Documentation CLIP

1303

X11 et cloisonnement graphique

Version	Date	Auteur	Commentaires
1.1.3	28/11/2008	Vincent Strubel	Correction de coquilles.
1.1.2	19/11/2008	Vincent Strubel	Ajout d'une référence à la documentation de conception *Xsecurity*.
1.1.1	06/08/2008	Vincent Strubel	Correction de certaines références obsolètes à SWM.
1.1	31/07/2008	Vincent Strubel	Réécriture au format OpenOffice, mise à jour et suppression des éléments redondants. A jour pour CLIP_03.00.07.
1.0	10/10/2007	Vincent Strubel	Version initiale, à jour pour *xorg-server-1.1.1-r4, xdm-1.0.5-r7*, *core-services-2.1.9*.

Table des matières

Introduction

Le serveur d'affichage X11 est traditionnellement un processus très privilégié au sein d'un système UNIX, avec des privilèges généralement équivalents à ceux du noyau lui-même. Cependant, un tel niveau de privilège est inacceptable sur un poste CLIP, au sein duquel les privilèges noyaux sont très finement contrôlés. Laisser ses privilèges habituels au serveur X11 introduirait un risque de voir les clients graphiques de la cage $USER_{clip}$, eux-mêmes potentiellement corrompus, exploiter une vulnérabilité dans ce serveur à fin d'obtenir des privilèges suffisant à remettre en cause toutes les fonctions de sécurité du système CLIP. Il est de ce fait nécessaire de réduire les privilèges du serveur X11 à un niveau tolérable pour la sécurité de CLIP. Cette réduction ne doit toutefois pas pour autant entraîner une vulnérabilité accrue de ce serveur à des attaques que pourraient lancer certains de ces clients. En effet, le serveur d'affichage demeure dans CLIP privilégié de fait, dans la mesure où il interagit avec des clients appartenant aussi bien au socle CLIP (le démon XDM) qu'avec des clients a priori moins intègres de la cage $USER_{clip}$, et participe ainsi au cloisonnement de cette cage. Ces deux impératifs contradictoires conduisent dans CLIP à la mise en œuvre d'un serveur X11 à privilèges réduits, isolé au sein d'une cage spécifique, selon une approche proche du concept de « zone démilitarisé » dans un déploiement réseau.

Par ailleurs, des contraintes supplémentaires pèsent sur le serveur graphique au sein d'un poste client CLIP-RM, dans la mesure où les cages RM_H et RM_B sont destinées à être utilisées simultanément en mode graphique, depuis la cage $USER_{clip}$ (cf. [CLIP_1304]). L'interdiction des transferts d'information entre les clients X11 liés à deux cages distinctes constitue alors une composante incontournable du cloisonnement applicatif. Un tel cloisonnement, contrairement à celui plus généralement assuré par *vserver* (cf. [CLIP_1202]), ne peut reposer uniquement sur une implémentation noyau, car il fait intervenir une sémantique propre (protocole *X11*) qui est inconnue du noyau. En d'autres termes, il est nécessaire de prolonger le cloisonnement « système » réalisé par *Vserver* par un cloisonnement graphique réalisé directement par le serveur graphique (lequel doit de se fait être considéré comme de confiance au même titre que le noyau pour ce qui est de la confidentialité des informations manipulées par les deux cages[1]).

Le cloisonnement graphique nécessite de labelliser différemment les fenêtres appartenant aux différentes cages et au socle, et d'interdire tout transfert d'information graphique (par copier-coller, capture d'écran, etc.) entre fenêtres de compartiments différents. Ce cloisonnement doit de plus être « lisible » : la labellisation des fenêtres doit être explicite pour l'utilisateur, et la fonction d'affichage de ces labels doit elle-même être de confiance, donc assurée par le socle.

Le cloisonnement graphique dans CLIP suit le même principe de « grosse granularité » que le cloisonnement système : plutôt que de labelliser individuellement chaque fenêtre, l'ensemble des fenêtres correspondant aux clients graphiques d'une cage est géré comme un « bureau » distinct (avec

[1] Mais pas en revanche, pour ce qui concerne l'intégrité du système en général, qui ne doit toujours reposer que sur le seul noyau.

sa fenêtre racine et son gestionnaire de fenêtres), lequel est lui-même représenté – par un médiateur dont la nature est précisée en 2.3Erreur : source de la référence non trouvée – dans une fenêtre unique du serveur X11 du socle. C'est cette fenêtre qui est cloisonnée par le serveur X11 du socle, et labellisée par le gestionnaire de fenêtre du socle (ces deux applicatifs étant considérés comme de confiance vis-à-vis du cloisonnement). La labellisation est transcrite visuellement par un bandeau de couleur (rouge pour la fenêtre RM_H, vert pour la fenêtre RM_B, et marron pour les clients du socle) et par l'ajout d'un mot clé (« RM_H : », « RM_B : » et « CLIP : », respectivement) au titre de la fenêtre. Ces différents éléments sont repris dans la Figure 1.

