Documentation CLIP

1001 a

Périmètre fonctionnel CLIP-RM

Version	Date	Auteur	Commentaires
1.0.4	02/12/2008	Vincent Strubel	Correction de coquille dans le titre de la fonction de configuration des mises a jour.
1.0.3	01/10/2008	Vincent Strubel	Nouvelle convention de nommage des fonctions. Convention plus lisible pour les références.
1.0.2	18/06/2008	Vincent Strubel	Correction des références.
1.0.1	30/05/2008	Vincent Strubel	Numérotation 1001 a, pour attribuer le même numéro (1001 b, etc) aux autres descriptions fonctionnelles CLIP.
1.0	28/05/2008	Vincent Strubel	Actualisation pour CLIP v3.00.01 et passage au format OpenOffice.
0.91	29/06/2007	Vincent Strubel	Prises en compte des remarques d'Olivier Grumelard.
0.9	25/06/2007	Vincent Strubel	Première version préliminaire

Table des matières

Introduction

Le présent document constitue une description des différentes fonctions de base que doit assurer un système CLIP-RM, composé d'un socle CLIP et de deux cages RM accédant à des réseaux de niveau de sensibilité différents. Les fonctions individuelles sont identifiées par **F.XXX Nom de la fonction**, avec **XXX** un identifiant symbolique de la fonction. Ces fonctions sont classées selon les différents compartiments logiciels d'un système CLIP-RM, dont la composition est rappelée dans la première section du document. On notera que ces fonctions sont supposées automatiques (c'est-à-dire lancées sans intervention d'un utilisateur), sauf lorsqu'il est spécifié explicitement qu'elles sont interactives (lancées à l'initiative d'un utilisateur).

1 Description générale du système

1.1 Organisation locale du système

Le système est composé d'un **socle**, et d'un ensemble de **compartiments** logiciels. Ces compartiments sont de deux types : on appellera **cage** un compartiment correspondant à une instance *vserver*[1], et de ce fait cloisonné dans son accès à tous les types de ressources et d'objets du système, et **vue** un compartiment correspondant à un cloisonnement *chroot* qui se limite au contrôle de l'accès au système de fichiers. Le socle correspond au contexte *vserver* dit *ADMIN* et englobe tous les autres compartiments[2], mais est limité au plan fonctionnel au démarrage et à l'arrêt du système, à la gestion des ouvertures de session utilisateurs, et à la réalisation des fonctions de sécurité fondamentales du système. Le système est par ailleurs composé d'un ensemble de paquetages, répartis en deux catégories : les **paquetages secondaires**, qui peuvent être mis à jour au fil de l'eau durant le fonctionnement normal du système, et les **paquetages primaires**, dont la mise à jour requiert une interruption de ce fonctionnement normal et un redémarrage, soit du système complet, soit de la cage à laquelle ils s'appliquent.

Le système CLIP-RM comporte les cages suivantes[3] :

- Une cage **USER$_{clip}$** dédiée aux sessions utilisateurs. Les fonctionnalités de cette cage sont essentiellement réduites à l'ouverture de session dans les autres compartiments.
- Une cage **AUDIT$_{clip}$** dédiée à la collecte centralisée, à la consultation et à l'analyse des journaux de l'ensemble du système.
- Une cage **ADMIN$_{clip}$** dédiée à la configuration de certains paramètres fonctionnels de l'ensemble du système.
- Une cage **UPDATE$_{clip}$** dédiée au téléchargement des mises à jours de l'ensemble du système, et à l'application des mises à jours de paquetages secondaires du socle et des cages USER$_{clip}$, AUDIT$_{clip}$, ADMIN$_{clip}$ et UPDATE$_{clip}$. Cette cage fonctionne de manière entièrement automatique.
- Une cage **X11** dédiée au serveur graphique du poste, qui fonctionne de manière entièrement automatique.
- Une cage **RM_B**, destinée au traitement d'informations d'un niveau de sensibilité « bas ».
- Une cage **RM_H**, destinée au traitement d'informations d'un niveau de sensibilité « haut ».
- Deux cages **SECURE_UPDATE_RM_B** et **SECURE_UPDATE_RM_H**, superposées aux cages RM_B et RM_H respectivement, et utilisées uniquement pour la mise à jour des

[1] C'est-à-dire confiné dans un contexte de sécurité et un contexte réseau *vserver* propres à ce compartiment, et dans les espaces de nommage et le contexte *veriexec* associé.

[2] Dans la mesure où les arborescences de fichiers des autres compartiments sont incluses dans l'arborescence de fichiers du socle.

[3] Sauf mention explicite du contraire, ces cages sont toutes actives séparément lors du fonctionnement normal du système.

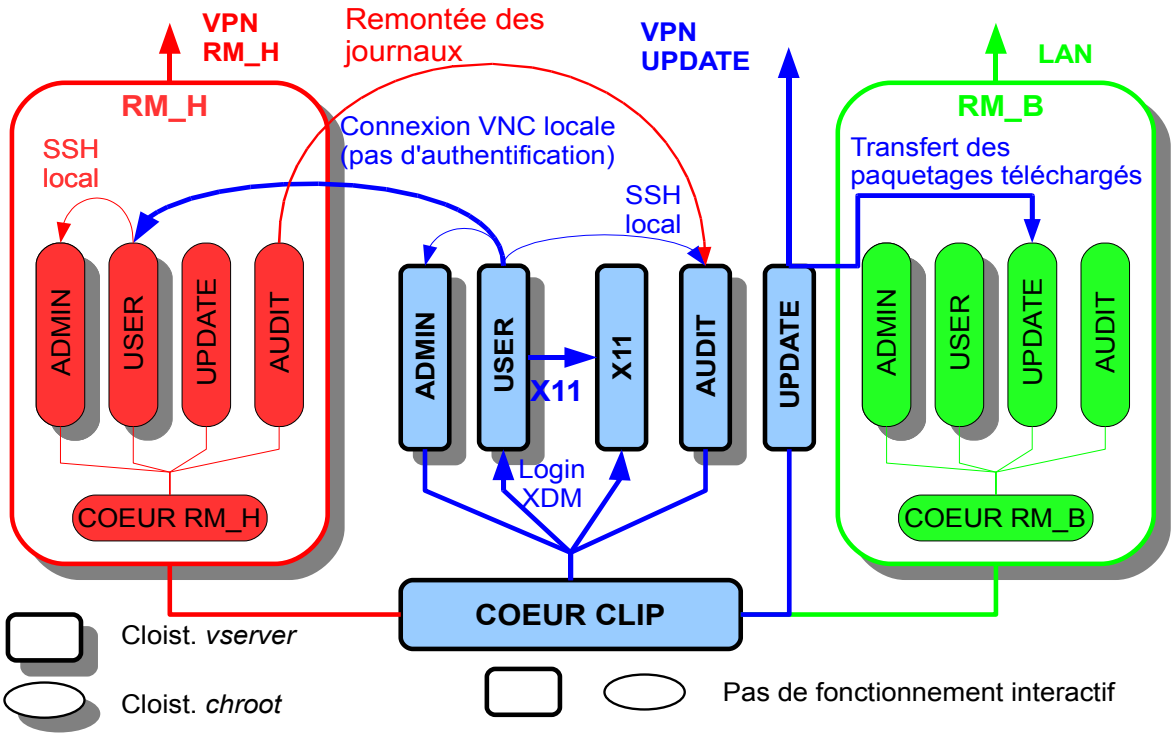
paquetages primaires de ces cages. L'activation de ces cages est exclusive de celle des cages RM, qui sont stoppées pendant leur mise à jour. Leur fonctionnement est entièrement automatique.

Chaque cage RM_H ou RM_B comporte de plus quatre vues :

- Une vue **USER**, dédiée à l'utilisation de la cage par un utilisateur final, incluant un environnement logiciel riche.
- Une vue **AUDIT**, dédiée à la collecte des journaux de la cage, toutes vues confondues, et à leur transfert vers la cage $AUDIT_{clip}$. Cette vue fonctionne de manière entièrement automatique.
- Une vue **ADMIN**, dédiée à la configuration de certains paramètres fonctionnels de la cage, toutes vues confondues.
- Une vue **UPDATE**, dédiée à l'application des mises à jour des paquetages secondaires de la cage, toutes vues confondues. Cette vue fonctionne de manière entièrement automatique.

Les paquetages du socle et des cages $USER_{clip}$, $AUDIT_{clip}$, $ADMIN_{clip}$, $UPDATE_{clip}$ et X11 sont issus d'une distribution commune[4], appelée **distribution CLIP**. Par extension, les cages $USER_{clip}$, $AUDIT_{clip}$, $ADMIN_{clip}$, $UPDATE_{clip}$ et X11 sont communément désignées **cages CLIP**. Les paquetages des deux cages RM_H et RM_B, toutes vues confondues, sont issus d'une seconde distribution, appelée **distribution RM**. Ces deux cages peuvent être désignées comme les **cages RM**.

[4] Les fichiers installés par un paquetage CLIP peuvent être utilisés à la fois par le socle et par plusieurs cages CLIP, le partage étant assuré par le partage des systèmes de fichiers sous-jacents. Le même principe permet de partager les paquetages RM entre toutes les vues d'une cage RM. Il n'y a en revanche pas de partage entre les deux cages RM : chaque paquetage RM est installé deux fois sur le système, une fois dans la cage RM_H et une fois dans la cage RM_B.



1.2 Environnement réseau

A chaque poste CLIP-RM sont associées sept adresses IP, qui correspondent toutes à des alias attribués à l'unique interface réseau du système. Chaque adresse est attribuée spécifiquement à une cage, pour laquelle elle constitue la seule adresse autorisée par le contexte réseau *vserver* associé à la cage. L'adresse réservée au socle[5] l'est par défaut, dans la mesure ou aucune des cages n'y a accès du fait du cloisonnement *vserver*. Ces sept adresses sont les suivantes :

- **CORE_ADDR**, réservée au socle du système.
- **USER_ADDR**, réservée à la cage $USER_{clip}$.
- **UPDATE_ADDR**, réservée à la cage $UPDATE_{clip}$.
- **AUDIT_ADDR**, réservée à la cage $AUDIT_{clip}$.
- **ADMIN_ADDR**, réservée à la cage $ADMIN_{clip}$.
- **RMH_ADDR**, réservée à la cage RM_H.

[5] Comme pour les autres ressources, le socle a implicitement accès à toutes les adresses locales, mais son périmètre fonctionnel se limite à l'utilisation de l'adresse qui lui est réservée.

- **RMB_ADDR**, réservée à la cage RM_B.

S'y ajoute l'adresse principale de la boucle locale, *127.0.0.1/8*, qui est aussi réservée au socle. L'adresse CORE_ADDR est la seule adresse routable sur le réseau local de déploiement.

