les montages de */lib*, */usr/lib* et */usr/local/lib*. Les répertoires */etc/admin* et */etc/admin/ike2*, partagés avec le socle (et avec les autres cages CLIP pour ce qui est du premier) sont montés en lecture-écriture afin de permettre l'administration des paramètres qu'ils contiennent. Les fichiers de */etc/admin* sont normalement propriété de l'utilisateur *_admin* et du groupe *admin,* avec des permissions discrétionnaires *0664*, ce qui permet à l'administrateur local (qui agit pour l'heure uniquement sous le compte *_admin* dans la cage) de modifier directement ces paramètres. La permission en écriture pour le groupe *admin* permet de plus d'envisager à terme l'utilisation de plusieurs comptes d'administration au sein de la cage ADMIN$_{clip}$[10]. En revanche, les fichiers de */etc/admin/ike2* (clés, mots de passe et certificats ACID pour IKE) doivent être propriété de *root*, avec des permissions *0400*, du fait des vérifications effectuées par le démon *iked* (*net-firewall/racoon2*). Ces fichiers ne peuvent être

[10] A la différence du principe mis en oeuvre à ce stade, qui consiste à associer un ou plusieurs comptes d'administration au sein de USER$_{clip}$ à un unique compte d'administration au sein de ADMIN$_{clip}$.

manipulés par l'administrateur local que par l'intermédiaire de l'utilitaire *install_ccsd* (*app-clip/install-ccsd*), qui garantit le respect de ces exigences. Enfin, le répertoire */etc/core* n'est exposé qu'en lecture seule, et le répertoire */etc/tcb* du socle n'est pas accessible dans la cage, ce qui signifie que toutes les opérations d'administration des comptes utilisateurs doivent être réalisées dans le socle, en réponse à une demande issue de la cage $ADMIN_{clip}$ et transmise au démon *usersrvadmin* par l'intermédiaire de la *socket unix /var/run/useradmin* de la cage (cf. 1.2). Enfin, l'import et l'export de paramètres sont permis par la possibilité de monter dans la cage, sur */mnt/usb,* un support USB sécurisé de niveau CLIP.

2.1.3 Configuration *Veriexec*

La cage $ADMIN_{clip}$ se voit associer un contexte *veriexec,* dont le niveau de sécurité est défini avec les drapeaux suivants (cf. [CLIP_1201]) par le script */etc/init.d/veriexec* (*app-clip/veriexec*) :

- *VRXLVL_ACTIVE*
- *VRXLVL_SELF_IMMUTABLE*
- *VRXLVL_LVL_IMMUTABLE*
- *VRXLVL_ENFORCE_MNTRO*

Ces drapeaux interdisent notamment toute opération d'administration de la base *veriexec* depuis la cage.

Le masque de capacités du contexte *veriexec* réduit les capacités attribuables dans la cage par *veriexec* à :

- *CAP_CHOWN*
- *CAP_DAC_OVERRIDE*
- *CAP_SYS_CHROOT*
- *CAP_NET_BIND_SERVICE*
- *CAP_SYS_TIME*

De même, le masque de privilèges CLSM du contexte *veriexec* réduit les privilèges attribuables dans la cage à :

- *CLSM_PRIV_PROCFD*
- *CLSM_PRIV_NETSERVER*

Les entrées *veriexec* suivantes sont définies dans le contexte de la cage :

- */bin/date* se voit attribuer *CAP_SYS_TIME*
- */bin/install_ccsd* se voit attribuer *CAP_CHOWN* et *CAP_DAC_OVERRIDE*
- */sbin/sshd* se voit attribuer *CAP_NET_BIND_SERVICE* et *CAP_SYS_CHROOT*, ainsi que *CLSM_PRIV_NETSERVER*, lorsqu'il est exécuté par *root* uniquement.

2.2 Fichiers

2.2.1 Périphériques et exécutables

La cage dispose de ses fichiers *devices* attitrés, montés depuis */mounts/admin_priv/dev*. Ces fichiers sont limités à :

- *zero, null, full*
- *urandom*, ainsi qu'un lien symbolique *random* pointant sur *urandom*[11]
- *ptmx, tty* et un montage de type *devpts* sur */dev/pts*, fournissant les pseudo-terminaux nécessaires au fonctionnement de *sshd*.
- *socket /dev/log* créée par le démon *syslog-ng* de $AUDIT_{clip}$ (cf. 3)
- répertoire */dev/mapper* permettant la copie temporaire (par un appel *tar* lancé dans le socle, cf. [CLIP_DCS_15088]) d'un *device dm-crypt stockage_amovible_<user>_clip*, correspondant à l'image déchiffrée d'un support amovible sécurisé.

Les exécutables de la cage sont tous propres à celle-ci, et en quasi totalité fournis par un paquetage *busybox* (cf. [BUSYBOX]), dans une configuration spécifique. Cette configuration est automatiquement utilisée lorsque le paquetage *sys-apps/busybox* est compilé avec la variable d'environnement *DEB_NAME_SUFFIX* (cf. [CLIP_1101]) positionnée à *"admin"* (le paquetage *Debian* résultant porte le nom de *busybox-admin*). Ce paquetage installe un unique exécutable, */bin/busybox*, qui apporte les fonctionnalités de différents utilitaires de base UNIX en fonction du nom sous lequel il est appelé, ainsi qu'un ensemble de liens symboliques vers *busybox* correspondant aux noms des différents utilitaires émulés. La configuration *admin* de *busybox* incorpore dans celui-ci un ensemble d'utilitaires nécessaires à la lecture et à la modification interactives de fichiers de configuration, notamment un *shell* compatible *ash* incluant un historique et une complétion automatique des commandes, et un éditeur *vi*. Le paquetage inclut aussi un script spécifique à CLIP, */bin/login_shell.sh*, qui positionne les variables d'environnement LC_ALL et LANG à *"fr_FR"* avant d'appeler */bin/sh*. Ce script est utilisé comme *shell* de *login* de l'utilisateur *_admin* (configuration dans */etc/core/passwd*), et permet à ce dernier de disposer d'une interface partiellement localisée en français. Le *shell busybox* utilise le drapeau *O_MAYEXEC*, spécifique à CLIP ([CLIP_1201]), lors de l'ouverture de fichiers, ce qui interdit l'exécution ou le "sourçage" de scripts depuis des montages portant l'option *noexec*. Seuls quatre exécutables sont installés en complément de *busybox* :

- */bin/date*, installé par le paquetage *sys-apps/coreutils* du socle (qui inclut ainsi deux copies de l'exécutable, */bin/date* et */mounts/admin_root/bin/date*). Un exécutable distinct de *busybox* est dans ce cas nécessaire, afin de pouvoir lui attribuer une entrée *veriexec* propre (cf. [CLIP_1201]).
- */sbin/sshd*, installé par le paquetage *net-misc/openssh* du socle[12], utilisé pour attendre les connexions d'administrateurs au sein de la cage.
- */bin/install_ccsd*, installé par le paquetage *app-clip/install-ccsd*, et spécifique à cette cage. Cet

[11] Le véritable *device random* (source d'aléa "matériel", éventuellement bloquante) n'est jamais exposé dans les cages, afin d'éviter qu'une cage ne puisse épuiser la "réserve d'entropie" de l'ensemble du système.

[12] Paquetage qui inclut une copie de ce démon pour chacune des cages $ADMIN_{clip}$ et $AUDIT_{clip}$, grâce au mécanisme de répertoire d'installation déporté (*CLIP_VROOTS*) de la génération de paquetages CLIP, cf. [CLIP_1101].

utilitaire permet la mise à jour de fichiers ACID utilisés pour l'authentification IKEv2, en leur attribuant les permissions adaptées (cf. 2.2.2).

- */bin/nano*, installé par le paquetage *nano-admin* (*app-editors/nano*) fournit un éditeur de texte plus simple que le *vi busybox*.

2.2.2 Fichiers de configuration

Les fichiers de configuration localement modifiables par l'administrateur sont rassemblés au sein de la cage dans l'arborescence montée sur */etc/admin*. Les fichiers modifiables directement sont attribués à l'utilisateur *_admin*, groupe *admin,* avec des permissions *0640*. Les fichiers de cette arborescence sont les suivants (en chemin relatif par rapport à */etc/admin/;* sauf mention explicite du contraire, les fichiers appartiennent à *_admin*) :

- *rollback-list* : liste des fichiers à conserver lors de l'écrasement d'une configuration par une restauration de configuration sauvegardée. Voir aussi [CLIP_DCS_15094].
- *hosts* : fichier de résolution de noms statique, correspondant à */etc/hosts* pour les cages CLIP (mais pas pour le socle, qui n'emploie aucune résolution de noms).
- *resolv.conf* : fichier de résolution de noms dynamique, correspondant à */etc/resolv.conf* pour les cages CLIP (mais pas pour le socle, qui n'emploie aucune résolution de noms).
- *issue* : message d'avertissement à afficher lors de l'ouverture de session utilisateur. Aucun message n'est affiché lorsque ce fichier est vide. Voir aussi [CLIP_1302].
- *conf.d/hostname* : nom d'hôte du système
- *conf.d/logfiles* : configuration de la rotation des fichiers de journaux, cf. 3.3.1.
- *conf.d/net* : configuration des adresses réseau, cf. [CLIP_1501].
- *conf.d/netfilter* : configuration du filtrage réseau, cf. [CLIP_1501].
- *conf.d/ntp* : configuration *ntp*, cf. 4.2.
- *clip_install/conffiles.list* : liste des fichiers de configuration installés par les paquetages, qui sont automatiquement copiés d'une installation CLIP à l'installation CLIP alternative lors d'une mise à jour du coeur CLIP. Voir aussi [CLIP_1101]. Ce fichier n'est pas inscriptible par l'administrateur local, du fait des permissions discrétionnaires. Il est renseigné de manière automatique par les différents paquetages lors de leur installation.
- *clip_install/backup_list.txt* : liste complémentaires de fichiers à copier d'une installation CLIP à l'installation CLIP alternative lors d'une mise à jour du coeur CLIP. Ce fichier, à la différence de *conffiles.list*, est sous le contrôle de l'administrateur local. Voir aussi [CLIP_DCS_13006].
- *clip_install/clip_install_clip_apps.conf* : paramètres de l'outil d'installation de paquetages (prise en compte des paquetages à fort impact) pour les paquetages secondaires CLIP. Voir aussi [CLIP_DCS_13006].
- *clip_install/clip_install_clip_core.conf* : paramètres de l'outil d'installation de paquetages (prise en compte des paquetages à fort impact) pour les paquetages primaires CLIP. Voir aussi [CLIP_DCS_13006].
- *clip_download/clip_download.conf* : paramètres généraux de l'outil de téléchargement de mises

à jour, en particulier pour l'activation du téléchargement initial au démarrage, cf. 4.2.

- *clip_download/clip_download_clip.conf* : paramètres de l'outil de téléchargement de mises à jour pour le téléchargement des paquetages CLIP. Voir aussi [CLIP_DCS_14009]. Dans le cas d'un système CLIP-RM, ce fichier est complété d'un fichier *clip_download/clip_download_rm_X.conf* pour chaque cage RM_X.
- *clip_download/sources.list.clip* : définition des miroirs distants pour le téléchargement des paquetages CLIP. Voir aussi [CLIP_DCS_14009]. Dans le cas d'un système CLIP-RM, ce fichier est complété d'un fichier *clip_download/sources.list.rm_X* pour chaque cage RM_X.
- *clip/download/cert/cacert.pem* : certificat de l'autorité racine certifiant le serveur HTTPS de téléchargement de mises à jour. Ce fichier peut être modifié directement par l'administrateur local, mais sa mise à jour nécessite la mise à jour d'un lien symbolique dans le même répertoire, qui peut être réalisée en appelant le script *install_cert*. Voir aussi [CLIP_DCS_14009].
- *ike2/cert/* : répertoire des clés ACID (et mots de passes associés) d'authentification IKEv2 modifiables par l'administrateur local. Voir aussi [CLIP_1501]. Ce répertoire contient au moins deux fichiers, *ccsd.pvr* et *ccsd.pwd,* contenant respectivement la clé privée du poste et son mot de passe. Par ailleurs, le répertoire contient systématiquement un fichier *lock*, utilisé uniquement pour la synchronisation des accès aux fichiers par *racoon2*. Les fichiers de ce répertoire doivent tous appartenir à *root*, et n'être accessibles que par ce dernier. De ce fait, leur mise à jour par l'administrateur local ne peut être faite qu'à l'aide de la commande *install_ccsd*, qui se voit attribuer les privilèges nécessaires par *veriexec*.

2.3 Démarrage et fonctionnement de la cage

La cage est configurée et démarrée par le script de démarrage */etc/init.d/clip_admin*, invoqué après la configuration des interfaces réseau et le verrouillage du système (cf. [CLIP_1301]). Ce script importe de manière sécurisée les variables *USER_ADDR* et *USER_MASK* depuis le fichier */etc/admin/conf.d/net* , sauf lorsqu'une configuration réseau sans échec (cf. [CLIP_1501]) est détectée à la présence d'un fichier */var/run/nonetwork.* Dans ce dernier cas, aucune variable n'est importée et l'adresse de la cage est fixée à *127.0.0.1/8*.

La cage est ensuite démarrée par une simple commande *vsctl -a <adresse> admin start* (cf. [CLIP_1202]), ce qui lance dans la cage un unique démon *sshd*. Ce dernier est configuré (dans */usr/local/etc/ssh/sshd_config* dans l'arborescence de la cage) de la manière suivante :

- Écoute sur le port 22, toutes adresses confondues (l'adresse d'écoute étant de toutes manières fixée par la configuration de la cage).
- Protocole SSHv2 uniquement. Les clés d'authentification du serveur sont générées à la première installation du paquetage *net-misc/openssh,* et stockées dans */usr/local/etc/ssh/* au sein de l'arborescence de la cage (montage en lecture seule). Une mesure spécifique appliquée lors de l'installation du poste garantit que les clés d'authentification du serveur sont les mêmes pour les deux installations CLIP normalement présentes sur le poste.
- Authentification limitée au compte *_admin*, et uniquement à l'aide de la méthode *PubkeyAuthentication* (cf. [CLIP_1302]), les clés publiques autorisées étant stocké dans */home/admin/.ssh/authorized_keys* dans l'arborescence de la cage. Le répertoire */home/admin* de la cage étant monté depuis le */home/adminclip* du socle, ces clés sont partagées entre les deux

installations CLIP du poste.

- Mise en oeuvre de la séparation de privilèges. Ce mode de fonctionnement est détaillé en [PRIVSEP] et [PRIV].

Après une authentification réussie sous le compte *_admin*, différentes opérations d'administration sont possibles :

- Soit en modifiant directement des fichiers de */etc/admin/*. Les références [CLIP_DCS_15093], [CLIP_DCS_13006], [CLIP_DCS_14009], [CLIP_1501] et [CLIP_1301] décrivent ces différents paramètres.
- Soit en invoquant des exécutables privilégiés au sein de la cage : *install_ccsd* pour l'installation ou la désinstallation de fichiers ACID pour IKEv2 dans */etc/admin/ike2/cert* (cf. [CLIP_1501]), et *date* pour la modification directe de l'heure ou de la date.
- Soit enfin en invoquant un client dialoguant avec un démon du socle (cf.) pour y faire réaliser une opération plus privilégiée :
  - *userclt*, pour dialoguer avec *usersrvadmin* et gérér les comptes utilisateurs ([CLIP_DCS_15093]).
  - *downloadrequest,* pour dialoguer avec *downloadmaster* et gérer les téléchargements de mises à jour ([CLIP_DCS_14009]).
  - *backupclt* et *databackupclt* pour dialoguer respectivement avec *backupsrv* et *databackupsrv* et piloter les opérations de sauvegardes / restaurations du système et des données ([CLIP_DCS_15094]).
  - *mdamclt*, pour dialoguer avec *mdamd* et gérer les synchronisations RAID ([CLIP_DCS_15093]), sur une configuration dotée de tels disques uniquement (CLIP-GTW à ce stade).

En revanche, aucune administration des supports USB sécurisés n'est possible directement depuis $ADMIN_{clip}$. Cette administration doit être réalisée depuis la session $USER_{clip}$ associée à toute session dans $ADMIN_{clip}$ (cf. 5).

L'arrêt de la cage est réalisé par un appel *vsctl admin stop*.

3 Cage AUDIT$_{clip}$

La cage AUDIT$_{clip}$ a un double rôle de collecte centralisée des journaux du système d'une part, et de consultation de ces journaux d'autre part. Elle dispose à ce titre d'un accès en lecture-écriture à la partition de journaux, montée en */var/log* dans le socle et en */log* dans la cage. Elle fonctionne à la fois de manière non interactive (pour la collecte de journaux) et de manière interactive (pour leur consultation), et est accessible depuis une session AUDIT dans la cage USER$_{clip}$.