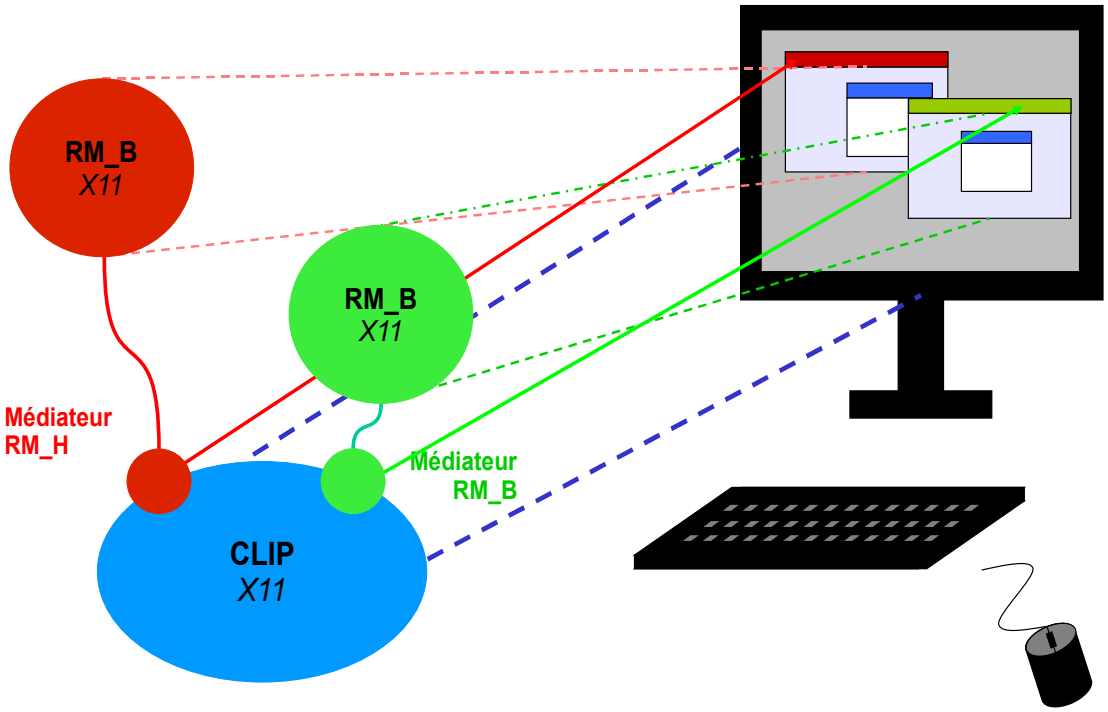

Figure 1: Principe du cloisonnement graphique sous CLIP.
La labellisation des fenêtres principales par un bandeau de couleur est assurée par le socle, et ne peut être contrefaite par les cages.

1 Cloisonnement du serveur X11 au sein du système

Cette première section décrit les mesures de réduction de privilèges et d'isolation du serveur X11 au

sein de CLIP, qui visent à protéger le reste du système dans le cas d'une compromission de ce serveur. Ces mesures sont communes à tous les types de systèmes CLIP.

1.1 Fonctionnement sans IOPL

Le serveur X11 du socle de CLIP est basé sur le serveur X11R7 (X11R7.3 à ce stade) de la fondation *X.org* ([XORG]). Celui-ci est modifié par deux patchs spécifiques à CLIP, assurant respectivement une restriction des privilèges nécessaires au lancement du serveur et une extension des possibilités de cloisonnement graphique.

Les serveurs X11 *Xorg* ou *Xfree* fonctionnent normalement sous l'identité *root,* et mettent en œuvre des privilèges élevés, en particulier la capacité *CAP_SYS_RAWIO*. Cette dernière leur est nécessaire afin d'obtenir par un appel *iopl(3)* les droits d'accès aux ports d'entrée-sortie PIO. Ces droits d'accès permettent notamment l'utilisation de l'ouverture graphique (*graphic aperture*), ainsi que des mécanismes de configuration spécifiques au *chipset* graphique. Cependant, un tel niveau de privilège est problématique du point de vue de la sécurité (cf. notamment [SMM]), et est interdit a priori par CLIP (qui masque *CAP_SYS_RAWIO* à travers le *sysctl cap-bound*, et bloque spécifiquement – par l'option *Grsecurity GRKERNSEC_IO*, cf. [CLIP_1203] – les appels système *iopl()/ioperm()*). Par ailleurs, ces droits d'accès aux ports d'entrée-sortie, bien qu'indispensables aux pilotes spécifiques de *chipsets* graphiques (*nvidia, radeon, etc…*) inclus dans *Xorg,* et aux fonctionnalités avancées qui leur sont associées (accélération matérielle 2D ou 3D par exemple), ne sont en revanche pas nécessaires au fonctionnement en mode *framebuffer* (pilote X11 *fbdev*), dans lequel toutes les opérations d'accès direct au matériel sont déléguées au pilote *framebuffer* du noyau. Ainsi, le serveur *Xorg* utilisé en mode *framebuffer* requiert par défaut – et refuse de se lancer s'il ne les obtient pas – des privilèges très élevés, qu'il ne met finalement pas en œuvre.

Un premier patch spécifique à CLIP, *xorg-server-*<version>*-clip-hwaccess.patch*, appliqué au paquetage *x11-base/xorg-server*, corrige cette incohérence, en supprimant du code d'initialisation du serveur X11 les fonctions de *probing* du bus PCI et d'initialisation matérielle, qui seules requièrent les privilèges *iopl()*. Le serveur *Xorg* compilé après application de ce *patch* ne peut fonctionner qu'avec le pilote *fbdev* (défini dans la section *Device* du fichier */usr/local/etc/X11/xorg.conf*), mais ne nécessite pas dans ce cas la capacité *CAP_SYS_RAWIO*. Il est cependant encore lancé sous l'identité *root*, afin d'obtenir des capacités plus bénignes qui demeurent nécessaires à son lancement.

