Documentation CLIP

1204

Privilèges Linux

Version	Date	Auteur	Commentaires
1.0.1	23/01/2009	Vincent Strubel	Passage en priorité d'I/O *idle* autorisée sans privilège pour le processus appelant. A jour pour *clip-kernel-2.6.22.24-r23*.
1.0	10/12/2008	Vincent Strubel	Version initiale, à jour pour le noyau *clip-kernel-2.6.22.24-r20*.

Table des matières

Introduction

La réduction et le contrôle strict des privilèges système constituent des propriétés de sécurité fondamentales des systèmes CLIP. Le présent document vise à référencer et détailler les différents privilèges intervenant au sein du noyau CLIP, afin de faciliter la compréhension de l'architecture de sécurité du système.

Le noyau CLIP correspond à un noyau Linux, auquel sont appliqués un ensemble de *patches*, dont les principaux sont le LSM CLIP (cf. [CLIP_1201]), *vserver* ([CLIP_1202]), et *grsecurity* ([CLIP_1203]). Ces différents *patches* ajoutent des mécanismes de sécurité complémentaires, et les privilèges associés, au noyau Linux standard. Ces éléments complémentaires sont identifiés dans la suite du document par des mots clés [CLIP-LSM], [VSERVER], et [GRSEC], respectivement. En l'absence d'un tel mot clé, les informations présentées sont valables pour un noyau Linux standard, de même version que celle décrite dans le document. Les autres *patches* appliqué au noyau CLIP ne sont pas décrits spécifiquement, dans la mesure où ils n'ont pas d'impact sur la gestion de privilèges, ou bien constituent des compléments aux *patches vserver* et *grsec*, et sont à ce titre décrits au même titre que ces derniers. Par ailleurs, un certain nombre de privilèges applicables au noyau Linux en général n'interviennent jamais au sein des noyaux CLIP, soit parce qu'ils se rapportent à des pilotes ou options de compilation noyau qui ne sont jamais activées sur ces noyaux, soit parce qu'ils sont spécifiques à des architectures de microprocesseurs sur lesquelles CLIP n'est pas supporté à ce jour. Ces éléments non applicables sont néanmoins décrits dans le document (uniquement pour les architectures *i386, x86_64, powerpc, mips, arm, sparc, sparc64* et *ia64*), mais identifiés par une couleur de police grisée.

Les capacités POSIX et LINUX constituent le principal mécanisme de gestion des privilèges au sein des noyaux Linux, depuis la version 2.2. La première section du présent document est ainsi consacrée à une description individuelle de chacune de ces capacités. La section 2 liste quant à elle les privilèges supplémentaires, dits "privilèges CLSM", introduits par le *patch* [CLIP-LSM], tandis que la section 3 recense les quelques rares cas au sein du noyau Linux où des opérations privilégiées restent attachées directement à l'identité *root,* et non à une capacité particulière. On notera enfin que le *patch* [VSERVER] introduit lui aussi des privilèges spécifiques, sous la forme de capacités de contexte, qui peuvent se substituer dans certains cas aux capacités POSIX. Ces capacités sont identifiées au fil de la description des capacités POSIX en section 1, mais ne font pas l'objet d'une section dédiée, dans la mesure où elle ne sont pas mises en oeuvre au sein des systèmes CLIP. Le lecteur est invité à consulter le document de référence [CLIP_1202] pour une description plus détaillées de ces capacités de contexte *vserver*.

1 Capacités

1.1 Capacités POSIX sur les fichiers

CAP_CHOWN

Changement du propriétaire ou du groupe propriétaire d'*inodes* lorsque le *fsuid* de l'appelant est différent de l'*uid* de l'*inode* – inclut les queues de messages POSIX et les *futex* mais pas les *IPC SYSV* (*CAP_SYS_ADMIN*).

NB : un tel changement supprime les bits 's' sauf pour les répertoires.

CAP_DAC_OVERRIDE

- Permissions UNIX et ACL (sauf drapeau *immutable*) :
  - *r,w,x* pour tout répertoire
  - *r,w* pour tout *inode* (sauf IPC *SystemV,* controlées par *CAP_IPC_OWNER*)
  - *x* pour tout *inode* qui a au moins une permission *'x'*.
- Lecture et suivi des liens du */proc.*

CAP_DAC_READ_SEARCH

Permissions UNIX et ACL :

- *r,x* pour tout répertoire
- *r* pour toute *inode* (sauf IPC *SystemV,* controlées par *CAP_IPC_OWNER*).

CAP_FOWNER

Permissions du propriétaire de tout *inode*, sauf cas d'application de *CAP_FSETID* (bits 's'):

- *chmod()*
- mettre un bit *setuid* quelconque
- changement arbitraire du *atime*/*mtime*
- accès en mode "*noatime*" (par *open()* ou par *fcntl()*)
- suppression de fichier dans un répertoire 't'
- modifier les drapeaux de fichier (sauf *immutable* et *append-only* : il faut en plus *CAP_LINUX_IMMUTABLE*)
- déclarer une ACL ext2 / ext3 / ext4
- modifier la "version" *ext2 / ext3 / ext4* (pour *nfs*)
- modifier la réservation d'espace *ext3 / ext4*
- [GRSEC] créer un lien dur vers un *inode* d'*uid* différent du *fsuid* courant, si l'*uid* courant est

différent de 0 (*CONFIG_GRKERNSEC_LINK*)

CAP_FSETID

- Mettre ou garder le bit *setgid* pour un groupe quelconque (il faut aussi CAP_FOWNER pour faire le *chmod()* si l'on n'est pas propriétaire du fichier).
- Pas de suppression des bits *setuid* (et *setgid* s'il y a au moins un 'x') lors de l'écriture sur un fichier projeté en mémoire.

Cette capacité ne permet pas de conserver les bits 's' à travers un *chown()*, contrairement à la norme POSIX.1e.

1.2 Capacités POSIX sur les processus

CAP_KILL

- Sur *exit()*, utilisation d'un signal pré-configuré (par *clone()*) à la place de *SIGCHLD* même si le père ou le fils a fait *exec()* depuis le *clone()* (NB : le nombre d'*exec()* est mesuré par un compteur, dont le débordement est possible)
- Appel *ioctl() KDSIGACCEPT* sur un terminal virtuel différent du terminal de contrôle de l'appelant (envoi d'un signal sur réception d'une combinaison de touches). Alternativement, la capacité *CAP_SYS_TTY_CONFIG* permet aussi de réaliser cette opération.
- Envoi d'un signal quelconque à un processus quelconque du même contexte *vserver* [VSERVER], sous réserve que le statut [CLIP-LSM] de ce dernier le permette (seule une tâche "*capraised*" disposant de *CAP_KILL* peut envoyer un signal à une tâche "*raised*").
- [CLIP-LSM] Envoi, par les signaux d'entrée-sortie sur les fichiers, d'un signal quelconque à un processus quelconque du même contexte *vserver*, sous réserve de compatibilité des statuts *raised / capraised*.
- [CLIP-LSM] Modification du numéro de signal ou du processus destinataire des signaux pour la signalisation des entrées-sorties sur un fichier, lorsque ces informations ont été précédemment modifiées par un processus quelconque du même contexte *vserver*.