3.1 Configuration

3.1.1 Configuration *Vserver*

La configuration de la cage est fournie par l'arborescence */etc/jails/audit*, installée par le paquetage *app-clip/core-services*. Les principaux éléments de cette configuration sont les suivants :

- numéro de contexte *vserver* : 504.
- racine : */audit.*
- commande initiale non définie.
- pas d'adresse IP fixe, l'adresse (unique) de la cage est définie à la valeur du paramètre dynamique USER_ADDR (adresse commune de ADMIN$_{clip}$, AUDIT$_{clip}$ et USER$_{clip}$) lors du lancement de la cage.
- Les capacités maximales autorisées dans la cage sont celles généralement autorisées dans les cages CLIP (cf. [CLIP_1202]), complétées de *CAP_NET_BIND_SERVICE* et *CAP_SYS_CHROOT*, qui sont nécessaires au fonctionnement du démon *sshd*, soit en résumé :
  - *CAP_CHOWN*
  - *CAP_DAC_OVERRIDE*
  - *CAP_DAC_READ_SEARCH*
  - *CAP_FOWNER*
  - *CAP_FSETID*
  - *CAP_KILL*
  - *CAP_SETGID*
  - *CAP_SETUID*
  - *CAP_NET_BIND_SERVICE*
  - *CAP_SYS_CHROOT*
- Les drapeaux de contexte *vserver* sont les drapeaux par défaut listés dans [CLIP_1202].

3.1.2 Montages

Les montages constituant l'arborescence de la cage sont répertoriés dans le Tableau 4. La cage possède sa racine propre montée depuis */mounts/audit_root,* qui fournit l'ensemble des exécutables disponibles dans la cage. En revanche, l'ensemble des bibliothèques du socle sont rendues accessibles en lecture seule par les montages de */lib*, */usr/lib* et */usr/local/lib*. La partition de journaux est montée en lecture-écriture sous */log*. L'export des journaux est permis par la possibilité de monter dans la cage, sur */mnt/usb,* un support USB sécurisé de niveau CLIP.

Source	Point de montage	Type	Options
/mounts/audit_root	*/*	*bind*	*ro,nodev,noatime*
/lib	*/lib*	*bind*	*ro,nosuid,nodev,noatime*
/usr/lib	*/usr/lib*	*bind*	*ro,nosuid,nodev,noatime*
/usr/local/lib	*/usr/local/lib*	*bind*	*ro,nosuid,nodev,noatime*
/etc/core	*/etc/core*	*bind*	*ro,nosuid,nodev,noexec,noatime*
/etc/shared	*/etc/shared*	*bind*	*ro,nosuid,nodev,noexec,noatime*
/mounts/admin_priv/etc.admin	*/etc/admin*	*bind*	*ro,nosuid,nodev,noexec,noatime*
/var/log	*/log*	*bind*	*ro,nosuid,nodev,noexec,noatime*
/mounts/audit_priv/home	*/home*	*bind*	*rw,nosuid,nodev,noexec,noatime*
/home/auditclip	*/home/audit*	*bind*	*rw,nosuid,nodev,noexec,noatime*
/mounts/audit_priv/var	*/var*	*bind*	*rw,nosuid,nodev,noexec,noatime*
/mounts/audit_priv/dev	*/dev*	*bind*	*ro,nosuid,noexec,noatime*
<audtmp>	*/tmp*	*tmpfs*	*rw,nosuid,nodev,noexec,noatime, mode=1777,size=16m*
/proc	*/proc*	*bind*	*ro,nosuid,nodev,noexec,noatime,nolock*
<none>	*/dev/pts*	*devpts*	*rw,nosuid,noexec,noatime,nolock, gid=5,mode=620*
/dev/mapper/stockage_amovible _<user>_clip	*/mnt/usb* (à la demande)	*vfat*	*rw,nosuid,nodev,noexec,uid=<user>, gid=<user>*

Tableau 4: Montages de la cage AUDIT$_{clip}$.

Les sources sont relatives à la racine du système, les points de montages à la racine de la cage (*/audit*).

3.1.3 Configuration *Veriexec*

La cage AUDIT$_{clip}$ se voit associer un contexte *veriexec,* dont le niveau de sécurité est défini avec les drapeaux suivants (cf. [CLIP_1201]) par le script */etc/init.d/veriexec* (*app-clip/veriexec*) :

- *VRXLVL_ACTIVE*
- *VRXLVL_SELF_IMMUTABLE*
- *VRXLVL_LVL_IMMUTABLE*

- *VRXLVL_ENFORCE_MNTRO*

Ces drapeaux interdisent notamment toute opération d'administration de la base *veriexec* depuis la cage.

Le masque de capacités du contexte *veriexec* réduit les capacités attribuables dans la cage par *veriexec* à :

- *CAP_SYS_CHROOT*
- *CAP_NET_BIND_SERVICE*

De même, le masque de privilèges CLSM du contexte *veriexec* réduit les privilèges attribuables dans la cage à :

- *CLSM_PRIV_PROCFD*
- *CLSM_PRIV_NETSERVER*

Les entrées *veriexec* suivantes sont définies dans le contexte de la cage :

- */sbin/sshd* se voit attribuer *CAP_NET_BIND_SERVICE* et *CAP_SYS_CHROOT*, ainsi que *CLSM_PRIV_NETSERVER*, lorsqu'il est exécuté par *root* uniquement.

3.2 Fichiers

3.2.1 Périphériques et exécutables

La cage dispose de ses fichiers *devices* attitrés, montés depuis */mounts/audit_priv/dev*. Ces fichiers sont limités à :

- *zero, null, full*
- *urandom*, ainsi qu'un lien symbolique *random* pointant sur *urandom*
- *ptmx, tty* et un montage de type *devpts* sur */dev/pts*, fournissant les pseudo-terminaux nécessaires au fonctionnement de *sshd*.
- *socket /dev/log* créée par le démon *syslog-ng* de la cage.
- répertoire */dev/mapper* permettant la copie temporaire (par un appel *tar* lancé dans le socle, cf. [CLIP_DCS_15088]) d'un *device dm-crypt stockage_amovible_<user>_clip*, correspondant à l'image déchiffrée d'un support amovible sécurisé.

Les exécutables de la cage sont tous spécifiques à celle-ci, et en quasi totalité fournis par un paquetage *busybox* (cf. [BUSYBOX]), dans une configuration spécifique. Cette configuration est automatiquement utilisée lorsque le paquetage *sys-apps/busybox* est compilé avec la variable d'environnement *DEB_NAME_SUFFIX* (cf. [CLIP_1101]) positionnée à *"audit"* (le paquetage *Debian* résultant porte le nom de *busybox-audit*). La configuration *audit* de *busybox* incorpore dans celui-ci un ensemble d'utilitaires nécessaires à la lecture et à l'analyse de fichiers de journaux, notamment un *shell* compatible *ash* incluant un historique et une complétion automatique des commandes, un éditeur *vi* et les utilitaires *grep, awk* et *sed.* Le paquetage inclut aussi un script spécifique à CLIP, */bin/login_shell.sh*, qui positionne les variables d'environnement LC_ALL et LANG à *"fr_FR"* avant d'appeler */bin/sh.* Ce script est utilisé comme *shell* de *login* de l'utilisateur *_audit* (configuration dans

/etc/core/passwd), et permet à ce dernier de disposer d'une interface partiellement localisée en français. Le *shell busybox* utilise le drapeau *O_MAYEXEC*, spécifique à CLIP ([CLIP_1201]), lors de l'ouverture de fichiers, ce qui interdit l'exécution ou le "sourçage" de scripts depuis des montages portant l'option *noexec*. Un seul exécutable est installé en complément de *busybox* : */sbin/sshd*, installé par le paquetage *net-misc/openssh* du socle[13], et utilisé pour attendre les connexions d'auditeurs au sein de la cage.

3.2.2 Fichiers de journaux

Les journaux sont répartis entre plusieurs fichiers prédéfinis de */log*, en fonction du type d'événement auquel ils se rapportent et de leur origine. Ces fichiers sont les suivants :

- *audit.log* : journaux du sous-système d'audit du noyau, non utilisé à ce jour.
- *auth.log* : journaux des actions d'authentification, contenant notamment les succès et échecs des traitements par les modules PAM *pam_tcb* et *pam_exec_pwd*, ainsi que ceux des démons *sshd* des cages $ADMIN_{clip}$ et $AUDIT_{clip}$, et du démon *pwcheckd* utilisé pour déverrouiller les sessions graphiques dans $USER_{clip}$ (cf. [CLIP_1302]).
- *clsm.log* : journaux du LSM CLIP (cf. [CLIP_1201]), issus du noyau et identifiés par leur préfixe *"CLSM:"*.
- *cron.log* : journaux du démon *cron* de $UPDATE_{clip}$ (cf. [CLIP_DCS_13006] et [CLIP_DCS_14009]).
- *daemon.log* : journaux des différents démons du système, notamment :
  - *iked* et *spmd* : établissement et expiration d'associations et de politiques de sécurité IPsec, erreurs de négociation IKEv2, cf. [CLIP_1502] .
  - *pwcheckd* : authentifications ou échecs d'authentification des utilisateurs.
  - *xdm* : ouvertures et fermetures de sessions.
  - *usbadmin_*, useradmin*, databackupd, backupd* et *mdadmd* : journaux des erreurs de traitement.
- *debug* : journaux utilisateur de niveau de priorité *debug*.
- *fw.log* : journaux des paquets rejetés par le pare-feu réseau local (cf. [CLIP_1501]).
- *grsec.log* : journaux *Grsecurity*, hors opérations de montage (cf. [CLIP_1203]).
- *grsec_mount.log* : journaux *Grsecurity* des opérations de montage et de démontage VFS (cf. [CLIP_1203]).
- *kern.log* : ensemble des journaux du noyau.
- *mail.log* : journaux d'un éventuel serveur de *mail* local. Non utilisé à ce stade.
- *messages* : journaux divers du système, notamment :
  - opérations de téléchargement réussies ou échouées dans $UPDATE_{clip}$ (journaux *clip-download,* cf. [CLIP_DCS_14009]).

[13] Paquetage qui inclut une copie de ce démon pour chacune des cages $ADMIN_{clip}$ et $AUDIT_{clip}$, grâce au mécanisme de répertoire d'installation déporté (*CLIP_VROOTS*) de la génération de paquetages CLIP, cf. [CLIP_1101].

  - opérations de mise à jour des paquetages primaires et secondaires de la distribution CLIP (respectivement dans le socle et dans $UPDATE_{clip}$, cf. [CLIP_DCS_13006]).
  - journaux des commandes lancées à l'ouverture et à la fermeture de session utilisateur par *pam_exec_pwd* (cf. [CLIP_1302]).
- *pax.log* : journaux des violations PaX (cf. [CLIP_1203]).
- *ssp.log* : journaux des violations SSP (cf. [CLIP_1101]) dans le socle et les cages CLIP uniquement.
- *syslog* : journaux internes de *syslog* (non utilisé à ce stade).
- *user.log* : journaux de la *facility syslog "user"*, en particulier ceux des scripts *hotplug* associés à la gestion de supports amovibles sécurisés (cf. [CLIP_DCS_15088])
- *veriexec.log* : journaux *veriexec* (échecs de vérification, cf. [CLIP_1201])
- *vserver.log* : journaux *vserver* (avertissements concernant notamment les opérations bloquées par *vserver*, cf. [CLIP_1202])

S'y ajoutent, dans le cas d'un système CLIP-RM, les fichiers suivants (pour chaque cage RM *rm_X*) :

- *rm_X_auth.log* : journaux d'authentification au sein de la cage RM_X, contenant principalement les authentifications réussies ou échouées vis-à-vis du démon *sshd* de la vue ADMIN de la cage.
- *rm_X_cron.log* : journaux du démon *crond* de la vue UPDATE de la cage.
- *rm_X_daemon.log* : journaux des autres démons de la cage, composés principalement des messages d'ouverture de session du démon *jailmaster*.
- *rm_X_messages.log* : autres journaux de la cage, notamment ceux des mises à jour de la cage.
- *rm_X_ssp.log* : violations SSP au sein de la cage.

Les différentes opérations correspondantes sont documentées dans [CLIP_1401].

Ces fichiers, renseignés par le démon *syslog-ng*, portent tous l'attribut *APPEND_ONLY*, qui interdit leur suppression et la modification de leur contenu (hors ajout en fin de fichier), et ce y compris à *root* du socle une fois le système verrouillé, dans la mesure où la capacité *CAP_LINUX_IMMUTABLE* est globalement masquée au sein du système. Ces fichiers sont par ailleurs complétés par deux autres fichiers, générés directement :

- *dmesg* : contient les journaux noyau durant la phase de démarrage. Généré par le script de démarrage */etc/init.d/logfiles*. Ce fichier se voit affecter un attribut *IMMUTABLE* après sa création, ce qui interdit sa modification ou sa suppression avant l'arrêt du système.
- *xdm.log* : contient la sortie standard du démon *xdm* et des commandes qu'il lance (par exemple, commandes lancées par le module PAM *pam_exec_pwd*). Les messages du démon sont aussi transmis au démon *syslog-ng*, et sauvegardés dans le fichier *daemon.log*. Cette journalisation *syslog* par *xdm* résulte d'un patch spécifique à CLIP appliqué à ce démon. Le fichier *xdm.log* ne porte aucun attribut spécifique.

L'ensemble de ces fichiers est créé avec l'utilisateur *syslog* et le groupe *syslog* comme propriétaires, et les permissions en lecture-écriture pour l'utilisateur et en lecture seule pour le groupe (*0640*), ce qui permet au démon *syslog-ng* d'écrire dans ces fichiers (en ajout uniquement), et à l'utilisateur *_audit*, utilisé pour la consultation de ces journaux et membre du groupe *syslog*, de lire leur contenu.

Enfin, le répertoire */log/keep* contient les anciennes versions de fichiers de journalisation, archivées lors de la rotation des journaux au démarrage (cf. 3.3). Seuls les fichiers produits par *syslog-ng* et le fichier de journaux noyau (*dmesg*) sont ainsi archivés. Un nombre configurable d'archives est conservé, sous forme compressée par *gzip*, pour chaque type de fichier. Les archives sont numérotées, sous la forme *<fichier>.gz.<numéro>,* de 1 au nombre maximal d'archives conservées, et de la plus récente à la plus ancienne. Les archives locales sont toutes protégées par un attribut IMMUTABLE.

Remarque 1 : journaux du serveur X11

Le serveur X11 d'un système CLIP ne journalise pas à ce stade ses messages à travers l'interface SYSLOG. Les seuls journaux du serveur sont constitués par sa sortie standard, stockée dans le fichier /var/log/Xorg.0.log *au sein de la cage X11. Ce fichier étant propre à la cage, il n'est pas consultable depuis* $AUDIT_{clip}$*. Il serait souhaitable à terme de modifier le serveur X11 de manière similaire au démon xdm, de manière à lui faire journaliser ses messages sur l'interface SYSLOG.*

3.3 Démarrage et fonctionnement de la cage

Le démarrage de la cage $AUDIT_{clip}$ est assuré par trois scripts de démarrage successifs : *logfiles* (*softlevel boot*, cf. [CLIP_1301]) assure la rotation des fichiers de journaux, *clip_audit* (*softlevel nonetwork*) crée la cage et lance la collecte des journaux, et *clip_sshd* (*softlevel default*) démarre le démon *sshd* de la cage afin de permettre des sessions de consultation de ces journaux. Cette séparation en plusieurs scripts est rendue nécessaire par le verrouillage progressif du système. La rotation des journaux doit être réalisée avant le masquage de la capacité *CAP_LINUX_IMMUTABLE* dans tout le système, car cette capacité est nécessaire pour supprimer les archives de journaux protégées par des attributs IMMUTABLE. En revanche, la cage elle-même ne doit être démarrée qu'après ce masquage, afin d'éviter de lancer un démon disposant de cette capacité. Cependant, ce lancement de la cage doit être réalisé suffisamment tôt dans la séquence de démarrage, afin de disposer d'une fonction de journalisation lors du lancement des scripts suivants. En particulier, la journalisation doit être activée avant le filtrage réseau (qui est susceptible de produire de nombreux messages de journalisation), et donc avant l'activation des interfaces réseau. Le démon *sshd* ne peut alors pas encore être lancé dans la cage, du fait de l'impossibilité de se mettre en écoute sur le réseau avant l'activation d'au moins une interface réseau.

3.3.1 Rotation des journaux

Le script */etc/init.d/logfiles* (*clip-layout/baselayout-clip*) est lancé avant le masquage de la capacité *CAP_LINUX_IMMUTABLE* par */etc/init.d/sysctl*. Il assure la création et la rotation éventuelles des fichiers de journaux décrits en 3.2 (à l'exception de *xdm.log*, qui est créé directement au lancement du démon *xdm*), et le positionnement correct des attributs et permissions UNIX.

