24 août 2008

SELinux, l'agence de sécurité du noyau

Catégorie : Sécurité Tags : lmhs

~~Retrouvez cet article dans :~~ Linux Magazine Hors série 17

SELinux ou Security Enhanced Linux est une solution de sécurité développée par la NSA qui s'intègre au noyau linux.

Généralités

SELinux est une solution de sécurité offrant un blindage du noyau Linux, en lui apportant des fonctionnalités de protection poussées supplémentaires comme le Mandatory Access Control, etc., que nous allons détailler dans l'article (cf Architecture).
Ce projet correspond à une initiative américaine et a été lancé par la NSA : National Security Agency. Cette agence de renseignement qui compte près de 40000 personnes est assez connue du grand public depuis les révélations concernant le projet Echelon, réseau planétaire de surveillance et d'interception de communications [Assemblée2000]. La NSA s'appuya initialement sur des piliers industriels comme MITRE, Secure Computing Corporation et NAI, avec une participation du tissu universitaire américain.
La première mouture publique de SELinux a été donnée à la communauté du logiciel libre (licence GPL) vers la fin 2000 par le Information Assurance Research Office qui pilote le projet. Cette entité de la NSA demeure responsable de la recherche et du développement de solutions de sécurité dans le domaine des technologies de l'information, pour des infrastructures sensibles avec

une vocation gouvernementale et industrielle (projet à prendre au sérieux, donc).
Cette première version de SELinux était fournie sous forme de patch du noyau Linux et lança une impulsion significative (supplémentaire) concernant la sécurité bas niveau du fameux kernel. La communauté Linux (et Linus Torvalds en particulier) décida alors de créer un framework de sécurité dans le noyau qui donna plus tard naissance à Linux Security Module (voir l'article dédié dans ce hors série). Avec NAI, la NSA changea alors son implémentation de SELinux pour passer d'un patch à un simple module noyau comme nous le verrons plus loin dans l'article.
En effet, nous commencerons par introduire l'architecture globale de SELinux (Flask...) et son intégration au noyau existant (LSM...), pour ensuite détailler son fonctionnement interne et ce qui façonne ses atouts (TE, RBAC) avant d'aborder rapidement des aspects pratiques comme l'installation ou la configuration. Nous conclurons enfin sur ce que cela apporte concrètement en sécurité, les limites, et terminerons sur le potentiel futur de ce projet.

Architecture

SELinux a été conçu sur le modèle de l'architecture Flask [Spencer99]. Au début, l'implémentation avait été faite sous forme de patch pour le noyau Linux. Par la suite, avec la création de l'architecture Linux Security Module (LSM), SELinux a progressivement évolué vers un module de sécurité utilisant les nouvelles fonctions du noyau. Remarquons au passage que l'implémentation de SELinux sous forme de module a donné lieu à une énorme contribution au projet LSM [Smalley01].

Mandatory Access Control

La protection apportée par SELinux est fondée sur le principe de Mandatory Access Control (MAC), en français : contrôle d'accès obligatoire. Contrairement au contrôle d'accès typique d'un système Unix, Discretionary Access Control (DAC), qui laisse le problème du contrôle d'accès à une ressource à la discrétion de son propriétaire, le but de MAC est de vérifier la légitimité de tous les accès par des sujets (processus) à des objets (fichiers, sockets...), et ceci à partir d'une politique de sécurité définie au préalable (cf [HSLM16/LSM].
Cependant, le MAC, que l'on retrouve dans de nombreuses solutions de sécurité, n'est pas l'apport principal de SELinux. En fait, ce qui caractérise vraiment la réalisation de la NSA est l'architecture Flask, déployée dans le noyau pour implémenter le MAC.
Nous allons maintenant voir plus en détail l'architecture **Flask**.