Le schéma de déploiement réseau envisagé pour CLIP repose sur les propriétés suivantes :

- Le poste est déployé sur un réseau local de niveau « bas », **LAN_RM_B**.
- USER_ADDR, AUDIT_ADDR et ADMIN_ADDR sont égales, et purement locales. Les seules communications autorisées par le pare-feu du poste pour cette adresse sont les échanges sur la boucle locale.
- La cage RM_B a accès au réseau local de déploiement. Les paquets qui en sont issus font l'objet d'une transformation locale de type *NAT* (*Network Address Translation*), visant à leur donner l'adresse source routable CORE_ADDR, au lieu de l'adresse source d'origine RMB_ADDR. Le réseau local de niveau « bas » comporte un ou plusieurs serveurs avec lesquels interagissent les utilisateurs de la cage RM_B, en particulier un serveur de messagerie *IMAP(S)/SMTP* et un annuaire *LDAP(S)* (adresse **SVC_RMB**).
- La cage RM_H a accès à un réseau de niveau « haut » **LAN_RM_H**, à travers un tunnel IPSec établi sur le réseau LAN_RM_B entre le poste CLIP (adresse CORE_ADDR sur LAN_RM_B) et une passerelle chiffrante **GW_RM_H** (adresse **RMH_GW** sur LAN_RM_B). L'adresse RMH_ADDR est routable sur le réseau LAN_RM_H. Ce réseau comporte un ou plusieurs serveurs avec lesquels interagissent les utilisateurs de la cage RM_H, soit au minimum un serveur de messagerie *IMAP(S)/SMTP* et un annuaire *LDAP(S)* (adresse **SVC_RMH**).
- La cage $\text{UPDATE}_{\text{clip}}$ a accès à un réseau de mise à jour (de niveau équivalent au niveau « haut ») **LAN_UPDATE**, à travers un tunnel IPSec établi sur le réseau LAN_RM_B entre le poste CLIP (adresse IP_CORE sur LAN_RM_B) et une passerelle chiffrante **GW_UPDATE** (adresse **UPDATE_GW** sur LAN_RM_B). L'adresse UPDATE_ADDR est routable sur le réseau LAN_UPDATE. Ce réseau comporte les serveurs de services utilisés par la cage $\text{UPDATE}_{\text{clip}}$[6], dont au moins un serveur HTTPS de mise à disposition des mises à jour (adresse **SVC_UPDATE**), éventuellement complété d'un serveur NTP permettant la synchronisation horaire des postes clients (toutes cages confondues).

[6] Et par le reste du système, pour lequel la cage $\text{UPDATE}_{\text{clip}}$ joue le rôle de mandataire de téléchargement.

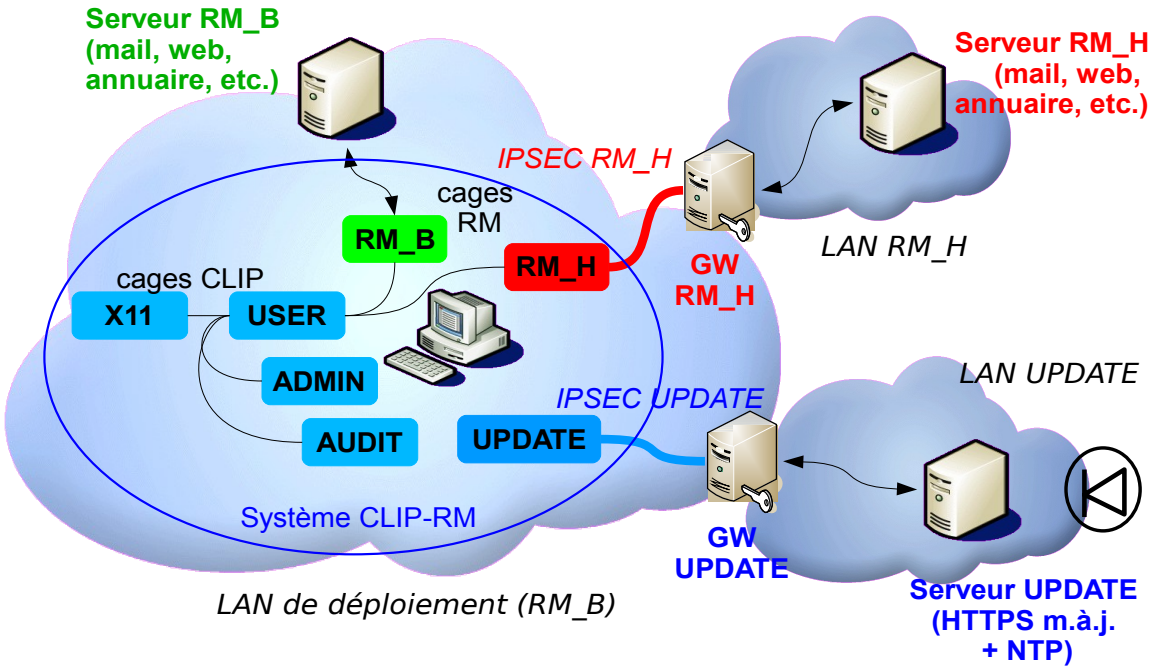


Figure 2: Environnement réseau d'un poste CLIP-RM.

1.3 Organisation sur le disque

A un instant donné, deux systèmes CLIP-RM complets sont installés sur le disque dur d'un poste, un en version (de la distribution CLIP) *n*, et l'autre en version *n-1*. Cette double installation permet de garantir une possibilité de retour en arrière après un échec dans la mise à jour du socle. Les deux installations partagent quatre partitions, stockant respectivement:

- les noyaux Linux et le chargeur de *boot* pour la première,
- l'ensemble des données utilisateur pour la deuxième,
- les journaux du système (toutes cages confondues) pour la troisième,
- le *swap* (chiffré) du système, toutes cages confondues, pour la quatrième.

Chaque système CLIP-RM dispose par ailleurs en propre de quatre partitions logiques sur le disque : une réservée aux fichiers installés par les paquetages primaires du socle, une partagée pour les fichiers installés par les paquetages secondaires du socle et les cages $USER_{clip}$, $AUDIT_{clip}$, $ADMIN_{clip}$, $UPDATE_{clip}$ et X11 (montée en */mounts* dans le système) et une pour chacune des cages RM_H et

RM_B (montées respectivement en */vservers/rm_h* et */vservers/rm_b*). Cette organisation sur le disque est résumée dans la Figure 3.

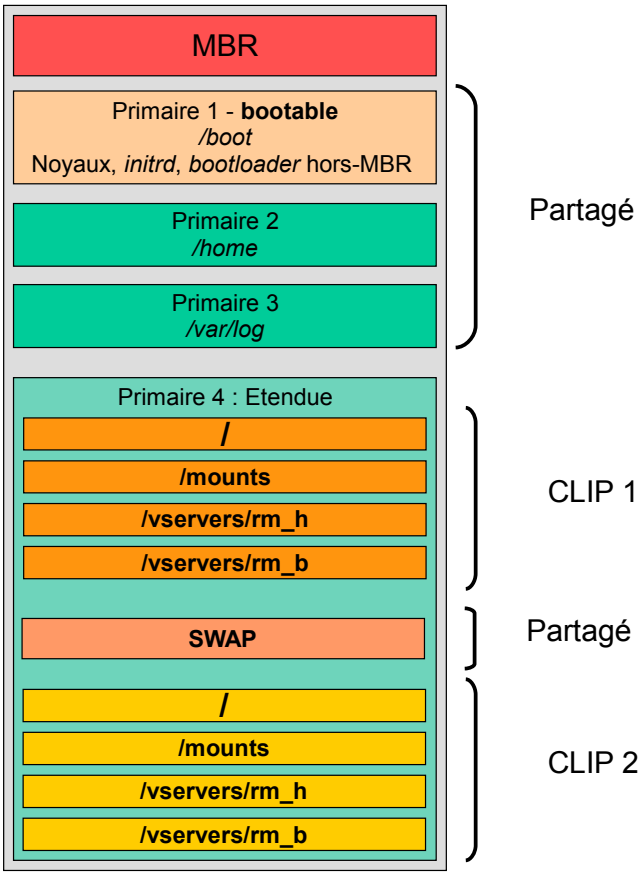


Figure 3: Organisation de deux systèmes CLIP-RM sur un disque dur.

2 Socle du système

Le socle intervient principalement lors du démarrage et de l'arrêt du système, ainsi que comme mandataire de sécurité pour les autres compartiments. Il possède aussi ses propres fonctions de mise à jour, et déclenche la mise à jour des paquetages primaires RM.

2.1 Démarrage du système

F.CORE_START_BOOT **Choix de l'installation de démarrage (interactif)**

Démarrer par défaut sur l'installation CLIP-RM la plus à jour présente sur le disque, mais proposer aussi une option de retour en arrière permettant le démarrage sur l'installation alternative.

F.CORE_START_INITSCRIPTS **Gestion des services**

Respecter les dépendances entre services de démarrage. Ces dépendances déterminent l'ordre de lancement des services, mais garantissent aussi qu'un service n'est pas lancé si ses dépendances ne sont pas satisfaites, et est arrêté si une de ses dépendances est arrêtée.

F.CORE_START_MOUNT **Montage des partitions**

Monter les différentes partitions du système après vérification éventuelle de l'intégrité[7] des systèmes de fichiers, réaliser les montages du socle.

F.CORE_START_SWAP **Création du swap**

Configurer et activer un *swap* chiffré avec une clé tirée aléatoirement.

F.CORE_START_LOGROTATE **Gestion initiale des journaux**

Créer les fichiers de journaux avec les droits et attributs appropriés, si de tels fichiers n'existent pas encore. Le cas échéant, effectuer une rotation des journaux existants et éventuellement la suppression des journaux les plus anciens. Les attributs des fichiers de journaux doivent interdire leur modification (y compris à l'utilisateur *root* du socle) autrement qu'en mode *append* après l'application de [F.CORE_START_SECLEVEL].

F.CORE_START_BACKUP **Sauvegarde du système ou des données**

Si une sauvegarde du système ou des données a été demandée (cf. [F.ADMIN_CONF_BACKUP]) par l'administrateur avant l'arrêt précédant du poste, réaliser cette sauvegarde et la stocker dans la partition dédiée aux sauvegardes système. Au besoin, réaliser la sauvegarde du système courant depuis l'installation alternative (cf. 1.3), en configurant le chargeur de démarrage pour démarrer par défaut sur la partition racine alternative, et en redémarrant immédiatement. La partition de démarrage originale est dans ce cas rétablie après traitement de l'opération de sauvegarde sur la partition alternative.

[7] Au sens *fsck*, et non cryptographique.

F.CORE_START_ROLLBACK Restauration du système ou des données

Si une restauration du système ou des données a été demandée (cf. [F.ADMIN_CONF_BACKUP]) par l'administrateur avant l'arrêt précédant du poste, réaliser cette restauration, sur les partitions courantes ou alternatives. Au besoin, réaliser la restauration sur les partitions courantes depuis l'installation alternative (cf. 1.3), en configurant le chargeur de démarrage pour démarrer par défaut sur la partition racine alternative, et en redémarrant immédiatement. La partition de démarrage originale est dans ce cas rétablie après traitement de l'opération de sauvegarde sur la partition alternative. Lors d'une opération de restauration, les fichiers localement modifiés par un administrateur (CLIP ou RM) sont sauvegardés dans leur version courante avant restauration, avant d'être écrasés par la restauration.

F.CORE_START_UPDATE Mise à jour initiale du socle

Le cas échéant, appliquer une mise à jour du socle avec basculement vers l'installation CLIP-RM alternative, et redémarrage immédiat (cf. [F.CORE_UPDATE_CORE]).

F.CORE_START_VERIEXEC Configuration de *veriexec*

Créer tous les contextes *veriexec* du système, charger les entrées *veriexec* du socle et des cages CLIP. Activer *veriexec* dans le socle et les cages CLIP.

F.CORE_START_VSERVER Configuration *vserver* initiale

Configurer la visibilité des fichiers du */proc*. Seul */proc/uptime* et les fichiers de */proc/acpi* nécessaires à la surveillance de la charge batterie sont rendus visibles dans les cages.