Le script importe en premier lieu deux variables depuis le fichier de configuration */etc/admin/conf.d/logfiles* :

- *KEEP_FILES_NUMBER* : définit le nombre de fichiers d'archives à conserver pour chaque fichier dans */log/keep*. Cette variable doit être définie comme un entier entre 5 et 99. En cas d'échec de l'import (y compris dans le cas d'une variable sortant des valeurs autorisées), la

valeur par défaut (5) est utilisée.

- *KEEP_SIZE* : définit la taille (en octets) au dessus de laquelle un fichier est susceptible d'être archivé. Cette variable doit être définie comme un entier entre 4096 et 9 999 999. En cas d'échec de l'import (y compris dans le cas d'une variable sortant des valeurs autorisées), la valeur par défaut (4096) est utilisée.

Le fichier */etc/admin/conf.d/logfiles* étant sous le contrôle de l'administrateur local du poste, sa lecture est réalisée à l'aide des fonctions d'import sécurisé de */lib/clip/import.sub* (cf. [CLIP_1301]).

Une fois ces deux variables définies, le script teste chacun des fichiers de journaux prédéfinis (hors fichier *dmesg*). Lorsqu'un fichier existe et dépasse la taille KEEP_SIZE, les archives existantes du fichier sont renumérotées, en supprimant la version la plus ancienne si elle dépasse KEEP_FILES_NUMBER, de la manière suivante :

```
<fichier>.gz.N          =>    supprimé si ( N >= KEEP_FILES_NUMBER )
ou
<fichier>.gz.N          =>    <fichier>.gz.(N+1) si ( N < KEEP_FILES_NUMBER)
<fichier>.gz.(N-1)      =>    <fichier>.gz.N
...
<fichier>.gz.1          =>    <fichier>.gz.2
```

Chaque renommage nécessite de supprimer au préalable l'attribut IMMUTABLE du fichier, et de le re-positionner à l'issue. Le fichier non compressé existant est ensuite compressé en *<fichier>.gz.1*, sur lequel sont positionnés les permissions et attributs appropriés (*syslog:syslog,* mode *0640*, attribut IMMUTABLE). Enfin, le fichier non compressé est supprimé (en retirant au préalable son attribut APPEND_ONLY), et un nouveau fichier vide est créé avec les permissions adaptées et l'attribut IMMUTABLE.

Le fichier *dmesg* subit un traitement similaire, à ceci près qu'il est systématiquement archivé (quelle que soit sa taille), et régénéré par un appel à la commande *dmesg*, et qu'il porte directement l'attribut IMMUTABLE plutôt que APPEND_ONLY.

3.3.2 Collecte des journaux

La cage $AUDIT_{clip}$ est créée et démarrée par le script *clip_audit* (*app-clip/core-services*), à partir de la configuration présente dans */etc/jails/audit*. Seul le démon de collecte de journaux, *syslog-ng*, est lancé dans la cage à cette étape du démarrage. Ce démon doit créer une socket de collecte dans le socle, et dans chaque cage CLIP du système, ainsi que dans les éventuelles cages RM. Ces *sockets* doivent être créées dans les arborescences de leurs cages respectives, plutôt que dans celle de la cage $AUDIT_{clip}$. Le démon *syslog-ng* doit de plus ouvrir le fichier */proc/kmsg*, qui n'est normalement exposé dans aucune cage, afin d'y lire les journaux du noyau. Afin de ne pas exposer ces différents fichiers directement dans la cage $AUDIT_{clip}$, l'approche adoptée consiste à démarrer le démon *syslog-ng* hors de la cage, et à le faire s'enfermer dans celle-ci après l'ouverture des différents descripteurs de fichiers sur lesquels il effectue sa collecte. Le paquetage *app-admin/syslog-ng* est ainsi patché de manière spécifique à CLIP, de telle sorte que le démon soit lié à la bibliothèque *clip-libs/clip-libvserver* (cf. [CLIP_1202]) et accepte un argument supplémentaire sur sa ligne de commande, *-x <xid>*. Quand cet argument est présent, le démon fait appel aux fonctions de *libclipvserver.so* pour s'enfermer dans la cage d'identifiant numérique *<xid>* (qui doit avoir été créée et maintenue active par ailleurs) lors de son démarrage, après avoir ouvert ses différentes sources de collecte, mais avant d'ouvrir ses sorties (fichiers de

journaux, *sockets* réseau, etc.).

Une autre problématique spécifique au script *clip_audit* est celle de la prise en compte du mode sans échec de la configuration réseau (cf. [CLIP_1501]). Dans la mesure où ce script est lancé avant tous les scripts de configuration réseau, il ne peut pas, à la différence de *clip_admin*, adapter son comportement et modifier l'adresse de la cage $AUDIT_{clip}$ en cas de présence du fichier */var/run/nonetwork*. Afin de maintenir un accès à la consultation des journaux dans le cas d'une configuration réseau sans échec, il est donc nécessaire de configurer systématiquement la cage pour qu'elle soit joignable depuis l'adresse *127.0.0.1/8*, qui est celle donnée à la cage $USER_{clip}$ en mode sans échec. De ce fait, la cage est systématiquement configurée avec l'adresse *127.0.0.1/8*, à laquelle s'ajoute[14] l'adresse *USER_ADDR* si celle-ci a pu être importée. Dans la mesure où l'adresse *127.0.0.1/8* n'est accessible à aucune cage en dehors du mode sans échec, cette approche n'introduit pas d'accès supplémentaire à la cage $AUDIT_{clip}$ en mode nominal.

Ces contraintes étant établies, le traitement réalisé par le script *clip_audit* est le suivant :

- Import sécurisé des variables *USER_ADDR* et *USER_MASK* depuis le fichier de configuration */etc/admin/conf.d/net*. En cas d'échec de cet import, un message d'erreur est affiché sur la console mais l'exécution se poursuit.
- Génération d'un *cookie vsctl* (cf. [CLIP_1202]) par la commande *vsctl audit cookie*.
- Configuration de la cage par une commande *vsctl audit setup*, en utilisant le *cookie* précédemment généré. Cette commande attribue à la cage l'adresse *127.0.0.1/8* (argument *-a 127.0.0.1/255.0.0.0*), et lorsque *USER_ADDR* et *USER_MASK* ont pu être importés, la deuxième adresse correspondant à ces variables, par un deuxième argument *-a <adresse>*. La commande *vsctl setup* maintient la cage active le temps de lancer *syslog-ng*.
- Lancement de *syslog-ng* sous l'identité *root* dans le socle, avec l'argument *-x 504*, pour le forcer à s'enfermer dans la cage nouvellement créée, et les arguments *-u syslog* et *-g syslog*, pour le forcer à s'exécuter, après son enfermement, sous l'identité *syslog:syslog*. Le démon ouvre ses *sockets* de collecte de journaux dans le socle et les cages du système avant de s'enfermer dans $AUDIT_{clip}$. Cette séquence de démarrage est résumée dans la Figure 2.
- Terminaison du processus *vsctl setup* dans la cage, par une commande *vsctl endsetup* utilisant le même *cookie* pour son authentification.

Le démon *syslog-ng* se voit attribuer les privilèges suivants (dans le socle, et non dans la cage) par une entrée *veriexec* :

- Privilège CLSM *CLSM_PRIV_KSYSLOG*, permettant au démon de lire le tampon de journaux noyau sans disposer de la capacité *CAP_SYS_ADMIN*, et depuis un contexte *vserver* différent de ADMIN.
- Privilège CLSM *CLSM_PRIV_IMMORTAL*, qui protège le démon contre tout signal en dehors de la séquence d'arrêt du système, et le protège contre une terminaison par le *out of memory killer* du noyau, afin de maintenir la disponibilité du service de collecte des journaux.
- Privilège CLSM *CLSM_PRIV_KEEPPRIV*, permettant au démon de conserver les deux privilèges ci-dessus (mais pas des capacités POSIX) lors de son enfermement dans la cage puis

[14] On se souviendra qu'une cage CLIP peut se voir attribuer jusqu'à quatre adresses IP distinctes. La premières de ces adresses est celle qui est utilisée en substitut de *INADDR_LOOP* et *INADRR_ANY*. Cf. [CLIP_1202].

de son changement d'identité.

En revanche, cette entrée *veriexec* n'attribue aucune capacité POSIX spécifique au démon. Le script *clip_audit* étant normalement invoqué avant *reducecap*, le démon disposera automatiquement lors d'un lancement nominal des capacités *CAP_SYS_ADMIN* et *CAP_CONTEXT* nécessaires au changement de contexte *vserver*. Ces capacités ne seront par contre plus disponibles une fois le système verrouillé (cf. [CLIP_1301]).

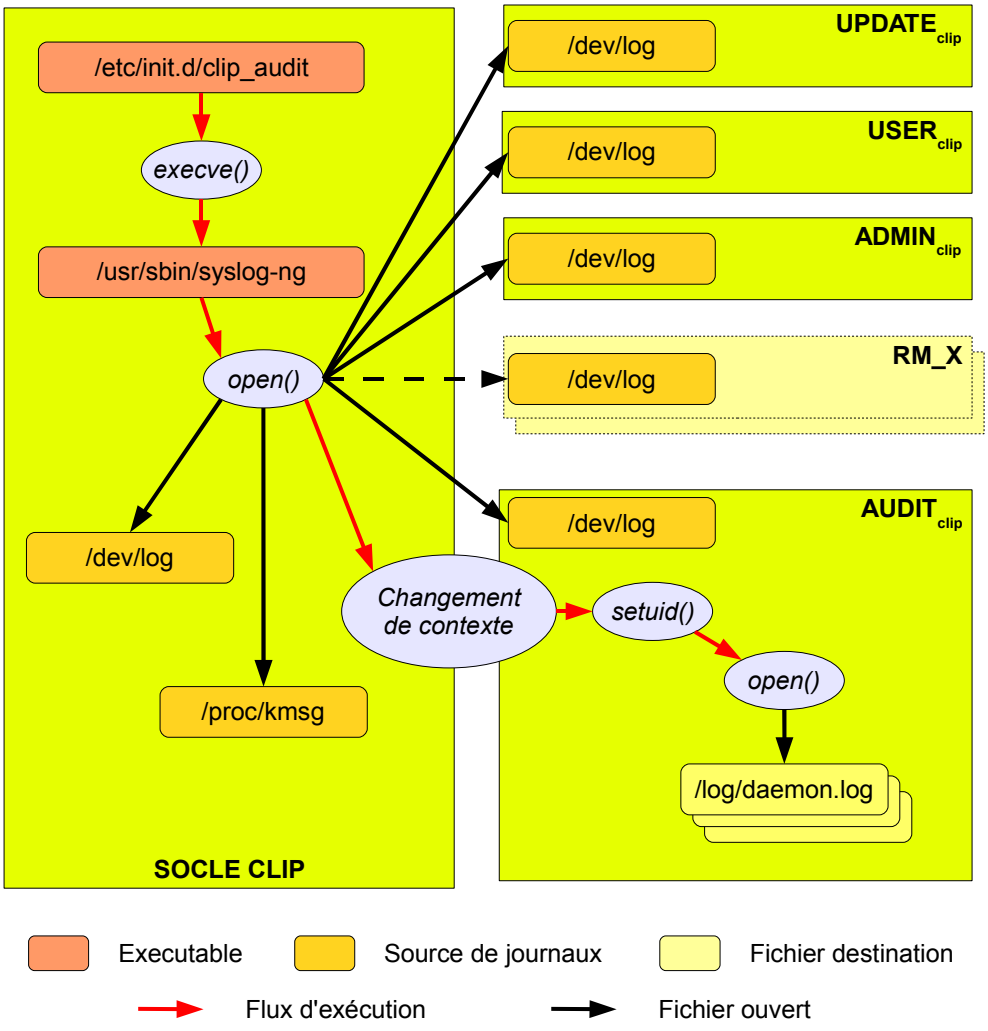

Figure 2: Séquence de démarrage du démon de collecte de journaux CLIP.

Remarque 2 : lancement d'un deuxième démon* syslog-ng *dans le socle

La non attribution des capacités CAP_SYS_ADMIN et CAP_CONTEXT n'interdit

cependant pas à un attaquant, qui parviendrait à lancer des commandes arbitraires sous l'identité root *dans le socle, le lancement d'un deuxième démon* syslog-ng *dans le socle afin de priver le démon nominal de journaux. Faute de pouvoir tuer ce dernier, le démon "parasite" pourrait créer des sockets de collecte par dessus les sockets nominales, et tenter de gagner une* race-condition *pour lire plus vite que le démon nominal le fichier* /proc/kmsg. *Cependant, l'ouverture de* /proc/kmsg, *contrairement à sa lecture, n'est pas possible avec le seul privilège CLSM_PRIV_KSYSLOG (sans CAP_SYS_ADMIN). De ce fait, un démon* syslog-ng *parasite, lancé après le verrouillage du système, ne pourrait pas priver le démon légitime de son accès aux journaux noyaux. Il serait néanmoins souhaitable de mettre en place à terme une protection comparable pour les autres sources de journaux.*

3.3.3 Sessions de consultation

Le script de démarrage *clip_sshd* (*app-clip/core-services*) assure le lancement d'un démon *sshd* dans la cage $AUDIT_{clip}$, après l'initialisation des interfaces réseau. Ce lancement est effectué par une simple commande *vsctl audit enter -- /sbin/sshd*, la cage *audit* étant déjà active à ce stade du démarrage.

Le démon *sshd* est configuré (dans */usr/local/etc/ssh/sshd_config* dans l'arborescence de la cage) de la manière suivante :

- Écoute sur le port 23, toutes adresses confondues (l'adresse d'écoute étant de toutes manières fixée par la configuration de la cage).
- Protocole SSHv2 uniquement. Les clés d'authentification du serveur sont générées à la première installation du paquetage *net-misc/openssh,* et stockées dans */usr/local/etc/ssh/* au sein de l'arborescence de la cage (montage en lecture seule). Une mesure spécifique appliquée lors de l'installation du poste garantit que les clés d'authentification du serveur sont les mêmes pour les deux installations CLIP du poste.
- Authentification limitée au compte *_audit*, et uniquement à l'aide de la méthode *PubkeyAuthentication* (cf. [CLIP_1302]), les clés publiques autorisées étant stocké dans */home/audit/.ssh/authorized_keys* dans l'arborescence de la cage. Le répertoire */home/audit* de la cage étant monté depuis le */home/auditclip* du socle, ces clés sont partagées entre les deux installations CLIP normalement présentes sur le poste.
- Mise en oeuvre de la séparation de privilèges. Ce mode de fonctionnement est détaillé en [PRIVSEP] et [PRIV].

Après une authentification réussie sous le compte *_audit*, la consultation des journaux est possible à l'aide des utilitaires standard disponibles dans la cage, notamment *vi* et *less.* Par ailleurs, le changement du mot de passe de l'utilisateur courant (utilisateur de la session $USER_{clip}$ ayant permis l'ouverture d'une session AUDIT) est possible depuis la cage $AUDIT_{clip}$, en invoquant le client *userclt* pour dialoguer avec le démon *usersrvaudit* sur la *socket unix /var/run/useradmin*. Ce démon, contrairement au démon *usersrv* joignable depuis la cage $ADMIN_{clip}$, ne permet que la modification du mot de passe de l'utilisateur courant, et non les autres opérations d'administration des comptes utilisateurs (cf. [CLIP_DCS_15093]). En revanche, aucune administration des supports USB sécurisés n'est possible directement depuis $AUDIT_{clip}$. Cette administration doit être réalisée depuis la session $USER_{clip}$ associée à toute session dans $AUDIT_{clip}$ (cf. 5).

3.3.4 Configuration de la collecte de journaux

La configuration du démon *syslog-ng* (*/etc/syslog-ng/syslog-ng.conf*) définit systématiquement deux "sources" de journaux :

- une source "noyau" *kernsrc*, limitée au fichier */proc/kmsg*
- une source "clip" *clipsrc*, rassemblant la source interne du démon, et les *sockets unix* en mode *stream* suivantes :
  - */dev/log* dans le socle
  - */mounts/audit_priv/dev/log* (*/dev/log* dans $AUDIT_{clip}$), limitée à 256 connexions simultanées.
  - */mounts/admin_priv/dev/log* (*/dev/log* dans $ADMIN_{clip}$), limitée à 256 connexions simultanées.
  - */mounts/update_priv/dev/log* (*/dev/log* dans $UPDATE_{clip}$), limitée à 256 connexions simultanées.
  - */mounts/user_priv/dev/log* (*/dev/log* dans $USER_{clip}$), limitée à 256 connexions simultanées.