Flask

Flask est un modèle très flexible autorisant l'implémentation de différents modèles de sécurité. Il repose sur une séparation complète entre le code de décision et le code d'application de la politique de sécurité. Par conséquent, SELinux constitue une sorte de gouvernement où le pouvoir

législatif et le pouvoir exécutif sont détenus par des entités distinctes.
La partie décisionnelle de l'architecture (policy decision-making code) de SELinux est contenue dans un composant particulier du noyau appelé le security server (en français, référentiel de sécurité). Cette partie constitue le "pouvoir législatif" dans SELinux. Elle établit les lois à partir de la politique de sécurité à appliquer sur le système.
La partie d'application (policy enforcement code) de la politique de sécurité est directement intégrée dans les composants du noyau gérant les processus, les systèmes de fichiers, les IPC et le réseau. C'est le "pouvoir exécutif" de SELinux, responsable de l'application des lois dans le système.
Enfin, SELinux comporte également un système de cache, pour un accès rapide aux décisions d'accès qui ont déjà été calculées par le référentiel de sécurité. Il s'agit de l'Access Vector Cache (AVC).
Pour que ce modèle soit indépendant de la politique de sécurité à appliquer, les interactions entre sujets et objets sont modélisées. Chaque entité du système reçoit un ~~label~~, qui est en fait un contexte de sécurité. Celui-ci va contenir des informations sur ce qu'est cette entité, et permettra de déterminer avec quelles autres entités elle peut interagir et comment. En outre, chaque contexte de sécurité est associé à un entier appelé ~~SID~~ (Security IDentifier ou identifiant de sécurité), pour simplifier leur manipulation. Les significations des labels et SID sont connues uniquement par le référentiel de sécurité, mais c'est la partie "enforcement" qui va attribuer les SID aux entités.
Les objets du système ont également une ~~classe~~, par exemple "file" ou "socket", à laquelle sont associées différentes permissions, comme "open" ou "link" pour un fichier, ou "listen" et "send" pour un socket.
Les décisions prises par le référentiel de sécurité reposent sur les contextes de sécurité attribués au couple (sujet, objet) en interaction, et sur la classe de l'objet. Elles sont de deux types : accès et transitions. Toutes ces décisions sont calculées en fonction de la politique appliquée sur le système (cf Modèles de sécurité).
Les décisions d'accès (access decisions) spécifient si une certaine permission est accordée ou refusée pour le couple de SID et la classe de l'objet cible. Les SID sont appelés SID source (celui du sujet) et SID cible (celui de l'objet). Concrètement, le référentiel de sécurité fournit un tableau de l'ensemble des permissions associées à la classe de l'objet, pour le couple (SID source, SID cible). Ce tableau est enregistré dans l'AVC, puis la décision d'accès est transmise par l'AVC à la partie "enforcement" pour être appliquée. Le référentiel de sécurité fournit également deux tableaux spécifiant les besoins de journalisation (audit) des permissions associées à la classe d'objet. Un des tableaux (auditallow) concerne l'audit en cas de succès, l'autre (auditdeny) concerne les cas d'échec.
Les décisions de transition (labeling decisions) attribuent des labels aux nouveaux objets. Une décision de transition de processus est faite lors de l'exécution d'un nouveau programme, à partir du SID du processus demandant l'exécution et du SID du fichier à exécuter. Une décision de transition d'objet est faite lors de la création d'un nouvel objet, à partir du SID du programme créant l'objet et du SID d'un objet "parent", typiquement le dossier parent s'il s'agit d'un fichier. Par ailleurs, une application peut spécifier directement le label attribué à un nouvel objet, en utilisant des fonctions de SELinux. Par exemple, un utilisateur qui se connecte par SSH va recevoir un SID qui lui correspond, et non pas le SID par défaut. Il est important de noter, premièrement, que l'application doit être autorisée par SELinux à modifier le contexte de sécurité, et d'autre part, qu'une telle transition doit être autorisée par une décision d'accès correspondante (cf Modèles de sécurité).
Le cas des fichiers enregistrés sur des mémoires de masse (disques durs, etc.) est particulier. En effet, ceux-ci possèdent des SID persistants, ou PSID, avec un format identique aux SID normaux. Ces ~~PSID~~ sont enregistrés dans le système de fichiers, et ne sont donc pas perdus lors d'un

redémarrage du système.

