Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

PREMIER MINISTRE

Secrétariat général
de la défense
et de la sécurité nationale

*Agence nationale de la sécurité
des systèmes d'information*

Paris, le 8 octobre 2015

N° DAT-NT-28/ANSSI/SDE/NP

Nombre de pages du document
(y compris cette page) : 51

Note technique

Recommandations de configuration d'un système GNU/Linux

Public visé:

Développeur	
Administrateur	✓
RSSI	✓
DSI	
Utilisateur	

Informations

Avertissement

Ce document rédigé par l'ANSSI présente les « **Recommandations de configuration d'un système GNU/Linux** ». Il est téléchargeable sur le site www.ssi.gouv.fr. Il constitue une production originale de l'ANSSI. Il est à ce titre placé sous le régime de la « Licence ouverte » publiée par la mission Etalab (`www.etalab.gouv.fr`). Il est par conséquent diffusable sans restriction.

Ces recommandations sont livrées en l'état et adaptées aux menaces au jour de leur publication. Au regard de la diversité des systèmes d'information, l'ANSSI ne peut garantir que ces informations puissent être reprises sans adaptation sur les systèmes d'information cibles. Dans tous les cas, la pertinence de l'implémentation des éléments proposés par l'ANSSI doit être soumise, au préalable, à la validation de l'administrateur du système et/ou des personnes en charge de la sécurité des systèmes d'information.

Personnes ayant contribué à la rédaction de ce document:

Contributeurs	Rédigé par	Approuvé par	Date
DTO, ST, DAT	BSS	SDE	8 octobre 2015

Évolutions du document :

Version	Date	Nature des modifications
1.0	8 octobre 2015	Version initiale

Pour toute remarque:

Contact	Adresse	@mél	Téléphone
Bureau Communication de l'ANSSI	51 bd de La Tour-Maubourg 75700 Paris Cedex 07 SP	communication@ssi.gouv.fr	01 71 75 84 04

Table des matières

1 Glossaire

AIDE	*Advanced Intrusion Detection Environment*, environnement de détection d'intrusion avancé
API	*Application Programming Interface*, interface de programmation
ASLR	*Address Space Layout Randomisation*, distribution aléatoire de l'espace d'adressage
AVC	*Access Vector Cache*, cache de vecteur d'accès
BIOS	*Basic Input/Output System*, système entrée/sortie basique
CERT	*Computer Emergency Response Team*, équipe d'intervention d'urgence informatique
DAC	*Discretionary Access Control*, contrôle d'accès discrétionnaire
GID	*Group IDentifier*, identifiant de groupe
HIDS	*Host Intrusion Detection System*, système de détection d'intrusion sur l'hôte
HSM	*Hardware Security Module*, module de sécurité matériel
IOMMU	*Input/Output Memory Management Unit*, unité de gestion de mémoire d'entrée/sortie
IPC	*Inter-Process Communication*, communication inter-processus
LDAP	*Lightweight Directory Access Protocol*, protocole léger d'accès léger à un répertoire
LSM	*Linux Security Module*, module de sécurité Linux
MAC	*Mandatory Access Control*, contrôle d'accès obligatoire
NSS	*Name Service Switch*, service de gestion de noms
NX	*No-eXecute*, pas d'exécution
OS	*Operating System*, système d'exploitation
PAM	*Pluggable Authentication Module*, module d'authentification enfichable
PIC	*Position Independent Code*, code indépendant de la position
PID	*Process IDentifier*, identifiant de processus
PIE	*Position Independent Executable*, exécutable indépendant de la position
RBAC	*Role Based Access Control*, contrôle d'accès basé sur rôles
RSS	*Really Simple Syndication*, réseau de syndication simple
UID	*User IDentifier*, identifiant utilisateur
XD	*eXecute Disable*, désactivation de l'exécution

2 Préambule

2.1 Avant-propos

Aujourd'hui les systèmes d'exploitation Unix et dérivés, et notamment GNU/Linux, jouent un rôle important dans l'écosystème des équipements, systèmes, réseaux et télécommunications.

Leur diversité et leur composition font qu'ils sont utilisés suivant un grand nombre de combinaisons possibles. Il ne serait pas efficace d'aborder dans le détail chacun des différents cas d'usage. Cependant des règles de configuration permettent d'obtenir un système raisonnablement sûr du moment que certains principes fondamentaux sont respectés, et de vérifier méthodologiquement qu'elles sont correctement appliquées à l'image d'une liste de vérification.

Le guide se concentre principalement sur des directives de configuration système génériques et des principes de bon sens qu'il convient de vérifier lors du déploiement des services hébergés. La configuration de ces services eux-mêmes peut faire l'objet d'une note technique dédiée, comme par exemple les *Recommandations pour la sécurisation des sites web* [1] ou les *Recommandations pour un usage sécurisé d'OpenSSH* [2] sur le site de l'ANSSI.

Il convient d'étudier l'applicabilité et la maintenabilité de chaque recommandation au cas d'usage considéré. Il est par ailleurs vivement conseillé d'avoir recours aux compétences d'un expert en système GNU/Linux pour la mise en œuvre de ces bonnes pratiques.

2.2 Niveau de durcissement

Les distributions GNU/Linux représentant un ensemble de systèmes d'une grande hétérogénéité, la maîtrise du socle système est une tâche complexe ; une expertise devient réellement nécessaire au fur et à mesure que le nombre de services et de serveurs augmente. Cependant certaines mesures de durcissement peuvent être mises en place en fonction du niveau de sécurité attendu, qui va dépendre de la sensibilité des données manipulées ou hébergées par le système et de la robustesse des contrôles d'accès réalisés en vue d'accéder aux ressources.

Un service exposé publiquement avec un contrôle d'accès faible et manipulant des données sensibles (serveur de transferts de courriers électroniques, serveur web d'entreprise, etc.) demande un niveau de sécurité renforcé, voire élevé. À l'inverse, un serveur de sauvegarde résidant sur un réseau isolé et uniquement accessible à quelques personnes pourra demander un niveau de sécurité moindre.

Les recommandations de ce guide sont données en fonction d'un niveau de durcissement estimatif. Ce niveau ne dépend pas forcément de la difficulté et du temps requis pour déployer une recommandation donnée, choses qui ne peuvent être appréciées qu'après un audit dans un contexte donné. Ces niveaux doivent être vus comme des fils conducteurs pour aider à la lecture et à l'administration système.

En fonction du niveau de durcissement retenu, les recommandations s'appliquent du niveau minimal jusqu'au niveau envisagé. Par exemple :

1. http://www.ssi.gouv.fr/securisation-sites-web/
2. http://www.ssi.gouv.fr/nt-ssh/

Niveau	Description
M I R E	Recommandation de niveau **minimal**. À mettre en œuvre systématiquement sur tout système.
I R E	Recommandation à appliquer dès le niveau **intermédiaire**. Correspond généralement à des services protégés par plusieurs couches de sécurité de niveau supérieur.
R E	Recommandation s'appliquant dès le niveau **renforcé**. Généralement pour des systèmes exposés à des flux non authentifiés ou de sources nombreuses.
E	Recommandation valide au niveau **élevé**. Correspond à des systèmes hébergeant des données sensibles accessibles depuis des réseaux non authentifiés ou peu contrôlés.

TABLE 1 – Grille de lecture des niveaux de durcissement

FIGURE 1 – Niveaux de sécurité système

3 Principes généraux de sécurité et de durcissement

Les principes décrits ci-après, essentiels, doivent guider l'installation et la configuration d'un système d'exploitation.

3.1 Principe de minimisation

Sous ce terme vont se trouver des concepts qui s'appliquent bien à l'ingénierie système. Les systèmes conçus et installés doivent éviter autant que possible la complexité en vue de :

- réduire la surface d'attaque au strict minimum ;
- permettre une mise à jour et un suivi du système efficace ;
- rendre l'activité de surveillance des systèmes plus accessible, le nombre de composants à surveiller étant réduit.

Ce principe peut se retrouver rapidement en contradiction avec d'autres, tout aussi importants. Seule une étude de cas avec l'aide d'une expertise système et sécurité permettra de faire des choix raisonnables. Bien que facile à énoncer, sa mise en œuvre est moins directe. Les chapitres suivants donneront des recommandations ciblées suivant les parties envisagées du système.

R1 - M I R E Minimisation des services installés

Seuls les composants strictement nécessaires au service rendu par le système doivent être installés.

Tout service (particulièrement en écoute active sur le réseau) est un élément sensible. Seuls ceux connus et requis pour le fonctionnement et la maintenance doivent être résidents. Ceux dont la présence ne peut être justifiée doivent être désactivés, désinstallés ou supprimés.

Un cas courant est la présence de services résidents d'auto-configuration, tels DHCP ou ZeroConf. Outre le fait qu'ils peuvent perturber le réseau sur lequel ils sont connectés, il est difficile pour ces services de s'assurer de la légitimité des éléments qu'ils reçoivent.

Dans la mesure du possible et sauf besoin opérationnel, ceux-ci ne doivent pas s'exécuter sur un serveur.

R2 - I R E Minimisation de la configuration

Les fonctionnalités configurées au niveau des services démarrés doivent être limitées au strict nécessaire.

Les services sont souvent installés avec des configurations par défaut qui activent des fonctionnalités potentiellement problématiques d'un point de vue sécurité. Citons, par exemple, les redirections de port SSH qui sont souvent acceptées puis utilisées pour contourner des règles de pare-feu, ou un serveur Apache avec l'indexation des répertoires activée et permettant de naviguer dans l'arborescence du système.

3.2 Principe de moindre privilège

Ce principe définit que tout objet ou entité gérée par un système ne dispose que des droits strictement nécessaires à son exécution, et rien de plus.

L'objectif est à la fois un gain en sécurité et sûreté :
- les conséquences de dysfonctionnements ou vulnérabilités sont limitées aux privilèges octroyés ;
- l'altération et/ou la compromission du système nécessitent une escalade de privilèges, moins triviale et discrète à réaliser dans les cas où plusieurs couches de protection sont mises en place.

R3 - R E Principe de moindre privilège

> Les services et exécutables disponibles sur le système doivent faire l'objet d'une analyse afin de connaître les privilèges qui leurs sont associés, et doivent ensuite être configurés et intégrés en vue d'en utiliser le strict nécessaire.

Les Unix et distributions GNU/Linux ont une séparation de privilèges qui repose principalement sur la notion d'utilisateur à deux niveaux : les utilisateurs « classiques » et les administrateurs (communément appelés `root`).

Si cette séparation manque de finesse, en particulier aujourd'hui où un utilisateur dispose d'un accès assez large aux primitives système (tout en étant de plus en plus exposé à des attaques : vol de comptes et d'identifiants, fuite d'informations, etc.), l'analyse approfondie d'un expert est souvent requise pour mettre en exergue les vulnérabilités résiduelles dans le but d'en minimiser autant que possible les conséquences.
D'autres mécanismes sont apparus en vue de permettre un contrôle additionnel plus fin des droits (SELinux, AppArmor, grsecurity), mais la séparation des privilèges basée sur l'utilisateur (et ses groupes) reste encore la plus usitée. Elle peut s'accompagner de mécanismes de cloisonnement (LXC, VServer, chroot, etc.) ou de filtrage (SECCOMP) en vue de limiter les accès au système d'exploitation sous-jacent.

R4 - E Utilisation des fonctionnalités de contrôle d'accès

> Il est recommandé d'utiliser les fonctionnalités de contrôle d'accès obligatoire (MAC) en plus du traditionnel modèle utilisateur Unix (DAC), voire éventuellement de les combiner avec des mécanismes de cloisonnement.

Les mécanismes de cloisonnement sont nombreux, et chacun offre des avantages et des inconvénients. Certains comme `chroot` sont universels et fonctionneront sur tous les systèmes Unix connus, mais se cantonnent à ne cloisonner qu'une partie du système (pour `chroot` : les accès au système de fichiers).

Des fonctions de cloisonnement et de contrôle des privilèges plus évoluées que `chroot` existent sous Linux ; citons à titre d'exemple les capacités (*POSIX capabilities*), les espaces de nommage (namespaces), les filtres *SECCOMP*, ou les solutions à base de *containers* (*LXC*, *VServer*).

3.3 Principe de défense en profondeur

Le principe de défense en profondeur repose sur la conception de plusieurs couches de sécurité indépendantes et complémentaires en vue de retarder un attaquant dont la mission est la compromission

du système.

Chaque couche de sécurité est donc un point de résistance que l'attaquant doit franchir. La mise en défaut d'une couche s'accompagne de signaux, d'alarmes ou de messages de journalisation permettant de détecter une activité suspecte et de pouvoir y réagir. L'étape de rémédiation s'en trouve aussi facilitée grâce aux informations supplémentaires agrégées sur le contexte de la compromission.

Ce principe a donc un réel avantage : détection, facilité de rémédiation, et amélioration de la sécurité.

R5 - M I R E Principe de défense en profondeur

Sous Unix et dérivés, la défense en profondeur doit reposer sur une combinaison de barrières qu'il faut garder indépendantes les unes des autres. Par exemple :
- authentification nécessaire avant d'effectuer des opérations, notamment quand elles sont privilégiées ;
- journalisation centralisée d'évènements au niveau systèmes et services ;
- priorité à l'usage de services qui implémentent des mécanismes de cloisonnement et/ou de séparation de privilèges ;
- utilisation de mécanismes de prévention d'exploitation.

R6 - R E Cloisonnement des services réseau

Les services réseau doivent autant que possible être hébergés sur des environnements distincts. Cela évite d'avoir d'autres services potentiellement affectés si l'un d'eux se retrouve compromis sous le même environnement.