CAP_SETGID

- *setfsgid* hors *gid,egid,sgid* de l'appelant
- *set*gid* : changement *gid,egid,sgid* hors *gid,egid,sgid* de l'appelant
- Changement des groupes auxiliaires (*setgroups*)
- Forger des "*credentials*" de s*ocket PF_UNIX* avec un *gid* hors *gid,egid,sgid* de l'appelant

CAP_SETUID

- *setfsuid* hors *uid,euid,suid* de l'appelant
- *set*uid* : changement *uid,euid,suid* hors *uid,euid,suid* de l'appelant

- Forger des "*credentials*" de socket PF_UNIX sur un *uid* hors *uid,euid,suid* de l'appelant
- Bénéfice des bits *setuid/setgid* pour un processus tracé sans *CAP_SYS_PTRACE* (hors attribution des capacités, qui relève de *CAP_SETPCAP*) ou partageant des éléments de son *struct task_struct*: *CLONE_FS*, *CLONE_FILES* et/ou *CLONE_SIGHAND*.

1.3 Capacités supplémentaires Linux

CAP_SETPCAP

- Changement des capacités pour un autre *pid* (sans excéder l'union des capacités de la cible et du donneur)
- Bénéfice des capacités supplémentaires offertes par un exécutable pour un processus tracé sans *CAP_SYS_PTRACE* (pas de *PT_PTRACE_CAP*) ou partageant des éléments de son *struct task_struct* : *CLONE_FS*, *CLONE_FILES* et/ou *CLONE_SIGHAND*.

Cette capacité est masquée dès le démarrage du noyau.

CAP_LINUX_IMMUTABLE

- Changement des drapeaux *immutable* et *append-only*.
- [VSERVER] Modification des attributs *vserver* (*BARRIER, HIDDEN*) d'un fichier par l'appel système *vc_set_iattr()* (avec *CAP_SYS_ADMIN* et *CAP_CONTEXT*)

Cette capacité est masquée par */etc/init.d/sysctl.*

CAP_NET_BIND_SERVICE

- Appel *bind()* d'une *socket INET* (ou autre : *bluetooth, sctp, rose*, ...) sur un port privilégié (<1024 pour *INET*).

CAP_NET_BROADCAST

Inutilisé.

CAP_NET_ADMIN

- Appel *ioctls SIOCGMIIPHY*, *SIOCGMIIREG* : modification des propriétés des interfaces en couches basses, lecture des paramètres WEP, et aussi modification par l'API du *sysfs*.
- Appels *ioctl()* de la couche Ethernet (*ethtool*).
- Appels *ioctl()* de la couche IP (*ifconfig*).
- Lecture/écriture de règles *iptables*.
- Lecture/écriture de règles *arptables*.
- Lecture/écriture des IP *virtual server* (*ipvs*).
- Options de *socket* (*sockopt*) *SO_DEBUG*, *SO_SNDBUFFORCE* et *SO_RCVBUFFORCE*,

option *SO_PRIORITY* avec une valeur <0 ou >6.

- Option *IP_TOS* avec une précédence haute.
- Définition de politique IPsec par *socket*.
- Options *MRT_** (sauf *MRT_INIT*) ailleurs que sur la *socket* pour le routage *multicast.*
- Ajout/modification/suppression de tunnel GRE ou IPIP.
- Ajout/suppression d'une entrée ARP.
- Ajout/suppression de route IP.
- Ajout/modification/suppression de *bridge* Ethernet.
- Création de *sockets PF_KEY* pour la configuration IPsec globale.
- Accès à la couche *XFRM* par *NETLINK*.
- *Drivers*:
  - *Drivers wireless*: configuration de la carte, accès RIDs, lecture clé WEP, etc.
  - *Drivers* PCI: modification des registres matériels MII PHY (gestion de la couche physique).
  - *Driver "bonding"* : *ioctls* de configuration.
  - *Driver "tun"*: changement de drapeaux par un non-propriétaire pour un *tun* alloué.
  - *Driver "eql"*: configuration.
  - Ouverture de */dev/ppp.*
  - Choix du *timeout plip.*
  - Ouverture de canal *slip* (discipline de ligne du terminal série).

Note: cette capacité contrôlait aussi historiquement la modification des contextes réseau *vserver*. Cependant, elle n'intervient plus dans les contextes réseaux de l'interface *vserver* utilisée dans CLIP à ce stade, dans laquelle la configuration des contextes réseau est contrôlée par *CAP_SYS_ADMIN*.

CAP_NET_RAW

- Création de *socket PF_PACKET.*
- Création de *socket PF_INET/SOCK_RAW.*
- Option *sockopt BINDTODEVICE* (attachement d'une *socket* à une interface).
- Choix d'options IP: *IPOPT_SEC, IPOPT_SID,* option non supportée ou drapeaux TS non supportés (option *timestamp*).

CAP_IPC_LOCK

- Utilisation de pages *hugetlbfs* conduisant au dépassement de la limite *RLIMIT_MEMLOCK*, lorsque l'appelant n'est pas membre du groupe défini par le *sysctl vm.hugetlb_shm_group* (*gid 0* par défaut).

- Utilisation de *mlock()* ou *mlockall()* conduisant à un dépassement de la limite *RLIMIT_MEMLOCK*.
- Appel *mmap()* ou *mremap()* sur une projection couverte par un *mlock MCL_FUTURE*, conduisant au dépassement de *RLIMIT_MEMLOCK*.
- Segments de mémoire partagée *SystemV* (*ipc/shm*) : commandes *SHM_LOCK* et *SHM_UNLOCK* sur un segment d'*uid* et *cuid* différents de l'*euid* de l'appelant. Commande *SHM_LOCK* entraînant un dépassement de la limite *RLIMIT_MEMLOCK*.
- *Driver infiniband* : réservation de plage DMA entraînant le dépassement de la limite *RLIMIT_MEMLOCK*.