F.CORE_START_NETWORKING Configuration réseau initiale

Configurer le filtrage réseau (*iptables*), les variables *sysctl* réseau et les politiques de sécurité (*SP*) IPSec. Les adresses IP à utiliser pour cette configuration sont importées de manière sécurisée depuis un fichier de configuration modifiable par l'administrateur local dans la cage $ADMIN_{clip}$ (cf. [F.ADMIN_CONFIG_NET_ADDR]).

F.CORE_START_ETHERNET Activation des interfaces réseau

Activer la boucle locale et l'interface *ethernet* en leur attribuant les différentes adresses décrites en 1.2. Les adresses IP à utiliser pour cette configuration sont importées de manière sécurisée depuis un fichier de configuration modifiable par l'administrateur local dans la cage $ADMIN_{clip}$ (cf. [F.ADMIN_CONFIG_NET_ADDR]).

F.CORE_START_SECLEVEL Verrouillage du système

Configurer les *sysctl kernel.clip.** et *kernel.cap-bound* pour verrouiller le niveau de sécurité du système.

F.CORE_START_JAILS_CLIP Lancement des cages CLIP sauf X11

Configurer les cages $USER_{clip}$, $AUDIT_{clip}$, $ADMIN_{clip}$, $UPDATE_{clip}$ (création des espaces de nommage et montages, configuration des contextes *vserver*).

F.CORE_START_VIEWERS Configuration des vues visionneuses RM

Configurer dans la cage $USER_{clip}$ les vues visionneuses utilisées pour la mise en oeuvre des cages RM, en créant les montages correspondant (cf. [F.USER_SESSION_RM_START]).

F.CORE_START_JAILS_RM Lancement des cages RM

Après avoir procédé à l'éventuelle mise à jour des coeurs de ces cages (cf. [F.CORE_UPDATE_RM_CORE]), charger les entrées *veriexec* de ces cages, y activer *veriexec*, configurer les cages (montages, contextes *vserver*) et lancer le démon *jailmaster* dans chacune d'entre elles.

F.CORE_START_IKE Négociation des associations de sécurité

Lancer le démon *racoon2* et négocier les associations de sécurité nécessaires aux tunnels IPSEC_RMH et IPSEC_UPDATE. Les adresses IP utilisées pour la configuration de *racoon2* sont importées de manière sécurisée depuis un fichier de configuration modifiable par l'administrateur local dans la cage $ADMIN_{clip}$ (cf. [F.ADMIN_CONFIG_NET_ADDR]). De même, la clé privée CCSD utilisée pour authentifier le poste auprès des passerelles est lue depuis un fichier susceptible d'être modifié par l'administrateur local. Cette lecture s'accompagne de la mise en oeuvre d'un verrouilllage permettant d'éviter de lire un fichier en cours de modification.

F.CORE_START_XDM Lancement du démon d'ouverture de session

Lancer le serveur graphique dans la cage X11 et le démon *xdm* dans le socle.

2.2 Arrêt ou redémarrage du système

F.CORE_STOP_JAILS Arrêt des cages

Terminer les cages RM_H, RM_B, $USER_{clip}$, $ADMIN_{clip}$, $UPDATE_{clip}$ et X11, et enfin $AUDIT_{clip}$, cette dernière n'étant terminée que juste avant le [F.CORE_STOP_MOUNT] et l'arrêt du système.

F.CORE_STOP_MOUNT Démontage des partitions

Démonter les partitions montées, ou remonter en lecture-seule celles qui ne peuvent pas être démontées.

2.3 Fonctions réseau

F.CORE_NETWORK_ROUTE Routage des flux sortants

Router au besoin les flux issus du système (toutes cages confondues) vers la passerelle par défaut.

F.CORE_NETWORK_SNAT Traduction d'adresse des flux RM_B

Réaliser la traduction d'adresse des flux issus de la cage RM_B, pour leur donner l'adresse source CORE_ADDR.

2.4 Ouverture et fermeture de session

F.CORE_SESSION_PASS Vérification de mot de passe *(interactif)*

Vérifier le mot de passe d'un utilisateur en le comparant à son empreinte cryptographique, sauvegardée par le socle.

F.CORE_SESSION_MOUNT Montage des partitions utilisateur

Lorsqu'un utilisateur valide et membre du groupe *crypthome* entre le mot de passe correct dans la fenêtre d'ouverture de session *xdm*, monter les partitions suivantes :

- Les partitions chiffrées RM_H et RM_B de l'utilisateur, montées sur les répertoires */home/user* des arborescences des vues USER des cages RM_H et RM_B (dans leurs *namespaces VFS* respectifs).
- Deux systèmes de fichiers *tmpfs* sur les répertoires */tmp* des arborescences des vues USER des cages RM_H et RM_B respectivement (dans leurs *namespaces VFS* respectifs).
- La partition chiffrée CLIP de l'utilisateur, montée sur le répertoire */home/user* de l'arborescence de la cage $USER_{clip}$ (dans le *namespace VFS* du socle, la cage $USER_{clip}$ n'étant pas encore créée à cet instant).
- Un système de fichiers *tmpfs* sur le répertoire */tmp* de l'arborescence de la cage $USER_{clip}$ (similairement monté dans *namespace VFS* du socle).

Les montages de partitions chiffrées sont effectués en utilisant le mot de passe de l'utilisateur, tel qu'il a été entré dans la fenêtre d'ouverture de session *xdm*, pour déchiffrer la clé de chiffrement de la partition. La partition n'est pas montée si ce déchiffrement échoue.

F.CORE_SESSION_OPEN Ouverture de session

Créer la cage $USER_{clip}$ et lancer la session graphique CLIP dans la cage $USER_{clip}$ lorsqu'un utilisateur valide entre le mot de passe correct dans la fenêtre d'ouverture de session *xdm* dans le socle et que [F.CORE_SESSION_MOUNT] a été réalisée avec succès. La connexion doit être refusée de manière explicite si le nom d'utilisateur n'est pas valide ou si le mot de passe est erroné. Dans ce cas, aucun programme n'est lancé et la cage $USER_{clip}$ n'est pas créée.

F.CORE_SESSION_CLOSE Fermeture de session

Lorsque la session graphique CLIP se termine dans la cage $USER_{clip}$, démonter tous les montages réalisés par [F.CORE_SESSION_MOUNT] et [F.CORE_USB_MOUNT], au besoin en tuant les processus les utilisant encore.

2.5 Fonctions de sécurité

F.CORE_SEC_JAIL_LOCAL Cloisonnement local des cages

Les ressources mises en oeuvre dans une cage ne sont pas visibles ni accessibles depuis les autres cages, sauf en ce qui concerne les fichiers explicitement exposés dans plusieurs cages.

F.CORE_SEC_JAIL_PRIV Limitation des capacités par cages

Les capacités POSIX dont peut disposer un utilisateur, y compris *root*, d'une cage, sont bornées par la limite définie dans la configuration de la cage, et par la limite globale (*cap-bound*) du système.

F.CORE_SEC_JAIL_NET Cloisonnement réseau des cages

Les processus d'une cage ne peuvent recevoir que les paquets destinés à l'adresse de la cage, et ne peuvent pas émettre de paquets ayant une adresse source différente de l'adresse de la cage.

F.CORE_SEC_JAIL_VIRT Virtualisation par cage

Les ressources partagées par tout le système (mémoire disponible, temps) sont virtualisées en fonction de la configuration de la cage.

F.CORE_SEC_MEM Gestion par défaut des droits sur la mémoire

Un processus ne peut pas exécuter une zone mémoire à laquelle il a ou a eu accès en écriture, sauf dérogation spécifique inscrite dans les drapeaux *PT_PAX_FLAGS* d'un exécutable.

F.CORE_SEC_RANDMMAP Randomisation des projections mémoire

Les projections en mémoire par *mmap()* sont par défaut réalisées à des adresses aléatoires.

F.CORE_SEC_MEMZERO Effacement de la mémoire après utilisation

La mémoire est remise à zéro dès sa désallocation.

F.CORE_SEC_ROOTPRIV Limitation des privilèges par défaut de *root*

Les capacités POSIX attribuées par défaut à *root* sont limitées à la valeur configurée par le *sysctl kernel.clip.rootcap*, sauf lors de l'exécution de binaires privilégiés par *veriexec*. Aucun privilège CLSM n'est attribué par défaut.

F.CORE_SEC_VERIEXEC Attribution de privilèges par *veriexec*

Veriexec permet d'attribuer des capacités *POSIX*, sans dépasser les limites de [F.CORE_SEC_JAIL_PRIV], ou des privilèges CLSM aux exécutables, et ce de manière indépendante d'une cage à l'autre. Cette attribution est conditionnée à la vérification de l'empreinte cryptographique (*hash*) par rapport à l'empreinte stockée dans l'entrée *veriexec* correspondante, et éventuellement à d'autres restrictions (processus appelant d'identité *root*, lancement de l'exécutable par un *thread* noyau, vérification de l'empreinte de l'éditeur de liens et des bibliothèques mises en oeuvre par l'exécutable.

F.CORE_SEC_DEVCTL Contrôle des écritures sur le disque

Une fois le système démarré (dans le *runlevel « default »*), il est rigoureusement impossible de modifier le contenu de la partition racine, de celles de l'installation CLIP alternative et des sauvegardes des données ou du système, ou de celle contenant les noyaux et le chargeur de démarrage.

F.CORE_SEC_NET_ACCESS Contrôle de l'accès au réseau

Seuls les exécutables qui y sont explicitement autorisés par *veriexec* peuvent accéder au réseau.

F.CORE_SEC_NET_FILTER Filtrage réseau

Filtrer les flux réseau cage par cage, aussi bien sur l'interface *ethernet* que sur la boucle locale. Les seules connexions sortantes autorisées sont celles définies dans le fichier de configuration du pare-feu, qui est modifiable localement au sein de $ADMIN_{clip}$ (cf. [F.ADMIN_CONFIG_NET_FILTER]), et importé de manière sécurisée. Aucune connexion entrante n'est autorisée sur l'interface *ethernet*. Seuls les processus exécuté dans RM_B ou dans le socle peuvent émettre des paquets en clair (c'est-à-dire sans encapsulation IPSec).

F.CORE_SEC_NET_IPSEC Chiffrement des flux réseau

Établir et maintenir les associations de sécurité nécessaires aux tunnels IPSEC_RMH et IPSEC_UPDATE. Réaliser l'encapsulation IPSec de tous les flux issus des cages RM_H et $UPDATE_{clip}$.

F.CORE_SEC_DISK_ENCRYPT Chiffrement de disque

Chiffrer le *swap* du système. Chiffrer utilisateur par utilisateur les partitions de données. Ne déchiffrer une partition utilisateur que durant une session de l'utilisateur correspondant. Chiffrer les supports amovibles. Voir aussi [F.CORE_START_SWAP], [F.CORE_SESSION_MOUNT] et [F.CORE_USB_MOUNT].

F.CORE_SEC_LOG_KERN Journalisation noyau

Journaliser les événements sensibles au niveau du noyau, et les rendre consultables par l'interface */proc/kmsg*. Les principaux événements journalisés sont :

- Initialisation des différents pilotes de périphériques
- Montages et démontages de systèmes de fichiers
- Créations et suppressions de ressources IPC (*SystemV*)
- Modification de l'heure système
- Envois de certains signaux, en particulier *SIGSEGV*, par le noyau
- Violations des contraintes *PaX*
- Violations des contraintes *vserver*
- Appels système interdits par *CLIP-LSM*
- Erreurs de vérification *veriexec*

F.CORE_SEC_LOG_USER Journalisation système

Journaliser sur une *socket syslog* les événements sensibles au niveau des démons et applicatifs du socle, en particulier :

- Authentifications *PAM* réussies et échouées (ouverture de session, déverrouillage de session)
- Ouvertures et fermetures de sessions *PAM* et *xdm*
- Lancement et arrêt des démons
- Arrêt du système
- Négociations d'associations de sécurité IPSec
- Changements des mots de passe utilisateurs
- Arrêt d'un processus suite à une violation *SSP/Propolice*

Cette fonction est aussi réalisée indépendamment dans chaque cage du système.