3.4 Activité de veille et maintenance

Tout système doit être surveillé afin d'en contrôler les dérives. Il en va de même pour son maintien en conditions de sécurité.

R7 - R E Journalisation de l'activité des services

Les activités du système et des services en cours d'exécution doivent être journalisées et archivées sur un système externe, non local.

Les correctifs logiciels peuvent apporter de nouvelles fonctionnalités à l'environnement logiciel, contribuant ainsi à réhausser son niveau de sécurité. D'autres visent à rectifier des vulnérabilités présentes sur le système. Une activité de veille sur les mesures correctives à appliquer est essentielle, car elle permet de connaître les vulnérabilités auxquelles celui-ci est ou a été exposé, et donc d'y prêter une attention particulière le temps qu'un correctif soit disponible.

R8 - M I R E Mises à jour régulières

Il est recommandé d'avoir une procédure de mise à jour de sécurité régulière et réactive.

L'activité de veille peut se faire par l'inscription à des listes de diffusion (équipes sécurité des composants et applications installés et de leurs éditeurs, flux RSS de CERT [3], etc.).

4 Configuration matérielle avant installation

Certaines options matérielles doivent être, suivant les cas, activées ou désactivées afin de durcir le socle qui va servir à l'installation. Ce paramétrage doit être fait de préférence avant l'installation pour que le système soit capable d'en tenir compte le plus tôt possible.

Les recommandations suivantes s'appliquent aux architectures x86. L'approche reste cependant applicable à d'autres architectures à ceci près que les mécanismes et directives de configuration seront très certainement différents.

4.1 Règlage général du BIOS

Le BIOS (et son pendant moderne l'UEFI) est l'interface principale de configuration matérielle du système. Cette interface n'est souvent accessible que lors des premiers instants du démarrage de la machine au travers de la frappe d'une combinaison de touches.

La configuration matérielle de la machine dépend plus de l'usage qui en sera fait que du système d'exploitation installé. Il est cependant nécessaire de préciser que la désactivation (ou l'activation) de fonctionnalités au niveau du BIOS peut bloquer l'utilisation de celles-ci par le système. La note « *Recommandations de configuration matérielle de postes clients et serveurs x86* [4] »aborde les différentes options que l'on peut trouver sur une machine x86 contemporaine.

R9 - I R E Configuration matérielle

Il est conseillé d'appliquer les recommandations de configuration mentionnées dans la note technique « *Recommandations de configuration matérielle de postes clients et serveurs x86* [4] ».

4.2 Mode 32 ou 64 bits

L'architecture x86 a évolué au fil du temps. Aujourd'hui la quasi-totalité des processeurs x86 est capable de fonctionner à la fois en mode 32 bits et en mode 64 bits. D'autres modes (mode réel par exemple) existent encore pour des raisons de rétro-compatibilité. Les distributions GNU/Linux et Unix grand public offriront surtout deux types de modes :

- le mode protégé (`protected mode`), cantonné à un adressage virtuel sur 32 bits (4 octets) ;
- le mode long (`long mode`), qui permet un adressage plus large sur 64 bits [5](8 octets).

3. Site du CERT-FR : http://www.cert.ssi.gouv.fr/
4. http://www.ssi.gouv.fr/nt-x86/
5. Seuls 48 bits sont effectivement exploités, les 16 bits non utilisés faisant partie des adresses « non-canoniques ».

Le mode 64 bits présente quelques avantages par rapport au mode 32 bits, et ce malgré l'absence de certains mécanismes permettant l'implémentation de protections (segmentation notamment) :

- adressage relatif par registre %rip, plus efficace pour du code relogeable comme PIC et PIE ;
- espace d'adressage plus grand, donc plus de bits sont accessibles pour l'ASLR ;
- la taille de mémoire allouable a un processus est beaucoup plus importante ;
- présence systématique de bits de protections NX/XD.

R10 - I R E Architecture 32 et 64 bits

Lorsque la machine le supporte, préférer l'installation d'une distribution GNU/Linux en version 64 bits plutôt qu'en version 32 bits.

4.3 Service d'IOMMU (virtualisation entrée/sortie)

L'activation du service d'IOMMU permet de protéger la mémoire du système vis-à-vis d'accès arbitraires réalisés par des périphériques. L'efficacité de cette protection dépend grandement de la façon dont le système est construit. Même si sa conception est imparfaite[6], l'IOMMU contribue à réduire les risques d'accès arbitraire à la mémoire par les périphériques.

L'activation de la fonctionnalité va dépendre de la configuration matérielle ainsi que du réglage du système d'exploitation : suivant la situation (présence ou absence d'une IOMMU fonctionnelle, quantité de mémoire présente sur le système, etc.) Linux peut décider de ne pas initialiser l'IOMMU. Il faut donc lui indiquer d'en forcer l'usage au travers d'une directive de configuration passée en paramètre lors du démarrage.

R11 - E Directive de configuration de l'IOMMU

La directive `iommu=force` doit être rajoutée à la liste des paramètres du noyau choisi lors du démarrage en plus de celles déjà présentes dans les fichiers de configuration du bootloader (`/boot/grub/menu.lst` ou `/etc/default/grub`).

Le changement de configuration peut nécessiter l'exécution d'un utilitaire tiers pour être pris en compte dans la configuration (`update-grub` par exemple) ainsi que le redémarrage complet du système. Par ailleurs, l'activation de l'IOMMU sur le système peut engendrer des instabilités matérielles, il convient donc de s'assurer du bon fonctionnement de celui-ci avant de déployer une telle mesure en production.

```
# Exemple pour un fichier /etc/default/grub

...
GRUB_CMDLINE_LINUX=" iommu=force"
```

5 Installation du système

Ce chapitre traite des recommandations à suivre lors de la première installation du système. Chaque système d'exploitation et distribution GNU/Linux adopte un cheminement qui lui est propre pendant cette étape. Quelques éléments de configuration vont cependant s'appliquer quasi-universellement.

6. https://www.sstic.org/2010/presentation/Analyse_de_l_efficacite_du_service_fourni_par_une_IOMMU/

5.1 Partitionnement

Il est usuel de réserver des partitions dédiées aux services pouvant générer beaucoup de volumétrie afin d'éviter de saturer les partitions système. L'espace à réserver pour chaque partition dépend des cas d'usage : un serveur de fichiers aura besoin d'une volumétrie importante pour `/srv` ou `/var/ftp/`, tandis qu'un serveur de journaux sera plutôt concerné par la volumétrie utilisable pour `/var/log`.

Le partitionnement doit ainsi permettre de protéger et isoler les différents composants du système de fichiers. Il est par défaut souvent insatisfaisant : il ne tient pas suffisamment compte des options `nosuid` (ignore les bits setuid/setgid), `nodev` (ignore les fichiers spéciaux caractère ou bloc), et `noexec` (ignore les droits d'exécution).

Il faut noter que suivant les systèmes et distributions, certaines des options de montage ne seront pas applicables transitoirement ; par exemple des utilitaires, installeurs ou produits estimeront que les fichiers écrits dans `/tmp` ou `/var` peuvent être exécutables. Dans ces cas exceptionnels il est nécessaire d'adapter le partitionnement. Un de ceux les plus fréquemment rencontrés est celui des distributions dérivées de Debian dont le `/var/lib/dpkg` nécessite des droits d'exécution. Une alternative est d'implémenter une procédure de maintenance durant laquelle les mises à jour sont installées, à l'image de ce que l'on trouve sur d'autres systèmes d'exploitation.

R12 - (I) (R) (E) Partitionnement type

Le partitionnement type recommandé est le suivant :

Point de montage	Options	Description
/	<sans option>	Partition racine, contient le reste de l'arborescence
/boot	nosuid,nodev,noexec (noauto optionnel)	Contient le noyau et le chargeur de démarrage. Pas d'accès nécessaire une fois le boot terminé (sauf mise à jour)
/opt	nosuid,nodev (ro optionnel)	Packages additionnels au système. Montage en lecture seule si non utilisé
/tmp	nosuid,nodev,noexec	Fichiers temporaires. Ne doit contenir que des éléments non exécutables. Nettoyé après redémarrage
/srv	nosuid,nodev (noexec,ro optionnels)	Contient des fichiers servis par un service type web, ftp, etc.
/home	nosuid,nodev,noexec	Contient les *HOME* utilisateurs. Montage en lecture seule si non utilisé
/proc	hidepid=1	Contient des informations sur les processus et le système
/usr	nodev	Contient la majorité des utilitaires et fichiers système
/var	nosuid,nodev,noexec	Partition contenant des fichiers variables pendant la vie du système (mails, fichiers PID, bases de données d'un service)
/var/log	nosuid,nodev,noexec	Contient les logs du système
/var/tmp	nosuid,nodev,noexec	Fichiers temporaires conservés après extinction

La partition `/boot` contient le noyau de démarrage ainsi que le(s) fichier(s) `System.map` contenant la table des symboles utilisée par celui-ci. Ce fichier est souvent parcouru par différents programmes malveillants afin de construire plus facilement des « exploits » de code noyau. Le partitionnement idéal demande à ce que cette partition ne soit pas montée automatiquement au démarrage (option `noauto`), et que le montage de celle-ci soit un évènement critique d'un point de vue système (pour une mise à jour ou un correctif noyau par exemple). Mais cette mesure demande d'adapter les outils système à cette configuration particulière, et donc demande une expertise système pointue lors de son déploiement.

R13 - (R) (E) Restrictions d'accès sur les fichiers System.map

Quand la partition `/boot` ne peut être démontée (ou qu'elle n'existe pas), le(s) fichier(s) `System.map` doivent être restreints en lecture à `root` uniquement.

Il faut noter que certains services peuvent toujours avoir besoin d'accéder à une arborescence donnée après une opération de `chroot`, sans que cette arborescence ne soit rattachée et visible directement depuis la cage `chroot`. Dans de tels cas, l'usage de points de montage *bind* est à envisager.

5.2 Choix des paquets à installer

L'installation des paquets est l'étape cruciale qui va déterminer l'ensemble des fichiers qui seront présents sur le système, les services qu'il va rendre ainsi que les paquets qui devront être maintenus dans le temps.

Il est plus facile d'obtenir une installation minimaliste en retirant tous les paquets présélectionnés, et de ne choisir que ceux nécessaires au contexte d'utilisation. Par exemple, l'exploitation d'un serveur ne requiert pas systématiquement l'installation d'une interface graphique locale (serveur X).

R14 - I R E Installation de paquets réduite au strict nécessaire

Le choix des paquets doit conduire à une installation aussi petite que possible, se bornant à ne sélectionner que ce qui est nécessaire au besoin.

Certaines distributions fournissent des « rôles » préconfigurés. Il est déconseillé de baser son installation sur lesdits rôles étant donné que les choix des mainteneurs de la distribution ne correspondent pas forcément aux besoins propres, ce qui nécessitera l'installation de paquets supplémentaires.

R15 - M I R E Choix des dépôts de paquets

Seuls les dépôts officiels à jour de la distribution doivent être utilisés.

R16 - R E Dépôts de paquets durcis

Lorsque la distribution fournit plusieurs types de dépôts, la préférence doit aller à ceux contenant des paquets faisant l'objet de mesures de durcissement supplémentaires.

Entre deux paquets fournissant le même service, ceux faisant l'objet de mesures de durcissement (à la compilation, à l'installation ou dans la configuration par défaut) doivent être privilégiés.

5.3 Configuration du chargeur de démarrage

Pour des raisons équivalentes à la configuration du BIOS, le chargeur de démarrage (aussi appelé *bootloader*) est un élément important de la chaîne de démarrage. Ceux d'aujourd'hui (GRUB, GRUB 2, etc.) sont fonctionnellement riches : ils permettent d'accéder aux systèmes de fichiers (et éventuellement de modifier les données), de booter sur des périphériques USB ou de changer les options de démarrage du noyau sélectionné.

R17 - R E Mot de passe du chargeur de démarrage

Un chargeur de démarrage permettant de protéger le démarrage par mot de passe doit être privilégié. Ce mot de passe doit empêcher un utilisateur quelconque de modifier ses options de configuration.

Quand le chargeur de démarrage n'offre pas la possibilité de lui adjoindre un mot de passe, une mesure technique (et le cas échéant organisationnelle) doit être mise en place afin de bloquer tout utilisateur dans ses tentatives de modification du paramétrage.

GRUB et GRUB 2 (chargeurs de démarrage pour architecture x86) offrent tous les deux la possibilité de leur rajouter un mot de passe de déverrouillage. Consultez leurs documentations respectives [7] pour de plus amples informations.

5.4 Mot de passe root et comptes administrateur

Le mot de passe `root` doit être choisi avec le plus grand soin et conformément aux recommandations actuelles. Sa connaissance doit être limitée aux seules personnes ayant le besoin d'en connaître.

R18 - M I R E Robustesse du mot de passe administrateur

Le mot de passe root doit être suffisamment robuste compte tenu des recommandations présentes dans la note « *Recommandations de sécurité relatives aux mots de passe* » disponible sur le site de l'ANSSI [8].
Ce mot de passe doit être unique et propre à chaque machine.

Il est souvent d'usage de définir différents niveaux de privilèges sur le système en fonction des prérogatives des administrateurs. Certains peuvent avoir des responsabilités qui concernent uniquement le site web, d'autres à l'infrastructure de journalisation, ou encore aux bases de données.

Le compte `root` est impropre à ces usages. Il donne plein pouvoir à la personne qui y a accès. L'usage d'un tel compte générique ne facilite pas l'imputabilité lors d'un incident, et ne favorise pas un modèle de séparation fine des privilèges (par exemple entre différentes équipes d'administration).