S'y ajoutent, dans le cas d'un système CLIP-RM, une source *rmXsrc* pour chaque cage RM_X, associée à la *socket unix* en mode *datagram* */vservers/rm_X/var/run/logger*, limitée à 16 connexions simultanées. Cette *socket* n'est pas utilisée directement par les clients *syslog* de la cage RM considérée, mais uniquement par le démon *syslog-ng* de la vue AUDIT de la cage (cf. [CLIP_1401]), qui y écrit les journaux collectés dans les différentes vues de la cage.

Ces différentes sources sont connectées aux destinations correspondant aux fichiers de journaux énumérés en 3.2, via des filtres basés sur le niveau et la *facility syslog* des messages, mais aussi sur des expressions régulières portant sur le contenu du message. On notera plus particulièrement les filtres suivants, qui définissent les conventions à respecter pour les différentes sources de messages :

- expression régulière *"^CLSM:.*"* sur la source *kernsrc* pour l'écriture dans *clsm.log*
- expression régulière *"^VERIEXEC:.*"* sur la source *kernsrc* pour l'écriture dans *veriexec.log*
- expression régulière *"^FW: .*"* sur la source *kernsrc* pour l'écriture dans *fw.log*
- expression régulière *"^PAX:.*"* sur la source *kernsrc* pour l'écriture dans *pax.log*
- expression régulière *"^vxW:.*"* sur la source *kernsrc* pour l'écriture dans *vserver.log*
- expression régulière *"^grsec:.*"* et non *"^grsec: (un|)mount.*"* sur la source *kernsrc* pour l'écriture dans *grsec.log*
- expression régulière *"^grsec: (un|)mount.*"* sur la source *kernsrc* pour l'écriture dans *grsec_mount.log*
- expression régulière *"^SSP:.*"* sur la source *clipsrc* pour l'écriture dans *ssp.log* (et le cas échéant le même filtre sur chaque source *rmXsrc* pour l'écriture dans *rm_X_ssp.log*

La connexion des sources aux fichiers destination assure de plus la traçabilité des origines des journaux : une destination donnée ne peut contenir que des messages issus d'une seule source. On distingue ainsi

les messages issus du noyau, ceux issus du *userland* CLIP (socle et cages CLIP confondues), et, dans un système CLIP-RM, ceux issus des cages RM (cage par cage).

3.3.5 Arrêt de la cage

L'arrêt de la cage est réalisé par la fonction *stop()* de *clip_audit()*, appelée tard dans la séquence d'arrêt du système ([CLIP_1301]), la gestion de dépendances entre scripts de démarrage assurant notamment que cet arrêt n'intervient qu'après celui de toutes les autres cages et de tous les démons du socle. La terminaison de $AUDIT_{clip}$ est compliquée par le privilège *CLSM_PRIV_IMMORTAL*, attribué au démon *syslog-ng*. Ce privilège interdit la terminaison du démon en dehors de la séquence d'arrêt. Même durant celle-ci, la terminaison du démon n'est pas possible depuis l'intérieur de la cage $AUDIT_{clip}$, car l'envoi d'un signal au démon nécessite de disposer de *CAP_SYS_BOOT* dans son masque héritable, alors que cette capacité est globalement masquée dans la cage. En particulier, une commande *vsctl audit stop* ne peut pas parvenir à tuer *syslog-ng*, et ne termine donc pas la cage. Il est ainsi nécessaire de tuer le démon depuis le contexte ADMIN. Plus précisément, la fonction *stop()* de *clip_audit()* réalise un traitement en deux temps :

- Dans un premier temps, une commande *vsctl audit stop* permet de terminer tous les processus de la cage autres que *syslog-ng*, en particulier le démon *sshd*[15].
- Dans un second temps, une commande *vsctl audit enter* permet d'entrer dans la cage (qui existe toujours du fait de la présence de *syslog-ng*) pour y invoquer *ps* et récupérer le numéro de *pid* du démon *syslog-ng*. Ce numéro n'est en effet pas visible directement dans le socle. Ensuite, le démon est tué par une commande *kill* lancée dans le socle sur son *pid*[16].

[15] On notera que ce démon n'est pas terminé par la fonction *stop()* du script *clip_sshd* qui assure son lancement.

[16] Cette opération est rendue possible par une exception au cloisonnement *vserver* des signaux, introduite par *clip-lsm* et *clip-patches* dans le noyau CLIP, autorisant l'envoi de signaux depuis le contexte ADMIN vers un autre contexte.

4 Cage UPDATE$_{clip}$

Le principal rôle de la cage UPDATE$_{clip}$ est le téléchargement des mises à jour CLIP (socle et cages), et l'application des mises à jour de paquetages secondaires. Dans le cas d'un système CLIP-RM, ce rôle est étendu au téléchargement de mises à jour des cages RM. Par ailleurs, son accès au VPN IPsec UPDATE lui confère un rôle secondaire de synchronisation horaire, par NTP, vis à vis de ce réseau. Elle dispose d'un accès aux montages stockant les miroirs locaux de paquetages (CLIP, et le cas échéant RM), ainsi qu'aux montages stockant les méta-données du système de gestion de paquetages. Elle fonctionne entièrement de manière non interactive : aucune session utilisateur n'y est possible.

4.1 Configuration

4.1.1 Configuration *Vserver*

La configuration de la cage est fournie par l'arborescence */etc/jails/update*, installée par le paquetage *app-clip/core-services*. Les principaux éléments de cette configuration sont les suivants :

- numéro de contexte *vserver* : 501.
- racine : */update.*
- commande définie à */sbin/crond*..
- pas d'adresse IP fixe, l'adresse (unique) de la cage est définie à la valeur du paramètre dynamique UPDATE_ADDR lors du lancement de la cage.
- Les capacités maximales autorisées dans la cage sont celles généralement autorisées dans les cages CLIP (cf. [CLIP_1202]), complétées de *CAP_NET_BIND_SERVICE* et *CAP_SYS_TIME*, qui sont nécessaires au fonctionnement du démon *ntp*, ainsi que de *CAP_SYS_CHROOT*, qui est pour l'heure réservée à des développements futurs, par exemple la création de vues au sein de la cage. En résumé, les capacités autorisées sont :
  - *CAP_CHOWN*
  - *CAP_DAC_OVERRIDE*
  - *CAP_DAC_READ_SEARCH*
  - *CAP_FOWNER*
  - *CAP_FSETID*
  - *CAP_KILL*
  - *CAP_SETGID*
  - *CAP_SETUID*
  - *CAP_NET_BIND_SERVICE*
  - *CAP_SYS_CHROOT*

- *CAP_SYS_TIME*

- Les drapeaux de contexte *vserver* sont les drapeaux par défaut listés dans [CLIP_1202].

4.1.2 Montages

Les montages constituant l'arborescence de la cage sont répertoriés dans les Tableau 5 et Tableau 6. La cage possède sa racine propre montée depuis */update_root*. On notera qu'à la différence des autres cages CLIP, cette racine est montée depuis la partition racine du système plutôt que depuis */mounts,* et est donc protégée en écriture, même depuis le socle, pendant le fonctionnement normal du système. Le coeur de la cage $UPDATE_{clip}$ est en effet associé au coeur du socle pour former le coeur CLIP, composé des paquetages primaires du système et uniquement mis à jour par le socle lors d'un redémarrage, et sur une partition alternative. Cela signifie en particulier que la cage $UPDATE_{clip}$ ne met jamais à jour sa racine par elle même.

Les bibliothèques du coeur CLIP (*/lib* et */usr/lib*) sont exposées en lecture seule dans la cage, de même que les répertoires de configuration partagés (*/etc/core*, */etc/shared* et */etc/admin*), comme cela est le cas pour les autres cages CLIP. L'ensemble de l'arborescence */usr* du socle est exposée en lecture-seule dans la cage, plutôt que le seul répertoire */usr/lib*, afin d'exposer aussi dans la cage les exécutables de gestion des paquetages, par exemple */usr/bin/dpkg* ou */usr/bin/apt-get*. Par ailleurs, $UPDATE_{clip}$ dispose d'un accès en lecture-écriture à l'ensemble de l'arborescence */usr/local*, ce qui lui permet de mettre à jour les paquetages secondaires CLIP[17]. De même, les racines dédiées des cages $ADMIN_{clip}$, $AUDIT_{clip}$ et $USER_{clip}$ et des éventuelles vues visionneuses (CLIP-RM) sont exposées en lecture-écriture dans la cage, afin de permettre leur mise à jour. Les arborescences privées de ces mêmes cages (arborescences montées depuis */mounts/admin_priv, /mounts/audit_priv* et */mounts/user_priv*) ne sont en revanche pas exposées, afin de préserver la confidentialité des données privées de ces cages. La seule exception à cette règle est l'exposition de l'arborescence */var* de $ADMIN_{clip}$ (*/mounts/admin_priv/var*) dans la cage, afin de permettre le partage de la *socket /var/run/download*, utilisée pour la communication entre le démon *downloadmaster* (dans $UPDATE_{clip}$) et son client *downloadrequest* (dans $ADMIN_{clip}$) (cf. 4.2).

***Remarque 3 : exposition de l'ensemble du* /var *de* $ADMIN_{clip}$ *dans* $UPDATE_{clip}$**

L'exposition de l'ensemble de l'arborescence /var *de* $ADMIN_{clip}$ *dans* $UPDATE_{clip}$ *n'est pas strictement nécessaire, et réduit sensiblement l'isolation des deux cages, notamment en permettant aux processus de* $UPDATE_{clip}$ *de contacter les différents démons du socle qui créent une socket de commande dans l'arborescence* /var *de* $ADMIN_{clip}$ *(cf. 2.3). Il serait de ce fait préférable de limiter la partie partagée de l'arborescence* /var *à un sous-répertoire dédié, contenant uniquement la socket de* downloadmaster.

[17] On notera que cette exposition de */usr/local* rompt, dans le cas de $UPDATE_{clip}$, le principe d'exclusivité des droits en écriture et exécution qui est appliqué dans les autres cages. */usr/local* ne peut en effet pas être monté avec un drapeau *noexec* dans la cage, du fait de la nécessité d'invoquer certains de ses exécutables depuis des *maintainer scripts* de paquetages secondaires.

Source	Point de montage	Type	Options
/update_root	/	bind	ro,nodev,noatime
/lib	/lib	bind	ro,nosuid,nodev,noatime
/usr/lib	/usr/lib	bind	ro,nosuid,nodev,noatime
/mounts/usr	/usr/local	bind	rw,nosuid,nodev,noatime
/etc/core	/etc/core	bind	ro,nosuid,nodev,noexec,noatime
/etc/shared	/etc/shared	bind	ro,nosuid,nodev,noexec,noatime
/mounts/admin_priv/etc.admin	/etc/admin	bind	ro,nosuid,nodev,noexec,noatime
/mounts/update_priv/var	/var	bind	rw,nosuid,nodev,noexec,noatime
/mounts/update_priv/root	/root	bind	rw,nosuid,nodev,noexec,noatime
/mounts/update_priv/dev	/dev	bind	ro,nosuid,noexec,noatime
/mounts/update_priv/tmp	/tmp	bind	rw,nosuid,nodev,noexec,noatime, mode=1777
/mounts/update_priv/pkgs	/pkgs/clip	bind	rw,nosuid,nodev,noexec,noatime
/var/pkg	/var/pkg	bind	rw,nosuid,nodev,noatime
/mounts/admin_root	/mounts/ admin_root	bind	rw,nosuid,nodev,noexec,noatime
/mounts/admin_priv/var	/mounts/ admin_rootlvar	bind	rw,nosuid,nodev,noexec,noatime
/mounts/audit_root	/mounts/ audit_root	bind	rw,nosuid,nodev,noexec,noatime
/mounts/user_root	/mounts/ user_root	bind	rw,nosuid,nodev,noexec,noatime
/proc	/proc	bind	ro,nosuid,nodev,noexec,noatime,nolock

Tableau 5: Montages de la cage UPDATE$_{clip}$.

Les sources sont relatives à la racine du système, les points de montages à la racine de la cage (*/update*).

Source	Point de montage	Type	Options
/mounts/viewers	/viewers	bind	rw,nosuid,nodev,noexec,noatime
/vservers/rm_h/update_priv/pkgs	/pkgs/rm_h	bind	rw,nosuid,nodev,noexec,noatime
/vservers/rm_b/update_priv/pkgs	/pkgs/rm_b	bind	rw,nosuid,nodev,noexec,noatime
/vservers/rm_h/update_priv/ var/pkg/lib/dpkg	/var/rm_h/pkg/ lib/dpkg	bind	rw,nosuid,nodev,noexec,noatime
/vservers/rm_b/update_priv/ var/pkg/lib/dpkg	/var/rm_b/pkg/ lib/dpkg	bind	rw,nosuid,nodev,noexec,noatime

Tableau 6: Montages supplémentaires de la cage UPDATE$_{clip}$ au sein d'un système CLIP-RM.

Les sources sont relatives à la racine du système, les points de montages à la racine de la cage (*/update*).

Les répertoires montés en */pkg/clip* et */var/pkg* dans la cage sont partagés entre le socle et la cage UPDATE$_{clip}$ pour la gestion des mises à jour. Le premier contient le miroir local de paquetages CLIP,

utilisé aussi bien pour les mises à jour de paquetages secondaires (par la cage) que primaires (par le socle). Le second contient les méta-données du système de gestion de paquetages (*apt*), et permet en particulier d'assurer de manière unifiée le suivi des dépendances. On notera aussi que, à la différence des cages CLIP interactives, le */tmp* de $UPDATE_{clip}$ ne prend pas la forme d'un système de fichiers *tmpfs* entièrement en mémoire, mais est monté en *bind* depuis un répertoire du disque. Ce montage est rendu nécessaire par l'utilisation que fait *dpkg* du */tmp* pour stocker les fichiers des paquetages pendant leur mise à jour, qui peut nécessiter une taille disponible importante, ne pouvant généralement pas être offerte en mémoire.

Dans un système de type CLIP-RM, deux répertoires similaires sont partagées par la cage $UPDATE_{clip}$ et la vue UPDATE de chaque cage RM_X :

- le répertoire monté sur */pkgs/rm_X* contient le miroir de paquetages de la cage RM_X, téléchargés par $UPDATE_{clip}$ mais installés par la vue UPDATE de la cage RM..
- le répertoire monté sur */var/rm_X/pkg/lib/dpkg* contient les méta-données des paquetages installés dans la cage RM_X, permettant à $UPDATE_{clip}$ de déterminer les paquetages à télécharger pour cette cage.

Ces deux répertoires sont accessibles en lecture-écriture dans la cage RM_X concernée. Il est donc nécessaire de les monter depuis la cage RM_X dans la cage $UPDATE_{clip}$, plutôt que l'inverse, afin de maintenir la séparation des systèmes de fichiers accessibles en lecture-écriture depuis les cages RM, sans laquelle des canaux de communication pourraient être créés entre ces cages (cf. 1.1). Ces montages sont complétés, dans un système CLIP-RM, par une exposition en lecture-écriture de */mounts/viewers,* qui contient les racines de vues visionneuses de la cage $USER_{clip}$ (cf. 5.3), afin de permettre la mise à jour des paquetages secondaires composant ces racines.

4.1.3 Configuration *Veriexec*

La cage $AUDIT_{clip}$ se voit associer un contexte *veriexec,* dont le niveau de sécurité est défini avec les drapeaux suivants (cf. [CLIP_1201]) par le script */etc/init.d/veriexec* (*app-clip/veriexec*) :

- *VRXLVL_ACTIVE*
- *VRXLVL_LVL_IMMUTABLE*
- *VRXLVL_ENFORCE_MNTRO*

Ces drapeaux interdisent notamment tout retrait de drapeaux de sécurité de la base *veriexec* depuis la cage, mais permettent en revanche l'administration depuis la cage des entrées *veriexec* de son contexte. Par ailleurs, le *xid* de la cage est définit comme contexte *veriexec* UPDATE du système, ce qui autorise la cage à modifier les entrées de tous les contextes qui ne portent pas le drapeau *VRXLVL_UPDATE_IMMUTABLE*, c'est-à-dire du socle et de toutes les cages CLIP, mais pas des éventuelles cages RM. Ces modifications sont par ailleurs conditionnées à la possibilité d'accéder en écriture au montage sous-jacent, du fait du drapeau *VRXLVL_ENFORCE_MNTRO*, ce qui interdit toute mise à jour des entrées correspondant aux paquetages primaires CLIP.