2.6 Fonctions de mise à jour

F.CORE_UPDATE_CORE Mise à jour du socle

Mettre à jour les paquetages primaires du socle et des cages CLIP du système en installant les nouvelles versions sur les partitions de l'installation CLIP alternative. La double signature des paquetages est systématiquement vérifiée. Les paquetages utilisés sont ceux mis à disposition par la cage $\text{UPDATE}_{\text{clip}}$ (cf. [F.UPDATE_MIRROR_SYNC] et [F.UPDATE_MIRROR_CDROM]). Cette opération n'est réalisée qu'au démarrage du système, lors du traitement de [F.CORE_START_UPDATE].

F.CORE_UPDATE_BOOT Mise à jour du chargeur de démarrage

Mettre à jour le chargeur de démarrage de manière à démarrer automatiquement sur la version la plus à jour du système. Cette opération n'est réalisée qu'au démarrage du système, lors du traitement de [F.CORE_START_UPDATE].

F.CORE_UPDATE_RECOVER Reprise sur erreur

Reprendre le traitement des fonctions [F.CORE_UPDATE_CORE] à [F.CORE_UPDATE_BOOT] à la suite d'une erreur ou d'une interruption dans leur traitement. Cette opération n'est réalisée qu'au démarrage du système, lors du traitement de [F.CORE_START_UPDATE].

F.CORE_UPDATE_RM_CORE Mise à jour des paquetages primaires RM

Lancer la mise à jour des paquetages primaires des cages RM avant le démarrage de ces cages, en créant des cages SECURE_UPDATE_RM similaires aux cages RM mais avec des droits plus importants sur le système de fichier et en y lançant la procédure de mise à jour adaptée.

Remarque 1 : Mise à jour périodique des paquetages primaires des cages RM

Outre la mise à jour au démarrage de la cage, il serait souhaitable de procéder périodiquement à la mise à jour des paquetages primaires de cages RM, en testant la disponibilité de telles mise à jour et, le cas échéant, en arrêtant temporairement les cages

RM pour relancer les cages SECURE_UPDATE_RM associées. Une telle mise à jour périodique devrait cependant n'être lancée que lorqu'aucun utilisateur n'a de session $USER_{clip}$ ouverte sur le système.

F.CORE_UPDATE_LOG Journalisation des mises à jour

Journaliser les opérations de mise à jour réussies, ainsi que les échecs, en particulier les échecs de vérification de signature (ces derniers doivent être facilement détectables par des outils d'analyse automatique des journaux).

2.7 Gestion des supports amovibles

F.CORE_USB_GEN_KEY Génération des clés de supports (*interactif*)

Générer, sur demande de l'utilisateur, une paire de bi-clés RSA utilisés pour, respectivement, signer des en-têtes de supports amovible, et chiffrer une clé symétrique stockée dans un tel en-tête et permettant de déchiffrer le reste du support. Ces clés sont associées à un niveau de sensibilité (CLIP, RM_H ou RM_B), et stockées dans la partition chiffrée de même niveau de l'utilisateur. Les deux mots de passes par lesquels sont protégées les deux clés privées de ces bi-clés sont saisis et confirmés par l'utilisateur à l'aide d'une série de quatre *pop-ups* graphiques lancés dans $USER_{clip}$.

F.CORE_USB_EXPORT_KEY Export des clés de supports (*interactif*)

Exporter, sur demande de l'utilisateur, la paire de bi-clés RSA de support amovibles de cet utilisateur à un niveau donné. Les clés sont exportées sous la forme d'une archive chiffrée en mode symétrique, avec une clé symétrique tirée aléatoirement. Cette archive est copiée sur un support amovible (non chiffré et non authentifié, contenant une partition formatée en FAT32) connecté au poste, de même que sa clé symétrique de chiffrement, elle-même chiffrée en mode asymétrique à l'aide d'une clé publique du poste, commune à tous les utilisateurs mais propre à un niveau. Une confirmation explicite est demandée à l'utilisateur avant l'export, sous la forme d'un *pop-up* graphique lancé dans la cage $USER_{clip}$. L'utilisateur doit naturellement disposer d'une session en cours dans $USER_{clip}$ avant tout export.

F.CORE_USB_AUTH_DECRYPT Authentification et déchiffrement de support

Alors qu'un utilisateur dispose d'une session dans $USER_{clip}$, détecter la connexion d'un support amovible sécurisé, et vérifier l'authenticité de son secteur de méta-données avec la clé publique de signature RSA de l'utilisateur, de niveau correspondant au niveau écrit dans les méta-données du support. En cas de succès de cette vérification, déchiffrer, à l'aide de la clé privée de chiffrement RSA de l'utilisateur au même niveau, la clé symétrique de chiffrement du support stockée parmi les méta-données de celui-ci, puis créer, à l'aide de cette dernière clé symétrique, une projection déchiffrée (non-montée automatiquement) du support. Le mot de passe de la clé privée RSA de chiffrement de l'utilisateur est demandée à l'aide d'un *pop-up* graphique lancé dans $USER_{clip.}$

F.CORE_USB_MOUNT Montage d'un support authentifié (*interactif*)

Monter, sur demande de l'utilisateur, un support amovible sécurisé, précédemment authentifié et

déchiffré par [F.CORE_USB_AUTH_DECRYPT]. Le montage d'un support de niveau RM_H ou RM_B est réalisé dans la vue USER de la cage RM correspondante. Le montage d'un support de niveau CLIP est réalisé dans la cage $AUDIT_{clip}$, $ADMIN_{clip}$ ou $USER_{clip}$, selon que l'utilisateur courant possède un profil auditeur, administrateur, ou simple utilisateur.

F.CORE_USB_UMOUNT — Démontage d'un support (*interactif ou non*)

Démonter, sur demande de l'utilisateur ou automatiquement lors de la fin de session $USER_{clip}$ de ce dernier (cf. aussi [F.CORE_SESSION_CLOSE]), tout support amovible sécurisé, précédemment monté par [F.CORE_USB_MOUNT].

F.CORE_USB_UNMAP — Suppression de la projection claire d'un support

Supprimer automatiquement lors de la fin de session $USER_{clip}$ d'un utilisateur, toute projection claire de support amovible chiffré créée par [F.CORE_USB_AUTH_DECRYPT] pendant la session de cet utilisateur, afin d'éviter d'exposer ces projections à un utilisateur suivant.

F.CORE_USB_INIT — Initialisation de supports amovibles (*interactif*)

Initialiser à un niveau donné, sur demande de l'utilisateur, un support de stockage amovible connecté au système. Cette initialisation consiste à :

- Générer un clé symétrique aléatoire.
- Initialiser un système de fichiers chiffré à l'aide de cette clé symétrique.
- Créer un secteur de méta-données, contenant le niveau du support et la clé symétrique, chiffrée à l'aide de la clé privée RSA de chiffrement de l'utilisateur au niveau concerné, ainsi qu'une signature de ces deux premiers éléments par la clé privée RSA de signature de l'utilisateur au niveau concerné.
- Inscrire ce secteur de méta-données en début et en fin du support amovible.

Les mots de passes protégeant les clés privées RSA (signature et chiffrement) sont demandés à l'utilisateur par des *pop-ups* graphiques lancés dans $USER_{clip}$.

3 Cages CLIP

3.1 Cages USER$_{clip}$ et X11

Ces cages servent uniquement de relais pour l'accès à d'autres cages et l'affichage des sessions correspondantes. On distingue trois types de sessions possibles au sein de la cage USER$_{clip}$: la session AUDIT, permettant l'accès exclusif et interactif à la cage AUDIT$_{clip}$, la session ADMIN, permettant l'accès exclusif et interactif à la cage ADMIN$_{clip}$, et la session USER, permettant l'accès simultané et interactif aux vues USER des cages RM_H et RM_B.

F.USER_SESSION_SELECT **Choix de Session**

Choisir le type de session selon les groupes auxquels appartient l'utilisateur qui lance une session dans la cage. Si l'utilisateur appartient au groupe *core_admin*, une session ADMIN est lancée. Sinon, si l'utilisateur appartient au groupe *core_audit*, une session AUDIT est lancée. Par défaut, si l'utilisateur n'appartient à aucun de ces deux groupes (c'est-à-dire, dans le cas d'un utilisateur « normal », ou d'un utilisateur ayant le rôle d'administrateur RM), une session USER est lancée.

F.USER_SESSION_START **Lancement de session**

Les trois types de sessions correspondent à des sessions graphiques, clientes du serveur X11 lancé par *xdm* dans la cage X11. La cage USER$_{clip}$ est créée par *xdm* à chaque ouverture de session, et disparaît à la fermeture de session, à la différence des autres cages du système qui sont maintenues actives en permanence par au moins un démon.

F.USER_SESSION_ENV **Configuration de l'environnement**

Configurer certaines variables d'environnement, avant d'exécuter la session choisie. En particulier, *$HOME* doit être défini à */home/user*, qui correspond au point de montage de la partition chiffrée utilisateur. Avant de lancer une session USER, les variables *$PATH, $LANG* et *$LC_ALL* sont aussi configurées.

F.USER_SESSION_AUDIT **Session utilisateur AUDIT *(interactif)***

La session AUDIT se résume à un émulateur de terminal en plein écran et sans gestionnaire de fenêtre, qui exécute directement un client *ssh* pour ouvrir une session AUDIT$_{clip}$ (connexion *audit* sur le port 23 sur l'adresse locale, et à un menu graphique permettant la gestion des supports amovibles sécurisés (cf. fonctions interactives de 2.7). L'utilisateur doit immédiatement se ré-authentifier en déverrouillant sa clé privée *ssh*. La session se termine en même temps que le client *ssh*, c'est-à-dire lorsque la session dans la cage AUDIT$_{clip}$ se termine. Le terminal d'une session AUDIT utilise une police verte.

F.USER_SESSION_ADMIN **Session utilisateur ADMIN *(interactif)***

La session ADMIN se résume à un émulateur de terminal en plein écran et sans gestionnaire de fenêtre, qui exécute directement un client *ssh* pour ouvrir une session ADMIN$_{clip}$ (connexion *admin* sur le port

22 sur l'adresse locale), et à un menu graphique permettant la gestion des supports amovibles sécurisés (cf. fonctions interactives de 2.7). L'utilisateur doit immédiatement se ré-authentifier en déverrouillant sa clé privée *ssh*. La session se termine en même temps que le client *ssh*, c'est-à-dire lorsque la session dans la cage $ADMIN_{clip}$ se termine. Le terminal d'une session ADMIN utilise une police jaune.

F.USER_SESSION_USER Session utilisateur USER *(interactif)*

La session USER lance automatiquement un gestionnaire de fenêtre et le menu *fbpanel* permettant de lancer des sessions dans les cages RM. La session se termine lorsque le menu *fbpanel* est fermé, par le bouton « fermer la session CLIP » du menu. L'environnement graphique de la session USER dans $USER_{clip}$ comprend aussi un afficheur de charge batterie et un menu permettant à l'utilisateur de lancer les différentes opérations interactives sur les supports amovibles décrites en 2.7, ainsi que de changer son mot de passe ou de verrouiller sa session.