R19 - I R E Imputabilité des opérations d'administration

Chaque administrateur doit posséder un compte dédié (local ou distant), et ne pas utiliser le compte `root` comme compte d'accès pour l'administration du système.

Les opérations de changement de privilèges devront reposer sur des exécutables permettant de surveiller les activités réalisées (par exemple `sudo`).

7. http://www.gnu.org/s/grub/
8. http://www.ssi.gouv.fr/mots-de-passe/

5.5 Installation d'éléments supplémentaires : clés, certificats

L'étape d'installation est l'occasion pour mettre en place des éléments préalablement générés tels que des clés d'authentification ou des certificats.

R20 - R E Installation d'éléments secrets ou de confiance

Tous les éléments secrets ou ceux concourant aux mécanismes d'authentification doivent être mis en place dès l'installation du système : mots de passe de comptes et d'administration, certificats d'autorité racines, clés publiques, ou encore certificats de l'hôte (et leur clé privée respective).

6 Configuration et services système

La configuration d'un système d'exploitation s'accompagne du déploiement de services. Les recommandations suivantes visent à établir quelques bonnes pratiques d'administration et de configuration. Elles agissent à deux niveaux :

- identification de biens essentiels, et mesures de durcissement adaptées ;
- mise en place de solutions de cloisonnement et d'isolation pour retarder autant que possible la compromission large échelle du système.

6.1 Services exposés à des flux non maîtrisés

Une attention toute particulière doit être portée sur les services exposés à des flux non maîtrisés, c'est-à-dire tout service communiquant avec des sources qui ne sont pas de confiance (Internet, bornes Wifi publiques, etc.) ou non authentifiées. Ce sont ceux qui sont préférentiellement attaqués et compromis car leur accès est libre.

La terminologie « service » est à prendre au sens large ici. Toute faille ou vulnérabilité exploitable avant une étape d'authentification est particulièrement concernée par cette recommandation. Par exemple, un service Web reposant sur une authentification HTTP basique peut être exploitable aux niveaux :

1. matériel (firmware, drivers cartes réseaux, etc.) ;
2. réseau (Ethernet, IP, TCP) ;
3. applicatif (couche SSL/TLS, en-têtes HTTP émis et reçus pré-authentification).

Ce service doit donc être durci et surveillé, et ce malgré l'apparente opération d'authentification effectuée par le serveur.

R21 - I R E Durcissement et surveillance des services soumis à des flux arbitraires

Les services exposés à des flux non maîtrisés doivent être surveillés et particulièrement durcis.
La surveillance consiste à caractériser le comportement du service, et à reporter tout écart par rapport à son fonctionnement nominal (celui-ci étant déduit des spécifications initiales attendues).

Le durcissement relève de l'ensemble des mesures techniques qui visent à retarder voire empêcher la compromission dudit service. Cette démarche s'applique dès la phase de conception (étude de séparation

de privilèges, spécifications non ambiguës, etc.) jusqu'à la réalisation (validation des entrées/sorties, configuration sécurisée, etc.) et la maintenance.

6.2 Configuration système - `sysctl`

Les *sysctl* sont un ensemble de variables qui permettent d'adapter le paramétrage du système d'exploitation et plus particulièrement son noyau.

Cette interface s'enrichit constamment avec le temps, au gré des évolutions et des améliorations qui sont apportées au système. Leurs portée, usage et paramétrage doivent donc faire l'objet d'une veille régulière si un administrateur souhaite profiter au mieux de leurs fonctionnalités.

Les *sysctl* peuvent agir à tout niveau :
- mémoire : configuration de la pagination, des allocateurs, des propriétés des mappings, etc.
- réseau : IP, TCP, UDP, tailles et caractéristiques des tampons, fonctions évoluées (syn cookies, choix d'algorithmes, etc.) ;
- noyau : contrôle des caches, swap, scheduling, etc.
- systèmes de fichiers : setuid dumps, hard et soft links, quotas ;
- processus : ressources allouées, restrictions d'exécution, cloisonnement, etc.

Leur nombre est grand et leurs fonctionnalités diverses. Il est nécessaire de se rapporter à leur documentation (généralement sous `doc/Documentation/` [9] des sources du noyau Linux) pour obtenir une explication détaillée quant à leur rôle. Certaines présentent des fonctions de sécurité intéressantes.

9. https://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/tree/Documentation

R22 - I R E Paramétrage des sysctl réseau

Ces sysctl sont données pour un hôte « serveur » typique qui n'effectue pas de routage et ayant une configuration IPv6 minimaliste (adressage statique). Quand IPv6 n'est pas utilisé il convient de le désactiver en mettant l'option `net.ipv6.conf.all.disable_ipv6` à **1**. Elles sont présentées ici telles que rencontrées dans le fichier `/etc/sysctl.conf` :

```
# Pas de routage entre les interfaces
net.ipv4.ip_forward = 0
# Filtrage par chemin inverse
net.ipv4.conf.all.rp_filter = 1
net.ipv4.conf.default.rp_filter = 1
# Ne pas envoyer de redirections ICMP
net.ipv4.conf.all.send_redirects = 0
net.ipv4.conf.default.send_redirects = 0
# Refuser les paquets de source routing
net.ipv4.conf.all.accept_source_route = 0
net.ipv4.conf.default.accept_source_route = 0
# Ne pas accepter les ICMP de type redirect
net.ipv4.conf.all.accept_redirects = 0
net.ipv4.conf.all.secure_redirects = 0
net.ipv4.conf.default.accept_redirects = 0
net.ipv4.conf.default.secure_redirects = 0
# Loguer les paquets ayant des IPs anormales
net.ipv4.conf.all.log_martians = 1
# RFC 1337
net.ipv4.tcp_rfc1337 = 1
# Ignorer les réponses non conformes à la RFC 1122
net.ipv4.icmp_ignore_bogus_error_responses = 1
# Augmenter la plage pour les ports éphémères
net.ipv4.ip_local_port_range = 32768 65535
# Utiliser les SYN cookies
net.ipv4.tcp_syncookies = 1
# Désactiver le support des "router solicitations"
net.ipv6.conf.all.router_solicitations = 0
net.ipv6.conf.default.router_solicitations = 0
# Ne pas accepter les "router preferences" par "router advertisements"
net.ipv6.conf.all.accept_ra_rtr_pref = 0
net.ipv6.conf.default.accept_ra_rtr_pref = 0
# Pas de configuration auto des prefix par "router advertisements"
net.ipv6.conf.all.accept_ra_pinfo = 0
net.ipv6.conf.default.accept_ra_pinfo = 0
# Pas d'apprentissage du routeur par défaut par "router advertisements"
net.ipv6.conf.all.accept_ra_defrtr = 0
net.ipv6.conf.default.accept_ra_defrtr = 0
# Pas de configuration auto des adresses à partir des "router
    advertisements"
net.ipv6.conf.all.autoconf = 0
net.ipv6.conf.default.autoconf = 0
# Ne pas accepter les ICMP de type redirect
net.ipv6.conf.all.accept_redirects = 0
net.ipv6.conf.default.accept_redirects = 0
# Refuser les packets de source routing
net.ipv6.conf.all.accept_source_route = 0
net.ipv6.conf.default.accept_source_route = 0
# Nombre maximal d'adresses autoconfigurées par interface
net.ipv6.conf.all.max_addresses = 1
net.ipv6.conf.default.max_addresses = 1
```

R23 - I R E Paramétrage des sysctl système

Voici une liste de sysctl système recommandées (au format `/etc/sysctl.conf`) :

```
# Désactivation des SysReq
kernel.sysrq = 0
# Pas de core dump des exécutables setuid
fs.suid_dumpable = 0
# Interdiction de déréférencer des liens vers des fichiers dont
# l'utilisateur courant n'est pas le propriétaire
# Peut empêcher certains programmes de fonctionner correctement
fs.protected_symlinks = 1
fs.protected_hardlinks = 1
# Activation de l'ASLR
kernel.randomize_va_space = 2
# Interdiction de mapper de la mémoire dans les adresses basses (0)
vm.mmap_min_addr = 65536
# Espace de choix plus grand pour les valeurs de PID
kernel.pid_max = 65536
# Obfuscation des adresses mémoire kernel
kernel.kptr_restrict = 1
# Restriction d'accès au buffer dmesg
kernel.dmesg_restrict = 2
# Restreint l'utilisation du sous système perf
kernel.perf_event_paranoid = 2
kernel.perf_event_max_sample_rate = 1
kernel.perf_cpu_time_max_percent = 1
```

Il est possible d'interdire le chargement de nouveaux modules (y compris par `root`) au travers d'une sysctl. Cette mesure, efficace pour empêcher la modification du noyau par des modules tiers potentiellement malveillants, peut avoir des conséquences non triviales sur le fonctionnement du reste du système. Il faut donc s'assurer que l'activation de cette option n'entraîne pas d'impacts fonctionnels significatifs.

R24 - R E Désactivation du chargement des modules noyau

Le chargement des modules noyau peut être bloqué par l'activation de la sysctl `kernel.modules_disabled`, soit directement par la commande suivante (non sauvegardée après redémarrage) :

```
sysctl -w kernel.modules_disabled=1
```

ou par la modification du fichier `/etc/sysctl.conf` :

```
# Interdiction de chargement des modules (sauf ceux déjà chargés à
# ce point)
kernel.modules_disabled = 1
```

Un module de sécurité, Yama, permet de rajouter une sysctl qui contrôle les droits d'accès à l'appel système `ptrace`. Celui-ci est particulièrement sensible car il permet de tracer le fonctionnement d'un processus. Par défaut, tout utilisateur est en droit de déboguer tous ceux lui appartenant, ce qui inclut les processus stockant des éléments sensibles dans leur mémoire comme des clés ou des mots de passe (navigateur Internet, `ssh-agent`, etc.). La compromission d'un processus de l'utilisateur peut permettre par rebond de récupérer ces données.

R25 - R E Configuration sysctl du module Yama

Il est recommandé de charger le module de sécurité Yama lors du démarrage (par exemple en passant l'argument `security=yama` au noyau) et de configurer la sysctl `kernel.yama.ptrace_scope` à une valeur au moins égale à 1.

6.3 Gestion de comptes d'accès

6.3.1 Désactivation des comptes utilisateurs inutilisés

R26 - R E Désactivation des comptes utilisateurs inutilisés

Les comptes utilisateurs inutilisés doivent être désactivés au niveau du système.

Cette désactivation passe par l'invalidation du compte au niveau de son mot de passe (suppression du champ `pw_passwd` dans le *shadow* et shell de login à `/bin/false`).

```
# Verrouillage d'un compte
usermod -L <compte>
# Désactivation de son shell de login
usermod -s /bin/false <compte>
```

La grande majorité des systèmes Unix/Linux isolent les applications et services les uns des autres en les dédiant chacun à un compte utilisateur propre. Par exemple un serveur web, une fois démarré, utilise les privilèges d'un utilisateur « web » (appelé *www* ou *www-data*), tandis que le serveur de noms s'exécute sous un compte distinct (par exemple *named*).

Ces comptes de service sont assimilables à des comptes utilisateurs inutilisés.

R27 - I R E Désactivation des comptes de services

Les comptes de service doivent être désactivés.

La désactivation de ce compte n'a pas de conséquence pratique sur l'utilisation des identifiants (UID) associés. Cette mesure permet d'éviter l'ouverture d'une session utilisateur par un compte de service. Il est important de noter que certains services peuvent se déclarer avec le compte `nobody`. Quand ceux-ci sont plusieurs à adopter un tel comportement, ils se retrouvent à partager les mêmes identifiants (et privilèges) au niveau du système d'exploitation. Un serveur web et un annuaire utilisant le compte `nobody` peuvent donc se contrôler mutuellement et altérer leur exécution l'un l'autre : modification de configuration, envoi de signaux, privilèges `ptrace`, etc.

R28 - R E Unicité et exclusivité des comptes de services système

Chaque service doit posséder son propre compte système et lui être dédié exclusivement.

6.3.2 Délai d'expiration de sessions utilisateurs

R29 - R E Délai d'expiration de sessions utilisateurs

Les sessions utilisateurs distantes (accès shell, clients graphiques) doivent être fermées au bout d'un certain délai d'inactivité.

La plupart des administrateurs et utilisateurs ouvrent souvent des connexions distantes vers les systèmes qu'ils maintiennent et exploitent, mais peuvent oublier de fermer ces sessions. Ceci expose inutilement l'hôte à des risques de compromissions (réutilisation d'un accès déjà ouvert à partir d'un compte compromis, vol de session, etc.), en plus de consommer inutilement des ressources machine.

Cette fonctionnalité est souvent présente sous la forme d'expiration de session après inactivité (*idle timeout* en anglais). Certains shells (bash, ksh, zsh) offrent une variable d'environnement `TMOUT` qui indique en secondes le délai maximum d'inactivité pour une session shell donnée.

6.4 PAM et NSS

PAM est un ensemble de modules qui permettent de configurer « dynamiquement » les différents mécanismes d'authentification et de gestion de comptes sur un système GNU/Linux. La méthode traditionnelle Unix (*shadow*) se configure quant à elle au travers du fichier `/etc/login.defs`.

NSS est la couche système qui se charge de manipuler et d'interroger les bases de données administratives. Il en existe plusieurs (`passwd`, `group`, `hosts`, `services`, etc.). Seules les bases de gestion des comptes utilisateurs vont être étudiées ci-dessous (il s'agit de `passwd` et `group`).

6.4.1 PAM

La documentation de PAM est riche, tout comme l'ensemble des fonctionnalités offertes par ses modules. L'objet de cette note n'est pas d'en expliquer le fonctionnement. On suppose l'administrateur familier avec celui-ci.