Le masque de capacités du contexte *veriexec* réduit les capacités attribuables dans la cage par *veriexec* à :

- *CAP_SYS_CHROOT*
- *CAP_NET_BIND_SERVICE*
- *CAP_SYS_TIME*

De même, le masque de privilèges CLSM du contexte *veriexec* réduit les privilèges attribuables dans la cage à :

- *CLSM_PRIV_PROCFD*
- *CLSM_PRIV_NETSERVER*
- *CLSM_PRIV_NETCLIENT*
- *CLSM_PRIV_VERICTL*

Les entrées *veriexec* suivantes sont définies dans le contexte de la cage :

- */usr/lib/apt/methods/https* se voit attribuer *CLSM_PRIV_NETCLIENT*
- */usr/sbin/start-stop-daemon* se voit attribuer *CLSM_PRIV_PROCFD*, lorsqu'il est exécuté par *root*
- */usr/sbin/ntpdate* se voit attribuer *CAP_SYS_TIME, CAP_NET_BIND_SERVICE*, *CLSM_PRIV_NETCLIENT* et *CLSM_PRIV_NETSERVER* lorsqu'il est exécuté par *root*.
- */sbin/verictl* (exécutable propre à la cage) se voit attribuer *CLSM_PRIV_VERICTL* lorsqu'il est exécuté par *root*.

4.1.4 Fichiers

La cage dispose par ailleurs de ses fichiers *devices* attitrés, montés depuis */mounts/update_priv/dev*. Ces fichiers sont limités à :

- *zero, null, full*
- *urandom*, ainsi qu'un lien symbolique *random* pointant sur *urandom*
- *socket /dev/log* créée par le démon *syslog-ng* de la cage $\text{AUDIT}_{\text{clip}}$
- fichier */dev/veriexec* permettant la mise à jour de la base *veriexec* du système (cf. [CLIP_1201])

A la différence des cages interactives du système, $\text{UPDATE}_{\text{clip}}$ n'intègre aucun pseudo-terminal (pas de *ptmx*, pas de montage *devpts*), et ne dispose pas du répertoire */dev/mapper* utilisé pour le montage de supports amovibles sécurisés, qui ne sont pas employés dans cette cage. En revanche, la cage est la seule avec le socle parmi les compartiments logiciels CLIP à disposer du *device veriexec*.

Les exécutables de la cage sont tous spécifiques à celle-ci, et en quasi totalité fournis par un paquetage *busybox* (cf. [BUSYBOX]), dans une configuration spécifique. Cette configuration est automatiquement utilisée lorsque le paquetage *sys-apps/busybox* est compilé avec la variable d'environnement *DEB_NAME_SUFFIX* (cf. [CLIP_1101]) positionnée à *"update"* (le paquetage *Debian* résultant porte le nom de *busybox-update*). La configuration *update* de *busybox* incorpore dans celui-ci un ensemble d'utilitaires utilisables dans les *maintainer-scripts* des paquetages, notamment les

utilitaires standards de manipulation de fichiers (*cp, mv, rm, dd,* etc.) et les interpréteurs *sed* et *awk*, dans leurs versions "allégées" *busybox*. Le paquetage inclut de plus un *shell ash* minimaliste, dépourvus d'auto-complétion ou de gestion de l'historique. Le *shell busybox* utilise le drapeau *O_MAYEXEC*, spécifique à CLIP ([CLIP_1201]), lors de l'ouverture de fichiers, ce qui interdit l'exécution ou le "sourçage" de scripts depuis des montages portant l'option *noexec*. Outre ces utilitaires *busybox*, la racine de $\text{UPDATE}_{\text{clip}}$ incorpore les exécutables complémentaires suivants :

- */bin/tar*, dont une copie est installée dans la racine de $\text{UPDATE}_{\text{clip}}$ par le paquet *sys-apps/tar* du socle (généré avec *CLIP_VROOTS* positionné à "*/ /update_root*"). Sa présence est nécessaire au fonctionnement de *dpkg*, qui met en oeuvre des extensions GNU de la commande *tar* qui ne sont pas supportées par le *tar busybox*.
- */sbin/vericti*, installé dans la cage par un paquetage spécifique, *verictl-update* (généré à partir de l'*ebuild app-clip/verictl* en positionnant *DEB_NAME_SUFFIX* à *"update"* et en définissant le drapeau USE *verictl*).

4.2 Démarrage et fonctionnement de la cage

Le démarrage des différents services de la cage est réparti sur deux scripts de démarrage, *clip_update* et *clip_download*. A l'issue de cette phase de démarrage, l'essentiel des services est invoqué périodiquement par le démon *cron* de la cage.

4.2.1 Configuration initiale de la cage

La cage est créée et configurée par le script *clip_update* (*app-clip/core-services*). Ce script importe dans un premier temps les paramètres définissant l'adresse IP de la cage, *UPDATE_ADDR* et *UPDATE_MASK*, depuis le fichier de configuration */etc/admin/conf.d/net*. Il procède ensuite à une configuration en plusieurs étapes, à l'instar de ce qui est fait pour la cage $\text{AUDIT}_{\text{clip}}$:

- Un *cookie* d'authentification *vsctl* est créé par la commande *vsctl update cookie* (cf.)
- La cage est créée, puis maintenue active, par une commande *vsctl update setup*, en utilisant ce *cookie*.
- Une commande préalable est lancée dans la cage : le script */bin/cleanup_log.sh* de la cage est lancé dans celle-ci par une commande *vsctl enter*, afin de supprimer les occurrences précédentes de fichiers de journalisation temporaires, créés par les outils de gestion et de téléchargement des mises à jour dans le */var* privé de la cage (ces fichiers ne sont pas consultables depuis la cage $\text{AUDIT}_{\text{clip}}$; ils sont principalement utilisés pour le test des fonctions de mise à jour).
- Le démon *crond* (fournit par le paquetage *busybox-update*), démon principal de la cage, est ensuite lancé par une deuxième commande *vsctl enter.*
- Enfin, la cage pouvant désormais être maintenue active par le démon *crond*, le processus *vsctl setup* est terminé par une commande *vsctl endsetup*, utilisant le même *cookie* pour son authentification.

4.2.2 Téléchargements initiaux et gestion des téléchargements

Le script *clip_download* (*app-clip/clip-download*) est invoqué alors que la cage est déjà active (cf. [CLIP_1301]), et après le démarrage du démon IKE permettant l'établissement du VPN de téléchargement de mises à jour (cf. [CLIP_1501]). Son rôle est double : réaliser un éventuel téléchargement de mises à jour d'une part, et lancer le démon *downloadmaster,* permettant le pilotage des téléchargements depuis la cage $ADMIN_{clip}$, d'autre part. L'intérêt d'un téléchargement initial est de disposer au plus tôt d'éventuelles mises à jour lors d'un démarrage faisant suite à une longue période hors ligne. Le script ne retourne qu'une fois le téléchargement achevé (pas de téléchargement en arrière plan), et les dépendances entre scripts de démarrage garantissent que les scripts réalisant les mises à jour initiales des paquetages secondaires CLIP, et des éventuelles cages RM (paquetages primaires et secondaires) ne sont appelés qu'après ce téléchargement initial, et peuvent de ce fait disposer des paquetages fraîchement téléchargés (cf. [CLIP_DCS_13006] et [CLIP_1301]). En revanche, les paquetages primaires CLIP ne peuvent pas être mis à jour immédiatement, comme détaillé dans la remarque ci-dessous.

Remarque 4 : téléchargement initial et mises à jour du coeur CLIP

Le téléchargement initial est susceptible de rendre disponible localement une mise à jour du coeur CLIP (paquetage primaires). Cependant, à la différence des paquetages secondaires, une telle mise à jour ne sera pas appliquée immédiatement après son téléchargement, mais uniquement au redémarrage suivant, dans la mesure où les mises à jour du coeur CLIP ne sont réalisées qu'avant l'établissement de liens réseau. Il serait cependant souhaitable de prendre en compte cette problématique lors du téléchargement initial de mises à jour, en déclenchant par exemple un redémarrage automatique dès lors qu'une mise à jour du coeur a pu être entièrement téléchargée.

La réalisation d'un téléchargement initial des mises à jour impose cependant un important délai dans la séquence de démarrage, aussi bien du fait du temps d'établissement du VPN UPDATE (pour lequel des associations de sécurité n'ont normalement pas encore été négociées à cette étape du démarrage) que de celui des téléchargement proprement dits. Afin de ne pas imposer un tel délai systématique, en particulier sur un poste qui ne reste jamais longtemps éteint ou hors ligne (et qui peut donc disposer à chaque démarrage de paquetages récents téléchargés avant l'extinction précédente), la réalisation d'un téléchargement initial n'est qu'optionnelle, et peut être activée ou désactivée par l'administrateur local en positionnant à *'yes'* ou *'no'* respectivement la variable *run_at_boot* définie dans le fichier */etc/admin/clip_download/clip_download.conf*.

Le comportement détaillé du script *clip_download* est ainsi le suivant :

- Lecture du fichier de configuration */etc/admin/clip_download/clip_download.conf*, à la recherche de la variable *run_at_boot*. Plutôt que de mettre en oeuvre les fonctions d'import sécurisés, le script se contente de rechercher un motif *'run_at_boot=yes'* dans le fichier, par une simple commande *grep*. Si ce motif n'est pas trouvé, c'est-à-dire si *run_at_boot* n'est pas définie ou est définie à tout autre valeur que *'yes'* (et pas uniquement à *'no'*), le script saute directement à la dernière étape de la présente énumération.
- Téléchargement initial pour la distribution CLIP, en invoquant */usr/bin/clip_download -j clip* dans la cage à l'aide d'une commande *vsctl update enter*. Dans la mesure où il est nécessaire, pour procéder au téléchargement, d'établir au préalable les associations de sécurité IPsec

correspondant au VPN UPDATE, cette commande est répétée au maximum cinq fois, avec des temporisation de 0.5, 1, 2, 2 et 2 secondes respectivement, tant qu'elle ne renvoie pas un code d'erreur nul. On notera que les codes d'erreurs de *clip_download* ne permettent pas de distinguer la cause d'un échec de téléchargement, ce qui impose de répéter les cinq tentatives lorsqu'un problème autre que l'établissement du VPN (par exemple l'indisponibilité du serveur) fait échouer le téléchargement. Les sorties standard et d'erreur de la commande de téléchargement sont redirigées vers le fichier de journaux temporaire */var/log/clip_download.log* (au sein du */var* privé de la cage, non consultable depuis $AUDIT_{clip}$).

- Au sein d'un système CLIP-RM uniquement, lancement d'un téléchargement initial dans chaque cage RM_X, par une commande *vsctl update enter -- /usr/bin/clip_download -j rm_X*. Une seule tentative de téléchargement est réalisée pour chaque cage, le VPN UPDATE étant supposé établis à ce stade. Les sorties standard et d'erreur de chaque commande de téléchargement sont redirigées vers le même fichier de journaux */var/log/clip_download.log*.
- Enfin, indépendamment de la valeur attribuée à *run_at_boot*, le démon *downloadmaster* (*app-clip/clip-download*) est lancé dans la cage par une commande *vsctl update enter*. Ce démon ce met en écoute sur la *socket /mounts/admin_root/var/run/download* au sein de la cage, qui correspond à */mounts/admin_priv/var/run/download* dans le socle et à */var/run/download* dans la cage $ADMIN_{clip}$, ce qui permet à l'administrateur local de lancer, verrouiller et déverrouiller les téléchargement.

4.2.3 Actions périodiques

Les différents services réalisés par la cage sont lancés périodiquement par le démon *crond*, ou, pour certains uniquement, par le démon *downloadmaster* à des moments arbitraires. Les tâches périodiques de *crond* sont définies dans le fichier */etc/cron/crontab/root*, qui est renseigné par les paquetages *app-clip/clip-download*, *app-clip/clip-install-clip* et *net-misc/ntp*. Chacun de ces paquetages ajoute une ou plusieurs lignes au fichier dans sa phase de post-installation, et retire ces mêmes lignes lors de sa pré-désinstallation.

Téléchargements

Les téléchargements sont réalisés par invocation du script */usr/bin/clip_download*. Le *crontab* de la cage spécifie une invocation par heure du script pour les paquetages CLIP, complétées dans le cas d'un système CLIP-RM d'une invocation horaire (à des heures différentes) par cage RM.

L'heure exacte (nombre de minutes après l'heure pile) à laquelle le script *clip_download* est invoqué pour CLIP est tirée aléatoirement pour chaque client par le script *postinst* du paquetage *clip-download*. Les variations sont limitées à la plage H+10-H+49, afin d'éviter de lancer *clip_download* en même temps que *clip_install* (qui est lancé à l'heure pile), ce qui interdirait l'exécution de l'un de ces deux scripts du fait de la mise en oeuvre de verrous. Les éventuels lancements de *clip_download* pour des cages RM sont réalisés à des intervalles fixes après ce premier lancement, ces intervalles étant calculés modulo 40 afin de rester dans la même plage de valeurs. Cette variabilité de l'heure de lancement permet de limiter la congestion sur le serveur de mise à disposition des mises à jour, en répartissant la charge dans le temps.

Le script peut également être lancé arbitrairement pour CLIP ou pour une éventuelle cage RM par

l'administrateur local, par l'intermédiaire du démon *downloadmaster*.

Installation de mises à jour

Les mises à jour de paquetages secondaires CLIP sont appliquées par l'invocation du script */usr/bin/clip_install* avec les paramètres *-d clip -p apps*. Le *crontab* de la cage spécifie une invocation à chaque heure pile du script, ce qui garantit l'absence de collisions avec les invocations périodiques de *clip_download*. Les possibles collisions entre une invocation de *clip_download* par *downloadmaster* et le lancement périodique de *clip_install* sont prises en compte par les mécanismes de verrouillage mis en oeuvre par ces deux scripts.

Synchronisation horaire

La synchronisation horaire du poste est réalisée par un script spécifique à CLIP, */usr/bin/clip_ntpdate*, installé par le paquetage *net-misc/ntp*. Ce script est invoqué toutes les heures par le *crontab* de la cage $\text{UPDATE}_{\text{clip}}$. Il réalise, lors de son lancement, l'import sécurisé d'une variable *USE_NTP* depuis le fichier */etc/admin/conf.d/ntp,* avec deux valeurs autorisées : '*yes'* et *'no'*. Lorsque cet import échoue, ou si la variable est définie à *'no'*, le traitement s'interrompt immédiatement. Dans le cas contraire, le script importe une seconde variable, *NTP_SERVER*, depuis le même fichier de configuration, puis réalise une synchronisation au travers du VPN UPDATE par une commande *ntpdate "${NTP_SERVER}"*. Les erreurs éventuelles, et le résultat en cas de succès (ajustement horaire réalisé) sont journalisés par l'interface *syslog*, et donc consultables depuis $\text{AUDIT}_{\text{clip}}$ (dans le fichier */log/messages.log*).

L'heure exacte à laquelle ce script est invoqué (minutes après l'heure pile) est tirée aléatoirement sur chaque client parmi les 60 choix possibles, par le script *postinst* du paquetage *ntp*. Cette mesure limite la congestion sur le serveur NTP, en répartissant la charge dans le temps.

4.2.4 Arrêt de la cage

Au cours de la séquence d'arrêt du système, le script *clip_download* est appelé en premier lieu, et tue le démon *downloadmaster*. Dans un second temps, le script *clip_update* est arrêté, et termine l'ensemble de la cage par une commande *vsctl update stop*.

5 Cage USER$_{clip}$

La cage USER$_{clip}$ constitue la principale cage d'utilisation du système, dans laquelle tous les utilisateurs locaux ouvre une session. Elle fonctionne en mode purement interactif, sans aucun service permanent en dehors des sessions utilisateur. Elle n'est d'ailleurs pas active en dehors de ces sessions. Son environnement logiciel est plus riche que celui des autres cages CLIP, et son fonctionnement plus variable en fonction du type de système CLIP.

5.1 Configuration

La cage USER$_{clip}$ est, à la différence des autres cages CLIP, une cage temporaire : elle n'est créée que lors de l'ouverture d'une session utilisateur, et se termine immédiatement à la fin de la session. La création de la cage est réalisée directement par le démon *xdm* lors de l'ouverture de session. Cependant, afin de simplifier le code de création de cage de *xdm*, les montages constituant l'arborescence de la cage sont réalisés de manière permanente dans le socle (*namespace* principal) au démarrage, et hérités par la cage chaque fois qu'elle est créée. La cage USER$_{clip}$ est aussi caractérisée par la diversité des applications qui peuvent y être lancées, en fonction du type de session et de système CLIP.