Modèles de sécurité

SELinux offre la possibilité d'utiliser différents modèles de sécurité. Suivant les modèles utilisés, les possibilités seront différentes, mais les principes de fonctionnement restent les mêmes, comme décrits dans la partie précédente. La politique de sécurité que l'on va pouvoir définir à partir de ces différents modèles est destinée à piloter le mécanisme de Mandatory Access Control. Pour plus d'informations, la lecture de [Amoroso94] est une bonne base.
Actuellement, le référentiel de sécurité implémenté dans SELinux est une combinaison de Type Enforcement et de Role-Based Access Control. Une implémentation de Multi-Level Security est en cours, ainsi qu'une implémentation de la norme CIPSO/FIPS188 pour l'application de labels au trafic réseau.
Comme expliqué dans la partie précédente, afin de pouvoir modéliser les interactions entre sujets et objets de façon efficace, SELinux attribue des labels (contextes de sécurité) aux entités du système. Il va s'agir d'une combinaison des identifiants déclarés par les différents modèles de sécurité. Par exemple : ~~moutane:sysadm_r:sysadm_t~~ est le label d'un processus lancé par l'utilisateur moutane lorsqu'il agit en tant qu'administrateur système. La signification des différents éléments est expliquée dans la suite de cette partie. Ce label sera associé arbitrairement à un entier appelé SID lors de la compilation de la politique de sécurité.

Type Enforcement

Le modèle de sécurité Type Enforcement (TE) implémenté dans SELinux est dérivé du modèle Domain and Type Enforcement. Dans DTE, tous les objets et processus du système reçoivent des attributs de sécurité. Ils sont appelés ~~domaines~~ pour les processus, et ~~types~~ pour les objets. Les domaines et types correspondent à des classes d'équivalence, les processus d'un même domaine et les objets d'un même type étant traités de façon identique. Ce modèle permet donc un contrôle très fin des exécutions et des transitions de domaines.
Le modèle TE diffère de l'original par le fait qu'il n'a qu'un seul type d'attributs de sécurité. Les domaines sont en fait des types qui peuvent être associés à des processus. Ainsi, en interne, SELinux ne fait pas de différence entre domaines et types.
Voici par exemple deux déclarations de types : la première concerne un processus, et la seconde concerne un fichier.

```
attribute domain;
attribute file_type;
attribute sysadmfile;
type syslogd_t, domain;
type resolv_conf_t, file_type, sysadmfile;
```

Après le mot clé type, le premier identifiant est le type déclaré, puis les identifiants suivants sont des attributs associés au type (on peut déclarer autant d'~~attributs~~ que l'on veut). On construit ainsi des ensembles de types. Par exemple, l'attribut domain va être associé à tous les types de programmes. L'administrateur de sécurité peut alors distinguer domaines et types même si SELinux ne fait pas cette distinction.
Une autre différence de SELinux par rapport au modèle original est la notion de classe d'objet, qui provient de l'architecture Flask. Deux objets ayant le même type, mais des classes différentes, pourront être traités différemment. Par exemple, on pourra traiter différemment un socket TCP et un socket Raw IP.

Enfin, le modèle TE n'associe pas directement les utilisateurs à des domaines, mais plutôt à des rôles, comme expliqué dans la partie suivante.
TE permet de définir des lois d'accès et des lois de transition. Si aucune loi n'existe pour une demande d'accès donnée, celle-ci est refusée. Quant aux lois de transition, les comportements par défaut sont la conservation du domaine pour les processus, et l'héritage du type du parent pour les objets.

```
type syslogd_t, domain;
type syslogd_exec_t, file_type, sysadmfile, exec_type;
domain_auto_trans(initrc_t, syslogd_exec_t, syslogd_t)
allow syslogd_t self:process { fork signal };
```

Voici quelques lignes tirées de la configuration de ~~syslogd~~. En résumé, ~~syslogd_t~~ est le type associé au processus, ~~syslogd_exec_~~t est le type associé au fichier exécutable. La macro ~~domain_auto_trans~~ utilisée ici spécifie qu'un programme de type ~~initrc_t~~, typiquement un script de ~~/etc/rc.d~~, peut exécuter le programme ~~syslogd~~ (lois d'accès) et qu'alors celui-ci prend le type ~~syslogd_t~~. Enfin, la ligne débutant par ~~allow~~ est l'expression directe d'une loi d'accès, spécifiant ici que le programme a accès aux appels système ~~fork~~ et ~~signal~~, et ce sur toute entité possédant un label identique (~~self~~).