PAM va essentiellement fournir le service de gestion de comptes, c'est-à-dire permettre l'authentification de l'utilisateur, la création de sa session et éventuellement toute opération qui doit se dérouler lors de la tentative d'accès : création d'environnement, récupération de tickets ou de données, droits d'accès, changement de mot de passe, etc.

Lorsqu'un service fait appel à PAM afin d'authentifier un utilisateur, l'opération est directement réalisée par un module PAM.

Dans les grandes lignes, l'application soumet aux modules PAM les éléments d'authentification. Suivant la configuration de ceux-ci (située dans différents fichiers présents sous `/etc/` et `/etc/pam.d/`), PAM retourne le résultat (échec ou succès) à l'application, qui donnera ensuite l'accès (ou non) au système.

Deux éléments importants doivent être notés :
- les modules PAM sont appelés par l'application. L'application manipule, au moins partiellement, des éléments qui contribuent à authentifier l'utilisateur (ce qui inclut des données potentiellement

sensibles comme des mots de passe) ;
– suivant les modules PAM utilisés, PAM ira vérifier les éléments au travers de bases de données locales (comme `shadow`) ou distantes (requêtes LDAP, Kerberos, client SQL, etc.).

R30 - M I R E Applications utilisant PAM

Le nombre d'applications utilisant PAM doit être réduit au strict nécessaire afin de limiter l'exposition d'éléments d'authentification sensibles.

R31 - I R E Sécurisation des services réseau d'authentification PAM

Quand l'authentification se déroule au travers d'un service distant (réseau), le protocole d'authentification utilisé par PAM doit être sécurisé (chiffrement du flux, authentification du serveur distant, mécanismes d'anti-rejeu, etc.).

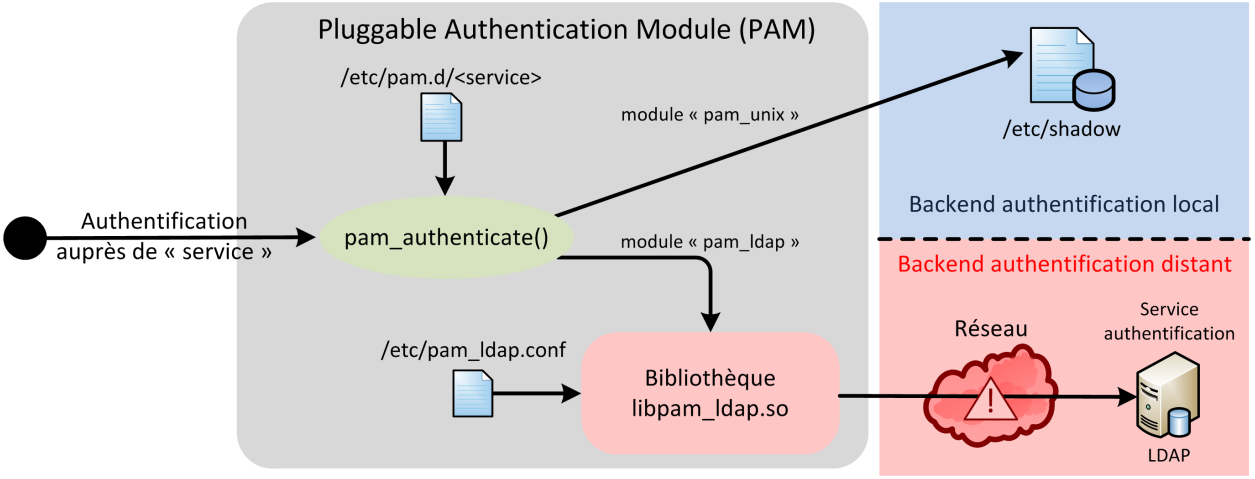

FIGURE 2 – Schéma d'une authentification PAM Unix + LDAP

Des protocoles comme Kerberos offrent de telles fonctions (c'est le cas de `pam_krb5` si le keytab *host* est renseigné pour la machine). D'autres modules, tels `pam_ldap` ou `pam_mysql`, ne chercheront pas à établir de lien sécurisé entre le client PAM et le service s'ils ne sont pas configurés explicitement en ce sens.

En plus de l'authentification, PAM va permettre de charger d'autres modules dont les fonctionnalités peuvent améliorer à la sécurité de l'architecture :

pam_time permet de restreindre les accès à une plage horaire ;

pam_cracklib permet de tester la difficulté des mots de passe ;

pam_passwdqc permet d'appliquer des contraintes suivant une politique de complexité de mots de passe (alternative à `pam_cracklib`) ;

pam_tally permet de bloquer temporairement un compte après un certain nombre d'échecs ;

pam_wheel permet de restreindre l'accès au compte `root` aux utilisateurs membre d'un groupe particulier (`wheel` par défaut).

Voici quelques exemples de fichiers de configuration PAM pour les modules décrits ci-dessus. Pour rappel ces fichiers sont situés sous `/etc/pam.d/` et portent le nom du service auquel ils sont associés. Seules les directives de configuration les concernant sont présentées.

Exemple avec `/etc/pam.d/su` et `/etc/pam.d/sudo` :

```
# Bloque l'accès à root aux membres du groupe 'wheel'
auth            required        pam_wheel.so
```

Exemple avec `/etc/pam.d/passwd` :

```
# Au moins 12 caractères, pas de répétition ni de séquence monotone,
# 3 classes différentes (parmi majuscules, minuscules, chiffres, autres)
password        required        pam_cracklib.so minlen=12 minclass=3  \
                                        dcredit=0 ucredit=0 lcredit=0 \
                                        ocredit=0 maxrepeat=1         \
                                        maxsequence=1 gecoscheck      \
                                        reject_username enforce_for_root
```

Exemple avec `/etc/pam.d/login` et `/etc/pam.d/sshd` :

```
# Blocage du compte pendant 5 min après 3 échecs
auth            required        pam_tally.so deny=3 lock_time=300
```

Le stockage des mots de passe en clair est proscrit car la compromission du système permet à un attaquant de les réutiliser sans effort vers d'autres services. La protection de ceux-ci ne doit pas reposer uniquement sur les droits d'accès à une base.

R32 - M I R E Protection des mots de passe stockés

Tout mot de passe doit être protégé par des mécanismes cryptographiques évitant de les exposer en clair à un attaquant récupérant leur base :
- hachage du mot de passe par une fonction de hachage considérée comme sûre (*SHA-256*, *SHA-512*), avec un sel et un nombre de tours assez grand (65536) ;
- fonction de dérivation de clé sur le mot de passe et combinée à une fonction de hachage considérée comme sûre (selon le *Référentiel Général de Sécurité*[10]) afin d'obtenir une empreinte valide pour un aléa donné ;
- chiffrement par une clé secrète (éventuellement protégée au travers d'un HSM auquel l'application accède).

PAM et `/etc/login.defs` peuvent être configurés afin d'utiliser *SHA-512* en suivant les recommandations ci-dessus :

Dans le fichier /etc/pam.d/common-password :

```
password    required    pam_unix.so obscure sha512 rounds=65536
```

Dans le fichier /etc/login.defs :

```
ENCRYPT_METHOD       SHA512
SHA_CRYPT_MIN_ROUNDS 65536
```

10. http://www.ssi.gouv.fr/rgs/

6.4.2 NSS

Un autre élément tout aussi critique pour la gestion des comptes utilisateurs est le sous-système NSS, qui gère l'ensemble des bases de données administratives.

Quand les comptes utilisateurs sont stockés dans un annuaire externe (fréquemment LDAP), NSS va se charger d'effectuer les requêtes auprès de l'annuaire dans le but de rendre les comptes visibles auprès du système.

Ces requêtes sont anonymes auprès de l'annuaire dans la configuration courante, avec un canal de communication non protégé. Il est donc aisé pour un attaquant de récupérer une liste de comptes valides auprès de l'annuaire voire même d'usurper le serveur annuaire auprès de NSS.

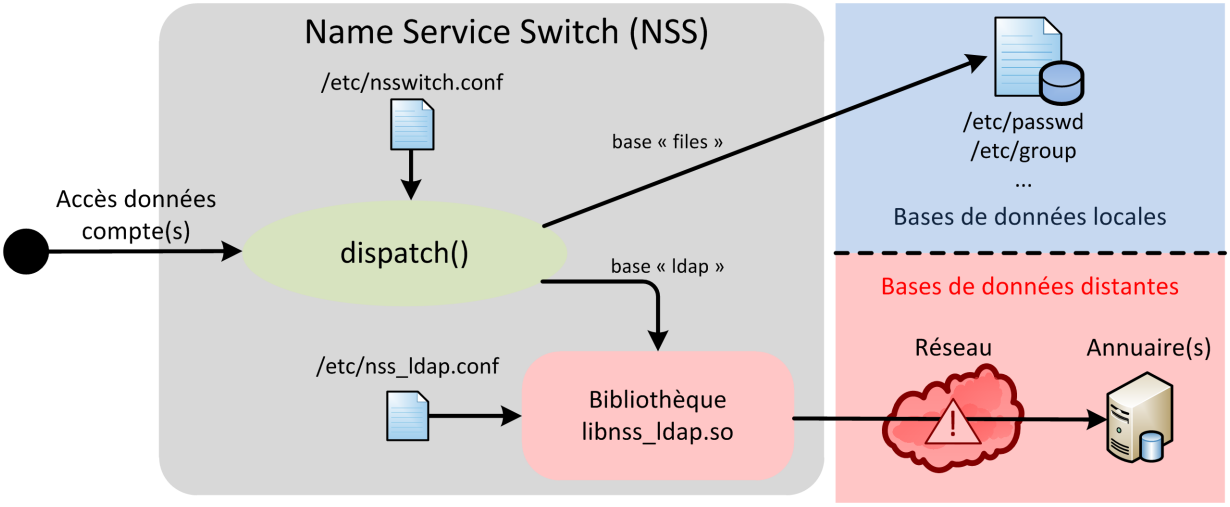

FIGURE 3 – Schéma des bases administratives de NSS

R33 - I R E Sécurisation des accès aux bases utilisateurs distantes

Lorsque les bases utilisateurs sont stockées sur un service réseau distant (type LDAP), NSS doit être configuré pour établir une liaison sécurisée permettant au minimum d'authentifier le serveur et de protéger le canal de communication.

R34 - I R E Séparation des comptes système et d'administrateur de l'annuaire

Il est recommandé de ne pas avoir de recouvrement de compte entre ceux utilisés par le système d'exploitation et ceux utilisés pour administrer l'annuaire.

L'usage de comptes administrateur d'annuaire pour effectuer des requêtes d'énumération de comptes par NSS doit être prohibé.

6.5 Vérification systèmes de fichiers et droits

Sous Unix, le modèle qui prévaut pour l'accès aux informations par un utilisateur est celui du DAC, ou contrôle d'accès discrétionnaire. C'est le propriétaire de la ressource (fichiers, répertoires, etc.) qui

en spécifie les droits d'accès.

6.5.1 `umask`

Quand l'utilisateur n'est pas explicite dans sa demande de droits, le système y applique un masque (le *umask*), qui est par défaut très permissif (généralement `0022`, soit : tout fichier créé est lisible par tous).

R35 - R E Valeur de umask

Le umask système doit être positionné à `0027` (par défaut, tout fichier créé n'est lisible que par l'utilisateur et son groupe, et modifiable uniquement par son propriétaire).

Le umask pour les utilisateurs doit être positionné à `0077` (tout fichier créé par un utilisateur n'est lisible et modifiable que par lui).

Jusqu'à récemment chaque distribution GNU/Linux adoptait sa propre façon de faire :

Distribution	Commentaire
Debian (et dérivés)	le umask système peut être directement modifié dans */etc/init.d/rc* et celui des utilisateurs via */etc/login.defs*.
CentOS (et dérivés)	le umask système est modifiable via */etc/sysconfig/init* et celui des utilisateurs via */etc/login.defs*.

Avec les migrations de nombreuses distributions sous *systemd*, le umask des services système doit être indiqué directement dans les fichiers de configuration du service, la valeur par défaut retenue étant `0022` (directive `UMask`). Le umask des utilisateurs devra quant à lui être spécifié dans */etc/profile*.

Une attention particulière doit être portée sur les fichiers et répertoires des types suivants :
- ceux contenant des éléments secrets, comme des mots de passe, des empreintes de mot de passe, des clés secrètes ou privées ;
- les exécutables possédant des bits particuliers, comme *setuid* ou *setgid* ;
- les répertoires de stockage temporaires auxquels tout le monde à accès ;
- les fichiers IPC nommés, comme les sockets ou les pipes, qui permettent à différents processus d'établir des canaux de communication entre eux.

Des droits adaptés vont devoir s'appliquer rationnellement à chacun de ces types.

6.5.2 Fichiers à contenu sensible

Les fichiers contenant clés privées, clés secrètes, mots de passe, empreintes, etc. (voire les journaux) sont considérés comme fichiers à contenu sensible.

Même lorsque ces fichiers ont leur contenu protégé par des mesures supplémentaires (chiffrement ou empreintes pour les mots de passe), il n'en demeure pas moins nécessaire d'en restreindre l'accès par principe de défense en profondeur.

R36 - I R E Droits d'accès aux fichiers de contenu sensible

Les fichiers à contenu sensible ne doivent être lisibles que par les utilisateurs ayant le strict besoin d'en connaître.

Quand ces fichiers contiennent des mots de passe (ou des empreintes de mots de passe) ils ne doivent être lisibles que par `root`. En revanche, les fichiers publics qui contiennent la liste des utilisateurs sont lisibles par tout le monde, mais sont éditables uniquement par `root`.