5.1.1 Configuration *Vserver*

La configuration de la cage est définie directement par *xdm* Certains de ses paramètres (adresse, racine, *xid*) sont ajustables sous la forme de paramètres *Xressources* ; les autres (capacités en particulier) sont définis "en dur" dans l'exécutable *xdm*. Les principaux éléments de cette configuration sont les suivants :

- numéro de contexte *vserver* : 502 (*Xressource clientXid*).
- racine : */user* (*Xressource clientRoot*).
- commande initiale non définie : un fils du démon *xdm* constitue le processus initial de la cage. Ce fils exécute ensuite un ensemble de clients, qui dépend du type de session.
- pas d'adresse IP fixe, l'adresse (unique) de la cage est définie à la valeur du paramètre dynamique USER_ADDR (adresse commune de ADMIN$_{clip}$, AUDIT$_{clip}$ et USER$_{clip}$) lors du lancement de la cage (*Xressource clientAddr*).
- Les capacités maximales autorisées dans la cage sont celles généralement autorisées dans les cages CLIP (cf. [CLIP_1202]), complétées de *CAP_SYS_CHROOT*, qui est nécessaire au fonctionnement de l'utilitaire *viewer-launch*, soit :
  - *CAP_CHOWN*
  - *CAP_DAC_OVERRIDE*
  - *CAP_DAC_READ_SEARCH*
  - *CAP_FOWNER*
  - *CAP_FSETID*

- *CAP_KILL*
- *CAP_SETGID*
- *CAP_SETUID*
- *CAP_SYS_CHROOT*

- De même, les drapeaux de contexte *vserver* sont les drapeaux par défaut listés dans [CLIP_1202].

5.1.2 Montages

Les montages, à l'exception des montages de session (partitions chiffrées et temporaires des utilisateurs) ou à la demande (supports USB sécurisés) sont réalisés à partir d'un fichier *fstab* dédié, */etc/fstab.user*. Ces montages sont résumés dans les Tableau 7 et Tableau 8.

Source	Point de montage	Type	Options
/mounts/user_root	*/*	*bind*	*ro,nosuid,nodev,noatime*
/lib	*/lib*	*bind*	*ro,nosuid,nodev,noatime*
/bin	*/bin*	*bind*	*ro,nodev,noatime*
/usr	*/usr*	*bind*	*ro,nodev,noatime*
/mounts/usr	*/usr/local*	*bind*	*ro,nosuid,nodev,noatime*
/etc/core	*/etc/core*	*bind*	*ro,nosuid,nodev,noexec,noatime*
/etc/shared	*/etc/shared*	*bind*	*ro,nosuid,nodev,noexec,noatime*
/mounts/admin_priv/etc.admin	*/etc/admin*	*bind*	*ro,nosuid,nodev,noexec,noatime*
/mounts/user_priv/var	*/var*	*bind*	*rw,nosuid,nodev,noexec,noatime*
/mounts/user_priv/dev	*/dev*	*bind*	*ro,nosuid,noexec,noatime*
/var/run/authdir	*/var/run/authdir*	*bind*	*ro,nosuid,noexec,noatime*
/proc	*/proc*	*bind*	*ro,nosuid,nodev,noexec,noatime,nolock*
<none>	*/dev/pts*	*devpts*	*rw,nosuid,noexec,noatime,nolock, gid=5,mode=620*
/dev/mapper/ _home_parts_<user>.part	*/home/user* (session)	*ext2*	*rw,nosuid,nodev,noexec,noatime*
<usertmp>	*/tmp* (session)	*tmpfs*	*rw,nosuid,nodev,noexec,noatime, mode=1777*
/tmp/.X11-unix	*/tmp/.X11-unix* (session)	*bind*	*ro,nosuid,nodev,noexec,noatime*
/dev/mapper/stockage_amovible _<user>_clip	*/mnt/usb* (à la demande)	*vfat*	*rw,nosuid,nodev,noexec,uid=<user>, gid=<user>*

Tableau 7: Montages de la cage $USER_{clip}$.

Les sources sont relatives à la racine du système, les points de montages à la racine de la cage (*/user*).

Source	Point de montage	Type	Options
/mounts/viewers	/viewers	bind	ro,nosuid,nodev,noatime,nolock
/mounts/xauth	/xauth	bind	rw,nosuid,nodev,noexec,noatime, mode=1777

Tableau 8: Montages supplémentaires de la cage $USER_{clip}$ au sein d'un système CLIP-RM.

Les sources sont relatives à la racine du système, les points de montages à la racine de la cage (*/user*).

La cage possède sa racine propre montée depuis */mount/user_root,* mais à la différence des autres cages CLIP, cette racine ne fournit pas les exécutables de base (*shell,* etc.) de la cage, qui sont au contraires fournis par le socle, dont le */bin* est exposé en lecture seule dans la cage. Les répertoires */lib*, */usr* et */usr/local* du socle sont aussi exposés en lecture seule, de manière à offrir à la cage un environnement logiciel riche. Outre les répertoires partagés standards, */etc/core*, */etc/shared* et */etc/admin*, la cage partage avec le socle et la cage X11 le répertoire */var/run/authdir*, dans lequel *xdm* place au lancement le *cookie Xauthority* principal du serveur X11, dont est dérivé le *cookie* de la session $USER_{clip}$ (lui-même stocké dans */tmp*). Aucun accès en écriture n'étant nécessaire à ce fichier, le montage est réalisé en lecture seule. De même, la cage partage en lecture seule avec le socle et la cage X11 le répertoire */tmp/.X11-unix*, dans lequel est créée la *socket unix* du serveur X11, qui est utilisée aussi bien par *xdm* que par les clients de $USER_{clip}$.

Des montages complémentaires sont réalisés dans le cas d'un système CLIP-RM, correspondant au répertoire */viewers* contenant les racines des vues visionneuses (en lecture-seule, comme les autres montages autorisant l'exécution de fichiers au sein de la cage), et au répertoire */xauth*, qui est exposé en lecture-écriture dans la cage et dont les sous-répertoires sont par ailleurs montés en lecture seule dans les différentes vues visionneuses, et qui est destiné à contenir les *cookies Xauthority* non privilégiés de ces différentes vues. Ce *répertoire xauth* est monté avec le mode *1777*, afin de permettre à tous les comptes utilisateurs de créer des *cookies* dans celui-ci. Voir aussi 5.3.

Plusieurs montages complémentaires sont ajoutés dans la cage à chaque ouverture de session (cf. [CLIP_1302]). Ces montages sont réalisés par le module PAM *pam_exec_pwd,* et sont définis dans le fichier de configuration */etc/fstab.session* (*app-clip/clip-user-mount*) :

- La partition chiffrée de niveau CLIP de l'utilisateur de la session est montée sur */home/user,* qui est défini comme le *$HOME* de l'utilisateur.
- Un système de fichiers en mémoire, *tmpfs*, est monté sur */tmp*, qui constitue le seul autre répertoire accessible à l'utilisateur au sens des permissions discrétionnaires[18]. Ainsi, l'utilisateur ne peut écrire aucune donnée claire sur le disque.
- Le montage de */tmp/.X11-unix* doit être créé à chaque ouverture de session, pour ne pas être masqué par le montage de */tmp*.

Enfin, l'import et l'export de données utilisateur sont permis par la possibilité de monter dans la cage, sur */mnt/usb,* un support USB sécurisé de niveau CLIP.

[18] Le répertoire */var/tmp* de la vue est en fait un lien symbolique pointant vers *../tmp*.

5.1.3 Configuration *Veriexec*

La cage $USER_{clip}$ se voit associer un contexte *veriexec,* dont le niveau de sécurité est défini avec les drapeaux suivants (cf. [CLIP_1201]) par le script */etc/init.d/veriexec* (*app-clip/veriexec*) :

- *VRXLVL_ACTIVE*
- *VRXLVL_SELF_IMMUTABLE*
- *VRXLVL_LVL_IMMUTABLE*
- *VRXLVL_ENFORCE_MNTRO*

Ces drapeaux interdisent notamment toute opération d'administration de la base *veriexec* depuis la cage.

Le masque de capacités du contexte *veriexec* réduit les capacités attribuables dans la cage par *veriexec* à :

- *CAP_SETGID*
- *CAP_SETUID*
- *CAP_SYS_CHROOT*

De même, le masque de privilèges CLSM du contexte *veriexec* réduit les privilèges attribuables dans la cage à :

- *CLSM_PRIV_PROCFD*
- *CLSM_PRIV_NETCLIENT*

Les entrées *veriexec* suivantes sont définies dans le contexte de la cage :

- */usr/local/bin/viewer-launch* se voit attribuer *CAP_SETGID, CAP_SETUID* et *CAP_SYS_CHROOT*.
- */usr/local/bin/ssh* se voit attribuer *CLSM_PRIV_NETCLIENT*
- */bin/fuser* se voit attribuer *CLSM_PRIV_PROCFD* lorsqu'il est invoqué par *root* uniquement.

5.1.4 Fichiers

La cage dispose de ses fichiers *devices* attitrés, montés depuis */mounts/user_priv/dev*. Ces fichiers sont limités à :

- *zero, null, full*
- *urandom*, ainsi qu'un lien symbolique *random* pointant sur *urandom*
- *ptmx, tty* et un montage de type *devpts* sur */dev/pts*, fournissant les pseudo-terminaux nécessaires au fonctionnement de *sshd*.
- *socket /dev/log* créée par le démon *syslog-ng* de $AUDIT_{clip}$ (cf. 3)
- répertoire */dev/mapper* permettant la copie temporaire (par un appel *tar* lancé dans le socle, cf.

[CLIP_DCS_15088]) d'un *device dm-crypt stockage_amovible_<user>_clip*, correspondant à l'image déchiffrée d'un support amovible sécurisé.

A la différence des cages CLIP présentées jusqu'ici, la cage ne contient pas de paquetage *busybox* attitré, mais a à sa disposition les principaux exécutables (hors utilitaires d'administration de */sbin*) du socle, en particulier les différents clients graphiques et leurs bibliothèques associées, au sein de l'arborescence */usr/local*. Le *shell* de la cage est le *shell bash* du socle CLIP. Ce *shell* utilise le drapeau *O_MAYEXEC*, spécifique à CLIP ([CLIP_1201]), lors de l'ouverture de fichiers, ce qui interdit l'exécution ou le "sourçage" de scripts depuis des montages portant l'option *noexec*.

5.2 Démarrage et fonctionnement de la cage

5.2.1 Démarrage de la cage

Le script de démarrage *clip_user* (*app-clip/core-services*) assure la création (dans l'espace de nommage du socle) des montages permanents de la cage conformément au fichier */etc/fstab.user*. Dans le cas d'un système CLIP-RM, ce script réalise également un "nettoyage" d'éventuels *cookies Xauthority* qui pourraient avoir été laissés dans le répertoire */xauth*[19]. En revanche, le script n'assure en aucun cas le démarrage de la cage, qui est à la charge du démon *xdm*.

Dans son fonctionnement normal, le démon *xdm* réalise deux appels *fork()*, respectivement dès son lancement et à l'ouverture de chaque session utilisateur, pour :

- Lancer le serveur X11 dès son démarrage dans le premier fils, afin de pouvoir afficher sa fenêtre d'ouverture de session. Un *cookie* X11 "principal" est généré à cette occasion afin d'authentifier *xdm* vis-à-vis du serveur. Dans la configuration CLIP, ce *cookie* est placé dans le répertoire */var/run/authdir/authfiles* du socle, qui est exposé sous le même nom dans les cages $USER_{clip}$ et X11. Le serveur X11 signale sa disponibilité à *xdm* par l'envoi d'un signal *SIGUSR* une fois son initialisation complète.
- Lancer dans le deuxième fils la session cliente X11 suite à l'authentification d'un utilisateur. Ce fils réalise un appel *setuid()* afin de prendre l'identité de l'utilisateur authentifié. Un *cookie Xauthority* est généré pour la session, et normalement placé dans */tmp/.XauthXXXXXX* (où les X sont remplacés par des caractères aléatoires). Enfin, le processus exécute la session X11 choisie, qui se présente comme un script placé normalement dans */etc/X11/Sessions*.

[19] Ces *cookies* sont supprimés automatiquement lorsque les visionneuses correspondantes se terminent. Cependant, une session X11 interrompue de manière non nominale pourrait avoir laissé en place ses *cookies*.

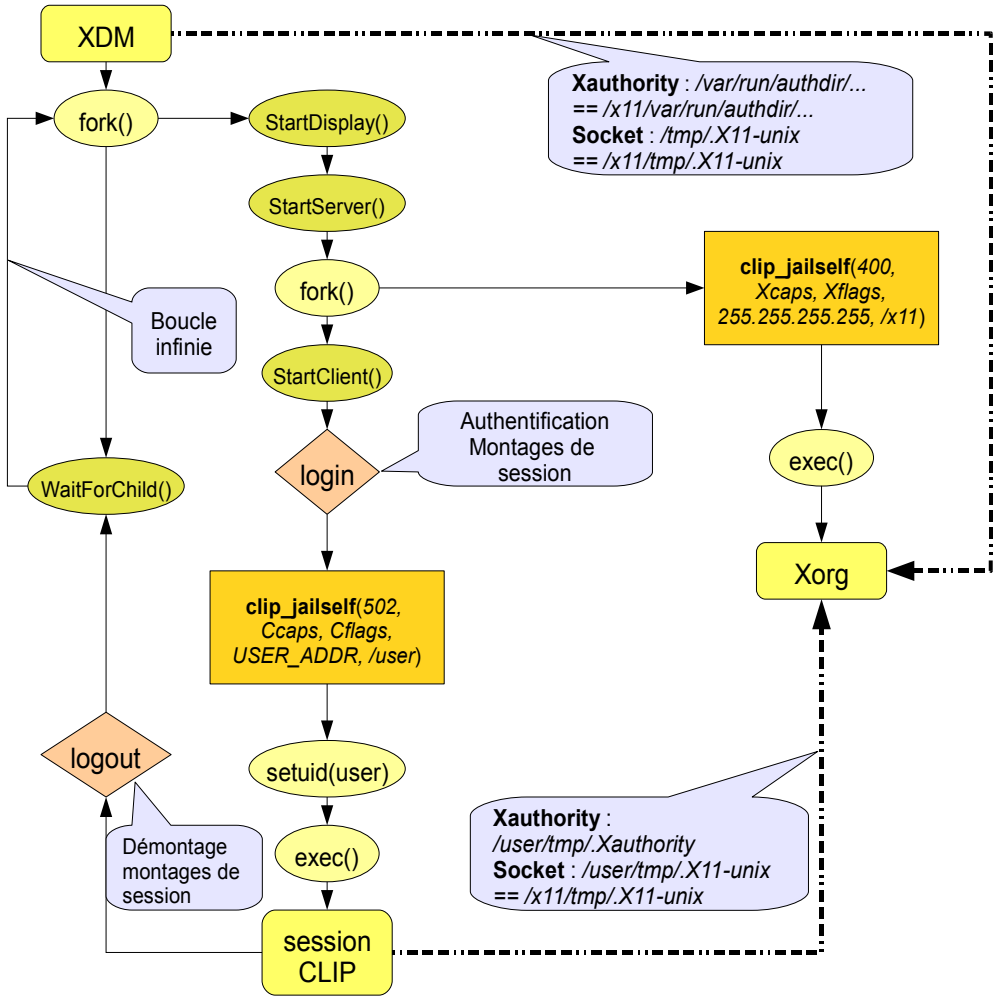

Figure 3: Principe du lancement des cages USER$_{clip}$ et X11 par XDM.

Ce démon *xdm* est spécifiquement modifié dans CLIP de manière à se lier dynamiquement à la bibliothèque *libclipvserver.so* (*clip-libs/clip-libvserver*) ([CLIP_1202]), afin d'enfermer chacun de ces deux fils dans une cage *vserver* distincte. Certains des paramètres de ces cages, en particulier leurs privilèges, sont définis directement dans le code du démon. D'autres en revanche, en particulier les racines, adresses IP et numéros de contexte de ces cages, sont définies comme des paramètres *Xressource* du démon. En particulier, l'enfermement de chaque processus fils dans une cage *vserver* n'est réalisé que si le numéro de contexte correspondant est défini (les autres paramètres prenant au besoin des valeurs par défaut). Les fils de *xdm* s'enferment dans leurs cages respectives avant de réaliser tout appel *exec()*, et, dans le cas du fils exécutant la session cliente, avant l'appel à *setuid()* révoquant éventuellement les privilèges *root*. Ce mode de fonctionnement est résumé dans la Figure 3.

Pour ce qui concerne la cage USER$_{clip}$, les paramètres suivants sont définis dans le fichier de configuration *Xressource /usr/local/etc/X11/xdm/xdm-config* :

- Le paramètre *DisplayManager.clientXid*, qui définit le numéro de contexte de la cage cliente, est positionné à *502*, numéro de contexte de $USER_{clip}$.
- Le paramètre *DisplayManager.clientRoot*, qui définit la racine de la cage cliente, est positionné à */user*.