CAP_IPC_OWNER

Accès à un partage IPC *SystemV* en dehors des permissions UNIX sauf :

- Commandes **_INFO* (*IPC_INFO, MSG_INFO, SEM_INFO, SHM_INFO*) : pas de contrôle.
- Commandes *IPC_RMID* ou *IPC_SET* : propriétaire ou *CAP_SYS_ADMIN*.
- Commandes *SHM_LOCK* ou *SHM_UNLOCK* sur les segments de mémoire partagée : propriétaire ou *CAP_IPC_LOCK*.

CAP_SYS_MODULE

- Initialisation ou déchargement d'un module noyau, sur les noyaux supportant ces opérations.
- Chargement automatique d'un module, lorsque le noyau supporte un tel chargement à la demande (option *CONFIG_KMOD*, non activée sur les noyaux CLIP), dans les cas suivants :
  - Chargement d'un pilote de carte réseau, lors d'un *ioctl()* sur une interface inexistante.
  - Chargement d'un protocole de verrouillage *gfs2*, lors du montage d'un volume *gfs2*.
  - Chargement d'un algorithme de contrôle de congestion *tcp*, suite à une modification du *sysctl net.ipv4.tcp_congestion_control*.
  - Chargement d'un pilote de *dongle irda*, lors de l'initialisation d'un périphérique de ce type.
- Lecture/écriture (réduction uniquement) du *sysctl kernel.cap-bound*.
- [CLIP-LSM] Lecture/écriture (réduction uniquement) des *sysctl kernel.clip.rootcap* et *kernel.clip.mount*.

Cette capacité est masquée par */etc/init.d/reducecap*.

CAP_SYS_RAWIO

- Projection mémoire en bloc d'un fichier (*ioctl() FIBMAP*), pour les systèmes de fichiers qui le supporte (dont *ext3*).
- Ouverture de */proc/kcore*, s'il est supporté.
- Réduction naturelle du risque de suppression de tâche par le *Out of Memory Killer* en cas de surcharge système : *"badness"* divisée par 4 (de manière complémentaire à la réduction offerte

par *CAP_SYS_ADMIN* ou un *uid / euid* nul).

- [GRSEC] Ouverture d'un fichier spécial en mode bloc depuis un processus enfermé dans une prison *chroot* (*CONFIG_GRKERNSEC_CHROOT_CAPS*, non activé sous CLIP)
- Ouverture de /dev/mem, /dev/kmem, /dev/port (entièrement bloqué sous CLIP par GRKERNSEC_IO [GRSEC])
- *Drivers* :
  - Appels *ioctl()* de commande IDE à bas niveau (les manipulations sur la *taskfile* IDE exigent aussi *CAP_SYS_ADMIN*).
  - Appels *ioctl()* de commande ATA à bas niveau (nécessitent aussi *CAP_SYS_ADMIN*).
  - Commande ou *reset* SCSI, exploitation de page mémoire sans remise à zéro, écriture sur */proc/scsi/sg/allow_dio* ou */proc/scsi/sg/def_reserved_size*,
  - Envoi de commandes SCSI autres que *read-safe* et *write-safe* par *ioctl()* sur un périphérique bloc (*read-safe* : avoir ouvert le périphérique suffit; *write-safe* : avoir ouvert en écriture suffit).
  - Certaines opérations de configuration des pilotes réseau *hamradio*, *cosa, bcm43xx* (*wifi*).
  - Certaines opérations de configuration des pilotes *v4l.*
  - Chargement de firmware externe (écriture sur /sys/class/firmware/*/data).
  - Certaines opérations sur les framebuffers vidéo *sis*.
  - Projections *mmap* des fichiers de */proc/bus/pci/*/** . Par ailleurs, les opérations de lecture directe de ces fichiers sont permises à tous les utilisateurs sur une plage standardisée. La capacité *CAP_SYS_ADMIN* permet la lecture en dehors de cette plage. Les accès en écriture à ces fichiers ne sont en revanche contrôlés que par les droits discrétionnaires positionnés à *644*.
- Architectures *i386* et *x86_64*, en l'absence du mécanisme [GRSEC] *GRKERNSEC_IO* : élévation d'*ioperms* (par appel *ioperm()* ou *ioctl() KDADDIO/KDENABIO*) ou d'*iopl.*
- *Architecture i386 : ouverture du device microcode, s'il est supporté.*

NB : *usbfs* n'est pas bridé hors des permissions sur les fichiers.

Cette capacité est masquée au démarrage par */etc/init.d/sysctl*.

CAP_SYS_CHROOT

Appel à *chroot()*. Les permissions *'x'* sont par ailleurs nécessaires sur le répertoire destination.

CAP_SYS_PTRACE

- Récupérer la liste des *futex* d'un processus d'*euid* et *uid* différents de l'*euid* de l'appelant.
- Pour un processus ayant un *uid* ou un *gid* différent des *uid, euid* et *suid* / *gid, egid* et *sgid* de la cible et/ou pour une cible non *dumpable* : attachement par *ptrace* ou accès à */proc/<pid>/environ, /proc/<pid>/fd/*, /proc/<pid>/maps* et */proc/<pid>/mem,* avec *<pid>* le PID (TGID) de la cible. L'accès à */proc/<pid>/mem* d'un autre processus est par ailleurs

restreint au processus auxquels le processus courant est déjà attaché par *ptrace*, et qui sont dans l'état stoppé. [CLIP-LSM] restreint par ailleurs ces accès lorsque le processus cible est *"raised"* : dans ce cas, le processus appelant doit, en plus de *CAP_SYS_PTRACE*, disposer de capacités POSIX permises et héritables, et de privilèges CLSM, au moins égaux à ceux du processus cible.

- Attribution (automatique) du drapeau *PT_PTRACE_CAP* à toute tâche qui se fait tracer par l'appelant, ce qui permet à ces tâches de traverser un *exec()* avec bit *suid*.
- Hors [CLIP-LSM] : tracer un enfant avec plus de capacités que soi ou accéder à son */proc/<pid>/mem, /proc<pid>/fd/*, /proc/<pid>/maps* ou */proc/<pid>/environ*, y compris lorsqu'il demande lui-même à être tracé.
- Architecture IA64 : attachement du *performance monitor* à une cible ayant des *uid, euid* et *suid* / *gid, egid* et *sgid* différents des *uid* et *gid* de l'appelant.

CAP_SYS_PACCT

Activation et désactivation du *process accounting* pour un fichier cible donné.