Quelques exemples de fichiers contenant des éléments sensibles :

```
-rw-r----- root root  /etc/gshadow
-rw-r----- root root  /etc/shadow
-rw------- foo  users /home/foo/.ssh/id_rsa
...
```

Le schéma d'analyse est le suivant :

1. les fichiers systèmes sensibles doivent avoir comme propriétaire le compte *root* afin d'éviter qu'un changement de droits puisse être effectué par un utilisateur non privilégié ;
2. quand ce fichier doit être accessible à un utilisateur non `root` (exemple : base de mot de passe de serveur web), l'utilisateur associé au serveur doit être membre d'un groupe dédié (exemple : *www-group*) qui aura un droit d'accès en lecture seule à ce fichier ;
3. le reste des utilisateurs ne doit posséder aucun droit.

6.5.3 Les fichiers exécutables *setuid* et/ou *setgid*

Ces fichiers sont particulièrement sensibles car ils s'exécutent avec les privilèges de leur utilisateur (ou groupe) propriétaire et non celui de l'utilisateur courant.

R37 - M I R E Exécutables avec bits setuid et setgid

Seuls les programmes spécifiquement conçus pour être utilisés avec les bits *setuid* (ou *setgid*) peuvent avoir ces bits de privilèges positionnés.

La présence d'un bit *setuid* (ou *setgid*) sur un exécutable nécessite que celui-ci adopte un certain nombre de précautions pour se prémunir de vulnérabilités liées au changement d'utilisateur. À titre d'exemple, de manière non exhaustive : nettoyer son environnement et réinitialiser un certain nombre d'éléments hérités du contexte antérieur (masques de signalisation, descripteurs de fichiers ouverts, etc.). La plupart des exécutables ne mettent pas en œuvre de telles précautions ; leur ajouter un bit *setuid* ou *setgid* introduirait par conséquent des possibilités d'escalade de privilèges.

Le cas le plus courant correspond aux exécutables *setuid root*, qui se lanceront avec les privilèges de root et non ceux de l'utilisateur appelant. Cela permet à un utilisateur d'effectuer des opérations pour lesquelles il n'a a priori pas de droit.

En présence de vulnérabilités, ces programmes sont exploités en vue de fournir un shell `root` à un utilisateur malveillant, ou tout du moins de détourner l'usage légitime du programme. Les exécutables

setuid (et accessoirement *setgid*) sont à étudier au cas par cas.

R38 - R E Exécutables setuid root

Les exécutables *setuid* doivent être le moins nombreux possible. Lorsqu'il est attendu que seuls les administrateurs de la machine les exécutent, il faut leur retirer le bit *setuid* et leur préférer des commandes comme `su` ou `sudo`, qui peuvent être surveillées.

Cette vérification doit avoir lieu après chaque mise à jour du système car leurs droits peuvent avoir été restaurés ; des programmes supplémentaires peuvent aussi avoir été installés lors de cette étape. Voici une liste non exhaustive de fichiers *setuid root* pouvant être rencontrés. Tout exécutable non mentionné dans cette liste devrait être examiné avec une attention particulière.

Exécutable	**Commentaire**
/bin/mount	À désactiver, sauf si absolument nécessaire pour les utilisateurs.
/bin/netreport	À désactiver.
/bin/ping6	(IPv6) Idem ping.
/bin/ping	(IPv4) Retirer droit setuid, sauf si un programme le requiert pour du monitoring.
/bin/su	Changement d'utilisateur. Ne pas désactiver.
/bin/umount	À désactiver, sauf si absolument nécessaire pour les utilisateurs.
/sbin/mount.nfs4	À désactiver si NFSv4 est inutilisé.
/sbin/mount.nfs	À désactiver si NFSv2/3 est inutilisé.
/sbin/umount.nfs4	À désactiver si NFSv4 est inutilisé.
/sbin/umount.nfs	À désactiver si NFSv2/3 est inutilisé.
/sbin/unix_chkpwd	Permet de vérifier le mot de passe utilisateur pour des programmes non root. À désactiver si inutilisé.
/usr/bin/at	À désactiver si *atd* n'est pas utilisé.
/usr/bin/chage	À désactiver.
/usr/bin/chfn	À désactiver.
/usr/bin/chsh	À désactiver.
/usr/bin/crontab	À désactiver si *cron* n'est pas requis.
/usr/bin/fusermount	À désactiver sauf si des utilisateurs doivent monter des partitions FUSE.
/usr/bin/gpasswd	À désactiver si pas d'authentification de groupe.
/usr/bin/locate	À désactiver. Remplacer par `mlocate` et `slocate`.
/usr/bin/mail	À désactiver. Utiliser un mailer local comme `ssmtp`.
/usr/bin/newgrp	À désactiver si pas d'authentification de groupe.

Exécutable	Commentaire
/usr/bin/passwd	À désactiver, sauf si des utilisateurs non root doivent pouvoir changer leur mot de passe.
/usr/bin/pkexec	À désactiver si PolicyKit n'est pas utilisé.
/usr/bin/procmail	À désactiver sauf si *procmail* est requis.
/usr/bin/rcp	Obsolète. À désactiver.
/usr/bin/rlogin	Obsolète. À désactiver.
/usr/bin/rsh	Obsolète. À désactiver.
/usr/bin/screen	À désactiver.
/usr/bin/sudo	Changement d'utilisateur. Ne pas désactiver.
/usr/bin/sudoedit	Idem `sudo`.
/usr/bin/wall	À désactiver.
/usr/bin/X	À désactiver sauf si le serveur X est requis.
/usr/lib/dbus-1.0/dbus-daemon-launch-helper	À désactiver quand D-BUS n'est pas utilisé.
/usr/lib/openssh/ssh-keysign	À désactiver.
/usr/lib/pt_chown	À désactiver (permet de changer le propriétaire des PTY avant l'existence de devfs).
/usr/libexec/utempter/utempter	À désactiver si le profil utempter SELinux n'est pas utilisé.
/usr/sbin/exim4	À désactiver si Exim n'est pas utilisé.
/usr/sbin/suexec	À désactiver si le `suexec` Apache n'est pas utilisé.
/usr/sbin/traceroute	(IPv4) Idem `ping`.
/usr/sbin/traceroute6	(IPv6) Idem `ping`.

La commande suivante permet de lister l'ensemble des fichiers setuid/setgid présents sur le système :

```
find / -type f \( -perm -2000 -o -perm -4000 \) -print 2>/dev/null
```

Retirer les droits *setuid* ou *setgid* se fait au travers de la commande `chmod` :

```
chmod u-s <fichier>  # Retire le bit setuid
chmod g-s <fichier>  # Retire le bit setgid
```

6.5.4 Fichiers sans utilisateur ou groupe propriétaire

Les fichiers présents dont le propriétaire ne fait pas partie de la base administrative `passwd` doivent être analysés et éventuellement corrigés afin d'avoir un propriétaire connu du système.

La commande suivante permet de lister l'ensemble des fichiers qui n'ont plus d'utilisateur ou de groupe associé :

```
find / -type f \( -nouser -o -nogroup \) -print 2>/dev/null
```

Des fichiers sans propriétaire connu peuvent être incorrectement attribués à un utilisateur lors de la création de son compte. Il faut donc s'assurer qu'aucun fichier n'est dans cette situation sur le système.

6.5.5 Les fichiers et répertoires accessibles à tous en écriture

Ces répertoires sont dans la majorité des cas utilisés comme zones de stockage temporaire, dans lequel un programme ira y enregistrer des données pour traitement. De nombreuses applications les utilisent de façon incorrecte. Les fichiers temporaires peuvent alors être détournés de leur fonction première et être exploités comme rebond (pour une escalade de privilèges par exemple).

R39 -  Répertoires temporaires dédiés aux comptes

Chaque compte utilisateur ou service doit posséder son propre répertoire temporaire et en disposer exclusivement.

Sur les distributions GNU/Linux récentes la méthode la plus directe pour créer un répertoire temporaire propre à chaque utilisateur est d'utiliser des modules PAM tels que `pam_mktemp` ou `pam_namespace`.

Il arrive que, pour diverses raisons techniques, l'exclusivité de ces fichiers temporaires ne puisse être garantie. Un palliatif (imparfait) est alors d'activer le *sticky bit* sur le répertoire afin que seul le compte qui a créé le fichier soit en droit de le supprimer.

Cela évite ainsi qu'un utilisateur (ou une application) puisse arbitrairement décider de supprimer et remplacer les fichiers temporaires d'un autre programme.

R40 -  Sticky bit et droits d'accès en écriture

Tous les répertoires accessibles en écriture par tous doivent avoir le *sticky bit* armé.

Cette mesure est imparfaite car elle n'empêche pas les situations de compétition entre deux programmes s'exécutant simultanément sous le même compte utilisateur.

la commande suivante permet de lister l'ensemble des répertoires modifiables par tous et sans *sticky bit* :

```
find / -type d \( -perm -0002 -a \! -perm -1000 \) -print 2>/dev/null
```

Il faut aussi s'assurer que le propriétaire du répertoire est bien `root`, sans quoi l'utilisateur propriétaire pourra à loisir modifier son contenu et ce malgré le *sticky bit*.

La commande suivante permet de lister l'ensemble des répertoires modifiables par tous et dont le propriétaire n'est pas `root` :

```
find / -type d -perm -0002 -a \! -uid 0 -print 2>/dev/null
```

Cependant aucun fichier régulier ne nécessite d'être modifiable par tous. Quand un fichier doit être modifiable par plusieurs utilisateurs ou programmes en même temps, un groupe doit être créé et seul ce groupe devra avoir des droits d'écriture sur ledit fichier.

La commande suivante permet de lister l'ensemble des fichiers modifiables par tout le monde :

```
find / -type f -perm -0002 -print 2>/dev/null
```

6.5.6 Les fichiers IPC nommés, sockets et/ou pipes

Les programmes peuvent échanger des informations au travers de `socket`. La plupart du temps ces sockets sont établies au niveau réseau (IP). Quand il s'agit d'établir des connexions entre processus locaux, des alternatives existent comme les *pipes* ou les *sockets* locales Unix.

Il faut prendre garde au fait que les droits d'accès qui s'appliquent à une socket locale sont ceux du répertoire la contenant et non ceux de la socket directement. Bien que certains systèmes honoreront tout de même ces permissions, POSIX ne l'impose pas.

R41 -  Sécurisation des accès pour les sockets et pipes nommées

Les sockets et pipes nommées doivent être protégées en accès par un répertoire possédant des droits appropriés.

Notamment, les sockets locales ne doivent pas être créées à la racine d'un répertoire temporaire accessible en écriture à tous.

Plusieurs commandes permettent d'obtenir des informations sur les sockets en cours d'utilisation sur le système. Suivant la politique de sécurité appliquée et les modules chargés, le résultat affiché peut ne pas être exhaustif.

Au travers de `sockstat` (ou `ss`), listant l'ensemble des sockets et les informations de processus associées pour des sockets locales :

```
ss -xp
```

Lister les mémoires partagées et leurs droits d'accès :

```
ipcs
ls /dev/shm
```

Obtenir des informations détaillées sur les I/O et IPC pour les processus du système :

```
lsof
```

6.6 Services réseau résidents

Les services résidents sont tous ceux en cours d'exécution sur la machine et pouvant être accédés par un processus local ou distant.

Ils sont tous autant de portes ouvertes sur le système permettant un accès illégitime à celui-ci lorsque le service est vulnérable ou mal configuré : site web permettant d'exécuter des commandes arbitraires, processus d'administration qui n'utilise pas un mécanisme d'authentification fiable, service réseau obsolète ayant une vulnérabilité exploitable, etc.

R42 - M I R E Services et daemons résidents en mémoire

Seuls les démons réseau strictement nécessaires au fonctionnement du système et du service qu'ils rendent doivent être résidents et n'être en écoute que sur les interfaces réseau adéquates.
Les autres démons doivent être désactivés et autant que possible désinstallés.

Les exemples les plus courants sont :
- les services de RPC (`portmap`, `rpc.statd`, etc.) qui ne sont en pratique utilisés que pour un serveur NFS ;
- les services bureautiques comme `dbus`, `hald`, `ConsoleKit`, `CUPS` ou `PolicyKit` ;
- le serveur X, rarement utile sur un serveur ;
- les services supplémentaires dont la configuration par défaut est souvent incomplète : SMTP (`Exim`, `Postfix`), NTP (`ntpd`) et DNS (`Bind`).

La liste des processus résidents et ceux en écoute sur le réseau peut être obtenue par les commandes suivantes :

```
ps aux
netstat -aelonptu
```

Une fois que les services non nécessaires sont désactivés, l'étape suivante est d'analyser les derniers programmes restants en y appliquant les règles suivantes :

1. mise à jour du programme ;
2. activation de mesures de cloisonnement (`chroot`, *containers*, filtres *SECCOMP* ou *Capsicum*, etc.) ;
3. retrait des privilèges quand ceux de `root` ne sont pas requis (en créant un compte dédié au service et en le configurant pour qu'il l'utilise) ;
4. toute directive de durcissement documentée pour le programme.

6.7 Configuration d'outils et services de monitoring

Plusieurs outils et services peuvent rapporter des éléments d'information sur le système. Une grande partie d'entre eux ne sont pas configurés de façon optimale à l'issue de l'installation du système. Les recommandations suivantes visent à combler cette carence.

6.7.1 Syslog

Cette partie vient en complément de la note technique *Recommandations de sécurité pour la mise en œuvre d'un système de journalisation* [11]11.

.

Le service *syslog* est le système de journalisation principal utilisé sous GNU/Linux. Il peut être décomposé en deux parties :

- un serveur (comme *syslog-ng* ou *rsyslog*) qui collecte l'ensemble des messages système au format `SYSLOG` qu'il reçoit ;
- plusieurs clients qui envoient des messages au serveur, majoritairement au travers de l'API `syslog()`.