En revanche, l'adresse de la cage (paramètre *DisplayManager.clientAddr*) n'est pas statiquement définie, mais passée sur la ligne de commande du démon par l'option *-addr <adresse>*. Cet argument est passé par le script */usr/local/etc/X11/startDM.sh,* qui est invoqué par */etc/init.d/xdm* pour lancer le démon *xdm*. Ce script *startDM.sh* réalise l'import sécurisé de l'adresse *USER_ADDR* depuis le fichier de configuration */etc/admin/conf.d/net*, et passe cette adresse par *-addr*. Le masque de sous-réseau associé à cette adresse est à ce stade défini "en dur" dans le code de *xdm* à *255.255.255.0*, et ne peut pas être modifié par un paramètre.

Ainsi, le démon *xdm* assure la création de la cage au sens *vserver* lors de chaque ouverture de session. On notera que cette création est précédée de la réalisation des montages de session dans l'arborescence de la cage (et toujours dans l'espace de nommage du socle) par le module PAM *pam_exec_pwd*. Ces montages, ainsi que les autres opérations réalisées préalablement à l'ouverture de session sont décrits en détail dans [CLIP_1302].

5.2.2 Arrêt de la cage

La cage se termine et est détruite à la fin de chaque session graphique. Cette terminaison est normalement assurée par le script de session cliente, qui réalise un appel *kill -15 -1* ou équivalent avant de se terminer. Si cet appel échoue, les processus de la cage peuvent encore être terminés par la boucle de démontage des montages de session, appelée par *pam_exec_pwd* à la fermeture de session ([CLIP_1302]), qui tue en particulier tous les clients X11 (dans la mesure où ceux-ci utilisent tous le montage de session */tmp/.X11-unix*). On notera enfin que l'ouverture d'une nouvelle session graphique est dans tous les cas impossible tant que la cage $USER_{clip}$ est active, car le démon *xdm* enferme ses fils par des appels *clip_jailself()*, qui ne fonctionnent que si la cage destination n'existe pas encore. On garantit ainsi une stricte séparation entre les sessions utilisateurs successives : il est impossible de laisser un processus actif dans $USER_{clip}$ d'une session à l'autre.

5.2.3 Processus de la cage

Après enfermement dans la cage $USER_{clip}$, le fils de *xdm* chargé de l'ouverture de session cliente exécute le script de session par défaut, */usr/local/etc/X11/Sessions/CLIP* (*x11-apps/xinit*). Ce script lance plusieurs commandes, dont certaines sont communes à toutes les sessions, tandis que les autres sont spécifique aussi bien à un type de session (utilisateur, auditeur, administrateur) qu'à un type de système CLIP.

Processus communs

Le seul processus commun à l'ensemble des sessions est à ce stade le démon *xscreensaver*, qui est lancé avec une configuration spécifiant :

- un verrouillage de session automatique après trois minutes d'inactivité ;

- l'utilisation de l'exécutable externe */usr/local/bin/xscreensaver-pwcheck* pour la vérification du mot de passe utilisateur lors du déverrouillage de session. Cet exécutable dialogue avec le démon *pwcheckd* du socle sur la *socket unix /var/run/pwcheck* (dans l'arborescence de la cage). Voir aussi [CLIP_1302].

Session ADMIN

Une session ADMIN est lancée lorsque l'utilisateur connecté est membre du groupe *core_admin*. Dans cette session, un fond d'écran uniformément noir est chargé par une commande *xsetroot*, et le menu dédié de gestion des supports USB sécurisés, *usbmenu*, est lancé en tâche de fond. Puis la session exécute un terminal *xterm* à fond noir et bordures (et police) jaunes, maximisé dans le sens vertical, et centré sur l'écran en laissant des marges libres sur les côtés de manière à ne pas interférer avec l'affichage de *usbmenu*. Le terminal étant lancé par la commande *exec*, sa terminaison entraîne celle de la session. Ce terminal exécute directement une commande de connexion *ssh* sur la boucle locale, vers la cage $ADMIN_{clip}$, plus précisément vers l'utilisateur *_admin* à l'adresse *127.0.0.1*[20] (port 22). Plutôt que d'exécuter directement */usr/local/bin/ssh*, un script */usr/local/bin/ssh-session.sh*, qui appelle cette commande, est exécuté par le terminal. Ce script fait appel à *kill -15 -1* et *kill -9 -1* pour terminer tous les processus de la cage lorsque la commande *ssh* rend la main, c'est-à-dire lors de la fin de session au sein de $ADMIN_{clip}$, ou directement en cas d'échec de connexion à cette dernière. On notera qu'aucun gestionnaire de fenêtres n'est utilisé au sein de la session.

Cette session est commune à tous les types de systèmes CLIP.

Plusieurs commandes qui peuvent être lancées depuis la cage $ADMIN_{clip}$ mettent en oeuvre des clients graphiques. Dans la mesure où cette cage n'a pas accès au serveur X11, le traitement de telles commandes passe obligatoirement par le socle, typiquement sous la forme d'une requête à un démon du socle comme *usersrvadmin* ou *usbadmin_clip*. Ce démon lance ensuite un script de traitement de la requête dans le socle, qui peut ensuite lancer des clients graphiques dans $USER_{clip}$ (à afficher sur le même écran que la session en cours) par la procédure suivante :

- Récupération de l'utilisateur connecté en consultant le fichier */var/run/utmp* du socle, qui est systématiquement renseigné par *xdm* à l'ouverture et à la fermeture de session.
- Récupération du fichier *Xauthority* associé à la session de l'utilisateur. Ce fichier est normalement le seul fichier correspondant à */user/tmp/.Xauth**, dans la mesure où les fichiers *Xauthority* des sessions successives sont supprimés en fin de session par le démontage du */tmp*.
- Lancement d'une commande graphique dans la cage $USER_{clip}$, sous l'identité de l'utilisateur et en passant le chemin du fichier *Xauthority* en argument. Cette action peut être réalisée par une commande du type :

```
vsctl user enter -u <uid> -- <commande> <xauthority> <args>
```

Session AUDIT

Une session AUDIT est lancée lorsque l'utilisateur connecté est membre du groupe *core_audit* et pas du groupe *core_admin*. Cette session est similaire à la session AUDIT, à trois différences près :

[20] Cette adresse est automatiquement convertie en l'adresse principale de la cage $USER_{clip}$, c'est-à-dire USER_ADDR, qui est aussi l'adresse des cages $ADMIN_{clip}$ et $AUDIT_{clip}$.

- Le terminal *xterm* utilise des bordures et une police vertes, plutôt que jaunes.
- Le port de connexion *ssh* est 23 plutôt que 22, afin de contacter le serveur *sshd* de $AUDIT_{clip}$ plutôt que celui de $ADMIN_{clip}$.
- La connexion *ssh* se fait sous le compte *_audit* plutôt que *_admin*.

Cette session est commune à tous les types de systèmes CLIP.

Lorsque l'utilisateur qui se connecte n'est membre ni de *core_audit* ni de *core_admin*, une session de type utilisateur est lancée, avec un environnement graphique plus riche et sans connexion *ssh* aux autres cages CLIP. Cette session est différente selon le type de système CLIP : CLIP-RM ou CLIP-single (aucune session USER n'est supportée sur les passerelles CLIP-GTW).

Session USER dans CLIP-RM

Dans cette session, un gestionnaire de fenêtre *openbox* ([OPENBOX]), et un menu *fbpanel* ([FBPANEL]) sont lancés, afin d'offrir à l'utilisateur un environnement ergonomique permettant le lancement de sessions dans les cages RM (et fonctionnellement limité pour l'essentiel à ces tâches). Ces deux exécutables sont de plus modifiés spécifiquement dans CLIP de manière à afficher systématiquement le niveau de sécurité (niveau *Xsecurity* étendu) des différents clients. Cette modification est détaillée dans [CLIP_1303]. Un fond d'écran spécifique est aussi chargé par une commande *xloadimage*.

La configuration utilisée par *fbpanel* diffère selon que l'utilisateur est membre du groupe *rm_admin* (administrateur RM) ou non. Dans le premier cas, le fichier de configuration */usr/local/share/fbpanel/rmadm* est utilisé, tandis que */usr/local/share/fbpanel/default* est utilisé dans le second cas. La différence entre ces deux configurations concerne uniquement le contenu du menu *fbpanel*.

Dans tous les cas, le panneau *fbpanel* incorpore les éléments suivants :

- Un menu, permettant de réaliser les opérations suivantes :
  - Quitter la session. Dans ce cas, *fbpanel* réalise un appel *kill()* afin de terminer tous les autres clients de la session avant de se terminer lui-même.
  - Verrouiller la session, en appelant la commande *xscreensaver-command -lock*.
  - Changer le mot de passe du compte courant, en invoquant */usr/local/bin/userclt modify*. Ce client transmet la demande de changement de mot de passe au démon *usersrv* du socle (cf. 1.2), qui exécute alors le script */sbin/modify_user.sh*. Ce dernier crée plusieurs fenêtres *pop-up* successives dans $USER_{clip}$, selon la même méthode que celle décrite pour la cage $ADMIN_{clip}$.
- Un bouton lanceur par cage RM. Ces boutons spécifiques (*plugins rmlaunch* de *fbpanel*) assurent un suivi de l'état des tâches qu'ils lancent, et évitent ainsi de tenter de lancer une session RM alors qu'une autre session est déjà en cours (tentative qui échouerait de toutes manières). Le lancement de session est réalisé en invoquant le script */usr/local/bin/rm_X_session.sh*, qui va à son tour invoquer le client *jailrequest*, selon une procédure détaillée dans [CLIP_1401].

- Une barre de tâche, reprenant les labels de niveau de sécurité des fenêtres.
- Dans le cas des postes disposant d'une (ou plusieurs) batteries, un indicateur de charge batterie totale, dérivé du plugin *batt* de *lxpanel* ([LXPANEL]). Ce *plugin* assure aussi le lancement d'une commande *Xdialog* d'alerte lorsque le niveau de charge descend à 5%.
- Une horloge.

S'y ajoutent, lorsque l'utilisateur n'est pas membre du groupe *rm_admin*, un sous-menu de gestion de supports USB par cage RM. Chaque sous-menu permet de lancer les différentes opérations de gestion de supports USB sécurisés supportées par le démon *usbadmin*, pour le niveau de la cage RM concernée, en lançant des commandes *usbclt_rm_X* pour contacter ce démon. On notera que les supports USB de niveau CLIP ne sont pas utilisables depuis une telle session USER CLIP-RM, contrairement aux sessions ADMIN, AUDIT ou USER CLIP-single.

Session USER dans CLIP-single

Dans le cas d'un système CLIP-single, une session KDE ([KDE]) complète est lancée dans $USER_{clip}$, en exécutant la commande *startkde* après définition d'un certain nombre de variables significatives (KDEDIR, KDEDIRS, QT_DIR, HOME, LANG, etc...).

La session KDE CLIP-single est adaptée par l'installation du paquetage *clip-data/kde-config-clip*, qui installe une arborescence partielle de configuration KDE dans le répertoire */usr/local/etc/kde* (qui est ajouté à KDEDIRS, afin d'être pris en compte par KDE, cf. [KDE_ADM]). Cette arborescence ajuste certains paramètres par défaut, notamment :

- utilisation du double-clic plutôt que du simple-clic par défaut
- ajout d'*applets* adaptés au panneau *kicker* et d'icônes sur le bureau
- fond d'écran par défaut

Elle assure de plus le lancement et l'arrêt de *xscreensaver*.

Par ailleurs, le script *startkde* est adapté de manière à ne pas chercher à exécuter de fichiers ou scripts du répertoire *$HOME* de l'utilisateur (ce qui serait de toutes manières interdit par l'option *noexec* du montage correspondant, et l'utilisation du *flag O_MAYEXEC* par le *shell* de la cage), et à ne jamais lancer l'application *kpersonnalizer* de personnalisation initiale de l'environnement.

5.3 Vues visionneuses

5.3.1 Configuration

Montages

Les vues visionneuses sont des vues de la cage $USER_{clip}$, présentes uniquement dans les systèmes CLIP-RM. Elles constituent l'environnement d'exécution des visionneuses VNC utilisées pour afficher le bureau d'une session RM (cf. [CLIP_1303]) à travers une *socket unix*. Un système CLIP-RM intègre une vue visionneuse pour chaque cage RM, avec des arborescences distinctes entre vues. Ces différentes arborescences sont montées sur des sous-répertoires *rm_X* de */mounts/viewers*, monté en

/viewers dans l'arborescence de $USER_{clip}$. Le Tableau 9 décrit les différents montages constitutifs d'une telle arborescence. On notera plus particulièrement les points suivants :

- Chaque vue dispose d'un accès en lecture seule aux bibliothèques du socle, montées depuis */lib, /usr/lib* et */usr/local/lib*. En revanche, les seuls exécutables de la cage sont ceux qui sont inclus dans la racine de celle-ci.
- Aucun point de l'arborescence des vues n'est accessible en écriture depuis les vues.
- Le répertoire */tmp/.X11-unix* est partagé en lecture seule avec le socle, comme c'est le cas pour la cage $USER_{clip}$, afin de donner aux clients VNC un accès à la *socket* du serveur X11.
- Chaque vue partage en lecture seule un répertoire avec la cage RM associée, monté depuis le répertoire */var/run* de la vue USER de la cage RM. Ce répertoire contient la *socket* créée par le serveur VNC de la cage lors du lancement d'une session par *jailmaster* (cf. [CLIP_1401]), et son exposition dans la vue visionneuse permet au client de la vue de s'y connecter. En revanche, le cloisonnement des vues interdit par construction à une vue donnée de se connecter à une cage RM autre que celle à laquelle elle est ainsi associée. Ce répertoire partagé contient aussi, dans un fichier, le mot de passe d'authentification VNC reconnu par le serveur[21].
- Un sous-répertoire dédié du répertoire */xauth* de la cage $USER_{clip}$ est monté dans chaque vue. Ce répertoire est destiné à contenir le *cookie Xauthority* de la vue, qui est généré hors de la vue, dans le domaine de sécurité X11 associé à la cage RM concernée. La vue visionneuse n'a pas accès en écriture à ce *cookie*, et n'a accès en lecture à aucun autre *cookie*. Il est donc impossible depuis la vue de lancer un client graphique dans un domaine X11 autre que celui nominalement attribué à la vue, ce qui constitue l'une des principales raisons d'êtres de ces vues[22].
- Aucun périphérique n'est exposé dans les vues, qui ne disposent pas de */dev*.

Source	Point de montage	Type	Options
/etc/viewers/rm_X/fstab.root			
/tmp/.X11-unix	*/tmp/.X11-unix*	*bind*	*ro,nosuid,nodev,noexec,noatime,nolock*
/lib	*/lib*	*bind*	*ro,nosuid,nodev,noatime,nolock*
/usr/lib	*/usr/lib*	*bind*	*ro,nosuid,nodev,noatime,nolock*
/usr/local/lib	*/usr/local/lib*	*bind*	*ro,nosuid,nodev,noatime,nolock*
/mounts/xauth/rm_X	*/xauth*	*bind*	*ro,nosuid,nodev,noexec,noatime,nolock*
/etclviewerslrm_X/fstab.vserver			
/vservers/rm_X/ user_priv/var/run	*/vserver/run*	*bind*	*ro,nosuid,nodev,noexec,noatime*

Tableau 9: Montages de la vue visionneuse associée à une cage RM_X dans $USER_{clip}$.

Les sources sont relatives à la racine du système, les points de montages à la racine de la vue (*/viewers/rm_X* dans l'arborescence de $USER_{clip}$).

[21] Ce mot de passe n'assure aucune fonction de sécurité, dans la mesure où la connexion est purement locale. Le même mot de passe est employé sur toutes les installations CLIP.

[22] La défense en profondeur, en réduisant l'exposition du système aux clients VNC, qui interagissent directement avec les cages RM, constitue une autre motivation majeure de la création de ces vues.

Les montages des vues visionneuses sont créés de manière statique dans l'espace de nommage du socle (comme le reste de l'arborescence $USER_{clip}$) par le script de démarrage */etc/init.d/clip_viewers* (installé par *app-clip/clip-vserver*). Les montages à créer pour chaque vue *rm_X* (c'est-à-dire destinée à afficher le bureau de la cage RM_X) sont définis dans deux fichiers, */etc/viewers/rm_X/fstab.root* et */etc/viewers/rm_X/fstab.vserver*. Le premier de ces fichiers définit les montages à réaliser depuis le socle (les sources de montages sont relatives à la racine du socle), tandis que le second définit ceux qui sont réalisés depuis l'arborescence de la cage RM associée (les sources des montages sont relatives à la racine de la cage).

Fichiers

L'ensemble des exécutables disponibles dans une vue visionneuses est fourni par deux paquetages, *busybox-viewer* et *tightvnc.*

Le paquetage *busybox-viewer* est généré à partir de *sys-apps/busybox*, en positionnant *DEB_NAME_SUFFIX* à *"-viewer"* (cf. [CLIP_1101]), ce qui sélectionne automatiquement une configuration adaptée. La configuration *viewer* de *busybox* est très limitée, incluant uniquement un *shell ash* minimaliste et les quelques commandes (*[[*, *rm*, *basename, etc.*) utilisées par le script *viewer.sh* décrit ci-dessous. Le shell *busybox* utilise le drapeau O_MAYEXEC, spécifique à CLIP ([CLIP_1201]), lors de l'ouverture de fichiers, ce qui interdit l'exécution ou le "sourçage" de scripts depuis des montages portant l'option *noexec*.