CAP_SYS_ADMIN

- Certaines opérations de maintenance *autofs* sans être dans le groupe de processus gérant le montage.
- Utilisation de l'appel système *bdflush()* (qui n'est plus honoré - travail repris par les *threads* noyau *pdflush*).
- Utilisation de l'appel système *lookup_dcookie()* (disponible si l'option *CONFIG_PROFILING* est activée).
- *Distributed Lock Manager* : création ou suppression d'un verrou.
- *Distributed Lock Manager* : écriture sur les *clusters* de configuration dans le système de fichiers *configfs*.
- Système de fichiers *gfs2* : configuration des *stores*.
- Système de fichiers *ncpfs* : modification du *charset* d'un montage, *ioctl() NPC_IOC_CONN_LOGGED_IN* et *NPC_IOC_SETROOT*.
- Système de fichiers *udf* : *ioctl() UDF_RELOCATE_BLOCKS*.
- Système de fichiers *smbfs* : *ioctl() NEWCONN,* lorsque l'*uid* appelant est différent de l'*uid* du serveur.
- Accès aux attributs étendus VFS de préfixe *"trusted."*.
- Système de fichiers *XFS* : accès aux attributs étendus de préfixe *"user."*, lorsque le fichier cible n'est ni un fichier régulier, ni un répertoire.
- Système de fichiers *XFS* : modification des *dmattrs*.
- Système de fichiers *XFS* : *ioctl()* de réservation d'espace et de changement de taille de partition, ainsi que certains autres *ioctl* de configuration (gestion des erreurs, etc.). Appels *ioctl()* de manipulation par *handle*.

- Dépassement du nombre maximum de fichiers ouverts (*sysctl fs.file-max*).
- Choix d'une priorité d'I/O *idle* ou *real-time*. [CLIP-LSM] Le choix d'une priorité *idle* pour le processus appelant est autorisé sans capacité particulière. Le choix d'une priorité *idle* pour un processus autre que l'appelant nécessite toujours *CAP_SYS_ADMIN*. Par ailleurs, *CAP_SYS_NICE* est aussi exigée pour le changement de priorité d'un processus d'identité différente de celle de l'appelant.
- Montage et démontage VFS, remontage et changement de type d'un montage. [VSERVER] Les mêmes opérations peuvent être alternativement autorisées par les capacités de contexte suivantes :
  - *VXC_SECURE_MOUNT* : montage avec l'option *nodev* (ajoutée automatiquement si l'appelant ne dispose pas de *CAP_SYS_ADMIN*), démontage d'un montage quelconque.
  - *VXC_SECURE_REMOUNT* : remontage avec l'option *nodev* (ajoutée automatiquement si l'appelant ne dispose pas de *CAP_SYS_ADMIN*).
  - *VXC_BINARY_MOUNT* : montage d'un système de fichiers "binaire" : *smbfs, nfs, coda*.
- Création d'un nouvel espace de nommage par *clone()* (*CLONE_NEWNS, CLONE_NEWUTS, CLONE_NEWIPC*) ou appel à *sys_unshare()* pour se désolidariser de l'espace de nommage hérité.
- Opérations de modification des quotas d'occupation VFS. [VSERVER] La capacité de contexte *VXC_QUOTA_CTL* autorise également ces opérations dans un contexte *vserver* donné.
- Lecture des quotas d'occupation VFS, pour un utilisateur différent de l'*euid* appelant dans le cas des quotas par utilisateur, et pour un groupe dont l'appelant n'est pas membre (au sens des groupes étendus) dans le cas des quotas par groupe. [VSERVER] La capacité de contexte *VXC_QUOTA_CTL* autorise également ces opérations dans un contexte *vserver* donné.
- IPC *SystemV* : modification (changement d'identifiant ou de permissions, ou destruction) lorsque l'appelant n'est pas propriétaire de l'IPC (*euid* appelant différent des *uid* et *cuid* de l'IPC).
- Dépassement de la limite *RLIMIT_NPROC* lors d'un *fork()* (*CAP_SYS_RESOURCE* ou l'identité *root* permettent aussi de dépasser cette limite).
- Appels *ioctl()* sur */dev/snapshot* (suspension sur disque).
- Changement du *hostname* et du *domainname*. [VSERVER] La capacité de contexte *vserver* permet aussi cet opération, portant sur les noms "virtualisés" d'un contexte *vserver* donné.
- Réduction naturelle du risque de suppression de tâche par l'*Out of Memory Killer* en cas de surcharge système. La *"badness"* de la tâche est divisée par quatre (la même réduction est aussi réalisée pour les tâches d'*uid* ou *euid* nul, qui ne disposent pas de *CAP_SYS_ADMIN*).
- Activation ou désactivation d'un fichier de *swap* (*swapon()/swapoff()*).
- Forger des "*credentials*" de *socket PF_UNIX* en utilisant un *pid* différent du *tgid* de l'appelant.
- Accès complet aux journaux noyau (sinon limité à la lecture des messages et de la taille). [CLIP-LSM] Le privilège *CLSM_PRIV_KSYSLOG* permet aussi cet accès. [VSERVER] De même, la capacité de contexte *VXC_SYSLOG* permet cet accès aux processus auxquels elle s'applique.

- Modification (augmentation uniquement) du *sysctl kernel.tainted.*
- [GRSEC] dépassement de la limite *RLIMIT_NPROC* lors d'un *exec()*. La capacité *CAP_SYS_RESOURCE* ou l'identité *root* permettent également de dépasser cette limite. (*CONFIG_GRKERNSEC_EXECVE*, non activé sous CLIP)
- [GRSEC] Accès quelconque aux journaux noyau (*dmesg*) (*CONFIG_GRKERNSEC_DMESG*, non activé sous CLIP à ce stade)
- [VSERVER] Appels systèmes privilégiés (qui nécessitent également *CAP_CONTEXT*), pour réaliser les opérations suivantes :
    - Création d'un contexte de sécurité, lecture des drapeaux et capacités d'un contexte de sécurité, changement de contexte de sécurité.
    - Modification des drapeaux et capacités (POSIX et de contexte) d'un contexte de sécurité (nécessite également *CAP_SYS_RESOURCE*).
    - Création ou destruction d'un contexte réseau, lecture des drapeaux et capacités d'un contexte réseau, changement de contexte réseau.
    - Ajout ou retrait d'adresses IP autorisées pour un contexte réseau, modification des drapeaux et capacités d'un contexte réseau (nécessite également *CAP_SYS_RESOURCE*).
    - Modification des noms (*hostname, domainname*, etc.) associés à un contexte (nécessite également *CAP_SYS_RESOURCE*).
    - Modification des attributs *vserver* (*BARRIER, HIDE, IUNLINK*, etc…) d'un fichier par l'appel système *vc_set_iattr()* (nécessite également *CAP_LINUX_IMMUTABLE*)
    - Lecture des masques de *rlimits* associées à un contexte.
    - Modification des *rlimits* associées à un contexte, modification, ajout ou suppresion de quotas d'espace disque par contexte, modification de l'ordonnanceur par contexte (nécessite également *CAP_SYS_RESOURCE*).
- [CLIP-LSM] Modification de la configuration *veriexec* en contexte ADMIN
- [CLIP-LSM] Ajout, modification ou suppression de contextes *veriexec* depuis le contexte ADMIN (avec *CAP_CONTEXT*)
- [CLIP-LSM] Ajout ou suppression d'entrées *devctl*. Ces opérations ne sont possibles que si le *sysctl kernel.clip.mount* a une valeur non nulle.
- *Drivers* :
    - Périphériques bloc: *ioctl()* d'ajout ou suppression de partition, de relecture des partitions, de modification de la taille du *read-ahead*, de changement de taille de bloc, de *flush* du cache, de déclaration en mode *ro/rw.*
    - Redéfinition de la géométrie associée à un type de disquette prédéfini, utilisation de l'ioctl FDSETDRVPRM.
    - Lecture de clé ou modification de l'état (si l'*uid* appelant est différent de celui qui a mis la clé) d'un */dev/loop cryptoloop*. [VSERVER] La capacité de contexte *vserver VXC_CLOOP_ADMIN* permet également de réaliser ces opérations.