Le serveur *syslog* est donc un élément fédérateur de journaux et accède à un grand nombre de données en provenance de sources systèmes diverses. N'importe quel service ou composant est susceptible de le solliciter en vue d'enregistrer un message, sans authentification.

R43 - ⬡ I R E Durcissement et configuration du service syslog

Le serveur *syslog* choisi doit être durci suivant les guides de sécurité associés à ce serveur. La configuration du service doit être réalisée suivant les *Recommandations de sécurité pour la mise en œuvre d'un système de journalisation* [12]11 accessible sur le site de l'ANSSI.

R44 - ⬡ I R E Cloisonnement du service syslog par chroot

Quand les moyens techniques et sa configuration le permettent, le service `syslog` doit être enfermé dans un environnement `chroot`.

R45 - ⬡ ⬡ ⬡ E Cloisonnement du service syslog par container

Le service `syslog` doit être isolé du reste du système dans un container dédié.

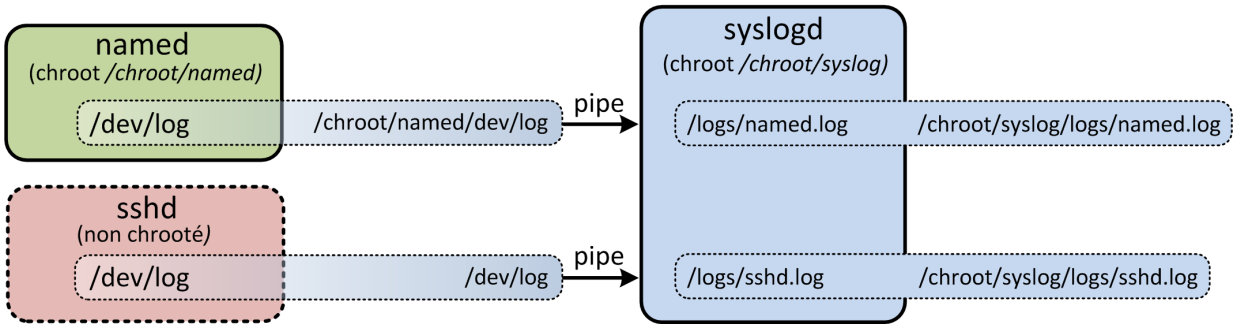

FIGURE 4 – Schéma syslog avec services chrooté et non-chrooté

11. http://www.ssi.gouv.fr/journalisation/

Il est d'usage d'avoir un serveur *syslog* résident qui ne gère que les messages soumis *localement* par les composants du système d'exploitation. Celui-ci jouera éventuellement le rôle d'émetteur quand ces messages doivent être envoyés à un serveur de centralisation de log.

R46 - I R E Journaux d'activité de service

Chaque service doit posséder un journal de log dédié sur le système. Ce fichier ne doit être accessible que par le serveur *syslog*, et ne doit pas être lisible, modifiable ou supprimable par le service directement.

L'objectif est de faire en sorte que deux niveaux de protection soient respectés :
- entre les services, afin qu'un service ne puisse ni manipuler ni accéder aux journaux enregistrés par un service différent ;
- au niveau du service lui-même, afin qu'en cas de compromission, celui-ci ne puisse trivialement lire, effacer ou altérer les traces enregistrées avant la compromission.

L'usage de l'API `syslog()` est une solution envisageable. L'envoi de messages peut être réalisé au travers de la commande `logger` en ligne de commande.

R47 - I R E Partition dédiée pour les journaux

Les journaux doivent reposer dans une partition séparée du reste du système.

La volumétrie réservée aux services de journalisation est toujours difficile à évaluer *a priori*. Il est préférable d'isoler les journaux du reste des volumes sur une partition dédiée afin d'éviter que le remplissage d'une partition ne puisse entraver la gestion des journaux qui y sont stockés (et réciproquement, qu'une saturation due aux journaux entraîne une indisponibilité de l'ensemble du système d'exploitation).

6.7.2 Mails et mails root

La messagerie est le deuxième service principal couramment utilisé par le système en vue d'informer sur certaines évolutions de son état. Cela se déroule généralement par l'envoi d'un message électronique à un destinataire spécifique, souvent un utilisateur humain.

Suivant les distributions, il arrive que le service de mail installé soit configuré en relai ouvert (accepte tout mail qui lui est soumis), mais avec la socket d'écoute uniquement liée à la boucle locale.

R48 - I R E Configuration du service local de messagerie

Quand un service de mail est installé sur la machine, celui-ci doit être configuré pour qu'il n'accepte que :
- les mails à destination d'un utilisateur local à la machine ;
- les connexions par la boucle locale (les connexions distantes au service de mail doivent être rejetées).

Dans la mesure du possible, préférer utiliser un service de redirection de mail léger (comme *ssmtp*).

Un cas fréquemment rencontré est le service `cron`, qui soumet systématiquement un mail lorsque la tâche exécutée affiche des données sur sa sortie d'erreur (stderr).

R49 - I R E Alias de messagerie des comptes de service

Pour chaque service exploité sur la machine, ainsi que le compte `root`, un alias mail vers un utilisateur administrateur doit être configuré afin qu'il reçoive les notifications et rapports expédiés par messagerie électronique.

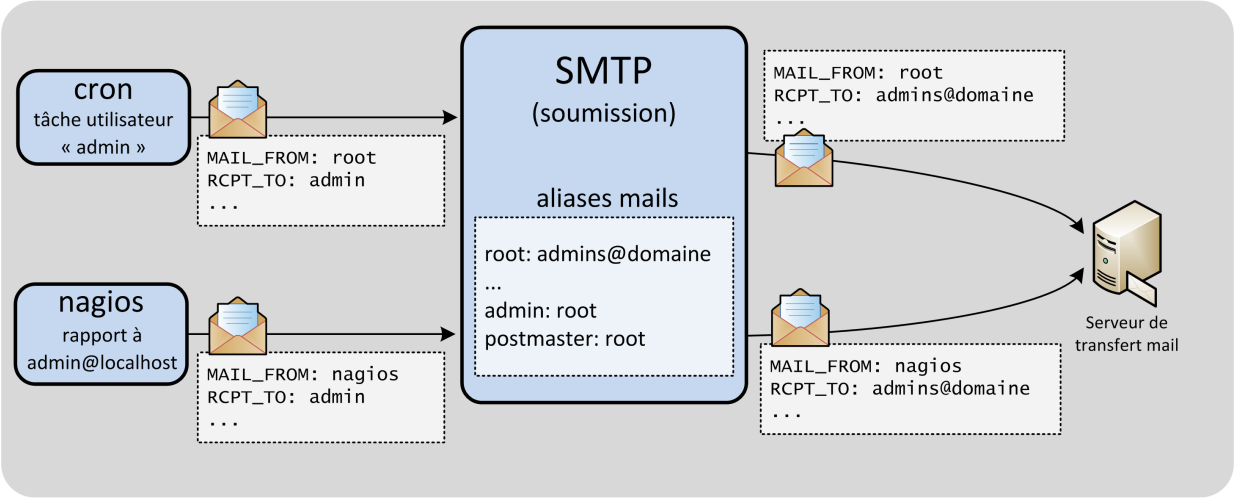

FIGURE 5 – Schéma d'envoi d'alertes mails à l'équipe d'administration

6.7.3 Surveillance du système par `auditd`

`auditd` est un service de journalisation avancé souvent présent sur les distributions GNU/Linux. Il permet d'enregistrer des opérations système spécifiques, voire d'alerter un administrateur lorsque des opérations privilégiées non prévues ont lieu.

Les évènements enregistrés dépendent des règles `auditd` écrites. Lorsqu'un enregistrement est déclenché, un second service, `audispd`, va se charger de son traitement : message SYSLOG, envoi de mail, écriture dans un fichier, etc.

`auditd` est capable de surveiller un grand nombre d'actions :

- appels système réalisés ;
- accès à une arborescence ou à des fichiers particuliers ;
- insertions de modules.

Consultez sa documentation pour plus de détails.

R50 - R E Journalisation de l'activité par auditd

La journalisation de l'activité du système doit être faite au travers du service `auditd`.

Voici un exemple de configuration `auditd` qui cherche à enregistrer des actions pouvant être dignes d'intérêt (voir fichier `/etc/audit/audit.rules`) :

```
# Exécution de insmod, rmmod et modprobe
-w /sbin/insmod   -p x
-w /sbin/modprobe -p x
-w /sbin/rmmod    -p x
# Journaliser les modifications dans /etc/
-w /etc/ -p wa
# Surveillance de montage/démontage
-a exit,always -S mount -S umount2
# Appels de syscalls x86 suspects
-a exit,always -S ioperm -S modify_ldt
# Appels de syscalls qui doivent être rares et surveillés de près
-a exit,always -S get_kernel_syms -S ptrace
-a exit,always -S prctl

# Rajout du monitoring pour la création ou suppression de fichiers
# Ces règles peuvent avoir des conséquences importantes sur les
# performances du système
-a exit,always -F arch=b64 -S unlink -S rmdir -S rename
-a exit,always -F arch=b64 -S creat -S open -S openat -F exit=-EACCESS
-a exit,always -F arch=b64 -S truncate -S ftruncate   -F exit=-EACCESS

# Verrouillage de la configuration de auditd
-e 2
```

Les journaux ainsi créés par `auditd` peuvent être verbeux en particulier quand de nombreuses activités sont rapportées. L'outil `aureport` permet de sélectionner les informations intéressantes en fonction de critères bien spécifiques : contexte sur des échecs d'authentification, rapport d'évènement sur certains fichiers ou répertoires, évènements anormaux (crash de programmes, reconfiguration de cartes réseaux), etc.

6.7.4 Surveillance du système de fichiers

L'opération de scellement d'un système de fichiers consiste en l'installation puis la configuration d'un service qui aura pour rôle de vérifier, au moins périodiquement, les modifications faites au niveau d'une arborescence.

C'est une fonctionnalité présente au sein de la plupart des HIDS. Le scellement permet de remonter des informations sur les modifications (écarts entre la version scellée et celle présente actuellement sur le système).

C'est une fonction utile aux administrateurs. Outre le fait qu'elle permet de conduire un audit périodique sur le système et de générer des rapports, elle permet aux équipes d'être informées sur les changements et donc d'obtenir un suivi des évolutions.

R51 - E Scellement et intégrité des fichiers

Tout fichier qui ne revêt pas un caractère transitoire (comme des fichiers temporaires, des bases de données, etc.) doit être surveillé par un programme de scellement.

Cela inclut notamment : les répertoires contenant des exécutables, des bibliothèques, des fichiers de configuration, ainsi que tout fichier pouvant contenir des éléments sensibles (clés cryptographiques, mots de passe, données confidentielles).

En plus de la fonction de scellement, certains outils sont capables de parcourir une arborescence et d'y repérer des situations potentiellement problématiques ou incorrectes (fichiers de clés ou de certificats avec des droits trop laxistes, fichiers de mots de passe lisibles par tout le monde, fichiers de log dont la taille diminue, etc.).

La compromission d'une machine pouvant s'accompagner d'une compromission de la base de scellement lorsque celle-ci est stockée localement, des mesures techniques doivent être mises en œuvre afin d'assurer que le contenu de la base reste autant que possible intègre.

R52 - E Protection de la base de données des scellés

La base de données de scellement doit être protégée de tout accès frauduleux par des mécanismes de signature cryptographique (avec la clé utilisée pour la signature non enregistrée localement en clair), ou être éventuellement stockée sur une machine distincte de celle sur laquelle le scellement est réalisé.

Il existe de nombreuses solutions. Les plus déployés sur les systèmes GNU/Linux sont Tripwire, Samhain et AIDE.

7 Solutions de cloisonnement et contrôle d'accès

7.1 Cloisonnement et virtualisation

Le cloisonnement est une technique qui, quand elle est correctement mise en place, permet de sécuriser les services les uns vis-à-vis des autres tout en rendant l'usage de ceux-ci plus souples (gestion multi-instances, application de restrictions aux ressources allouées, etc.).

R53 - R E Restriction des accès des services déployés

Les services déployés doivent voir leurs accès limités au strict nécessaire système, notamment au niveau fichiers, processus ou réseau.

Il est historiquement réalisé au travers de `chroot` qui permet de restreindre la vue du système de fichiers d'un processus à un répertoire donné (qui devient sa racine). En revanche, les accès aux processus et au réseau ne sont pas cloisonnés, le modèle Unix historique restant applicable.

Aujourd'hui d'autres solutions de cloisonnement existent et offrent différents niveaux d'abstraction. Elles se caractérisent généralement par le composant qui va mettre en place les interfaces de virtualisation :

- par *containers*, où le noyau est capable de gérer plusieurs instances systèmes (LXC, VServer, jails, zones) ;
- par émulation, où un émulateur reproduit une machine physique complète (QEMU, VirtualBox, Parallels) ;
- par hyperviseur léger *bare-metal* (Xen, Hyper-V), où l'hyperviseur va gérer (éventuellement avec l'aide d'un système hôte) différentes machines virtuelles ;
- par hyperviseur noyau (Linux KVM).

Ces techniques ne sont pas exclusives ; il doit en être fait un usage raisonnable en gardant à l'esprit que les interfaces qu'elles offrent sont autant de portes par lesquelles une intrusion est possible : un système à base de containers pour lequel le noyau est compromis verra l'ensemble des containers eux-mêmes compromis ; il en va de même avec les machines virtuelles et leur hyperviseur.

R54 - R E Durcissement des composants de virtualisation

Tout composant support de virtualisation doit être durci, notamment par l'application de mesures techniques contrant les tentatives d'exploitation.