Role-Based Access Control

Le modèle original de Role-Based Access Control (RBAC) assigne des rôles aux utilisateurs, et associe des permissions à ces rôles. Le modèle de SELinux associe à chaque utilisateur un ensemble de rôles, et associe à chaque rôle un ensemble de domaines TE. SELinux combine ainsi les facilités de gestion du RBAC, et la finesse des contrôles d□accès du modèle TE. Par exemple, la règle suivante indique que le rôle ~~system_r~~ (le rôle des processus du système) peut manipuler des objets de type

```
syslogd_t.
role system_r types syslogd_t;
```

Les rôles sont inclus dans les contextes de sécurité associés aux sujets et objets du système. Pour les objets, il existe un rôle générique ~~object_r~~. Pour les processus, la configuration de RBAC dans SELinux permet de définir des règles de transition de rôles reposant sur les rôles et les types. Il est préférable de limiter les possibilités de changement de rôle. En fait, la première des deux règles suivantes spécifie que seuls les domaines TE ayant l□attribut ~~privrole~~ peuvent effectuer cette opération. La seconde autorise la transition de ~~system_r~~ vers ~~sysadm_r~~.

```
constrain process transition ( r1 == r2 or t1 == privrole );
allow system_r sysadm_r;
SELinux User Identity
```

Les UID traditionnelles de Linux ne conviennent pas pour SELinux. En effet, un même utilisateur peut utiliser plusieurs UID au cours de son activité sur le système, par exemple lorsqu□il utilise des programmes nécessitant des privilèges. Un programme lancé par root peut prendre n□importe quelle UID via les fonctions set*uid. SELinux introduit donc un attribut reflétant l□identité de l□utilisateur dans le contexte de sécurité, complètement indépendant de l□UID. Il est ainsi possible pour SELinux d□effectuer des contrôles sur l□identité de l□utilisateur sans LMHS17.qxd 29/09/03 14:47 Page 40 Novembre/Decembre 2003 41 perturber le système de contrôle d□accès discret (DAC) de Linux. Dans la politique de sécurité standard de SELinux, seuls certains domaines ont la possibilité de modifier l□identité de l□utilisateur. Par exemple,

les programmes login et sshd ont été modifiés afin d□utiliser les fonctions des bibliothèques fournies avec SELinux. Ainsi, les identités appropriées sont attribuées aux utilisateurs qui se connectent sur le système. En outre, un utilisateur donné conservera en permanence son identité SELinux, même s□il utilise une commande telle que su, on obtient ainsi une meilleure traçabilité.

```
user root roles { user_r sysadm_r };
user moutane roles { user_r sysadm_r };
user lolo roles { user_r };
constrain process transition ( u1 == u2 or t1 == privuser );
```

Voici quelques déclarations d'utilisateurs avec les rôles qu'ils peuvent prendre. La dernière ligne se trouve dans le fichier ~~constraints~~ et limite la possibilité de changer l'utilisateur aux domaines ayant l'attribut ~~privuser~~.

Multi-Level Security

Le modèle de sécurité MLS est parmi les plus proches du modèle Bell-LaPadula [Bell73]. Il attribue des niveaux de confidentialité aux différents objets et utilisateurs du système. Assez proche de la notion du besoin d'en connaître du milieu militaire, ce modèle était par exemple implémenté dans certains systèmes Unix, notamment des systèmes d'exploitation Cray.
L'implémentation SELinux n'est pas encore terminée, mais il est déjà possible d'associer des niveaux de confidentialité à toutes les entités du système. Ensuite, le contrôle d'accès est simple : on n'a accès en lecture à un objet d'un certain niveau de classification que si l'on possède soi-même un niveau supérieur ou égal; et l'on n'a le droit de modifier un objet que si l'on possède un niveau égal.

```
sensitivity unclassified alias u;
sensitivity confidential alias c;
sensitivity secret alias s;
sensitivity top_secret alias ts;
dominance { u c s ts }
```

L'exemple donné ici est la déclaration des différents niveaux de classification utilisés dans MLS, avec la hiérarchie associée.