- ATA-SCSCI : *ioctl() HDIO_DRIVE_CMD* et *HDIO_DRIVE_TASK*. La capacité *CAP_SYS_RAWIO* est également exigée.
- *Reset*, déverrouillage (si utilisé) ou *debug* du lecteur CDROM.
- Accès privilégié DRM (*Direct Rendering Management*): *ioctl()* privilégiés de reconfiguration (drapeaux *DRM_ROOT_ONLY* et *DRM_AUTH*), projection de zones mémoires réservées ou projection en écriture de zones en lecture seule, *ioctl()* d'ajout de projections PCI, *sg* ou *framebuffer*.
- Modification de certains paramètres d'une interface série, ignorer des erreurs de lancement d'un */dev/ttyS*, avoir plus d'informations sur son état.
- Remise à zéro automatique des statistiques *lp* après lecture par l'*ioctl LPGETSTATS*.
- Réinitialisation et calcul de *checksum* sur la *nvram* (*ioctl()* sur */dev/nvram*)
- Ajout au *pool* d'entropie du générateur d'aléa ou modification de l'estimatif de sa qualité ou mise à zéro de l'entropie existante (*ioctl()* sur */dev/[u]random*)
- Association d'un */dev/raw* à un périphérique bloc.
- Ouverture d'un *tty* passé en mode exclusif (drapeau *TTY_EXCLUSIVE*), injection sur un terminal qui n'est pas celui de contrôle (mais l'écriture sur */dev/input/event** ne vérifie pas les capacités), redirection de la console système, vol de terminal de contrôle à une autre session.
- Modification des *locked termios* (*ioctl() TIOCSLCKTRMIOS*).
- Opération spéciale par *ioctl()* sur un terminal virtuel qui n'est pas celui de contrôle (copier-coller, etc.), redirection des messages noyau
- Définition ou modification de la *secure attention key* (SAK) via un *ioctl() KDSKBENT* sur le terminal virtuel. [GRSEC] La capacité *CAP_SYS_TTY_CONFIG* est également nécessaire pour réaliser un tel *ioctl()*.
- Modification de la discipline de ligne d'un *tty*.
- Accès aux variables EFI via *sysfs*.
- Ecriture sur */proc/ide/*/*/{driver,settings}* et *ioctl()* de commandes IDE bas niveau, différents de ceux contrôlés par *CAP_SYS_RAWIO* (état du bus, *nice*, *reset*; les manipulations sur la *taskfile* IDE nécessitent à la fois *CAP_SYS_ADMIN* et *CAP_SYS_RAWIO*).
- Configuration du *device mapper* LVM. [VSERVER] La capacité de contexte *vserver VXC_ADMIN_MAPPER* autorise également ces opérations.
- Configuration des */dev/md* (RAID) par *ioctl()* (tous) ou écriture sur les fichiers d'attributs correspondants dans le *sysfs*.
- Opérations de diagnostic et de *debug* sur certains périphériques accessibles par *sbus*.
- Différentes opérations de diagnostic et de configuration sur les périphériques *v4l* (options de *debug* génériques et opérations spécifiques aux différents pilotes).
- Ajout / suppression d'interfaces *DVB*.

- Accès en lecture à davantage d'informations (plages arbitraires) sur la configuration PCI par *sysfs* ou *procfs* (écriture: pas de contrôle), accès en lecture ou écriture à la configuration PCI par appel système.
- Certaines opérations de configuration PCMCIA : *ioctl()* en écriture (*IOC_OUT*), *DS_BIND_MTD, DS_BIND_REQUEST*.
- Opérations *ioctl()* de contrôle sur les périphériques SCSI. La capacité *CAP_SYS_RAWIO* est également exigée pour les opérations génériques (*sg* et *scsi*). Pour les autres opérations (couche *sd* et pilotes de contrôleurs spécifiques), *CAP_SYS_ADMIN* seule suffit.
- Carte son *SB Live* (*emu10k1*): arrêt carte son, certaines opérations bas niveau. Les autres cartes son ne sont pas contrôlées.

- Architecture *i386* :
  - Ouverture de */dev/apm*.
  - Commande *vm86 VM86_REQUEST_IRQ*.
  - Ecriture, et *ioctl() ADD/SET/DEL/KILL ENTRY/PAGE_ENTRY* sur les *mtrr*.
- Architecture *x86_64* :
  - Appels *ioctl()* de lecture et configuration du *Machine Check Exception*.
- Architecture *powerpc* :
  - Configuration de l'interface *Machine Facility* sur les systèmes *iSeries*
  - Commandes *RTAS* sur les systèmes de type *CHRP*.
  - *Macintosh* : mise en sommeil PMU.
  - *Macintosh* : contrôle des LEDs faciales.
- Architectures *sparc* et *sparc64* :
  - Montage par l'appel système *sunos_mount()* (en revanche, les montages NFS par *sunos_nfs_mount()* ne sont pas contrôlés).
- Architecture *ia64* :
  - Ouverture des fichiers de */proc/sal/**.
  - Gestion des erreurs du port série virtuel (*simserial*).
  - Contrôle du module *tiocx*.
- Architecture *arm* :
  - Ouverture en mode privilégié de */dev/apm* (émulé).
- Architecture *mips* :
  - Appel *prctl PR_SETSTACKSIZE* à travers la couche de compatibilité *IRIX*.

