PREMIER MINISTRE

Secrétariat général
de la défense
et de la sécurité nationale

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

Paris, le 27 juillet 2012

N° DAT-NT-002/ANSSI/SDE/NP

Nombre de pages du document : 7

Note technique

Recommandations de sécurité relatives à un système GNU/Linux

Informations

Personnes ayant contribué à la rédaction de ce document:

Contributeurs	Rédigé par	Approuvé par	Date
Division assistance technique, Bureau audits et inspections	Laboratoire architectures matérielles et logicielles	SDE	5 juillet 2012

Évolutions du document :

Version	Date	Nature des modifications
1.0	5 juillet 2012	Version initiale
1.1	26 juillet 2012	Corrections mineures

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

Préambule

Ce document ne se veut pas une procédure générique de sécurisation d'un système GNU/Linux, les bonnes pratiques à appliquer dépendant de la fonction de la machine à sécuriser (serveur, poste client, passerelle, etc.). Les éléments qui suivent ne sauraient donc aucunement avoir un caractère exhaustif. Il s'agit simplement d'énoncer les principaux axes de durcissement à explorer. Ces derniers viennent en complément de principes de base applicables à tout système d'eploitation comme par exemple l'application régulière des correctifs.

Il convient bien entendu d'étudier l'applicabilité de chaque recommandation au cas d'espèce considéré. Il est par ailleurs fortement conseillé d'avoir recours aux compétences d'un expert du système GNU/Linux pour la mise en oeuvre de ces bonnes pratiques.

Principe de base : Réduction de la surface d'attaque !

En premier lieu, « minimisation » est un leitmotiv qui doit guider la configuration sécurisée d'un système GNU/Linux quel que soit son usage : il faut supprimer tous les éléments logiciels superflus pour réduire la surface d'attaque (tout ce qui est inutile aux utilisateurs légitimes ne servira qu'à l'attaquant). Le principe de minimisation de la surface d'attaque s'applique non seulement à la couche utilisateur («*userland*»), mais aussi au noyau.

Au niveau *userland* :

Cela passe notamment par l'élimination des services inutiles en écoute sur le réseau [1], en priorité, ainsi que des services qui s'exécutent sous le compte *root* et qui ne sont pas nécessaires au fonctionnement du système (exemples : *pulseaudio* ou *dbus*). Dans le cas d'un serveur, cela passe notamment par l'élimination du serveur X, qui constitue un service particulièrement complexe et privilégié. On notera bien que le terme "élimination" pris dans ce contexte ne signifie pas nécessairement la désinstallation des programmes associés, qui peut s'avérer impossible du fait de dépendances entre paquetages. Dans un tel cas de figure, il peut s'avérer nécessaire de conserver le démon installé sur le système, mais en faisant en sorte qu'il ne soit pas lancé automatiquement. De manière complémentaire, on pourra chercher à désinstaller tous les programmes qui ne sont pas utiles dans le contexte d'utilisation légitime du système, et dont l'installation n'est pas strictement nécessaire au titre des dépendances entre paquetages. De tels programmes ne peuvent en effet servir qu'à un attaquant. A titre d'exemple, si la présence d'un compilateur ou d'un débogueur (*gdb*) sur le système cible n'est jamais strictement nécessaire à la mise en oeuvre d'une attaque, la disponibilité de tels outils peut souvent faciliter grandement la tâche de l'attaquant.

Lorsque la machine comporte plusieurs interfaces réseau, il est souhaitable de spécialiser celles-ci pour dissocier les différents type de flux métiers et les flux d'administration. Il convient alors d'imposer que les services soient en écoute uniquement sur les interfaces adaptées. Par exemple, dans le cas d'un serveur Web doté de deux interfaces, le serveur http ne sera en écoute que sur l'interface de production alors que le serveur ssh ne sera en écoute que sur l'interface d'administration.

Le principe de « moindre privilège » doit aussi être gardé à l'esprit tout au long de la configuration du système. On limitera ainsi autant que faire se peut l'ensemble des programmes privilégiés. En particulier, outre les démons exécutés en tant que *root* (décrits plus haut), les éventuels scripts lancés en arrière plan par un *cron* et les programmes *setuid* doivent être les moins nombreux possibles. [2] On

1. La commande `netstat --programs --listening` permet d'inventorier les programmes en écoute sur une socket réseau ou locale.
2. La commande `find / -user root -perm /u+s` permet de lister les exécutables setuid root du système.

notera en particulier qu'attribuer un bit *setuid* à un programme qui n'a pas été développé spécialement pour cela constitue généralement une grave erreur. A ce titre, lorsqu'il est nécessaire de déléguer des privilèges d'accès via un exécutable spécifique, il est préférable de réaliser un contrôle d'accès par groupe en s'appuyant sur le bit *setgid*, plutôt qu'un contrôle d'accès par utilisateur reposant sur un bit *setuid*. Lorsqu'elles sont disponibles, il conviendra enfin d'activer les options de certains programmes privilégiés (par exemple démons SSH ou IKE) permettant une réduction de privilèges [3] ou une séparation de privilèges [4].