1.2 Cloisonnement du serveur X11

Outre cette réduction de privilèges, il est souhaitable d'isoler autant que faire se peut le serveur X11 du reste du système, et ce pour deux raisons principales :

- Limiter les conséquences sur l'ensemble du système d'une compromission de ce serveur, qui interagit au moins au sens du protocole X11 avec les applications utilisateur et demeure de ce fait exposé. Outre les privilèges « intrinsèques » (capacités essentiellement) accordés au processus *X* proprement dit, il convient de limiter la vision que ce dernier a du système, afin de lui interdire toute interaction non maîtrisée avec les processus et fichiers réservés au socle CLIP.
- Améliorer la protection du serveur X11 vis-à-vis des applications de la cage $USER_{clip}$, en restreignant les interactions possibles avec ces applications au seul protocole X11. En effet, le serveur X11, même cloisonné vis-à-vis du socle CLIP, doit pour des raisons fonctionnelles

interagir avec des processus de ce dernier (XDM, *xmessage*), et doit donc être considéré comme « moins sacrifiable » que $USER_{clip}$.

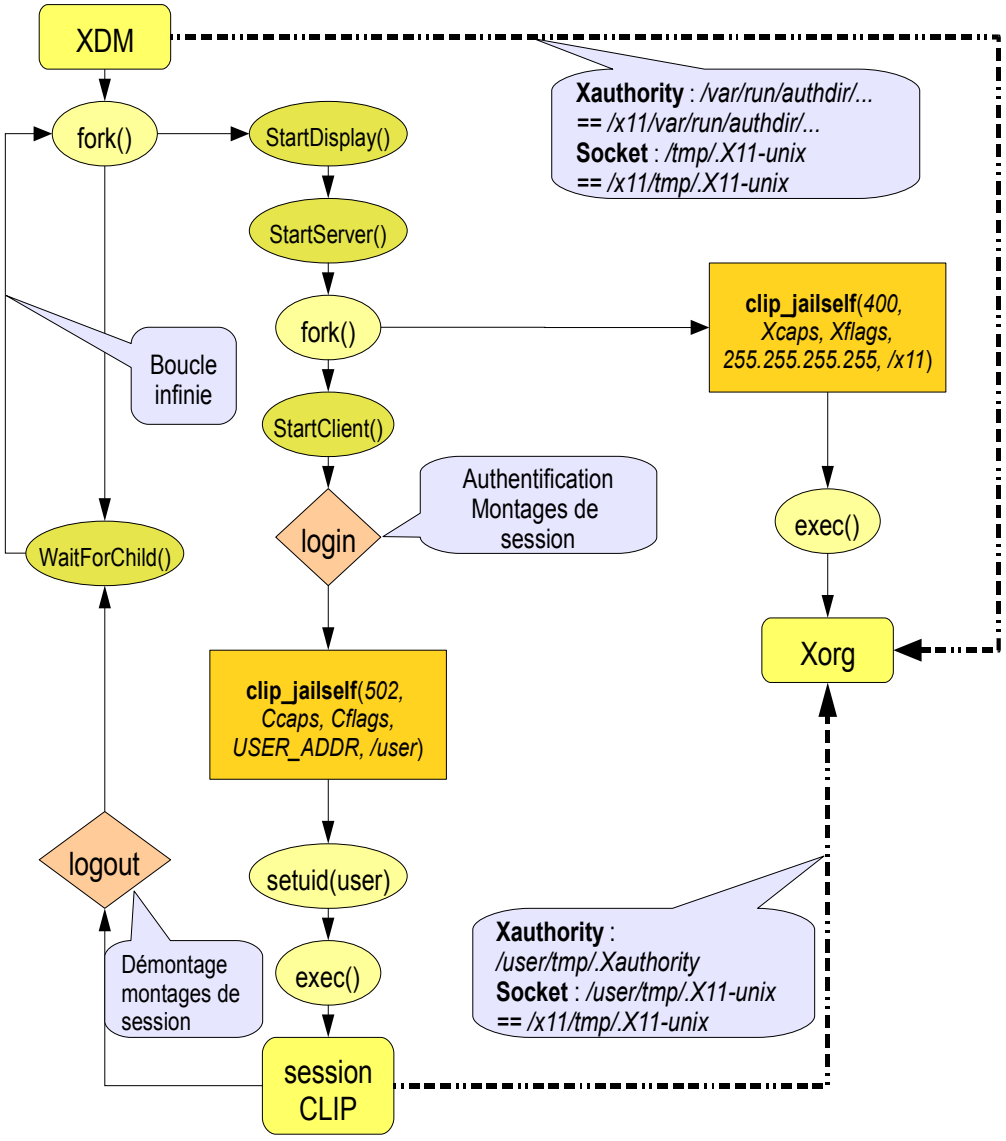

Figure 2: Création des cages X11 et $USER_{clip}$ par le démon XDM.