CAP_SYS_BOOT

- Utilisation de l'appel système *kexec_load()*.

- Utilisation de l'appel système *reboot()* (contrôle l'activation du raccourci Ctrl-Alt-Del, l'arrêt ou le redémarrage de la machine). [VSERVER] L'appel *reboot()* lancé avec *CAP_SYS_BOOT* dans un contexte non-ADMIN est automatiquement redirigé vers l'éventuel gestionnaire virtuel de *reboot,* et ne redémarre en aucun cas le poste.

CAP_SYS_NICE

- Choix d'une priorité d'I/O pour un processus d'*uid* différent des *euid* et *uid* de l'appelant. [CLIP-LSM] De plus, indépendamment de *CAP_SYS_NICE*, cette opération est interdite pour un appelant qui n'est pas *capraised* vis-à-vis d'une tâche *raised* disposant de capacités permises ou héritables supérieures à celles de l'appelant.
- Diminution du *nice* de la tâche appelante, réduction du *nice* en dessous de la limite *RLIMIT_NICE.*
- Passage de la tâche appelante en priorité *real-time* au dessus de la limite *RLIMIT_RTPRIO*.
- Changement de la priorité, de l'affinité CPU ou du *scheduler* d'un processus d'*euid* et *uid* différents de l'*euid* appelant. [CLIP-LSM] De plus, indépendamment de *CAP_SYS_NICE*, cette opération est interdite pour un appelant qui n'est pas *capraised* vis-à-vis d'une tâche *raised* disposant de capacités permises ou héritables supérieures à celles de l'appelant.
- Déplacement des pages (appel système *sys_move_pages()* ou *sys_migrate_pages()*) d'un processus de *suid* et *uid* différents des *uid* et *euid* de l'appelant (avec *CONFIG_MIGRATION* ou *CONFIG_NUMA*).
- Appels *sys_move_pages(), sys_mbind()* ou *sys_migrate_pages()* avec le drapeau *MPOL_MF_MOVE_ALL* (avec *CONFIG_MIGRATION* ou *CONFIG_NUMA*).
- Migration vers un *node* mémoire différent de ceux attribués à la tâche appelante (avec *CONFIG_NUMA*).
- Architecture *powerpc* (*cell*) : création d'un *inode spufs* avec le drapeau *SPU_CREATE_NOSCHED*.

CAP_SYS_RESOURCE

- Réglage de la fréquence des interruptions des horloges *HPET* (*event timer*) ou *RTC* au dessus du maximum.
- Création d'une console virtuelle au delà du nombre maximum.
- Dépassement du nombre de *mappings* clavier (correspondant aux états des touches modificatrices) lors d'un *ioctl() KDSKBENT* sur un terminal virtuel.
- Dépassement des quotas VFS d'*inodes* et/ou de blocs (uniquement en l'absence de drapeau *root squash*, et pour un format de quota différent de *QFMT_VFS_OLD*).
- Accès aux blocs réservés au propriétaire sur un système de fichiers *ext2 / ext3 / ext4* quasi-saturé. Le fait d'être propriétaire du système de fichiers, c'est-à-dire de posséder le même *fsuid* que l'*uid* propriétaire du système de fichiers, ou encore d'être membre (au sens des groupes étendus) du groupe propriétaire de celui-ci, lorsque ce groupe a un *gid* non nul, permet également d'accéder à ces blocs sans *CAP_SYS_RESOURCE*.

- Dépassement de la limite inférieure d'espace libre sur un système de fichiers *ufs*.
- Déverouillage des quotas sur un système de fichiers *gfs2*.
- Modification du drapeau *journal_data,* ajout ou extension d'un groupe sur un système de fichiers *ext3 / ext4*.
- Réduction du malus (*badness*) à la sélection pour suppression de tâche par l'*Out of Memory Killer* en cas de surcharge système (écriture sur */proc/<pid>/oom_adj,* pour une tâche quelconque). On notera que [CLIP-LSM] interdit par ailleurs toute augmentation de ce malus pour les tâches disposant du privilège *CLSM_PRIV_IMMORTAL*.
- Injection de faute dans une tâche quelconque, lorsque le sous-système d'injection de faute est compilé dans le noyau (option *CONFIG_FAULT_INJECTION*).
- Dépassement de la taille (nombre de messages supérieur au maximum courant, mais pas à la limite *HARD_MSGMAX*, taille de message supérieure à la taille maximale courante) pour une queue de messages POSIX.
- Dépassement du nombre maximum de queues de message POSIX.
- Dépassement de la taille maximale pour une queue de message IPC *SystemV* (commande *IPC_SET*)
- Dépassement de la limite *RLIMIT_NPROC* lors d'un appel *fork()*. La capacité *CAP_SYS_ADMIN* ou l'identité *root* permettent également un tel dépassement.
- Elévation d'une limite *rlimit* au dessus de la limite *hard* courante. [VSERVER] La capacité de contexte *VXC_SET_RLIMIT* permet également cette opération dans un contexte *vserver*.
- La capacité *CAP_SYS_RESOURCE* est la seule, en plus des capacités sur les fichiers (section 1.1), à ne pas être masquée automatiquement par *nfsd*.
- [GRSEC] Dépassement de la limite *RLIMIT_NPROC* lors d'un *execve(),* lorsque *CONFIG_GRKERNSEC_EXECVE* est activé. Dans ce cas, la capacité *CAP_SYS_ADMIN* permet également un tel dépassement.
- [VSERVER] Cette capacité est exigée, en plus de *CAP_CONTEXT* et *CAP_SYS_ADMIN*, pour les opérations suivantes :
  - Envoi d'un signal à tous les processus d'un contexte de sécurité.
  - Attribution des *namespaces* de la tâche appelante à un contexte de sécurité.
  - Modification des drapeaux et masques de capacités (POSIX ou de contexte) d'un contexte de sécurité ou d'un contexte réseau.
  - Modification des *rlimits* d'un contexte de sécurité.
  - Modification de l'ordonnanceur d'un contexte de sécurité.
  - Ajout, suppression ou modification du quota d'espace disque d'un contexte de sécurité.
  - Ajout ou suppression d'adresses IP à un contexte réseau.
  - Modification des noms (*hostname, domainname*, etc.) associés à un contexte.