Une solution de durcissement efficace sur un système GNU/Linux est d'appliquer le patch grsecurity [13] au noyau. Cette possibilité est abordée plus loin dans le présent document.

7.1.1 Chroot

`chroot` est originellement le tout premier mécanisme de cloisonnement qui a été utilisé pour des applications. C'est aussi le plus pauvre. Il consiste à changer le répertoire racine d'un programme donné : une fois *chrooté* il ne peut alors plus accéder aux répertoires autres que celui pris comme racine ainsi que les descendants. Le processus n'a qu'une vision partielle du système de fichiers.

`chroot` présente de nombreuses faiblesses sous GNU/Linux, parmi lesquelles :

- impossibilité d'interdire à un utilisateur `root` de sortir de sa cage ;
- impossibilité (suivant certains systèmes d'exploitation) d'interdire à un utilisateur non privilégié de sortir de sa cage avec la complicité d'un processus externe ;
- cloisonnement limité au système de fichiers ; les accès aux processus, au réseau, etc. ne sont pas bloqués ;
- nécessité de posséder initialement les privilèges de `root` afin de pouvoir être exécuté.

R55 - I R E Cage chroot et privilèges d'accès du service cloisonné

La cage `chroot` d'un service ne doit contenir que le strict minimum nécessaire à la bonne exécution de celui-ci.

Lorsqu'il est enfermé dans cette cage, le service doit impérativement s'exécuter avec les privilèges d'un utilisateur simple (non `root`), dédié (dont l'identité est utilisé uniquement par la cage) et ne pas avoir accès en écriture à cette nouvelle racine.

13. https://grsecurity.net

L'administrateur désireux d'utiliser une cage `chroot` doit prendre garde à sa construction : il faut s'assurer que le processus confiné n'accède pas en lecture/écriture directement à sa racine. Cela lui permettrait de créer puis contrôler des fichiers dont le chemin est sensible (comme `./etc/shadow`), et qui seraient potentiellement utilisés de façon erronée par tout binaire qui se retrouverait enfermé dans cette cage.

`chroot` est le plus souvent implémenté comme mécanisme interne de protection pour un service donné : une fois les opérations les plus privilégiées effectuées, le service utilise `chroot` pour modifier sa racine, puis change vers un utilisateur non privilégié afin de perdre ses droits d'administrateur `root`.

R56 -  Activation et utilisation de chroot par un service

`chroot` doit être utilisé et activé lorsque le service implémente ce mécanisme.

7.2 Contrôle d'accès et mécanismes de sécurité avancés

Le contrôle d'accès consiste à s'assurer qu'une entité (processus ou utilisateur) a des droits suffisants en vue d'accéder à une ressource donnée. Bien qu'un contrôle d'accès permette lui aussi d'effectuer du cloisonnement, l'approche choisie est généralement différente de celle adoptée au travers de mécanismes de virtualisation : le contrôle d'accès laisse souvent la référence à un objet système visible à l'application et retourne une erreur en cas de privilège insuffisant pour y accéder, tandis qu'un système reposant sur de la virtualisation cloisonnera l'application par l'absence de référence à cet objet (pointeur, chemin d'accès, etc.).

Historiquement le contrôle d'accès sous Unix/Linux repose sur la notion d'utilisateur. D'autres méthodes sont cependant applicables aujourd'hui au travers des LSM, dont `SELinux` et `AppArmor` sont les plus connus et utilisés. Il est important de noter que ces modèles d'accès viennent en supplément du modèle utilisateur traditionnel Unix et ne s'y substituent pas. Par ailleurs `AppArmor` et `SELinux` sont mutuellement exclusifs : les deux systèmes ne peuvent pas fonctionner conjointement sur le même noyau Linux.

Un patch additionnel au noyau Linux, grsecurity, propose des fonctionnalités plus poussées de durcissement que ne le permettent les modules de sécurité Linux [14]. Malheureusement ce patch est assez peu appliqué en standard par les distributions GNU/Linux ce qui requiert des efforts d'intégration et de maintenance supplémentaires.

7.2.1 Modèle traditionnel Unix

Le modèle d'accès historique repose sur la notion d'utilisateurs qui sont reconnus par le système au travers d'identifiants uniques (UID). Chaque processus, fichier, répertoire, ressource, etc. est associé à un UID (son propriétaire), qui couplé à des droits va autoriser ou refuser un accès (ou non) à une ressource.

Plusieurs UID peuvent être réunis en groupe auquel un identifiant unique peut être associé : un GID. Le principe de gestion des accès reste cependant similaire aux UID.

14. Le projet fournit une matrice de comparaison qui permet de s'en convaincre : https://grsecurity.net/compare.php

Ce modèle d'accès est un DAC, pour contrôle d'accès discrétionnaire. Il est discrétionnaire car l'utilisateur propriétaire d'une ressource est celui qui en spécifie les droits d'accès, avec des droits de lecture, écriture et exécution chacun donné pour :

- l'utilisateur propriétaire de la ressource ;
- le groupe propriétaire de la ressource ;
- le reste du monde.

Cette approche est encore courante aujourd'hui et reste celle appliquée par défaut sous GNU/Linux. Elle présente cependant d'importantes limitations :

- les droits sont à la discrétion du propriétaire, ce qui peut ne pas convenir aux environnements contraints par une politique de sécurité ;
- le propriétaire peut être incapable de donner les bons droits sur ses ressources, ce qui offre des possibilités d'accès à des données confidentielles (voire des compromissions) ;
- l'isolation entre les utilisateurs est grossière, les droits par défaut étant souvent laxistes ;
- il n'y a que deux niveaux de privilèges sur le système : administrateur `root`, et les utilisateurs simples, sans privilège. Le changement d'utilisateur requiert l'usage de binaires sensibles sur le système (ceux *setuid root*) ;
- il ne permet pas de retirer des privilèges particuliers à un processus suivant le contexte : le navigateur Web et le traitement de texte d'un même utilisateur auront autant de privilèges l'un que l'autre sur les ressources ;
- la surface d'attaque est grande, un utilisateur standard a accès à un grand nombre d'informations sur le système d'exploitation sous-jacent.

Ces limitations ont contribué à l'émergence d'outils de gestion de délégation de droits, le plus connu étant `sudo`. D'autres approches, comme les LSM, offrent plus de possibilités, mais au prix d'un investissement et d'une complexité plus élevés.

7.2.2 sudo

`sudo` est un utilitaire installé lorsqu'il y a un besoin de déléguer des droits et privilèges à différents utilisateurs. Cette délégation repose sur la possibilité pour un utilisateur donné d'exécuter une commande préalablement définie avec les privilèges d'un autre utilisateur. Afin de pouvoir réaliser cela, `sudo` est un exécutable *setuid root*. Il est donc important de se préoccuper de sa sécurité à deux niveaux :

- au niveau du durcissement, afin d'éviter à un utilisateur malveillant de pouvoir exploiter les vulnérabilités du binaire ;
- au niveau de sa configuration, où le fait de donner le droit à un utilisateur d'exécuter certaines commandes peut lui attribuer plus de privilèges et de prérogatives qu'initialement nécessaires.

Le fonctionnement de `sudo` repose en très grande partie sur le modèle traditionnel Unix pour fonctionner, qu'il enrichit avec une logique de transition plus fine que ne le permet originellement `su`. Les éléments ci-dessous visent à donner quelques recommandations et pistes à étudier en vue d'utiliser et configurer `sudo` de telle sorte qu'il offre le moins de contournement et d'effets de bords possibles.

Ces recommandations s'appliquent aussi bien pour les cas où `sudo` est utilisé pour l'exécution de commandes privilégiées par un utilisateur simple, ou la restriction de privilèges quand un utilisateur privilégié (tel que `root`) souhaite exécuter une commande avec des droits plus faibles, par précaution.

7.2.2.1 Droits d'accès

`sudo` est généralement installé par défaut avec des droits ouverts et permet à n'importe quel utilisateur de l'utiliser (droit d'exécution pour *tout le monde*). `sudo` étant un exécutable privilégié

et complexe de par sa configuration, il est préférable de restreindre ses droits d'exécution à un groupe utilisateur dédié. Cela réduit la surface d'attaque, notamment quand le système et le binaire lui-même sont victimes de vulnérabilités exploitables par n'importe quel utilisateur (CVE-2012-0809, CVE-2012-0864).

R57 - I R E Groupe dédié à l'usage de sudo

Un groupe dédié à l'usage de `sudo` doit être créé. Seuls les utilisateurs membres de ce groupe doivent avoir le droit d'exécuter `sudo`.

Un cas envisageable est de créer un groupe dédié (ici : `sudogrp`), et de lui attribuer les droits pour exécuter `sudo`. Seuls les membres de ce groupe pourront ainsi y faire appel :

```
# ls -al /usr/bin/sudo
-rwsr-x---. 2 root sudogrp [...] /usr/bin/sudo
```

Cette modification doit être vérifiée et éventuellement appliquée après chaque mise à jour, ces changements pouvant être écrasés par les scripts d'installation.

7.2.2.2 Configuration générale

La configuration se fait au travers de l'édition du fichier `/etc/sudoers` au travers de la commande `visudo`. Il contient un ensemble de directives qui vont permettre à `sudo` de connaître les commandes qu'un utilisateur aura le droit d'exécuter ou pas, ceci éventuellement en fonction du nom d'hôte de la machine.

Il est particulièrement difficile de fournir un ensemble de recommandations capables de couvrir l'ensemble des situations dans lesquelles `sudo` peut être utilisé. En effet, le nombre de commandes disponibles, l'architecture du système d'information, les services installés et leurs configurations, etc. rendent quasiment impossible l'établissement d'une configuration standard sécurisée.

Quelques règles de bonnes pratiques doivent être respectées afin d'éviter autant que possible des erreurs de configuration pouvant conduire à des contournements de la politique de sécurité. La lecture de la page man de `sudoers` est fortement recommandée, notamment la partie traitant des échappements shell.

R58 - I R E Directives de configuration sudo

Les directives suivantes doivent être activées par défaut :

noexec	appliquer le tag **NOEXEC** par defaut sur les commandes ;
requiretty	imposer à l'utilisateur d'avoir un tty de login ;
use_pty	utiliser un pseudo-tty lorsqu'une commande est exécutée ;
umask=0027	forcer umask à un masque plus restrictif ;
ignore_dot	ignorer le '.' dans `$PATH` ;
env_reset	réinitialiser les variables d'environnement ;
passwd_timeout=1	allouer 1 minute pour entrer son mot de passe.

Ce qui donne dans le fichier `/etc/sudoers` :

```
Defaults noexec,requiretty,use_pty,umask=0027
Defaults ignore_dot,env_reset,passwd_timeout=1
```

Le reste du fichier est ensuite majoritairement constitué de règles permettant de déclarer pour un utilisateur donné (ou un groupe) la liste des commandes qu'il est en mesure d'exécuter, le tout aidé de spécifications d'alias lorsque la politique de gestion des droits devient complexe (`User_Alias`, `Runas_Alias`, etc.). La vérification du droit d'exécution (ou pas) d'une commande repose sur une comparaison de chaîne entre la commande souhaitée par l'utilisateur et la spécification présente dans `sudoers`. Le modèle le plus simple est le suivant :

```
utilisateur   hostname = ( utilisateur-cible ) commande, [...]
```

R59 - (M) (I) (R) (E) Authentification des utilisateurs exécutant sudo

Une authentification de l'utilisateur appelant doit être effectuée avant toute exécution de commande par `sudo`.
Le mot-clé `NOPASSWD` ne doit pas être utilisé.

L'obligation d'authentification vise à retarder une escalade de privilèges évidente lorsqu'un compte est compromis, en particulier quand l'attaquant n'est pas en mesure de s'authentifier comme l'utilisateur légitime. Par commodité `sudo` ne redemande le mot de passe qu'après une période de temps configurable (15 min par défaut). Dans tous les cas, une authentification doit avoir été initialement faite.

R60 - (I) (R) (E) Privilèges des utilisateurs cible pour une commande sudo

Les utilisateurs cible d'une règle doivent autant que possible être un utilisateur non privilégié (c'est-à-dire non `root`).

Il est courant que la commande à exécuter ne requiert pas de droit super-utilisateur (édition d'un fichier dont le propriétaire n'est pas `root`, envoi d'un signal à un processus non privilégié, etc.). Afin de limiter toute tentative d'escalade de privilèges au travers d'une commande exécutée, il est préférable que des droits d'utilisateur simple lui soient appliqués.

R61 - (R) (E) Limitation du nombre de commandes nécessitant l'option EXEC

Les commandes nécessitant l'exécution de sous-processus (tag *EXEC*) doivent être explicitement listées et réduites autant que possible au strict minimum.

Des commandes fonctionnellement riches peuvent être exécutées au travers de `sudo`, tels des éditeurs textes (`vi`) ou des binaires (`tcpdump`). Elles peuvent permettre l'exécution de sous commandes dans l'environnement créé par `sudo` et ainsi donner la possibilité à un utilisateur d'exécuter d'autres programmes avec des privilèges non prévus à l'origine. Cela peut ainsi conduire à des contournements de la politique de sécurité, voire des escalades de privilèges.

À cette fin, `sudo` permet de surcharger les différentes fonctions permettant d'exécuter d'autres programmes. Cela permet par exemple de bloquer la création triviale d'un sous shell au travers de `vi`. Il est cependant important de noter que cette protection est imparfaite et ne fait que surcharger les fonctions permettant de créer ces nouveaux processus. Le processus reste libre d'exécuter les appels systèmes qu'il souhaite (`execve` notamment) et ainsi contourner ce mécanisme de protection.