De manière complémentaire, les mécanismes de contrôle d'accès obligatoires constituent un très bon moyen de limiter les privilèges de programmes. La difficulté de mise en œuvre de ces mécanismes réside souvent dans leur configuration. C'est particulièrement le cas de SELinux. Certaines alternatives moins complexes que SELinux sont alors envisageables (Tomoyo, AppArmor, ACL grsecurity, etc.).

L'utilisation des capacités (*capabilites*) permet aussi de réduire le pouvoir de nuisance de programmes privilégiés compromis. Ces capacités, qui consistent en une discrétisation des différents privilèges associés en général à root, peuvent être utilisées de deux manières distinctes. D'une part, pour les distributions qui le supportent, le mécanisme des capacités de fichiers (*file capabilities*) offre une alternative mieux maîtrisée à l'attribution de bit *setuid root* à des exécutables. D'autre part, certains programmes et démons exécutés par root offrent optionnellement (par une option de compilation ou de configuration selon le cas) la possibilité de réduire leurs privilèges au strict nécessaire - ces options doivent être activées lorsqu'elles sont supportées.

Enfin, il est important de limiter de manière générale les programmes réalisant une analyse (parsing) sur des données non maîtrisées de format complexe qu'elles soient issues du réseau ou locales appartenant à un utilisateur non privilégié. En effet, les fonctions de parsing associés à des formats complexes présentent fréquemment des vulnérabilités. À titre d'exemple, un navigateur web traite des données extrêmement complexes et, il est impossible de garantir l'absence de vulnérabilités dans les traitements effectués. Ainsi, un tel navigateur ne doit pas être mis en œuvre sur une machine autre qu'un poste client même sous une identité non privilégiée.

Au niveau du noyau :

Il est fortement recommandé de mettre en œuvre un noyau sur mesure vis-à-vis des périphériques matériels et des fonctions logiques dont aura besoin le système. Par exemple, un serveur n'aura probablement pas besoin de wifi, de bluetooth et de manière générale de toutes les piles protocolaires qui sont susceptibles de contenir des failles permettant à un attaquant d'obtenir des privilèges noyau. Un serveur n'aura pas non plus besoin de supporter plusieurs modèles de cartes réseau, de cartes graphiques, etc.

On préférera des noyaux non modulaires pour contrer des élévations de privilège via des modules malveillants. Faute de pouvoir désactiver complètement le support des modules, on bloquera leur insertion dans le noyau au terme du démarrage du système [5].

Cette dernière recommandation s'applique également dans les situations où la configuration d'un noyau sur mesure est trop complexe à mettre en œuvre, soit parce que le parc de machines est très hétérogène, soit parce qu'on ne dispose pas des moyens humains nécessaires au suivi des mises à jour du noyau. Un noyau de distribution générique à jour peut en effet être préférable à un noyau sur

3. Le programme révoque ses privilèges dès lors qu'il n'en a plus besoin.
4. Les traitements les plus complexes sont relégués à un programme esclave non privilégié.
5. écrire 1 dans `/proc/sys/kernel/modules_disabled`.

mesure sur lequel les correctifs de sécurité ne sont pas appliqués.

Enfin, la posture de méfiance à l'égard des fonctions de parsing, recommandée plus haut, s'applique à plus forte raison au niveau du noyau.

En complément, réaliser une défense en profondeur.

« Défense en profondeur » est également une autre règle qui s'applique à la configuration sécurisée d'un système : il convient d'éviter de satisfaire une exigence de sécurité par un seul mécanisme. Au contraire, il est recommandé de multiplier les barrières. Par exemple, la mise en place d'un pare-feu local en complément de la suppression des services inutiles est recommandée. Un tel pare-feu permettra aussi par exemple à contrôler les connexions initiées depuis un serveur (celui-ci a-t-il d'ailleurs vocation à ouvrir des connexions ?)