CAP_SYS_TIME

- Appel *ioctl()* sur l'horloge RTC (et sur les périphériques RTC spécifiques) pour changer l'heure, le coefficient correctif (PLL) ou l'origine des temps (*epoch*).
- Configuration de l'horloge système (*stime()*, *settimeofday()*, *adjtimex()*)

CAP_SYS_TTY_CONFIG

- Appels *ioctl()* sur les terminaux virtuels pour la configuration du clavier: *KDSKBENT / KDGKBENT* (*mapping*), *KDSKBSENT* (touches fonction), *KDKBDREP* (répétition), *KDSETKEYCODE* / *KDGETKEYCODE* (*scancodes*)
- Configuration de la commutation des terminaux virtuels: *ioctl() VT_LOCKSWITCH* et *VT_UNLOCKSWITCH.*
- Appels *ioctl()* spécifiques sur tout terminal virtuel qui n'est pas le terminal de contrôle du terminal courant : *KIOCSOUND, KDMKTONE, KDSETMODE, KDSKBMODE, KDSKBIACR, KDSKBLED, KDSETLED, KDSIGACCEPT, VT_SETMODE, VT_ACTIVATE, VT_WAITACTIVE, VT_RELDISP, VT_RESIZE, VT_RESIZEX, PIO_FONT, GIO_FONT, PIO_CMAP, PIO_FONTX, GIO_FONTX, PIO_FONTRESET, KDFONTOP, PIO_SCRNMAP, PIO_UNISCRNMAP, PIO_UNIMAPCLR, PIO_UNIMAP, GIO_UNIMAP.* La capacité *CAP_KILL* suffit par ailleurs à réaliser un *ioctl() KDSIGACCEPT* sur un terminal différent du terminal de contrôle.
- Appel système *vhangup()* sur le terminal de contrôle de l'appelant.
- Architectures 64 bits avec support des exécutables 32 bits (*x86_64* avec *CONFIG_IA32_EMULATION, powerpc* avec *CONFIG_PPC64, mips* avec *CONFIG_MIPS32_COMPAT*, *sparc64* avec *CONFIG_SPARC32_COMPAT, ia64* avec *CONFIG_IA32_SUPPORT*) : *ioctl()* 32 bits privilégiés sur un terminal qui n'est pas le terminal de contrôle.

CAP_MKNOD

- Création par *mknod()* d'un fichier spécial en mode caractère ou bloc.
- Système de fichiers *xfs* : appel *ioctl() XFS_IOC_FSSETDM_BY_HANDLE.*

CAP_LEASE

- Mettre un bail (*fcntl(... F_SETLEASE ...)*, aussi utilisé directement par *nfs4*) sur un *inode* dont l'*uid* est différent du *fsuid* de l'appelant.

CAP_AUDIT_WRITE

- Envoi de messages d'audit via *netlink,* lorsque le sous-système d'audit est intégré au noyau.

CAP_AUDIT_CONTROL

- Lorsque le sous-système d'audit est intégré au noyau, permet l'écriture sur */proc/<pid>/loginuid*, pour *<pid>* correspondant au PID (TGID) de l'appelant. L'écriture sur le

/proc/<pid>/loginuid d'une autre tâche n'est en aucun cas permise.

- Configuration du sous-système d'audit via *netlink*.

CAP_CONTEXT

- [VSERVER] Accès à l'appel système multiplexé *sys_vserver* (d'autres capacités peuvent être nécessaires selon la commande). Avec l'option *CONFIG_VSERVER_LEGACY*, l'appel système *NEW_S_CONTEXT* est néanmoins autorisé sans la capacité *CAP_CONTEXT* (ni aucune autre capacité).
- [VSERVER] Lecture ou écriture par *ioctl() EXT[3,4]_IOC_GETTAG, EXT[3,4]_IOC_SETTAG, FIOC_GETTAG, FIOC_SETTAG, FIOC_GETXFLG* et *FIOC_SETXFLG* des drapeaux de fichiers dans */proc* (équivalent aux attributs de fichiers pour */proc*, gestion de la visibilité dans le *procfs*. Ces *ioctl()* ne sont supportés que lorsque l'option de compatibilité ascendante *CONFIG_VSERVER_LEGACY* a été sélectionnée.
- [CLIP-LSM] Accès en lecture (écriture - ajout ou suppression d'entrées ou de contextes - si combiné à *CAP_SYS_ADMIN*) à un contexte *veriexec* autre que le contexte ADMIN, depuis le contexte ADMIN. Il est rappelé que l'accès à un autre contexte depuis un contexte autre que ADMIN est interdit, sauf si le contexte appelant est aussi le contexte UPDATE de la base *veriexec* (auquel cas, le privilège *CLSM_PRIV_VERIEXEC* suffit).

2 Privilèges CLIP-LSM

Le module de sécurité CLIP-LSM (cf. [CLIP_1201]), propre à CLIP, ajoute aux modèle de capacités POSIX standard un certain nombre de privilèges spécifiques, qui peuvent être répartis en deux catégories : les privilèges complémentaires du modèle de capacités, d'une part, qui contrôlent des opérations non couvertes par ce dernier, ou ajoutent des restrictions à certaines des opérations privilégiées normalement permises par une capacité donnée, et les privilèges "substituts" au modèle de capacités, permettant de réaliser certaines opérations nécessitant normalement une capacité, sans disposer de cette capacité.

2.1 Privilèges complémentaires

Les privilèges supplémentaires suivants sont à ce stade mis en oeuvre au sein du noyau CLIP :

CLSM_PRIV_CHROOT

Privilège nécessaire au contournement des différents contrôles spécifiques réalisés par le LSM CLIP lors d'une opération *chroot()*.

CLSM_PRIV_VERICTL

Privilège exigé pour toute opération d'administration de *veriexec* par *ioctl* sur */dev/veriexec*, lorsque le contexte *veriexec* courant est actif. On notera que la capacité *CAP_SYS_ADMIN* est de plus exigée dans le cas d'opérations réalisées dans le contexte 0.

CLSM_PRIV_NETCLIENT

Permet la création d'une *socket* de famille autre que *AF_UNIX* ou *AF_NETLINK*. Nécessaire pour réaliser un appel *connect()* sur une telle *socket*, ainsi qu'un appel *sendmsg()* sur une telle *socket* lorsqu'elle n'est pas encore connectée.

CLSM_PRIV_NETSERVER

Permet la création d'une *socket* de famille autre que *AF_UNIX* ou *AF_NETLINK*. Nécessaire pour réaliser un appel *bind(), accept()* ou *listen()* sur une telle *socket*.

CLSM_PRIV_NETOTHER

Permet la création d'une *socket* de famille autre que *AF_UNIX* ou *AF_NETLINK*.