Cette situation particulière du serveur X11 s'apparente à celle associée au concept de « zone démilitarisée » dans le domaine de la sécurité réseau. Dans un cas comme dans l'autre, un serveur (X11 dans le cas de CLIP, serveur de messagerie d'entreprise par exemple dans le cas réseau) est prestataire de services à la fois pour des éléments internes privilégiés (respectivement socle CLIP et intranet d'entreprise) et des éléments externes potentiellement menaçants ($USER_{clip}$ et internet respectivement). La solution adoptée dans les deux cas est elle aussi similaire : dans le cas réseau, le serveur de messagerie sera typiquement isolé à la fois de l'internet et de l'intranet par des pares-feux qui

restreignent ses échanges avec ces deux domaines aux seuls flux légitimes, tandis que dans le cas de CLIP le serveur X11 est isolé à la fois de $USER_{clip}$ et du socle en étant enfermé dans une cage dédiée, qui ne permet avec le reste du système que les interactions spécifiquement autorisées lors de la configuration de la cage.

La construction de cette cage X11 est détaillée dans le document [CLIP_1304], et son mode de lancement est rappelé dans la Figure 2.

2 Cloisonnement graphique interne au serveur X11

Les mesures décrites dans la présente section visent à assurer, au sein du serveur X11, un cloisonnement entre des clients de différents niveaux de sécurité. A ce titre, ces mécanismes ne sont mis en oeuvre que sur les postes CLIP traitant simultanément à l'écran des informations de niveaux de sécurité hétérogènes, c'est à dire à ce stade uniquement au sein de systèmes CLIP-RM. Les modifications correspondantes du serveur X11 et de ses bibliothèques sont néanmoins aussi intégrées aux autres types de systèmes CLIP, afin d'une part de maintenir une meilleure homogénéité entre les configurations, et d'autre part de faciliter d'éventuelles mises en oeuvre futures de ces extensions au sein de ces systèmes.

2.1 Modification de l'extension *Xsecurity*

Un deuxième ensemble de patches spécifiques à CLIP, nommés selon la convention *<paquetage>-<version>-clip-domains.patch*, est appliqué aux paquetages *x11-base/xorg-server* et *x11-libs/libXext*, ainsi qu'à l'utilitaire *x11-apps/xauth*. Ces patches apportent une amélioration de l'extension *Xsecurity*, qui permet la réalisation du cloisonnement graphique décrit en introduction de cette section. L'extension *Xsecurity*, telle qu'elle est incluse dans un serveur X11 standard, permet de répartir les clients X11 entre deux domaines, *Trusted* et *Untrusted*, selon le *cookie Xauthority* utilisé pour leur authentification auprès du serveur. Les clients du domaine *Untrusted* ne peuvent pas accéder aux fonctions de configuration « sensibles » du serveur, ni à certaines de ses extensions. Ce bridage des privilèges X11 est à ce stade jugé suffisant pour réaliser la protection en intégrité du serveur X11 mentionnée en introduction de la présente section. L'extension *Xsecurity* apporte de plus des propriétés de confidentialité, en interdisant aux clients *Untrusted* d'accéder à la plupart des ressources X11 des clients *Trusted*. En particulier, les opérations de copier-coller ou de capture d'écran depuis un client *Trusted* sont interdites aux clients *Untrusted*. La liste précise des interactions autorisées entre clients *Trusted* et *Untrusted* est donnée dans la documentation de conception de l'extension *Xsecurity* (cf. [XSECURITY]).

Le patch *clip-domains* réalise une adaptation des ces propriétés de confidentialité au contexte de CLIP, en permettant la définition de plusieurs domaines *Untrusted* distincts, qui ont entre eux les mêmes restrictions d'interaction qu'entre les domaines *Trusted* et *Untrusted*. Ainsi, le domaine *Untrusted X* ne peut pas réaliser une capture d'écran sur un client du domaine *Untrusted Y*, pour tous *X* et *Y* distincts. Par ailleurs, une commande supplémentaire, *QueryTrustLevel*, est ajoutée au protocole X11, permettant à un client *Trusted* d'interroger le serveur X11 afin de connaître le domaine d'appartenance d'un autre client. L'utilitaire *xauth* (*x11-apps/xauth*) est modifié de manière à pouvoir générer les *cookies*

Xauthority[2] correspondant à ces différents domaines. Le domaine *Untrusted 1* est associé a priori à RM_H, et le domaine 2 à RM_B, tandis que le domaine *Trusted* est réservé aux clients du socle. Ainsi, on obtient bien l'objectif recherché de protection en intégrité et confidentialité du socle vis-à-vis des cages, et des cages vis-à-vis l'une de l'autre.

La labellisation visible des fenêtres est quant à elle réalisée par la modification du gestionnaire de fenêtre *openbox* (cf. [OPENBOX]), qui est adapté de manière à interroger le serveur X11 – en utilisant l'interface *QueryTrustLevel* ajoutée à ce dernier – de manière à obtenir les domaines de sécurité auxquels appartiennent les différents clients pour lesquels il gère des fenêtres. La couleur de bandeau et le titre de fenêtre affiché par *openbox* sont adaptés en fonction de ces informations, de manière à retranscrire le domaine de sécurité de chaque fenêtre. Les conventions suivantes sont retenues :

- Les fenêtres du domaine de sécurité *Trusted* sont affichées avec un bandeau bleu, et leur titre est systématiquement précédé de *"CLIP: "*.
- Les fenêtres du domaine de sécurité *Untrusted* 1 sont affichées avec un bandeau rouge, et leur titre est systématiquement précédé de *"RM_H: "*.
- Les fenêtres du domaine de sécurité *Untrusted* 2 sont affichées avec un bandeau vert, et leur titre est systématiquement précédé de *"RM_B: "*.
- Les fenêtres de tout autre domaine de sécurité sont affichées avec un bandeau jaune, et un titre précédé de *"OTHER: "*. Aucune fenêtre de ce niveau n'est normalement utilisée dans une session CLIP.