F.USER_SESSION_CLOSE Fermeture de session *(interactif)*

Lors de la fermeture d'une session, quel que soit son type, la session *pam* ouverte auprès de *xdm* lors de l'authentification initiale de l'utilisateur est terminée, ce qui déclenche le démontage des partitions temporaires et chiffrées de l'utilisateur par la fonction 2.4. Par ailleurs, tous les processus s'exécutant dans la cage sont terminés, ce qui entraîne la terminaison de la cage $USER_{clip}$ elle-même.

F.USER_SESSION_LOCK Verrouillage de session

La session graphique dans $USER_{clip}$, quel que soit son type (USER, ADMIN, AUDIT), se verrouille automatiquement après trois minutes d'inactivité (clavier et souris). Par ailleurs, un déclenchement explicite du verrouillage est possible depuis une session USER, en cliquant sur l'un des éléments du menu utilisateur. L'écran d'une session verrouillée est noirci, et le clavier et la souris attribués uniquement à l'utilitaire de verrouillage, de telle sorte que la seule action possible en état verrouillé soit de déclencher une tentative de déverrouillage.

F.USER_SESSION_UNLOCK Déverrouillage de session

Lorsque la session graphique $USER_{clip}$ est en état verrouillé, toute action de l'utilisateur sur le clavier ou la souris déclenche l'apparition d'une fenêtre demandant la saisie du mot de passe de l'utilisateur courant, afin de déverrouiller la session. En cas de succès de l'authentification, l'utilisateur retrouve immédiatement sa session dans l'état où elle était avant verrouillage. Dans le cas contraire, l'échec est journalisé, et un délai de trois secondes est imposé avant toute nouvelle tentative de déverrouillage.La modification nécessite de saisir une fois l'ancien mot de passe, et deux fois le nouveau, dans des fenêtres *pop-up* lancées dans la cage $USER_{clip}$. La modification nécessite de saisir une fois l'ancien mot de passe, et deux fois le nouveau, dans des fenêtres *pop-up* lancées dans la cage $USER_{clip}$.

Les fonctions suivantes s'appliquent uniquement à la session USER :

F.USER_SESSION_RM_START Lancement de session RM (*interactif*)

Lancer une session dans la cage RM_H ou RM_B lorsque l'utilisateur clique sur le bouton correspondant du menu *clip-menu*. Ce lancement se fait en deux temps : d'abord une requête de création de session USER est transmise au *jailmaster* de la cage correspondante, à l'aide de *jailrequest*, puis une visionneuse VNC est lancé dans une cage *chroot* (« vue visionneuse » de la cage $USER_{clip}$)

afin de se connecter à cette session USER dans la cage. Le lancement doit être interrompu si la requête *jailmaster* échoue, en particulier si une session utilisateur est déjà lancée dans la cage RM. Ces deux étapes se font par ailleurs sans réauthentification de l'utilisateur. Le fichier correspondant à la socket UNIX VNC est créé lors du lancement de la session USER dans la cage RM (première étape), et supprimé dès la connexion d'une visionneuse (deuxième étape). La connexion à *jailmaster* comme au serveur est robuste vis-à-vis des accès concurrents.

F.USER_SESSION_RM_STOP — Fermeture d'une session RM (*interactif*)

La fermeture de session dans la cage RM peut être déclenchée de trois manières : fermeture depuis la session elle-même dans la cage RM (bouton déconnexion de l'environnement KDE), ou fermeture de la fenêtre visionneuse à l'aide du gestionnaire de fenêtres (plus précisément, par un clic droit dans la barre de tâches de la session $USER_{clip}$) de la session $USER_{clip}$, ou encore en quittant entièrement la session $USER_{clip}$. Dans les deux premiers cas, la fenêtre visionneuse est fermée, et une nouvelle session peut être immédiatement lancée dans la cage RM.

F.USER_SESSION_CHPASS — Modification du mot de passe – profil utilisateur (*interactif*)

Modifier le mot de passe d'authentification de l'utilisateur courant, en cliquant sur l'entrée « Changer le mot de passe » du menu *fbpanel*. La modification nécessite de saisir une fois l'ancien mot de passe, et deux fois le nouveau, dans des fenêtres *pop-up* lancées dans la cage $USER_{clip}$. Cette fonction est aussi disponible pour les utilisateurs disposant d'un profil administrateur RM. Le nouveau mot de passe choisi par l'utilisateur est soumis à des contraintes de qualité (nombre de caractères des différents types, absence du dictionnaire, etc.). Un mot de passe différent est demandé à l'utilisateur tant que celui-ci ne saisit pas un mot de passe acceptable.

F.USER_X11_LABEL_SET — Labellisation X11

Les fenêtres X11 sont labellisées en fonction du niveau de sensibilité. Le serveur graphique exécuté dans la cage X11 reconnaît au moins trois domaines de sécurité : CLIP (privilégié), RM_H et RM_B (non privilégiés). Au démarrage de la session, le gestionnaire de fenêtres et le menu *clip-menu* sont lancés dans le domaine CLIP. Lors du lancement d'une session dans une cage RM, un *cookie* d'autorisation est généré pour le domaine RM_H ou RM_B correspondant, et attribué à la visionneuse VNC utilisée pour se connecter à la cage. Le cloisonnement *chroot* de la visionneuse lui interdit l'accès aux autres *cookies Xauthority* du système.

F.USER_X11_LABEL_DISP — Affichage des labels X11

Le gestionnaire de fenêtres rend la labellisation X11 introduite par la fonction 3.1 visuellement apparente de deux manières. D'une part, la couleur du bandeau de chaque fenêtre est fixée par son domaine de sécurité : marron pour CLIP, rouge pour RM_H et vert pour RM_B. D'autre part, l'un des préfixes *<CLIP>*, *<RM_H>* ou *<RM_B>* est ajouté au titre de chaque fenêtre, selon son domaine. La couleur de labellisation est de plus reprise dans la barre de tâches *fbpanel* permettant la commutation entre les fenêtres.

F.USER_X11_DOMAINS Cloisonnement graphique

Le serveur X11 interdit toute interaction (capture d'écran, envoi d'événements, etc.) entre une fenêtre non privilégiée et une fenêtre d'un domaine différent, qu'il soit privilégié ou non. Par ailleurs, les fenêtres appartenant à un domaine non-privilégié ne peuvent pas lire ni écrire dans le tampon de copier-coller du serveur X11.

F.USER_X11_PROTECT Protection du serveur X11

Les fenêtres des domaines RM_H et RM_B n'ont accès qu'aux extensions « sécurisées » du serveur X11, et ne peuvent pas modifier les propriétés de ce dernier (résolution, etc.).

F.USER_X11_WM Gestion de fenêtres

Le gestionnaire de fenêtres ne permet pas de redimensionner une fenêtre, ni de la déplacer, ni de la réduire à son bandeau ou de masquer ce bandeau. Le focus est obtenu en cliquant sur une fenêtre, ou sur l'entrée correspondante dans la barre de tâches *fbpanel*, ce qui entraîne son passage au premier plan. La fenêtre de premier plan est clairement identifiée par un bandeau de couleur plus foncée. Les fenêtres dédiées aux visionneuses RM occupent l'intégralité de l'espace disponible sur l'écran, à l'exception de celui réservé à la barre *fbpanel* et aux bandeaux de couleur (cf. [F.USER_X11_LABEL_DISP]).

3.2 Cage AUDIT$_{clip}$

La cage a un double rôle, de collecte des journaux pour l'ensemble du système d'une part, et de consultation de ces journaux d'autre part.

F.AUDIT_LOG_COLLECT_KERN Collecte des journaux noyau

Collecter les journaux du noyau par lecture sur */proc/kmsg*.

F.AUDIT_LOG_COLLECT_CLIP Collecte des journaux du socle et des cages CLIP

Collecter directement les journaux du socle et des cages CLIP, par lecture sur différentes *sockets /dev/log*.

F.AUDIT_LOG_COLLECT_RM Collecte des journaux des cages RM

Collecter les journaux des cages RM_H et RM_B, envoyés par les vues AUDIT de ces cages. La provenance des journaux doit être clairement identifiée lors de la collecte, de manière à permettre leur écriture par [F.AUDIT_LOG_FILTER] dans les fichiers dédiés à la cage RM d'origine.

Remarque 2 : Uniformisation des heures de collecte

Il serait souhaitable que le démon de collecte des journaux de AUDIT$_{clip}$ inscrive dans chaque message une heure de référence (heure du socle) de sa collecte, en remplacement ou complément de l'heure d'émission (qui n'est pas nécessairement de confiance car sous le contrôle de l'émetteur).

F.AUDIT_LOG_FILTER **Répartition des journaux**

Répartir les journaux dans différents fichiers en fonction de leur nature et de leur provenance.

F.AUDIT_SESSION_OPEN **Ouverture de session de consultation AUDIT$_{clip}$**

Écouter sur la boucle locale (port 23) en attente de connexions. Lors d'une telle connexion, authentifier l'utilisateur selon le protocole *SSH2* (mode *PubkeyAuthentication* uniquement). Une session ne peut être ouverte que sous le compte *_audit*, et uniquement par un utilisateur disposant d'une clé privée à laquelle est associée une clé publique autorisée dans la cage AUDIT$_{clip}$. En cas de succès, une session de consultation est ouverte, offrant un *shell* sous l'identité *_audit* dans la cage AUDIT$_{clip}$ à un utilisateur d'une session AUDIT de la cage USER$_{clip}$.

F.AUDIT_SESSION_LAST **Rappel de la connexion précédente**

Afficher un message rappelant la session précédente (date et heure d'ouverture et adresse d'origine) lors de l'ouverture de session AUDIT$_{clip}$.

F.AUDIT_SESSION_CHPASS **Modification du mot de passe – profil auditeur (*interactif*)**

Modifier le mot de passe d'authentification de l'utilisateur courant, par invocation de l'utilitaire *userclt*. La modification nécessite de saisir une fois l'ancien mot de passe, et deux fois le nouveau, dans des fenêtres *pop-up* lancées dans la cage USER$_{clip}$. Le nouveau mot de passe choisi par l'utilisateur est soumis à des contraintes de qualité (nombre de caractères des différents types, absence du dictionnaire, etc.). Un mot de passe différent est demandé à l'utilisateur tant que celui-ci ne saisit pas un mot de passe acceptable.

F.AUDIT_SESSION_READ **Consultation des journaux (*interactif*)**

L'utilisateur *_audit* dispose des droits discretionnaires en lecture sur les fichiers créés par [F.AUDIT_LOG_FILTER]. Il peut les analyser lors d'une session de consultation à l'aide d'outils en ligne de commande, en particulier *vi, less, tail, grep*, *sed* et *awk*.

F.AUDIT_SESSION_CLOSE **Fermeture de session AUDIT$_{clip}$ (*interactif*)**

Fermer la session lorsque l'utilisateur *_audit* quitte son *shell* dans AUDIT$_{clip}$, ou lorsque la session AUDIT est terminée dans la cage USER$_{clip}$. Aucune action spécifique n'est réalisée à la fermeture.

F.AUDIT_AVAILABILITY **Protection en disponibilité de la collecte de journaux**

Le démon *syslog* réalisant les fonctions [F.AUDIT_LOG_COLLECT_KERN] à [F.AUDIT_LOG_FILTER] dispose de privilèges suffisants pour ne pas pouvoir être interrompu par un utilisateur quelconque du socle ou des cages, en dehors des séquences de démarrage ou d'arrêt du système.