Il est fortement recommandé de partitionner un système de manière rationnelle. Un partitionnement bien réfléchi a plusieurs vertus :

- éviter la saturation. Il est à ce titre utile de placer sur une partition distincte tout répertoire susceptible d'être rempli à la suite d'actions extérieures (journaux, "spool" d'impression, file d'attente de messages ou base de donnée associée à un serveur web), en particulier lorsque le service susceptible de remplir le répertoire tourne sous l'identité *root* ;
- simplifier la sauvegarde ;
- appliquer des options de montages nécessaires et suffisantes (`ro`, `nodev`, `noexec` et `nosuid`).

Le dernier point est souvent ignoré et pourtant il est extrêmement important et permet de contrer facilement un grand nombre d'attaques. Par exemple, les supports de stockage amovibles doivent impérativement être montés en `nodev,noexec,nosuid` (c'est normalement le cas par défaut, mais il est important de le vérifier). Cette même combinaison peut éventuellement s'appliquer aussi à `/home` (les utilisateurs ont-ils besoin d'exécuter du code présent dans leur home ?). `/tmp` et `/var/tmp` n'ont généralement pas besoin de `exec` (il convient d'examiner scrupuleusement les éventuels programmes qui imposeraient de contrevenir à cette règle : cela dénote fréquemment une mauvaise conception logicielle). En outre, la grande majorité des montages se fait en `nodev` mais lorsqu'elles le supportent, les partitions doivent être montées de préférence en lecture seule.

Il est utile de noter que certaines distributions Linux créent automatiquement des répertoires et points de montage accessibles en écriture universelle et disposant de l'option `exec` (par exemple `/dev/shm`, `/run/shm`, `/run/lock` etc.). Il est souhaitable d'ajouter l'option "noexec" à ces montages selon les modalités propres à chaque distribution. En général, le rajout d'une ligne correspondant au montage dans `/etc/fstab` suffit.

En lien avec le partitionnement des supports de stockage, il y a lieu de s'interroger sur le chiffrement des données. Le chiffrement du disque, de préférence avec des moyens qualifiés par l'ANSSI est évidemment indispensable sur un poste client (a fortiori si celui-ci est utilisé en condition de nomadisme). La question se pose plus sur une machine faisant office de serveur. Tout dépend du niveau de sensibilité des données qu'il héberge et de la protection physique de l'équipement. Dans le cas de données sensibles, même si le serveur est stocké dans un bunker, le chiffrement permet de « se rassurer » pour le cas où les procédures de recyclage de disques s'avèrent laxistes (concrètement, si les disques ne sont pas détruits physiquement après usage), ce pour un surcoût relativement modeste. En revanche, dans de rares cas qu'il convient d'apprécier, le chiffrement peut présenter un risque pour la disponibilité (si l'administrateur perd les clés, par exemple).

Il est aussi fortement recommandé d'appliquer des correctifs (*patchs*) de durcissement du noyau lors de la recompilation de ce dernier. Il est par exemple ici question des protections génériques du sous-système de gestion de la mémoire servant à protéger le système contre des attaques qui exploitent des vulnérabilités de type corruption mémoire (PaX), mais aussi des options de durcissement diverses proposées par grsecurity (masquage des entrées de `/proc`, listes blanches d'utilisateurs autorisés à créer des sockets, durcissement de chroots, etc.).

Dans tous les cas, une analyse fine des droits à positionner sur les fichiers devra être effectuée. En effet, l'absence de règles peut laisser la porte ouverte à des élévations de privilèges ! Il est notamment recommandé de limiter autant que possible le nombre de fichiers systèmes sur lesquels l'on attribue des droits d'écriture à tous les utilisateurs. On peut aisément réaliser une revue des fichiers présentant cette caractéristique avec la commande `find /etc /usr /bin /sbin /var -type f -perm /o+w`, afin de s'assurer que les recours à cette configuration sont tous réellement nécessaires. Pour une analyse plus fine de la configuration des droits, il est également possible de mettre en oeuvre des scripts spécialisés, comme ceux issus du paquetage *Bastille Linux*, qui sont en mesure de détecter et éventuellement corriger un certain nombre d'erreurs classiques de configuration.

Cloisonner les différents services.

Pour accompagner ces mesures (exceptionnellement en substitut), « cloisonnement » et « isolation » sont également des aspects primordiaux de la configuration sécurisée, tout particulièrement dans le cas d'un serveur.

Il est ainsi fortement recommandé d'enfermer les services dans des environnements d'exécution isolés du reste du système (socle et autres services) afin de circonscrire au mieux les conséquences d'une intrusion. On privilégiera pour ce faire les solutions de type *lxc*, *vservers* ou *openvz* qui sont globalement plus robustes qu'un *chroot* et dont l'isolation ne se cantonne pas au système de fichiers.