CLSM_PRIV_NETLINK

Nécessaire pour créer une *socket* de la famille *AF_NETLINK*.

CLSM_PRIV_IMMORTAL

Interdit la réception de signaux envoyés par une tâche qui ne possède pas la capacité *CAP_SYS_BOOT*

dans son masque héritable, et réduit la probabilité d'être tué par le *Out-Of-Memory Killer*.

CLSM_PRIV_KEEPPRIV

Entraîne la conservation de tous les privilèges CLSM de la tâche lors d'un changement d'identité de *root* à non-*root* (*setuid()* ou équivalent), ou d'un changement de contexte *vserver*.

CLSM_PRIV_XFRMSP

Nécessaire, en plus de *CAP_NET_ADMIN*, pour tout ajout ou suppression de politique de sécurité (SP) IPsec.

CLSM_PRIV_XFRMSA

Nécessaire, en plus de *CAP_NET_ADMIN*, pour tout ajout, mise à jour ou suppression d'association de sécurité (SA) IPsec.

CLSM_PRIV_SIGUSR

Permet à une tâche d'un contexte *vserver* non privilégié d'envoyer un signal *SIGUSR1* ou *SIGUSR2* à une tâche du contexte 0 (ADMIN) qui dispose du privilège *CLSM_PRIV_RECVSIG*.

CLSM_PRIV_RECVSIG

Permet à une tâche du contexte *vserver 0* de recevoir un signal *SIGUSR1* ou *SIGUSR2* envoyé par une tâche d'un autre contexte, disposant du privilège *CLSM_PRIV_RECVSIG*.

2.2 Privilèges substituts

Les privilèges substituts du LSM CLIP ont généralement pour but de permettre certaines opérations privilégiées depuis un contexte *vserver* non privilégié, sans pour autant autoriser la capacité contrôlant normalement ces opérations (et généralement d'autres, qui ne doivent pas être autorisées dans le contexte) dans ce même contexte. Ces privilèges s'apparentent ainsi aux capacités de contexte *vserver* (cf. [CLIP_1202]), à ceci près qu'ils sont une propriété d'une tâche donnée, et non du contexte dans son intégralité.

CLSM_PRIV_PROCFD

Permet l'accès aux descripteurs de fichiers ouverts d'une autre tâche (du même contexte *vserver*), à travers les fichiers */proc/<pid>/fd/** et */proc/<pid>/maps* (avec *<pid>* le PID de la tâche cible), sans disposer de la capacité *CAP_SYS_PTRACE*.

CLSM_PRIV_KSYSLOG

Permet un accès complet au *buffer* de journaux noyaux, sans posséder la capacité *CAP_SYS_ADMIN*, ni apartenir au contexte *vserver* 0.

3 Opérations contrôlées par un test direct sur l'identité

Les privilèges suivants sont accordés par test direct à toute tâche d'*euid* nul :

- Sans [CLIP-LSM] : envoi de signaux *SIGIO/SIGURG* via un fichier dont on a défini le *pid* propriétaire, mais dont le processus a un *suid* (respectivement *uid*) différent de l'*uid* (respectivement *euid*). Avec [CLIP-LSM], l'envoi de signaux via un fichier est soumis à des contrôles équivalents à ceux d'un envoi direct par *kill()* : contrôle de *CAP_KILL* pour l'envoi à un processus d'identité différente, contrôle des masques de capacités permises et héritables pour l'envoi à une tâche *raised* par une tâche non *capraised*, contrôle des contextes *vserver*.
- Attacher une tâche de *suid* et *uid* différents de l'*euid* appelant à un ensemble de CPUs, via le système de fichiers des *cpusets*.
- Réduction du risque de suppression de tâche par l'*OOM killer* en cas de surcharge système, lorsque la tâche concernée ne possède par ailleurs pas *CAP_SYS_ADMIN*.
- Les tests de permissions discrétionnaires sur les variables *sysctl* (aussi bien pour l'appel système *sysctl()* que pour l'accès à travers */proc*) considèrent l'appelant comme propriétaire d'une variable quelconque si et seulement si son *euid* est nul.

Les privilèges suivants sont accordés par test direct à toute tâche d'*uid* nul :

- Ouverture du *driver agpgart* (*uid* ou *suid*).
- Réduction du risque de suppression de tâche par l'*OOM killer* en cas de surcharge système, lorsque la tâche concernée ne possède par ailleurs pas *CAP_SYS_ADMIN*, et a un *euid* non nul.

Les privilèges suivants sont accordés par test direct sur le champ *user* de la *struct task_struct (user == &root_user)* :

- Dépassement de la limite de processus *RLIMIT_NPROC* lors d'un appel *fork()* ou d'un changement d'utilisateur, lorsque la tâche appelante ne possède par ailleurs aucune des capacités *CAP_SYS_ADMIN* et *CAP_SYS_RESOURCE*. On notera que ce privilège ne permet en revanche pas de dépasser la limite *RLIMIT_NPROC* globale d'un contexte de sécurité *vserver*.
- Dépassement de la limite de processus *RLIMIT_NPROC* d'un utilisateur, lors d'un changement d'identité d'un processus (changement d'*uid*). Dans ce cas, les tests sont réalisés sur le nouvel utilisateur (aussi bien le test d'identité que le test de limite). On notera que contrairement au *fork()*, les capacités *CAP_SYS_ADMIN* et *CAP_SYS_RESOURCE* ne sont pas prises en compte dans ce cas.

Par ailleurs, les *euid/uid/suid* nuls entraînent un traitement spécifique quant à la gestion des capacités POSIX et LINUX :

- Attribution du masque de capacité par défaut (*sysctl kernel.clip.rootcap* avec [CLIP-LSM], *CAP_FULL_SET* sinon) aux différents masques de capacité d'un exécutable, en fonction des

identités de l'exécutable : masques permis et héritable pour *euid* ou *uid* nul, masque effectif pour *euid* nul uniquement.

- Conservation des capacités permises et héritables lors d'un changement d'*euid* dès lors que *uid* ou *suid* reste nul.
- Copie du masque permis vers le masque effectif lors d'un changement d'*euid* lorsque le nouvel *euid* est nul.
- [CLIP-LSM] Copie des privilèges CLSM sauvegardés dans les privilèges effectifs lors d'un changement d'*euid*, lorsque le nouvel *euid* est nul.
- Copie des capacités « fichier » (*CAP_FS_MASK*) du masque permis vers le masque effectif lors d'un changement de *fsuid* lorsque le nouvel *fsuid* est nul.

Annexe A Références

[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*