Le paquetage *tightvnc* intégré aux vues est produit à partir de *net-misc/tightvnc*, en positionnant le drapeau USE *clip_vncviewer*. Ce drapeau limite l'arborescence installée par le paquetage à deux éléments, */bin/vncviewer* qui correspond à la visionneuse *TightVNC* ([TIGHTVNC]), et un script */bin/viewer.sh* qui effectue l'ensemble des traitements décrits en 5.3.2.

Un seul paquetage *tightvnc* (paquetage secondaire CLIP) fournit des fichiers disjoints pour l'ensemble des vues visionneuses, grâce à la mise en oeuvre du mécanisme *CLIP_VROOTS* ([CLIP_1101]).

5.3.2 Fonctionnement

La procédure complète de lancement d'une session RM est détaillée dans le document [CLIP_1401]. La présente description se limite au traitement effectué une fois une session RM_X lancée par une commande *jailmaster*. La vue RM_X correspondante est alors activée, en invoquant dans cette vue le script */bin/viewer.sh* qui réalise le traitement côté visionneuse. Ce script est lancé sous l'identité de l'utilisateur courant de $USER_{clip}$. Son lancement nécessite cependant de disposer de la capacité *CAP_SYS_CHROOT* nécessaire à l'opération *chroot()* d'enfermement dans la vue. A cette fin, la cage $USER_{clip}$ intègre un utilitaire *viewer-launch* (*app-clip/chroot-launch*), qui se voit attribuer la capacité *CAP_SYS_CHROOT* par une entrée *veriexec*. Cet utilitaire permet de lancer une commande "chrootée" dans un sous répertoire de */viewers*, selon la ligne de commande :

```
viewer-launch <path> <cmd> <args...>
```

avec :

- <*path*> le chemin relatif par rapport à */viewers* dans lequel lancer la commande. Le *chroot* est ainsi réalisé dans */viewers*<*path*>. Pour être accepté, <*path*> doit commencer par un '/', ne pas être limité à ce caractère, et ne pas contenir la chaîne "..".
- <*cmd*> la commande à lancer, qui doit être écrite comme un chemin absolu dans la prison *chroot*.
- <*args...*> une liste arbitraire d'arguments à passer à la commande.

Avant de réaliser l'appel *chroot()*, *viewer-launch* ferme tous les descripteurs de fichiers ouverts, crée une nouvelle session de processus et se détache de tout terminal de contrôle. Il révoque de plus toutes ses capacités après l'appel *chroot()* et avant d'exécuter <*cmd*>.

Afin de lancer une vue visionneuse, *viewer-launch* est appelé par le script */usr/local/bin/rm_X_session.sh* (*x11-misc/fbpanel*) correspondant à la cage RM cible, pour lancer */bin/viewer.sh* avec les paramètres suivants :

- Nom de la fenêtre visionneuse, typiquement "Visionneuse RM_X"
- Taille de la fenêtre visionneuse (fixée dans le script en fonction de la taille de l'écran, à l'aide d'un script *postinst* utilisant *screen-geom.eclass*, cf. [CLIP_1101])
- Un argument supplémentaire optionnel indique qu'une socket *vnc* existait dans le répertoire partagé entre la visionneuse et la cage RM, avant l'appel de *jailrequest* (ce qui signifie que l'existence du fichier n'équivaut pas à la présence d'un démon *Xvnc* en écoute)

Au préalable, le script *rm_X_session.sh* aura généré, dans */xauth/rm_X/xauthority*, un *cookie Xauthority* appartenant au domaine X11 approprié pour la cage RM_X.

Le script */bin/viewer.sh,* lorsqu'il est lancé sans son troisième paramètre indiquant la présence d'une *socket vnc* non significative, attend l'apparition de cette *socket* dans */vserver/run/vnc,* en répétant au maximum cinq fois une attente de deux secondes. Dans le cas contraire, le script "fait au mieux" en attendant trois secondes avant de poursuivre son exécution. Il positionne ensuite les variables *DISPLAY* (:0) et *XAUTHORITY* (*/xauth/xauthority*), avant de lancer *vncviewer* sur la socket *vnc /vserver/run/vnc/vnc1*, avec le mot de passe */vserver/run/vncpasswd*. Le script et la vue se terminent au retour de la commande *vncviewer.*

6 Cage X11

La cage X11 est entièrement réservée à l'exécution de serveur d'affichage X11 du système, qu'elle isole du reste du système afin de protéger symétriquement le démon vis-à-vis du reste du système, et le reste du système vis-à-vis du démon X11. Cette isolement s'apparente au concept de "zone démilitarisée" (DMZ) généralement employé dans le domaine de la sécurité réseau, pour isoler un serveur prestataire de service aussi bien pour un intranet sécurisé que pour un extranet non sécurisé. On pourra remarquer que le serveur X11 de CLIP est bien dans une situation comparable, puisqu'il est à la fois prestataire de service pour des clients du socle (*xdm*) que pour des clients de la cage $USER_{clip}$ (session utilisateur).

6.1 Configuration

6.1.1 Configuration *Vserver*

La configuration de la cage est définie directement par *xdm* Certains de ses paramètres (adresse, racine, *xid*) sont ajustables sous la forme de paramètres *Xressources* ; les autres (capacités) sont définis "en dur" dans l'exécutable *xdm*. Les principaux éléments de cette configuration sont les suivants :

- numéro de contexte *vserver* : 400 (*Xressource serverXid,* défini dans */usr/local/etc/X11/xdm-config*).
- racine : */x11* (*Xressource serverRoot,* défini dans */usr/local/etc/X11/xdm-config*).
- commande initiale non définie : un fils du démon *xdm* constitue le processus initial de la cage. Ce fils exécute ensuite un ensemble de clients, qui dépend du type de session.
- l'adresse de la cage est laissée à sa valeur par défaut *255.255.255.255* (*Xressource serverAddr*). Comme cette adresse n'est normalement attribuée à aucune interface, la cage X11 n'a aucun accès au réseau.
- Les capacités maximales autorisées dans la cage sont celles généralement autorisées dans les cages CLIP (cf. [CLIP_1202]), complétées de *CAP_SYS_BOOT et CAP_SYS_TTY_CONFIG,* qui sont nécessaires au fonctionnement de *Xorg*, soit :
  - *CAP_CHOWN*
  - *CAP_DAC_OVERRIDE*
  - *CAP_DAC_READ_SEARCH*
  - *CAP_FOWNER*
  - *CAP_FSETID*
  - *CAP_KILL*
  - *CAP_SETGID*
  - *CAP_SETUID*
  - *CAP_SYS_BOOT*

- *CAP_SYS_TTY_CONFIG*

- De même, les drapeaux de contexte *vserver* sont les drapeaux par défaut listés dans [CLIP_1202].

6.1.2 Système de fichiers

Les montages de la cage sont réalisés à partir d'un fichier *fstab* dédié, */etc/fstab.user*. Ces montages sont résumés dans le Tableau 10.

Source	Point de montage	Type	Options
/bin	*/bin*	*bind*	*ro,nosuid,nodev,noatime*
/lib	*/lib*	*bind*	*ro,nosuid,nodev,noatime*
/usr	*/usr*	*bind*	*ro,nodev,noatime*
/mounts/usr	*/usr/local*	*bind*	*ro,nodev,noatime*
/mounts/x11_priv/var	*/var*	*bind*	*rw,nosuid,nodev,noexec,noatime*
/var/run/authdir	*/var/run/authdir*	*bind*	*rw,nosuid,nodev,noexec,noatime*
/mounts/x11_priv/dev	*/dev*	*bind*	*rw,nosuid,noexec,noatime*
/tmp	*/tmp*	*bind*	*rw,nosuid,nodev,noexec,noatime, mode=1777*
/proc	*/proc*	*bind*	*ro,nosuid,nodev,noexec,noatime,nolock*

Tableau 10: Montages de la cage X11.
Les sources sont relatives à la racine du système, les points de montages à la racine de la cage (*/x11*)

A la différence des autres cages CLIP, la racine propre de la cage X11 est limitée à une arborescence de répertoires correspondants aux différents points de montage. Il n'y a en particulier pas de paquetage *busybox* ou *baselayout* dédié pour cette arborescence. La cage a accès en lecture seule à l'essentiel des exécutables et bibliothèques du socle. Elle dispose par ailleurs d'un */var* privé (contenant essentiellement le fichier de journaux */var/log/Xorg.0.log*, qui n'est pas consultable par la cage $AUDIT_{clip}$), et d'un */dev* privé. Ce dernier contient un ensemble de périphériques plus étendu que les autres cages CLIP, afin de permettre les accès matériels nécessaires au serveur X11. Ces périphériques, qui sont créés par le profil *generic-clip-x11* du script */sbin/MAKEDEV*, sont les suivants :

- *zero, null, full*
- *urandom*, ainsi qu'un lien symbolique *random* pointant sur *urandom*
- terminaux virtuels : *ttyX, vsc[a]X, console, tty*
- *framebuffer* vidéo : *fbX*
- périphériques d'entrées : *input/mouseX*

La cage partage avec le socle les répertoires */var/run/authdir* (en lecture seule, aussi exposé dans $USER_{clip}$), contenant le *cookie Xauthority* initial créé dans le socle par *xdm*, et */tmp* (en lecture-écriture,

dont le sous-répertoire *.X11-unix* est aussi exposé dans $USER_{clip}$ et ses vues), dans lequel *Xorg* crée sa *socket unix* d'écoute.

6.1.3 Configuration *Veriexec*

La cage X11 se voit associer un contexte *veriexec,* dont le niveau de sécurité est défini avec les drapeaux suivants (cf. [CLIP_1201]) par le script */etc/init.d/veriexec* (*app-clip/veriexec*) :

- *VRXLVL_ACTIVE*
- *VRXLVL_SELF_IMMUTABLE*
- *VRXLVL_LVL_IMMUTABLE*
- *VRXLVL_ENFORCE_MNTRO*

Ces drapeaux interdisent notamment toute opération d'administration de la base *veriexec* depuis la cage.

Le masque de capacités du contexte *veriexec* réduit les capacités attribuables dans la cage par *veriexec* à :

- *CAP_SYS_CHROOT*
- *CAP_SYS_BOOT*
- *CAP_SYS_TTY_CONFIG*

De même, le masque de privilèges CLSM du contexte *veriexec* réduit les privilèges attribuables dans la cage à :

- *CLSM_PRIV_PROCFD*
- *CLSM_PRIV_SIGUSR*

Les entrées *veriexec* suivantes sont définies dans le contexte de la cage :

- */usr/local/bin/Xorg* se voit attribuer *CAP_SYS_BOOT, CAP_SYS_TTY_CONFIG* et *CLSM_PRIV_SIGUSR* lorsqu'il est exécuté par *root* uniquement.

6.2 Démarrage et fonctionnement de la cage

6.2.1 Démarrage de la cage

Le script de démarrage *clip_x11* (*app-clip/core-services*) assure la création (dans l'espace de nommage du socle) des montages de la cage conformément au fichier */etc/fstab.x11*. La cage proprement dite est créée par le démon *xdm*, lors du lancement de celui-ci par le script */etc/init.d/xdm*. Le démon *xdm* crée un fils chargé du lancement du serveur d'affichage, qui configure immédiatement cette cage, avant de s'y enfermer et d'y exécuter le serveur *Xorg*. Ce dernier (défini dans */usr/local/etc/X11/xdm/Xservers*) est lancé avec les arguments *-br* (pour ne pas afficher la "mire" grise classique au démarrage) et *-nolisten tcp* (pour ne pas tenter d'ouvrir une *socket* d'écoute réseau, opération qui échouerait de toutes manières faute de privilèges CLIP-LSM suffisants). Le principe de lancement du serveur *Xorg* dans la cage X11 est résumé dans la Figure 3.

Le serveur X11 doit, après son lancement, signaler la fin de son initialisation par l'envoi d'un signal *SIGUSR* à son père (le démon *xdm*, ou dans d'autres cas *xinit*). Le changement de contexte *vserver* interdit a priori dans le cas de CLIP une telle opération. Afin d'éviter un longue attente de synchronisation entre *Xorg* et *xdm*, l'exécutable *Xorg* se voit attribuer par *veriexec* le privilège *CLSM_PRIV_SIGUSR*, permettant justement l'envoi d'un signal *SIGUSR* vers le contexte ADMIN. De manière symétrique, *xdm* se voit attribuer dans le socle le privilège *CLSM_PRIV_RECVSIG* lui permettant la réception d'un tel signal issu d'un autre contexte.

On notera que le serveur d'affichage *Xorg* est par ailleurs adapté de manière spécifique dans CLIP afin de permettre son lancement avec des privilèges réduits, en particulier sans l'accès direct au matériel qui lui permettrait sinon de sortir trivialement de sa cage.

6.2.2 Arrêt de la cage

L'arrêt complet de la cage est réalisé par une commande *vsctl x11 stop*, lancée par le script */etc/init.d/xdm* lors du traitement de sa fonction *stop()*.

Les montages de la cage X11 sont ensuite démontés dans le socle, lors de l'arrêt du script *clip_x11*.

6.2.3 Fonctionnement de la cage

En fonctionnement, la cage X11 est limitée au démon *Xorg*, qui reçoit des requêtes client sur la *socket unix /tmp/.X11-unix/X0*, qu'il expose dans le socle, la cage $USER_{clip}$ et les vues visionneuses de celle-ci.

Annexe A Références

[CLIP_1001] *Documentation CLIP – 1001 - Périmètre fonctionnel CLIP*

[CLIP_1002] *Documentation CLIP – 1002 – Architecture de sécurité*

[CLIP_1003] *Documentation CLIP – 1003 – Paquetages CLIP*

[CLIP_1101] *Documentation CLIP – 1101 – Génération de paquetages CLIP*

[CLIP_1201] *Documentation CLIP – 1201 – Patch CLIP-LSM*

[CLIP_1202] *Documentation CLIP – 1202 – Patch Vserver*

[CLIP_1203] *Documentation CLIP – 1203 – Patch Grsecurity*

[CLIP_1204] *Documentation CLIP – 1204 – Privilèges Linux*

[CLIP_1205] *Documentation CLIP – 1205 – Implémentation CCSD en couche noyau*

[CLIP_1301] *Documentation CLIP – 1301 – Séquences de démarrage et d'arrêt*

[CLIP_1302] *Documentation CLIP – 1302 – Fonctions d'authentification locale*

[CLIP_1303] *Documentation CLIP – 1303 – Cloisonnement graphique*

[CLIP_1401] *Documentation CLIP – 1401 – Cages RM*

[CLIP_1501] *Documentation CLIP – 1501 – Configuration réseau*

[CLIP_1502] *Documentation CLIP – 1502 – Mise en oeuvre de racoon2.*

[CLIP_DCS_13006] *Spécification fonctionnelle des outils de gestion des mises à jour, CLIP-ST-13000-006-DCS*

[CLIP_DCS_14009] *Spécification fonctionnelle des outils de mise à disposition des mises à jour, CLIP-ST-14000-009-DCS*

[CLIP_DCS_15088] *Conception de l'étude de la problématique de stockage sur support amovible, CLIP-DC-15000-088-DCS*

[CLIP_DCS_15093] *Conception de l'étude sur la définition des paramètres contrôlables par l'administrateur local, CLIP-DC-15000-093-DCS*

[CLIP_DCS_15094] *Conception de l'étude de retour à une configuration antérieure,*

CLIP-DC-1500-094-DCS

[BUSYBOX] *Busybox, http://busybox.net/*

[PRIVSEP] *Privilege Separated OpenSSH, http://www.citi.umich.edu/u/provos/ssh/privsep.html*

[PRIV] *Preventing Privilege Escalation,* Niels Provos, Markus Friedl et Peter Honeyman, *12th USENIX Security Symposium*. *http://www.citi.umich.edu/u/provos/papers/privsep.pdf*

[OPENBOX] *Openbox, http://icculus.org/openbox/*

[FBPANEL] *Fbpanel, http://fbpanel.sourceforge.net/*

[LXPANEL] *LXPanel, http://lxde.org/wiki/LXPanel*

[KDE] *K Desktop Environnement, http://www.kde.org*

[KDE_ADM] *KDE System Administration, http://techbase.kde.org/KDE_System_Administration*

[TIGHTVNC] *TightVNC, http://www.tightvnc.com*

Annexe B Liste des figures

Annexe C Liste des tableaux

Annexe D Liste des remarques