La couleur de chaque type de bandeau est adaptée selon que la fenêtre est au premier plan et a le focus (bandeau plus clair) ou non (bandeau plus sombre). On notera par ailleurs que les visionneuses VNC utilisées pour afficher les bureaux de cages RM (cf. [CLIP_1304] et [CLIP_1401]) sont configurées de telle sorte que leur fenêtre prenne tout l'espace disponible à l'écran, à l'exception de celui occupé par le menu *fbpanel* et par le bandeau de la fenêtre. Ainsi, toute confusion liée au focus est impossible entre les bureaux RM, dont un seul est visible à la fois. De plus, le niveau de sécurité est aussi repris dans une barre de tâches spécifique à CLIP, intégrée au menu *fbpanel* ([FBPANEL]) et dérivée du *plugin "taskbar"* de ce dernier. Cette barre des tâches, qui se présente sous la forme d'un *plugin "mltaskbar"* (*"multi-level taskbar"*), interroge elle aussi le serveur X11 par l'interface *QueryTrustLevel*, et affiche le niveau de chaque client graphique par l'intermédiaire d'un préfixe textuel et d'un carré de couleur, intégrés au *widget* associé au client dans la barre de tâche, et définis selon les mêmes conventions que celles adoptées pour *openbox*.

Openbox est par ailleurs configuré de manière à ne pas permettre à une fenêtre quelle qu'elle soit de s'afficher en plein écran, ni d'être redimensionnée ou déplacée.

2.2 Mise en oeuvre du cloisonnement graphique dans CLIP-RM

L'affichage par le serveur X11 du socle de CLIP des « bureaux » des deux cages RM_H et RM_B repose sur la mise en œuvre du protocole VNC de déport d'affichage. Un serveur VNC (paquetage *net-misc/tightvnc* ([TIGHTVNC]), réduit à sa composante serveur) joue le rôle de serveur X11 à l'intérieur

[2] Un *cookie Xauthority* contient l'équivalent d'une clé d'authentification, nécessaire à l'accès au serveur X11, cf. *man xauth(1)* ou *man Xsecurity(7)*. La méthode d'authentification utilisée sous CLIP est la méthode standard *MIT-MAGIC-COOKIE-1*, décrite dans *man Xsecurity(7)*.

de chaque cage. Un client VNC (composante client de *net-misc/tightvnc*) installé dans le socle vient se connecter à ce serveur (à travers une socket UNIX, cf. 2.3) sur ce serveur, et affiche le bureau complet de la cage comme un *bitmap* dans une fenêtre X11 du socle.