3.3 Cage ADMIN$_{clip}$

La cage ADMIN$_{clip}$ permet l'administration de certains paramètres fonctionnels du système, par édition de fichiers de configuration exposés en écriture dans cette cage, ou par lancement d'utilitaires spécifiques.

F.ADMIN_SESSION_OPEN — **Ouverture de session ADMIN$_{clip}$**

Ecouter sur la boucle locale (port 22) en attente de connexions. Lors d'une telle connexion, authentifier l'utilisateur selon le protocole *SSH2* (mode *PubkeyAuthentication* uniquement). Une session ne peut être ouverte que sous le compte *_admin*, et uniquement par un utilisateur disposant d'une clé privée à laquelle est associée une clé publique autorisée dans la cage ADMIN$_{clip}$. En cas de succès, une session d'administration est ouverte, offrant un *shell* sous l'identité *_admin* dans la cage ADMIN$_{clip}$ à un utilisateur d'une session ADMIN de la cage USER$_{clip}$.

F.ADMIN_SESSION_LAST — **Rappel de la connexion précédente**

Afficher un message rappelant la session précédente (date et heure d'ouverture et adresse d'origine) lors de l'ouverture de session ADMIN$_{clip}$.

F.ADMIN_SESSION_CHPASS — **Modification du mot de passe – profil administrateur (*interactif*)**

Modifier le mot de passe d'authentification de l'utilisateur courant, par invocation de l'utilitaire *userclt*. La modification nécessite de saisir une fois l'ancien mot de passe, et deux fois le nouveau, dans des fenêtres *pop-up* lancées dans la cage USER$_{clip}$. Le nouveau mot de passe choisi par l'utilisateur est soumis à des contraintes de qualité (nombre de caractères des différents types, absence du dictionnaire, etc.). Un mot de passe différent est demandé à l'utilisateur tant que celui-ci ne saisit pas un mot de passe acceptable.

F.ADMIN_SESSION_CLOSE — **Fermeture de session ADMIN$_{clip}$ (*interactif*)**

Fermer la session lorsque l'utilisateur *admin* quitte son *shell* dans ADMIN$_{clip}$, ou lorsque la session ADMIN est terminée dans la cage USER$_{clip}$. Aucune action spécifique n'est réalisée à la fermeture.

F.ADMIN_CONFIG — **Administration des paramètres système (*interactif*)**

Les fichiers pouvant être modifiés par le rôle d'administrateur sont projetés par le socle dans la cage avec des droits en écriture. Les droits discrétionnaires autorisent de plus l'utilisateur *_admin* à lire et modifier ces fichiers. Lors d'une session, l'utilisateur *_admin* modifie ces fichiers à l'aide d'outils adaptés, notamment *vi*.

F.ADMIN_CONFIG_NET_ADDR — **Configuration des adresses réseau (*interactif*)**

Modifier, en éditant un fichier de configuration, les différentes adresses IP locales, ainsi que les

adresses de la passerelle par défaut et des passerelles chiffrantes GW_RMH et GW_UPDATE. Ces modifications ne prennent effet qu'au redémarrage du système. Des mesures spécifiques permettent d'interdire même en cas d'erreur de configuration la fuite d'informations de RM_H en clair sur le réseau RM_B, ainsi que des entrées en clair dans la cage $UPDATE_{clip}$ depuis RM_B.

F.ADMIN_CONFIG_NET_DNS — Configuration de la résolution de nom CLIP (*interactif*)

Configurer la résolution de nom pour le socle et les cages CLIP. Cette configuration porte aussi bien sur la résolution statique (modification de */etc/hosts*) que sur la résolution dynamique (modification de */etc/resolv.conf*).

F.ADMIN_CONFIG_NET_FILTER — Configuration du filtrage réseau (*interactif*)

Configurer, en éditant un fichier, les ports TCP et UDP ouverts dans le sens sortant sur l'interface *ethernet*. Les ports ouverts sont définis cage par cage. Ces modifications ne prennent effet qu'au prochain redémarrage.

F.ADMIN_CONF_NET_IPSEC — Configuration IPsec (*interactif*)

Installer ou supprimer une clé privée CCSD, à utiliser pour l'authentification IKEv2 du poste auprès des passerelles RM_H et UPDATE. Ces opérations sont réalisées par le biais d'un utilitaire dédié, *install_ccsd*, qui attribue des permissions adaptées aux fichiers installés. Activer ou désactiver, par modification d'un fichier de configuration (prise en compte au prochain démarrage du système), le mode *NAT-Traversal* pour le démon IKEv2.

F.ADMIN_CONF_USER — Gestion des comptes utilisateurs (*interactif*)

Créer, lister ou supprimer, par invocation d'un utilitaire spécifique, des comptes utilisateurs. Attribuer, lors de la création d'un compte, les rôles d'administration (CLIP ou RM) ou d'audit à ce compte. Définir, lors de la création d'un compte, la taille des partitions chiffrées attribuées à ce compte. L'ensemble de ces opérations repose sur l'affichage de fenêtres *pop-up* (pour demander le mot de passe d'un compte, ou choisir la taille de ses partitions par exemple) qui sont lancés dans la cage $USER_{clip}$. Par ailleurs, les outils de gestion des comptes utilisateurs interdisent la suppression du compte de l'utilisateur courant. Enfin, le mot de passe attribué à un nouvel utilisateur est soumis à des contraintes de qualité (nombre de caractères des différents types, absence du dictionnaire, etc.). Un mot de passe différent est demandé à l'administrateur tant que celui-ci ne saisit pas un mot de passe acceptable.

F.ADMIN_CONF_DOWNLOAD — Configuration des téléchargements de mises à jour (*interactif*)

Configurer les sources de téléchargements de mises à jour, aussi bien pour CLIP que pour les cages RM. Activer ou désactiver le téléchargement initial (au démarrage du système) de ces mises-à-jour. Configurer les âges minimum et maximum des configurations téléchargées, ainsi que le délai de purge des miroirs locaux. L'ensemble de ces opérations est réalisé par édition de fichiers de configuration.

F.ADMIN_CONF_SSL — **Mise à jour du certificat HTTPS de téléchargement (*interactif*)**

Mettre à jour, par invocation d'un utilitaire spécifique *install_cert*, le certificat de l'autorité de certification utilisée pour les téléchargements *HTTPS* de mises à jour.

F.ADMIN_UPDATE_DOWNLOAD — **Pilotage des téléchargements de mises à jour (*interactif*)**

Lancer un téléchargement de mise à jour, au choix pour CLIP, RM_H ou RM_B. Verrouiller ou déverrouiller le téléchargement des mises à jour pour ces différents compartiments. Ces opérations sont réalisées à l'aide d'un utilitaire spécifique, *downloadctl*.

F.ADMIN_CONF_UPDATE — **Configuration des mises à jour CLIP (*interactif*)**

Autoriser ou interdire l'application automatique de mises à jour à fort impact pour le socle et les cages CLIP. Les autres paramètres des fonctions de mise à jour ne sont pas configurables.

F.ADMIN_CONF_DATE — **Configuration de l'heure et de la date (*interactif*)**

Ajuster l'heure, la date et la zone horaire du système. L'heure et la date sont ajustées par invocation de l'utilitaire *date*, tandis que la zone horaire est configurée par copie d'un fichier de définition.

F.ADMIN_CONF_NTP — **Configuration de la synchronisation horaire (*interactif*)**

Activer ou désactiver la synchronisation horaire *NTP*, et définir le nom du serveur *NTP* auprès duquel cette synchronisation est effectuée, en modifiant un fichier de configuration.

F.ADMIN_CONF_BACKUP — **Administration des sauvegardes et restaurations (*interactif*)**

Lister les sauvegardes système et données présentes sur le système. Demander la réalisation d'une opération de sauvegarde ou de restauration du système ou des données pour le prochain démarrage (cf. [F.CORE_START_BACKUP] et [F.CORE_START_ROLLBACK]). Ces opérations sont réalisées à l'aide d'utilitaires spécifiques, et reposent notamment sur des *pop-ups* graphiques lancés dans la cage $USER_{clip}$.

3.4 Cage UPDATE$_{clip}$

La cage UPDATE$_{clip}$ a un double rôle, de téléchargement des mises à jour pour l'ensemble du système d'une part, et d'application des mises à jour des paquetages secondaires CLIP d'autre part. Les téléchargements sont gérés par configurations (ensembles cohérents de paquetages), et permettent de créer des miroirs locaux (un pour le socle et les cages CLIP, un pour chaque cage RM_H ou RM_B) à partir desquels les mises à jours sont réalisées, de manière décorrélée du téléchargement.

F.UPDATE_DNS — **Résolution de noms statique et réseau**

Déterminer l'adresse IP du serveur de mise à jour dans LAN_UPDATE de manière statique (fichier */etc/hosts*) ou, à défaut, dynamique par requête DNS.

F.UPDATE_NTP — **Synchronisation horaire**

Vérifier une fois par heure si la synchronisation horaire a été activée par l'administrateur local (cf. [F.ADMIN_CONF_NTP]), et le cas échéant procéder à la synchronisation avec le serveur NTP défini par l'administrateur, dont le nom est importé de manière sécurisée. Cette synchronisation se fait obligatoirement à travers le VPN IPsec UPDATE, avec un serveur NTP situé au sein du LAN UPDATE.

F.UPDATE_MIRROR_LOOKUP — **Recherche de mises à jour disponibles**

Rechercher au démarrage et périodiquement ensuite (ou à la demande de l'administrateur, cf. [F.ADMIN_UPDATE_DOWNLOAD]) de nouvelles configurations disponibles, aussi bien dans la distribution RM que dans la distribution CLIP, sur le serveur HTTPS du réseau LAN_UPDATE à travers le tunnel IPSEC_UPDATE. Le serveur HTTPS est systématiquement authentifié par son certificat.

F.UPDATE_MIRROR_SYNC — **Synchronisation réseau des miroirs locaux**

Lorsque de nouvelles configurations sont détectées, procéder à leur téléchargement sur le serveur HTTPS de LAN_UPDATE, à travers le tunnel IPSEC_UPDATE, et à leur intégration aux miroirs locaux. Le serveur HTTPS est systématiquement authentifié par son certificat. Par ailleurs, la double signature de chaque configuration ou paquetage téléchargée est vérifiée immédiatement après son téléchargement.

F.UPDATE_MIRROR_LOG — **Journalisation des erreurs de téléchargement**

Journaliser les erreurs rencontrées lors du téléchargement des mises à jour, en particulier les erreurs d'authentification du serveur de mise à disposition ou de vérification des signatures (ces dernières doivent être facilement détectables par des outils d'analyse automatique des journaux)..

F.UPDATE_MIRROR_CDROM **Synchronisation des miroirs locaux depuis un CD-ROM**

Alternativement à [F.UPDATE_MIRROR_SYNC], récupérer de nouvelles configurations depuis un CD-ROM monté sous */mnt/cdrom* dans l'arborescence de la cage.

Remarque 3 : Montage de CD-ROM de mise à jour

Aucun mécanisme ne permet à ce jour le montage d'un CD-ROM contenant des mises à jour sous /mnt/cdrom dans l'arborescence de la cage. Un tel mécanisme, piloté depuis la cage $ADMIN_{clip}$, serait nécessaire à la mise en oeuvre de la fonction [F.UPDATE_MIRROR_CDROM].