R62 - I R E Du bon usage de la négation dans une spécification sudo

Les spécifications de commandes ne doivent pas faire intervenir de négation.

Spécifier des droits d'accès par négation (approche de type liste noire) est inefficace et peut être facilement contourné. On s'attend par exemple à ce qu'une spécification du type :

```
# A eviter absolument, cette règle est trivialement contournable!
user ALL = ALL,!/bin/sh
```

interdise l'exécution du shell, mais il n'en est rien : il suffit de copier le binaire `/bin/sh` sous un autre nom pour que celui-ci devienne exécutable par cette règle au travers du mot clé *ALL*.

Afin de déterminer si une commande est exécutable par un utilisateur, `sudo` réalise une comparaison de chaîne entre la commande désirée et les spécifications correspondantes dans le `sudoers` suivant les règles du *globbing* shell ; cette évaluation inclut les arguments.

R63 - I R E Arguments explicites dans les spécifications sudo

Toutes les commandes du fichier `sudoers` doivent préciser strictement les arguments autorisés à être utilisés pour un utilisateur donné.
L'usage de `*` (*wildcard*) dans les règles doit être autant que possible évité. L'absence d'arguments auprès d'une commande doit être spécifiée par la présence d'une chaîne vide (`""`).

Les arguments permettent de modifier de façon assez importante le comportement d'un programme, que cela soit en terme d'opérations réalisées (lecture, écriture, suppression, etc.) ou de ressources accédées (chemin d'accès dans une arborescence). Afin d'éviter toute possibilité de détournement d'une commande par un utilisateur, les ambiguïtés doivent être levées au niveau de sa spécification.

Sur certains systèmes, les messages noyau ne sont accessibles que par `root`. Si un utilisateur doit néanmoins posséder les privilèges lui permettant de les afficher, il faut restreindre ses arguments afin d'éviter qu'il puisse vider le tampon au travers de l'option `-c` :

```
user ALL = dmesg ""
```

Il est important de noter que permettre à un utilisateur non privilégié d'exécuter sous l'identité `root` un programme susceptible de réaliser une écriture arbitraire (rendue possible par l'écriture trop laxiste d'une règle) reviendra dans les faits à donner à cet utilisateur les privilèges de `root` car celui-ci

pourra, par diverses méthodes (manipulation des fichiers de mots de passe, modification de binaires *setuid root*,...), obtenir les privilèges de *root* sans le contrôle d'accès et d'exécution que permet `sudo`.

Il est assez courant de permettre l'édition d'un fichier par un utilisateur au travers de `sudo` en appelant directement un éditeur de texte. Un éditeur de texte est un programme fonctionnellement riche, pouvant ouvrir et éditer d'autres fichiers par des commandes internes voire même exécuter un script shell. Certains éditeurs peuvent donc offrir un environnement privilégié complet à un utilisateur. `sudoedit` s'affranchit de ces problématiques en laissant le soin à l'éditeur de modifier un fichier identique temporaire avec les droits de l'utilisateur courant, puis de remplacer le fichier pointé par ce même fichier temporaire une fois l'opération terminée. L'éditeur ne s'exécute donc qu'avec les droits de l'utilisateur courant et jamais avec les privilèges de l'utilisateur cible.

R64 -  Du bon usage de sudoedit

L'édition d'un fichier par `sudo` doit être réalisée au travers de la commande `sudoedit`.

7.2.3 AppArmor

AppArmor est un des LSM fréquemment rencontrés sous GNU/Linux, et celui utilisé par défaut par les variantes de SUSE (dont OpenSUSE) et Ubuntu. C'est un modèle MAC où une autorité décide des accès d'une application sans que celle-ci puisse les altérer.

Sa documentation est directement accessible sur le site du projet [15]. Sa configuration repose sur des spécifications de droits basées sur des chemins dans l'arborescence du système de fichiers, et appliquées pour chaque exécutable.

En plus des spécifications d'accès, AppArmor permet de bloquer l'usage des capacités POSIX, ainsi que restreindre l'usage réseau (directive `network`). Ces restrictions restent cependant sommaires et n'ont pas la richesse d'expression d'`iptables` [16].

Lorsqu'un exécutable sous AppArmor appelle un autre exécutable, son profil de sécurité permet de déclarer la façon dont la transition doit s'effectuer (héritage du profil actuel, activation d'un autre profil de sécurité, sortie du confinement, etc.).

Bien que simple à appréhender pour un administrateur habitué au modèle d'accès traditionnel Unix, AppArmor ne permet pas d'attacher de labels de sécurité à un flux ou à des messages. C'est un framework qui se concentre essentiellement sur la spécification d'accès et de privilèges pour des exécutables, la notion de label de sécurité en est absente.

Une grande partie des distributions GNU/Linux qui l'utilisent fournissent, au moins pour les exécutables les plus sensibles (`syslogd`, `ntpd`, binaires *setuid root*, etc.) des profils AppArmor par défaut dans le répertoire `/etc/apparmor.d/`, un fichier par exécutable. Les opérations (y compris les refus d'accès) sont enregistrées par `auditd` directement dans le fichier `/var/log/audit/audit.log`.

15. http://wiki.apparmor.net/index.php/Documentation
16. Utilitaire système permettant de configurer les règles de pare-feu sous GNU/Linux.

R65 - E Activation des profils de sécurité AppArmor

Tout profil de sécurité AppArmor présent sur le système doit être activé par défaut.

L'activation de AppArmor est différente suivant les distributions, et repose généralement sur l'exécution d'un script comme `/etc/init.d/apparmor`. Pour s'assurer que AppArmor est bien actif, utiliser `aa-status` une fois le démarrage terminé :

```
# aa-status
apparmor module is loaded.
5 profiles are loaded.
5 profiles are in enforce mode.
...
4 processes have profiles defined.
4 processes are in enforce mode.
...
0 processes are unconfined but have a profile defined.
```

Il faut notamment vérifier que les processus en cours d'exécution sur le système et pour lesquels un profil est déclaré soient confinés par AppArmor.

7.2.4 SELinux

SELinux (pour *Security-Enhanced Linux*) est un LSM largement utilisé dans les distributions Red Hat et CentOS.

SELinux repose sur l'usage de *labels* assignés à des objets (fichiers, processus, sockets, etc.) qui permettent, suivant le *domaine* auquel un processus est confiné, d'autoriser ou de refuser un accès à une ressource.

Au niveau du système de fichiers, les labels sont stockés comme attributs étendus et peuvent être obtenus via `ls -Z`. Ceux des processus sont visibles avec `ps -Z`.

L'usage des labels permet de déclarer des domaines de responsabilité et donc de connaître à tout instant les opérations que peut effectuer un processus sur une ressource donnée. SELinux permet aussi de définir les *transitions* d'un domaine à un autre lorsqu'un utilisateur requiert des privilèges d'accès spécifiques. La granularité d'accès est bien plus fine que dans AppArmor, au détriment d'une complexité plus importante.

La documentation de SELinux est extensive et accessible sur le page wiki du projet [17].

Cette section se borne à donner quelques vérifications pour que tout administrateur désireux d'utiliser les profils fournis par sa distribution puisse le faire correctement.

Les politiques SELinux disponibles par défaut sur les distributions sont les suivantes :

17. http://selinuxproject.org/page/Main_Page

minimale	(*minimum*) politique minimale qui ne confine qu'un nombre très restreint de services ;
ciblée	(appelée *targeted* ou *default*) politique cloisonnant chaque service système dans un domaine. Les utilisateurs interactifs ne sont pas cloisonnés ;
multi-niveau	(*mls*) politique incluant la politique ciblée et qui propose en sus l'utilisation de niveaux de classification hiérarchiques (sensibilité).

L'utilisation des politiques *minimale* et *multi-niveau* est déconseillée.

La suite de cette section va se concentrer sur la politique par défaut *targeted*. Chaque service système (démon) est confiné dans un domaine distinct. Plusieurs instances d'un même service exécuté dans un même domaine peuvent être séparées à l'aide des catégories. Les utilisateurs interactifs sont par défaut associés à un domaine non confiné (*unconfined_t*) qui n'impose pas de restrictions supplémentaires. Le modèle de contrôle d'accès reste dans ce cas très proche du DAC.

R66 - E Activation de la politique targeted avec SELinux

Il est recommandé d'activer la politique *targeted* quand la distribution en offre le support et qu'elle n'exploite pas de module de sécurité autre que SELinux.

L'activation de la politique *targeted* passe généralement par trois étapes :

1. Vérifier la valeur de `SELINUXTYPE` dans le fichier `/etc/selinux/config` :

   ```
   # grep SELINUXTYPE /etc/selinux/config
   SELINUXTYPE=targeted
   ```

2. Si la valeur n'est pas *targeted* (ou *default*), éditer le fichier pour déclarer la variable `SELINUXTYPE` à *targeted* puis recharger la politique de sécurité :

   ```
   # semodule -R
   ```

3. Puis s'assurer que SELinux est actif et avec les bons paramètres :

   ```
   # sestatus
   SELinux status:                 enabled
   ...
   Loaded policy name:             default
   Current mode:                   enforcing
   ```

Le comportement d'une politique de sécurité peut être modifié au travers de variables booléennes. Certaines sont utiles d'un point de vue sécurité.

R67 - E Paramétrage des booléens SELinux

Il est recommandé de mettre les valeurs booléennes suivantes à *off* :

allow_execheap	si à *off*, interdit aux processus de rendre exécutable leur tas (heap) ;
allow_execmem	si à *off*, interdit aux processus d'avoir une zone mémoire avec des droits w (écriture) et x (exécution) ;
allow_execstack	si à *off*, interdit aux processus le droit de rendre leur pile (stack) exécutable ;
secure_mode_insmod	si à *off*, interdit le chargement dynamique des modules par n'importe quel processus ;
ssh_sysadm_login	si à *off*, interdit les logins SSH de se connecter directement en rôle sysadmin.

La valeur par défaut de ces variables peut être modifiée avec la commande `setsebool` :

```
# setsebool -P allow_execheap=off
# setsebool -P allow_execmem=off
# setsebool -P allow_execstack=off
# setsebool -P secure_mode_insmod=off
# setsebool -P ssh_sysadm_login=off
```

Il en existe de nombreuses autres. La commande `semanage boolean --list` permet d'obtenir des informations sur les variables booléennes définies dans la politique utilisée sur un système. Une grande partie de ces variables sont aussi documentées dans le wiki de CentOS [18].

Tout comme AppArmor, les opérations sont enregistrées par `auditd` directement dans le fichier `/var/log/audit/audit.log` sous le mot clé AVC.

R68 - E Désinstallation des outils de débogage de politique SELinux

Les outils de manipulation et de débogage de politique SELinux ne doivent pas être installés sur une machine en production.

Leur usage doit être limité aux machines de développement, sur lesquelles l'interprétation d'une erreur remontée dans les logs d'audit SELinux peut être effectuée.

Le démon *setroubleshootd* doit être désactivé et les paquets suivants désinstallés : setroubleshoot, setroubleshoot-server, setroubleshoot-plugins.

Par défaut, les droits des utilisateurs interactifs ne sont pas restreints dans la politique *targeted*. Mais il est possible de confiner sélectivement les utilisateurs n'ayant pas besoin d'effectuer des tâches d'administration sur un système. Il faut pour cela leur associer l'utilisateur SELinux *user_u* avec la commande suivante :

18. http://wiki.centos.org/fr/TipsAndTricks/SelinuxBooleans

```
# usermod -Z user_u <utilisateur>
```

7.2.5 Grsecurity

Le patch grsecurity [19] est un patch offrant de nombreuses contre-mesures vis-à-vis d'exploits cherchant à compromettre le système d'exploitation et plus particulièrement son noyau. Il est très largement compatible avec les modules de sécurité Linux ; les deux peuvent être utilisées conjointement sur le même système.

À la différence de AppArmor et SELinux qui sont intégrés au noyau et qui utilisent l'interface LSM, grsecurity est un patch à appliquer directement sur les sources du noyau. Il doit ensuite être configuré (au travers d'un `make menuconfig` par exemple), puis recompilé.

Pour des raisons de commodité, des dépôts non-officiels existent et permettent à un administrateur de récupérer des noyaux Linux pré-patchés avec grsecurity. Il est cependant ***primordial*** de s'assurer que ces dépôts soient légitimes avant d'utiliser aveuglément le noyau qu'ils fournissent.

L'accès et la documentation associée à ces dépôts sont généralement référencés directement par les mainteneurs des distributions GNU/Linux sur leur wiki.

La démarche à suivre sera souvent la suivante :

1. obtenir la version du noyau actuellement utilisé (`uname -r`) ;
2. récupérer les sources de grsecurity [20] et leur signature pour la version la plus proche couvrant celle du noyau utilisé ;
3. vérifier la signature (à l'aide de `gpg`) ;
4. appliquer le patch grsecurity sur les sources du noyau fourni par la distribution (au travers de la commande `patch`) ;
5. reconfigurer le noyau pour la compilation, via un `make menuconfig` par exemple (éventuellement en réutilisant la configuration de l'ancien noyau par `make oldconfig`) ;
6. compiler les sources, puis installer le nouveau noyau en lieu et place de l'ancien.

Certaines distributions donnent des indications détaillées sur la marche à suivre pour obtenir un noyau modifié et le fournir en temps que paquet d'installation :

Distribution	Instructions
Debian	https://wiki.debian.org/grsecurity
Redhat/CentOS	https://wiki.centos.org/HowTos/Custom_Kernel

R69 - E Durcissement d'un noyau avec grsecurity

Il est recommandé d'utiliser un noyau durci avec le patch grsecurity quand la distribution GNU/Linux le permet.

19. https://grsecurity.net
20. https://grsecurity.net/download.php