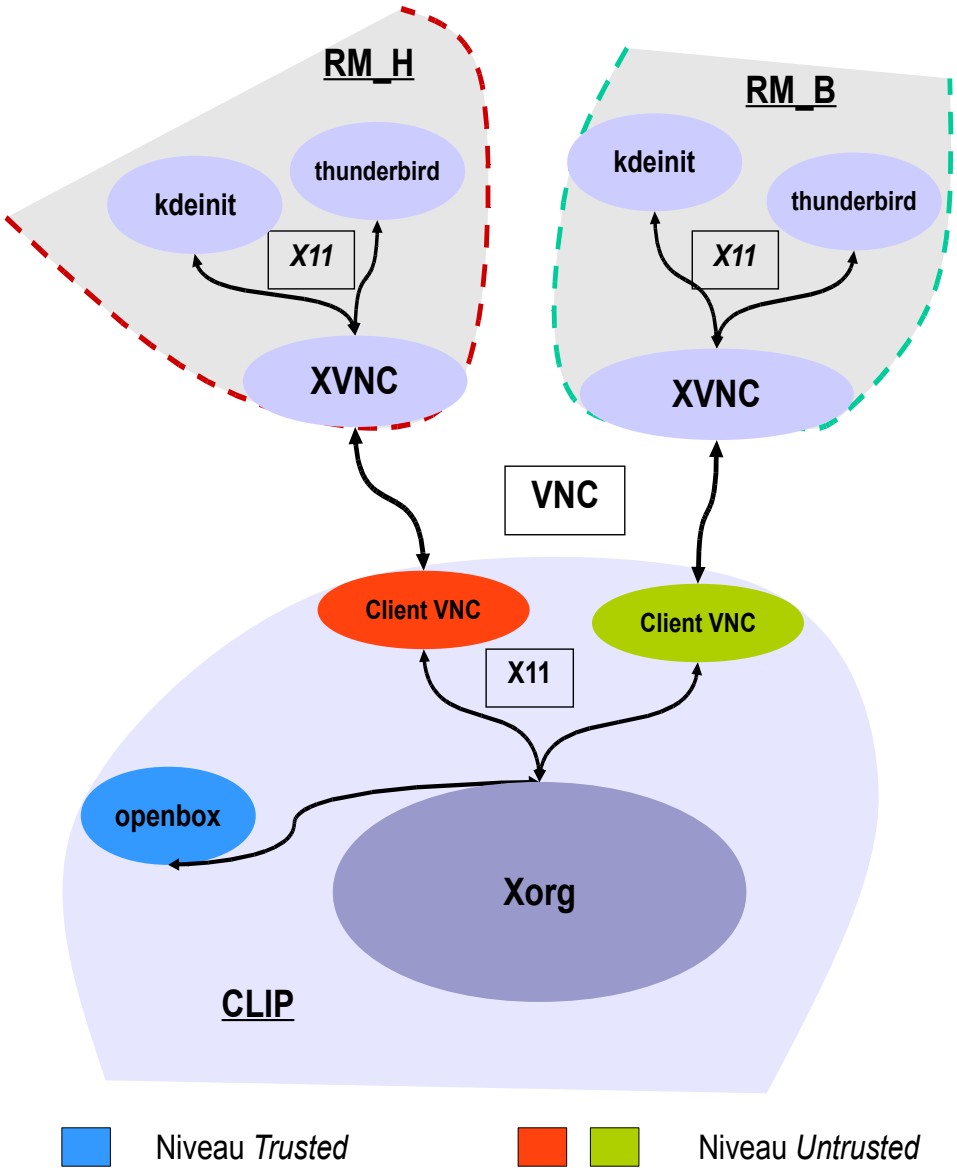

Figure 3: Principes de mise en oeuvre de VNC pour assurer le cloisonnement et la labellisation graphique. On notera bien que seul le client VNC est dans le domaine X11 du socle, et par conséquent labellisé au sens de l'extension *Xsecurity*. Les serveurs XVNC et leurs clients constituent des domaines X11 totalement disjoints, et dépourvus de labellisation spécifique.

Une telle approche présente plusieurs avantages (par rapport notamment à une solution reposant

uniquement sur le modèle réseau X11, avec ou sans proxy[3]) :

- Une rupture protocolaire VNC est imposée entre les domaines réseau X11 des cages et du socle, qui sont ainsi totalement décorrélés : les clients X11 internes à la cage n'existent pas en tant qu'entités X11 distinctes du point de vue du serveur *Xorg* du socle.
- Le serveur X11 des cages (qui est *a priori* exposé à des attaques en intégrité depuis les clients de la cage) est entièrement contenu dans la cage (au sens noyau : instance *Vserver*). En particulier, une compromission d'un client X11 de la cage qui serait propagée à son serveur ne permettrait pas de briser le cloisonnement.
- *A contrario*, le client VNC appartient entièrement au socle, et peut être considéré comme de confiance vis-à-vis du cloisonnement. Il intervient de plus comme médiateur obligatoire de l'affichage de la cage correspondante. Il est donc à même de remplir le rôle des médiateurs de confiance RM_H et RM_B représentés sur la Figure 2.

Dans ce modèle, le client VNC est aussi l'unique client X11 du socle correspondant à l'affichage d'une cage, il suffit donc de labéliser ce seul client pour labéliser « en bloc » toute la cage, et ce de manière complètement décorrélée de l'affichage X11 interne à la cage.

2.3 Vues visionneuses

L'utilisation d'un modèle client/serveur VNC, quoique très adapté au cadre d'emploi de CLIP, demeure toutefois problématique dans la mesure où les communications VNC se font normalement par sockets INET (TCP/IP). Outre les risques d'exposition à distance des serveurs VNC, il serait difficile de mettre au point un cloisonnement réseau sur la boucle locale qui puisse, sans hypothèses sur les cages, offrir de bonnes garanties d'authenticité de la communication client-serveur[4]. Ainsi, il demeurerait possible, suite à une erreur ou à une attaque, de se connecter au serveur VNC de RM_H avec le client VNC destiné à se connecter à RM_B (et labellisé en conséquence).

Cette problématique est prise en compte dans CLIP par un patch spécifique appliqué à *tightvnc* (sources communes au client et au serveur VNC), qui remplace toutes les sockets INET par des sockets UNIX nommées. Ce patch ajoute à *vncserver* une option *–rfbsock* permettant de préciser le chemin d'accès au fichier associé à la socket d'écoute VNC/RFB. De manière symétrique, la connexion par le client se fait par la syntaxe :

```
vncviewer <chemin du fichier socket> (-passwd <fichier de mot de passe> …)
```

plutôt que :

```
vncviewer <adresse IP / nom DNS du serveur>( :<port d'écoute>) (…)
```

[3] Par exemple la solution à base de proxy *Xnest* qui avait été retenue dans la maquette initiale CLIP FreeBSD.

[4] Une autre approche possible serait de confiner chaque client VNC dans un contexte réseau *vserver* associé à une adresse IP dédiée et configurée uniquement sur la boucle locale, pour laquelle des règles *iptables* de filtrage et SNAT n'autoriseraient la communication qu'avec le contexte réseau de la cage RM_H ou RM_B visée. Mais pour être robuste, cette solution nécessiterait d'associer un contexte de sécurité au contexte réseau, et ainsi de créer une cage *vserver* dédiée pour le client VNC. La solution retenue, qui repose sur un cloisonnement *chroot* limité au système de fichier, est plus légère, offre un niveau de robustesse satisfaisant, et présente l'avantage supplémentaire de ne pas exposer un serveur VNC au réseau depuis chaque cage.