F.UPDATE_MIRROR_CLEAN **Nettoyage des miroirs locaux**

Supprimer les copies locales des paquetages obsolètes (selon le paramètre de purge défini par [F.ADMIN_CONF_DOWNLOAD]).

F.UPDATE_MIRROR_RM **Mise à disposition des cages RM des miroirs RM**

Mettre à disposition de chaque cage RM_H ou RM_B une copie du miroir RM_H ou RM_B local.

Remarque 4 : Déclenchement des mises à jour de paquetages primaires CLIP

Il serait souhaitable que la cage $UPDATE_{clip}$ soit en mesure de signaler au socle la disponibilité dans le miroir local de tous les paquetages nécessaires à une mise à jour des paquetages primaires CLIP.

Remarque 5 : Déclenchement des mises à jour de paquetages primaires RM

Il serait souhaitable que la cage $UPDATE_{clip}$ soit en mesure de signaler au socle la disponibilité dans le miroir local de tous les paquetages nécessaires à une mise à jour des paquetages primaires RM.

F.UPDATE_INSTALL **Application des mises à jour de paquetages secondaires CLIP**

Appliquer au démarrage et périodiquement ensuite les mises à jour de paquetages secondaires CLIP disponibles dans le miroir local, pour les paquetages dont l'impact est autorisé par l'administrateur. La double signature des paquetages est systématiquement vérifiée avant leur installation.

F.UPDATE_INSTALL_LOG **Journalisation des mises à jour**

Journaliser les opérations de mise à jour réussies, ainsi que les échecs, en particulier ceux rencontrés dans la vérification des signatures (ces derniers doivent être facilement détectables par des outils d'analyse automatique des journaux).

F.UPDATE_RECOVERY **Reprise sur erreur**

Reprendre le traitement des fonctions [F.UPDATE_MIRROR_SYNC],

[F.UPDATE_MIRROR_CDROM] ou [F.UPDATE_INSTALL] suite à une erreur ou une interruption lors de leur invocation précédente.

4 Cages RM

Les fonctions des cages RM sont essentiellement réparties entre les différentes vues qui les composent. Les seules fonctions réalisées hors de toute vue sont les suivantes :

F.RM_VERIEXEC_LOAD **Configuration initiale de *veriexec***

Charger les entrées *veriexec* associées à la cage, dans le contexte *veriexec* de celle-ci, avant l'exécution de tout démon dans la cage. Cette fonction est réalisée au sein de la cage, mais invoquée directement par le socle (fonction [F.CORE_START_JAILS_RM]).

F.RM_VIEWS_START **Démarrage des vues**

Démarrer les quatre vues de la cage en lançant le démon approprié enfermé par *chroot* dans chaque cage. A l'issue de ce démarrage, aucun processus ne doit plus tourner dans la cage en dehors des vues. Cette fonction est elle aussi invoquée par la fonction [F.CORE_START_JAILS_RM] du socle.

F.RM_VERIEXEC_UNLOAD **Suppression des entrées *veriexec***

Supprimer toutes les entrées *veriexec* du contexte associé à la cage, avant de terminer celle-ci. Cette fonction est invoquée par le socle lors de l'arrêt de la cage (éventuellement dans le cadre de son redémarrage, qui n'est jamais réalisé à ce stade).

4.1 Vue USER

F.RM_USER_SESSION_START **Lancement de session dans la vue USER**

Attendre les connexions sur une *socket* UNIX de la vue USER. Lorsqu'une telle connexion est reçue (du fait de la réalisation de [F.USER_SESSION_RM_START]), lancer dans la vue un serveur VNC et un ensemble de clients X11 de ce serveur composant une session cliente (cf. [F.RM_USER_SESSION_SELECT]), sous l'identité de l'utilisateur ayant effectué la connexion. Ce lancement n'est réalisé que si aucune autre session n'est en cours d'exécution dans la vue. Une fois le lancement effectué, un acquittement est envoyé à l'initiateur de la connexion par l'écriture d'un caractère sur la *socket*. Par ailleurs, le lancement de session ne doit pas être possible si les partitions chiffrées de l'utilisateur n'ont pas été montées par [F.CORE_SESSION_MOUNT].

F.RM_USER_SESSION_SELECT **Choix d'un type de session cliente RM**

Choisir les clients lancés en plus du serveur VNC lors du démarrage d'une session. Deux types de sessions clientes sont possibles : si l'utilisateur de la session appartient au groupe *rm_admin*, une session USER-ADMIN est lancée, sinon, une session USER normale est lancée.

F.RM_USER_SESSION_ENV Configuration de l'environnement

Positionner un certain nombre de variables d'environnement avant de lancer la session cliente, en particulier *$HOME*, qui doit être positionné à */home/user* (point de montage de la partition chiffrée de l'utilisateur), le *$PATH*, et les variables de configuration linguistique *LC_ALL* et *LANG*, qui doivent être positionnées à *fr_FR*.

F.RM_USER_SESSION_ADMIN Session cliente USER-ADMIN *(interactif)*

La session cliente USER-ADMIN consiste en un unique émulateur de terminal *xterm*, en plein écran (pour le serveur X11 VNC de la cage), qui lance lui-même immédiatement un client *ssh* pour ouvrir une connexion sous l'identité *_admin* dans la vue ADMIN, sur la boucle réseau locale de la cage. La session USER-ADMIN se termine en même temps que ce client *ssh*.

F.RM_USER_SESSION_USER Session cliente USER *(interactif)*

La session cliente USER consiste en un environnement de bureau *KDE* complet. Elle se termine lorsque l'utilisateur quitte cet environnement, par exemple en sélectionnant l'option « Déconnexion » dans le menu *KDE*. Les fonctionnalités propres à cet environnement graphique font l'objet d'une description plus détaillée en 4.1.

F.RM_USER_SESSION_STOP Terminaison de la session USER

La session USER peut être terminée de deux manières : soit par terminaison de la session cliente, soit par déconnexion de la visionneuse VNC. Aucun processus ne continue à s'exécuter sous l'identité d'un utilisateur après la terminaison de sa session.

F.RM_USER_VNC_NET Connexion VNC

Le serveur VNC lancé par [F.RM_USER_SESSION_START] écoute uniquement sur une *socket* de type UNIX. Il accepte exactement une connexion sur cette *socket* (connexion d'une visionneuse VNC, initiée par [F.USER_SESSION_RM_START]), aucune connexion ultérieure n'est acceptée après cette première connexion. La terminaison de la connexion est détectée et entraîne la terminaison du serveur VNC (cf. [F.RM_USER_SESSION_STOP]).

F.RM_USER_VNC_X11 Serveur X11 VNC

Le serveur VNC se comporte comme un serveur X11 « standard » vis à vis des clients de la vue USER. Il offre notamment des fonctionnalités de copier-coller internes à la vue.

4.2 Session cliente USER

F.RM_USER_APP_MAIL **Messagerie (*interactif*)**

Un client de messagerie (*Mozilla Thunderbird*) permet d'accéder à des comptes POP, POPS, IMAP et IMAPS, et d'envoyer des messages par SMTP ou ESMTP (+TLS).

F.RM_USER_APP_MAIL_CRYPT **Chiffrement et signature de messages (*interactif*)**

Le client de messagerie permet de chiffrer et signer les messages composés, et de déchiffrer et de vérifier la signature des messages reçus, et ce aussi bien avec S/MIME que GPG.

F.RM_USER_APP_ANNUARY **Annuaire (*interactif*)**

Le client de messagerie permet la recherche de correspondants dans un annuaire LDAP ou LDAPS situé sur un serveur distant.

F.RM_USER_APP_BROWSER **Navigation web (*interactif*)**

Un navigateur (*Mozilla Firefox*) permet la navigation http, https et ftp. Il supporte notamment les langages HTML, XHTML, Javascript et SVG. Il intègre de plus les *plugins Adobe Flash* et *Java*.F.RM_USER_APP_SSL Chiffrement et authentification SSL/TLS

Le navigateur et le client de messagerie supportent les protocoles de chiffrement et d'authentification SSLv3 et TLS.

F.RM_USER_APP_DNS **Résolution de noms statique ou dynamique**

Déterminer les adresses IP des serveurs de services RM_H ou RM_B statiquement par le fichier */etc/hosts* ou dynamiquement par requête DNS.

F.RM_USER_APP_FILE_MANAGER **Exploration de fichiers (*interactif*)**

Une application graphique (*Konqueror*) permet la navigation dans l'arborescence de fichiers de la vue, la création, la suppression de fichiers, leur déplacement ou leur copie entre répertoires ou supports, et leur ouverture lorsqu'ils sont dans un format supporté par l'environnement de travail.

RM_USER_APP_IMG_DISP **Lecture de formats image (*interactif*)**

Une application graphique dédiée (*Kuickshow*) permet l'ouverture des formats d'image courants : JPEG, GIF, PNG, BMP.

F.RM_USER_APP_PDF_DISP **Lecture de fichiers PDF (*interactif*)**

Une application graphique dédiée (*KPDF*) permet la lecture de fichiers au format PDF. Elle supporte notamment l'affichage des signets d'un document PDF, la recherche de texte dans un tel document, et la

copie de texte ou d'images depuis un tel document vers le tampon de copier-coller de l'environnement.

F.RM_USER_APP_PS_DISP — Lecture de fichiers Postscript (*interactif*)

Une application graphique dédiée (*Kghostview*) permet la lecture de fichiers au format Postscript.

F.RM_USER_APP_TXT — Lecture, création et modification de fichiers textes (*interactif*)

Un éditeur graphique (*Kwrite*) permet la lecture et l'écriture de fichiers textes. Cet éditeur supporte au moins les codages ASCII, ISO-8859-1, ISO-8859-15 et UTF-8.

F.RM_USER_APP_OFFICE — Suite bureautique (*interactif*)

Une suite bureautique (*OpenOffice*) permet la lecture, la création et la modification de fichiers aux formats bureautiques. Elle comprend au moins un logiciel de traitement de texte, un de présentation, un tableur et un éditeur de schémas. Elle est compatible avec les formats de documents ODF, ainsi que, généralement, avec les différentes versions des formats *doc*, *xls* et *ppt* de Microsoft.

F.RM_USER_APP_FILE_ARCH — Ouverture et création d'archives compressées (*interactif*)

Une application graphique dédiée (*Ark*) permet l'ouverture d'archives compressées aux formats *rar, zip, tar.gz* et *tar.bz2*, et la création d'archives aux formats *zip, tar.gz* et *tar.bz2*.

F.RM_USER_APP_FILE_CRYPT — Chiffrement et déchiffrement de fichiers (*interactif*)

Une application graphique dédiée (*Kgpg*) permet le chiffrement GPG de fichiers, et le déchiffrement de tels fichiers, ainsi que la gestion d'un trousseau de clés privées et publiques, et la génération de nouveaux bi-clés GPG.

F.RM_USER_APP_FILE_PRINT — Impression dans des fichiers PDF et Postscript (*interactif*)

Les services d'impression de l'environnement permettent de générer un document au format PDF ou Postscript à partir de tout format supporté (formats bureautiques, images, texte et HTML).

F.RM_USER_APP_COPYPASTE — Copier-coller (*interactif*)

L'environnement graphique utilisateur permet le copier-coller de texte, d'images et de fichiers entre les différentes applications qui le composent.