Autres points de sécurisation !

Il faut évidemment sécuriser les modes de connexion et d'authentification au serveur, notamment pour les accès d'administration. On imposera par exemple l'utilisation de comptes non privilégiés et traçables [6] et un emploi correct de clés SSH, c'est à dire protégées par des passphrases dont la partie privée n'est pas copiée sur les serveurs facilitant ainsi des attaques par rebonds [7].

Comme déjà évoqué en préambule, les systèmes GNU/Linux n'échappent bien entendu pas aux thématiques de sécurité usuelles pour tous systèmes :

- procédure de mise à jour avec une attention particulière sur leur origine [8] (penser aux implications du durcissement du socle de base et du remontage de partitions en lecture/écriture, par exemple) ;
- génération et exploitation des journaux du système ;
- problématiques de sauvegarde ;
- supervision en temps réel, notamment pour les indicateurs de sécurité (échecs de connexion, utilisation du compte root, etc.) ;
- sécurité physique, notamment pour l'accès à la console locale.

6. et pas ssh root@server.
7. Dans le cas où les rebonds sont légitimes, on privilégiera le mécansime d'« *agent-forwarding* ».
8. La signature cryptographique des mises à jour permet par exemple de les authentifier.

Pour pousser la sécurisation un peu plus loin, il est possible d'utiliser certaines distributions qui proposent la recompilation systématique des logiciels pour maîtriser et durcir la chaîne de compilation des paquetages (imposer par exemple que les exécutables soient relocalisables (`CFLAGS="-fpie"`) pour que la randomisation d'adresse (ASLR) puisse être efficace, protéger le système contre les buffer overflow dans la pile (`CFLAGS="-fstack-protector-all"`), le durcir de manière générique (`CFLAGS="-D_FORTIFY_SOURCE=2"`) ou encore sécuriser le chargement dynamique (`LDFLAGS="-Wl,-z, relro -Wl,-z, now"`)). Cela nécessite toutefois un suivi et une organisation rigoureuse. Une alternative moins lourde peut consister à ne recompiler localement que certains paquetages d'une distribution classique, par exemple ceux correspondant à des démons réseau exposés à l'extérieur. À défaut, on privilégiera les distributions réputées qui proposent une politique de mise à jour de sécurité claire et sérieuse sur les distributions plus confidentielles. Il est en effet souhaitable de réduire le plus possible la fenêtre temporelle entre la découverte d'une faille dans un logiciel et le déploiement du correctifs dans la distribution considérée.

En synthèse, les recommandations minimales à respecter :

A minima, l'ANSSI estime que les 5 recommandations suivantes doivent être appliquées. Lorsque leur suivi n'est pas possible, il doit être justifié et les risques induits assumés.

R1	Réduire la surface d'attaque en : – mettant à jour régulièrement le système ; – supprimant les logiciels inutiles ; – limitant au juste besoin les services en écoute pour chaque interface réseau ; – désactivant le chargement dynamique des modules ; – appliquant le principe de moindre privilège aux logiciels ; – utilisant un mécanisme de contrôle d'accès reposant sur les rôles (Tomoyo, AppArmor, ACL grsecurity) ; – adaptant le noyau (suppression des éléments inutiles, application des patchs de durcissements) au contexte d'emploi envisagé.

Note : Certaines mesures nécessitent une certaine expertise technique et leur mise en œuvre peut impacter la disponibilité du système qu'il convient de bien apprécier au préalable.

R2	Appliquer le principe de défense en profondeur en : – utilisant un pare-feu local ; – implémentant un schéma de partitionnement et les options de montage adaptés au contexte d'emploi ; – positionnant les droits en lecture, écriture et exécution sur les fichiers après une analyse fine des besoins des utilisateurs mais aussi des services locaux. Une attention particulière sera portée sur les scripts lancés automatiquement ; – chiffrant les données.

Note : Là aussi, il convient de bien apprécier l'impact de ces mesures avant de les mettre en œuvre.

R3	Mettre en place des mesures de cloisonnement pour isoler les différents éléments applicatifs.

R4	Rédiger et appliquer les procédures d'administration sécurisées (enregistrement des données de connexion des exploitants, supervision, mises à jour, sauvegardes, etc.).

R5	Définir et mettre en place une politique de journalisation d'événements cohérente : – distinguant différents niveaux de criticité selon leur impact sur la sécurité du système ; – permettant la remontée d'alertes en fonction de ces niveaux.