Ainsi, le contrôle d'accès au serveur VNC de la cage RM_X peut être assimilé au contrôle d'accès aux fichiers de la cage[5], et plus particulièrement à la socket */vservers/rm_X/user/var/run/vnc/vnc1* normalement créée par le serveur VNC de la cage (lancé dans la vue USER de cette cage suite à une requête d'ouverture de session utilisateur, cf. [CLIP_1401]). Par ailleurs, il est tout aussi important de contrôler l'accès au fichier contenant le *cookie Xauthority* sur lequel repose la labellisation du client VNC, et de s'assurer que le client VNC qui se connecte à RM_X n'a accès qu'au *cookie* correspondant au domaine non privilégié affecté à RM_X.

[5] Dont il est rappelé qu'ils ne sont pas visibles par défaut dans la cage $USER_{clip}$. L'autorisation d'accès doit être donnée explicitement par remontage *bind* depuis l'arborescence de la cage dans celle de la vue $USER_{clip}$.

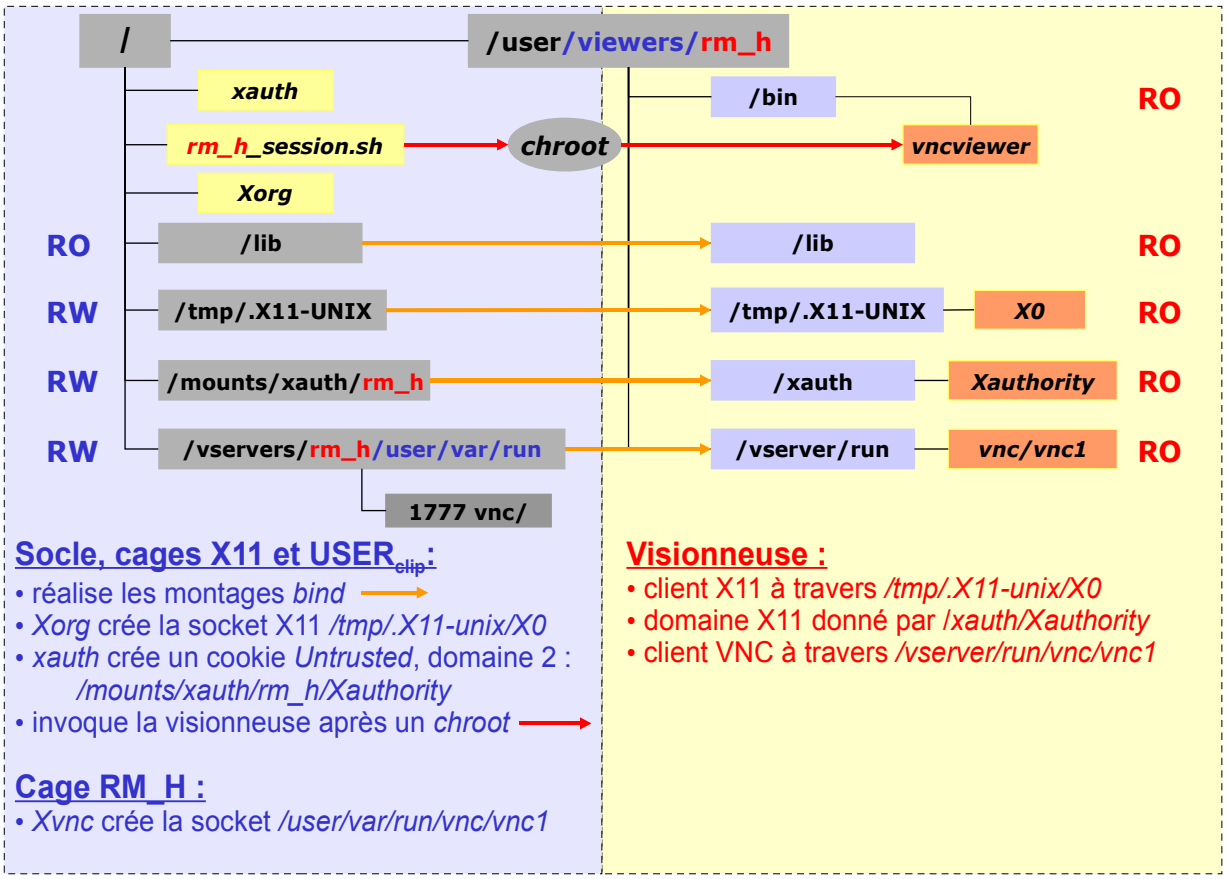

Figure 4: Principe de création d'une vue visionneuse.

Le *chroot* permet notamment de contrôler l'accès du client VNC aux fichiers sensibles identifiés en orange sur le schéma.

Les bibliothèques dynamiques nécessaires au client VNC sont en revanche fournies par les répertoires *<X>/lib* du socle. Le */tmp* du socle est exposé dans chacune des deux vues, tout comme il l'est déjà dans la cage USER$_{clip}$ elle-même, afin d'exposer la socket UNIX du serveur X11.