F.RM_USER_APP_FONTS — Polices de caractères

L'environnement graphique utilisateur supporte les polices de caractère *truetype*, et inclut notamment les polices *corefonts* couramment associées aux outils bureautiques Microsoft : *Times New Roman, Arial, etc…*

F.RM_USER_APP_I18N_FRENCH Localisation en français

L'environnement est entièrement localisé en français. En particulier, les menus, messages et rubriques d'aides de l'environnement et de ses applications sont affichés en français.

F.RM_USER_APP_SPELLCHECK Correcteur orthographique

Le client de messagerie, le navigateur web et les outils bureautiques incluent un correcteur orthographique français.

F.RM_USER_APP_CONFIG Personnalisation de l'environnement de travail (*interactif*)

L'environnement de travail peut être personnalisé par des outils graphiques, qui permettent au moins de modifier le fond d'écran et les polices de caractères par défaut, et de créer des icônes sur le bureau.

4.3 Vue AUDIT

Les vues AUDIT des cages RM fonctionnent de manière entièrement automatique, la consultation des journaux qu'elles collectent étant réalisée depuis la cage $AUDIT_{clip}$.

F.RM_AUDIT_LOG_COLLECT Collecte des journaux de la cage

Collecter les journaux des quatre vues de la cage en créant une *socket /dev/log* dans chacune de ces vues.

F.RM_AUDIT_LOG_SEND Transférer les journaux

Transférer les journaux au démon de collecte de la cage $AUDIT_{clip}$ (cf. [F.AUDIT_LOG_COLLECT_RM]).

4.4 Vue ADMIN

Les vues ADMIN des cages RM sont similaires dans leur rôle et leur fonctionnement à la cage $ADMIN_{clip}$. Elles permettent l'ouverture de sessions interactives par *ssh* sur la boucle locale depuis une session cliente USER-ADMIN dans la vue USER de la même cage. Ces vues sont dédiées à l'administration des paramètres propres à la cage (par exemple, résolution de nom, ou paramètres éditables de la procédure de mise à jour de la cage) et qui ne sont pas gérés ni utilisés par le socle (en particulier, pas les adresses ou les ports ouverts dans le pare-feu local, qui relèvent du socle et donc de $ADMIN_{clip}$).

F.RM_ADMIN_SESSION_OPEN Ouverture de session ADMIN RM

Ecouter sur la boucle locale de la cage RM (port 22) en attente de connexions. Lors d'une telle connexion, authentifier l'utilisateur selon le protocole *SSH2* (mode *PubkeyAuthentication* uniquement). Une session ne peut être ouverte que sous le compte *_admin*, et uniquement par un utilisateur disposant d'une clé privée à laquelle est associée une clé publique autorisée dans la vue ADMIN. En cas de succès de une session d'administration est ouverte, offrant un *shell* sous l'identité *_admin* dans la vue

ADMIN à un utilisateur d'une session USER-ADMIN de la vue USER de la même cage.

F.RM_ADMIN_SESSION_LAST — **Rappel de la connexion ADMIN RM précédente**

Afficher un message rappelant la session précédente (date et heure d'ouverture, adresse d'origine) lors de l'ouverture de session ADMIN.

F.RM_ADMIN_SESSION_CLOSE — **Fermeture de session ADMIN$_{clip}$ (*interactif*)**

Fermer la session lorsque l'utilisateur *_admin* quitte son *shell* dans ADMIN, ou lorsque la session USER-ADMIN est terminée dans la vue USER. Aucune action spécifique n'est réalisée à la fermeture.

F.RM_ADMIN_CONFIG — **Administration des paramètres de la cage (*interactif*)**

Les fichiers de la cage pouvant être modifiés par le rôle d'administrateur sont projetés dans la vue ADMIN avec des droits en écriture. Les droits discrétionnaires autorisent de plus l'utilisateur *_admin* à lire et modifier ces fichiers. Lors d'une session, l'utilisateur *_admin* modifie ces fichiers à l'aide d'outils adaptés.

F.RM_ADMIN_CONF_DNS — **Configuration de la résolution de nom RM (*interactif*)**

Configurer la résolution de nom pour la cage RM concernée. Cette configuration porte aussi bien sur la résolution statique (modification de */etc/hosts*) que sur la résolution dynamique (modification de */etc/resolv.conf*).

F.RM_ADMIN_CONF_UPDATE — **Configuration des mises à jour RM (*interactif*)**

Autoriser ou interdire l'application automatique de mises à jour à fort impact pour la cage RM concernée. Les autres paramètres des fonctions de mise à jour ne sont pas configurables.

4.5 Vue UPDATE

Les vues UPDATE des cages RM sont limitées à la mise à jour des paquetages secondaires de leur cage, toutes vues confondues. Elles disposent pour cela chacune d'un miroir de la distribution RM, tenu à jour par la cage $UPDATE_{clip}$, ainsi que d'un accès en écriture aux racines des autres vues de la cage. Ces vues UPDATE, fonctionnent de manière entièrement automatique, sans intervention d'un utilisateur local autre que la configuration limitée réalisée par [F.RM_ADMIN_CONF_UPDATE], et n'accèdent pas au réseau.

F.RM_UPDATE_LOOKUP — **Détection de mises à jour**

Détecter les nouvelles configurations mises à disposition dans le miroir local par la cage $UPDATE_{clip}$ (cf. [F.UPDATE_MIRROR_RM]).

F.RM_UPDATE_INSTALL — **Application des mises à jour de paquetages secondaires RM**

Appliquer au démarrage et périodiquement ensuite les mises à jour de paquetages secondaires RM disponibles dans le miroir local, pour les paquetages dont l'impact est autorisé par l'administrateur (cf. [F.RM_ADMIN_CONF_UPDATE]). La double signature des paquetages est systématiquement vérifiée.

F.RM_UPDATE_INSTALL_RECOVER — **Reprise sur erreur**

Reprendre le traitement de la fonction [F.RM_UPDATE_INSTALL] suite à une erreur ou une interruption lors de son invocation précédente.

F.RM_UPDATE_INSTALL_LOG — **Journalisation des mises à jour**

Journaliser les opérations de mise à jour réussies, ainsi que les échecs, en particulier ceux rencontrés dans la vérification des signatures (ces derniers doivent être facilement détectables par des outils d'analyse automatique des journaux).

4.6 Cages SECURE_UPDATE_RM

Ces deux cages sont superposées aux cages RM_H et RM_B. Leur rôle est limité à l'application des mises à jour de paquetages primaires au profit des cages RM. Elles sont constituées d'une unique vue, UPDATE, superposée à la vue UPDATE de la cage RM correspondante, mais disposant de droits en écriture plus étendus (en particulier, possibilité de modifier les fichiers à la racine de la vue).

F.SECRM_UPDATE_INSTALL — **Aplication des mises à jour de paquetages primaires RM**

Appliquer les mises à jour de paquetages primaires de la cage sur invocation par le socle (cf. [F.CORE_UPDATE_RM_CORE]). La double signature des paquetages est systématiquement vérifiée.

F.SECRM_UPDATE_RECOVER — **Reprise sur erreur**

Reprendre le traitement de la fonction [F.SECRM_UPDATE_INSTALL] suite à une erreur ou une interruption lors de son invocation précédente.

F.SECRM_UPDATE_LOG — **Journalisation des mises à jour**

Journaliser les opérations de mise à jour réussies, ainsi que les échecs, en particulier ceux rencontrés dans la vérification des signatures (ces derniers doivent être facilement détectables par des outils d'analyse automatique des journaux).

5 Fonctionnalités générales

5.1 Robustesse

F.GENERAL_AVAILABILITY **Robustesse aux interruptions**

Le système ne doit pas être rendu inutilisable ou non sécurisé, sauf déficience matérielle, par une interruption de son fonctionnement à quelque moment que ce soit. En particulier, les fonctions d'accès au réseau, d'installation des mises à jour et d'ouverture de session doivent être disponibles après un redémarrage, indépendamment de l'état du système avant redémarrage.

5.2 Installation

F.INST_INSTALL **Installation du système**

Installer un système CLIP-RM depuis un support amovible, en formatant le disque dur puis en installant des paquetages disponibles sur le support ou sur un serveur accessible par le réseau.

Remarque 6 : Vérification des signatures à l'installation

Il serait souhaitable de vérifier les signatures de paquetages lors de l'installation du système.

Remarque 7 : Respect du cloisonnement à l'installation

Il serait souhaitable de n'installer les distributions RM qu'avec des privilèges équivalents à ceux des cages RM finales, et les paquetages secondaires du socle avec des privilèges équivalents à ceux de la cage UPDATE$_{clip}$.

F.INST_PARAM **Personnalisation de l'installation**

Modifier les paramètres d'installation qui ne sont pas accessibles depuis le rôle d'administrateur local du poste, en particulier la taille et la désignation des partitions d'installations, et les configurations (au sens de groupe de paquetages) qui sont installées sur le système.

F.INST_KEYS **Mise à la clé**

Installer les éléments cryptographiques nécessaires au fonctionnement du poste qui ne peuvent pas être installés ou mis à jour par l'administrateur local :

- Clés CCSD de vérification des signatures de paquetages (clé développeur et clé validateur).
- Clés publiques CCSD d'authentification IKEv2 des passerelles UPDATE et RM_H.
- Clés publiques RSA d'export des clés RSA de gestion des supports amovibles (une clé publique par niveau : CLIP, RM_H et RM_B).

F.INST_CONFIG Configuration initiale du système

Réaliser une configuration initiale des paramètres administrables du système. Cette configuration inclut tous les éléments manipulés par les fonctions suivantes :

- [F.ADMIN_CONFIG_NET_ADDR]
- [F.ADMIN_CONFIG_NET_FILTER]
- [F.ADMIN_CONF_NET_IPSEC]
- [F.ADMIN_CONF_DOWNLOAD]
- [F.ADMIN_CONF_NTP]
- [F.ADMIN_CONF_DATE]
- [F.ADMIN_CONF_SSL]
- [F.ADMIN_CONF_USER]
- [F.ADMIN_CONFIG_NET_DNS]
- [F.RM_ADMIN_CONF_DNS]
- [F.ADMIN_CONF_UPDATE]
- [F.RM_ADMIN_CONF_UPDATE]

Annexe A Références

[CLIP_1002]	*Documentation CLIP – 1002 – Architecture de sécurité*
[CLIP_1101]	*Documentation CLIP – 1101 – Génération de paquetages*
[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_1206]	*Documentation CLIP – 1206 – Générateur d'aléa noyau*
[CLIP_1301]	*Documentation CLIP – 1301 – Séquences de démarrage et d'arrêt*
[CLIP_1302]	*Documentation CLIP – 1302 – Fonctions d'authentification CLIP*
[CLIP_1303]	*Documentation CLIP – 1303 – X11 et cloisonnement graphique*
[CLIP_1501]	*Documentation CLIP – 1501 – Configuration réseau*
[CLIP_DCS_13006]	*Spécification fonctionnelle des outils de mise à jour, CLIP-ST-13000-006-DCS*
[CLIP_DCS_15088]	*Document de conception de l'étude sur les supports amovibles, CLIP-DC-15000-088-DCS*
[CLIP_DCS_15093]	*Document de conception de l'étude sur les paramètres contrôlables par l'administrateur, CLIP-DC-15000-093-DCS*
[CLIP_DCS_15094]	*Document de conception de l'étude sur le retour à une configuration antérieure, CLIP-DC-15000-094-DCS (Ed0 Rev 3 ou ultérieure)*
[CCSD]	*Couche Cryptographique pour la Sécurité de Défense – Document d'Interface Client version 3.2*

Annexe B Liste des figures

Annexe C Liste des tableaux

Annexe D Liste des remarques