Le besoin de contrôler de manière incontournable l'accès en lecture à ces fichiers, conjugué à un principe général de réduction des privilèges et de la visibilité sur le système, conduit dans CLIP à enfermer chacun des deux clients VNC, affectés à l'affichage de RM_B ou RM_H respectivement, dans une prison *chroot* qui ne « voit » que les fichiers associés à cette cage. Ces deux prisons constituent des vues de la cage USER$_{clip}$ (par analogie avec les vues USER, UPDATE, etc. des cages RM), et sont construites de manière similaire à cette dernière par des montages *bind* depuis certains répertoires du socle. Le principe de configuration et de fonctionnement de ces vues visionneuses est

résumé par la Figure 4, et détaillé dans le document de référence [CLIP_1304].

Le principe général de cloisonnement graphique qui en résulte est résumé dans la Figure 5.

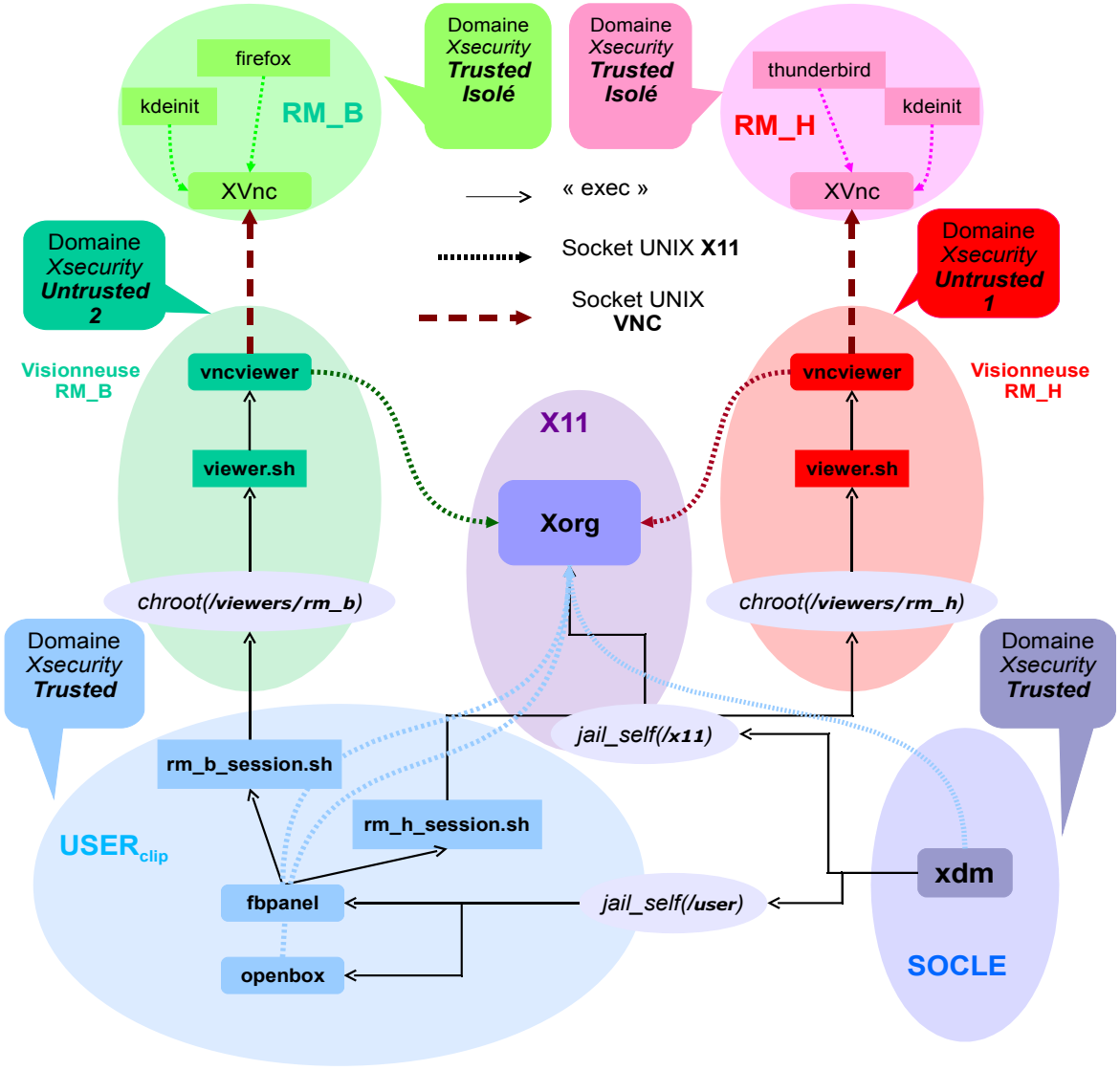

Figure 5: Vue d'ensemble du cloisonnement graphique au sein d'un système CLIP-RM.

Annexe A Références

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

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

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

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

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

[CLIP_1304] *Documentation CLIP – 1304 – Cages et socle CLIP*

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

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

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

[SMM] Loïc Duflot, Olivier Grumelard, Daniel Etiemble, *Utiliser les fonctionnalités des cartes mères ou des processeurs pour contourner les mécanismes de sécurité des systèmes d'exploitation, http://www.ssi.gouv.fr/fr/sciences/fichiers/lti/sstic2006-duflot-papier.pdf*

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

[XORG] *X.org Foundation, http://www.x.org*

[XSECURITY] David P. Wiggins, *Security Extension Specification Version 7.1 X11 Release 6.4*

Annexe B Liste des figures