Labeled Networking Support

SELinux comporte également un modèle de trafic réseau labellisé. Il repose sur une nouvelle option de l'en-tête IP décrite dans CIPSO/FIPS-188. L'objectif est de pouvoir associer les paquets IP à des SID, qui pourront être décodés par le destinataire du trafic. Ce décodage est fait par le protocole Security Context Mapping Protocol (SCMP).
Ce modèle peut être mis en place sur un parc de machines où les politiques SELinux sont équivalentes, i.e. les utilisateurs, les types, les rôles et les niveaux MLS sont les mêmes sur chaque machine. Ce parc de machines, aussi appelé "périmètre de sécurité", doit être déclaré dans la configuration de la politique. Les SID sont alors transmis de façon transparente à l'intérieur de ce périmètre.
En revanche, ce modèle ne définit aucune protection du trafic. Elle est pourtant nécessaire, et doit être apportée par des moyens annexes, afin de garantir confidentialité, intégrité, imputabilité et protection contre le rejeu.

Intégration de SELinux...

L'intégration de SELinux dans une distribution GNU/Linux ne se fait pas seulement au niveau du noyau. Comme mentionné dans la partie sur RBAC, plusieurs démons et utilitaires doivent être modifiés afin de prendre en compte l'environnement SELinux.

... dans le noyau

A l'intérieur du répertoire contenant les sources du noyau (souvent /usr/src/linux), on trouve le dossier security contenant les modules utilisant l'architecture LSM. Le code spécifique à SELinux se trouve dans security/selinux : dans le sous-dossier ss, on trouve la partie "Security Server"; le dossier selopt contient l'implémentation de la norme CIPSO/FIPS188.
La partie "enforcement" de SELinux est assurée par les ancres (hooks) fournis par l'infrastructure Linux Security Modules. SELinux implémente l'ensemble des ancres, et les fonctions correspondantes sont définies dans le fichier hooks.c. On se reportera à [HSLM16/LSM] pour tout ce qui concerne LSM.

... dans les démons et utilitaires

SELinux comporte une bibliothèque, libsecure, permettant aux programmes normaux d'utiliser facilement les fonctions de communication avec l'environnement SELinux. En effet, SELinux se trouve dans l'espace noyau, et nécessite donc des mécanismes spéciaux fournis par LSM pour interagir avec les processus standards. Actuellement, cette communication se fait par des entrées spécifiques dans /proc.
Deux catégories de programmes utilisent cette bibliothèque : des démons et utilitaires classiques modifiés (sshd, login, ls, ps...), et de nouveaux utilitaires fournis avec SELinux (runas, avc_toggle, checkpolicy, setfiles...). On se reportera à la partie "Outils et configuration" pour plus de détails.

Guide d'installation

Dans cette partie, seules les étapes les plus significatives de l'installation seront détaillées. Le fichier README de l'archive SELinux est un très bon guide. Les chemins cités sont relatifs au dossier d'extraction de l'archive SELinux.
Pour commencer, il faut télécharger l'archive contenant les sources de SELinux sur le site de la NSA [SELinux]. Plusieurs options sont possibles ; on choisit ici de télécharger séparément les sources du noyau Linux 2.4.21, le patch lsm-2.4 et l'archive selinux.
La première étape importante est l'application des patches du noyau. Le patch lsm-2.4 va insérer les ancres de l'infrastructure Linux Security Module dans les sources du noyau. L'intégration définitive de ces ancres devrait se faire dans le noyau 2.6. Il est ensuite nécessaire d'appliquer un second patch spécifique à SELinux, pour ajouter certaines ancres absentes du patch lsm-2.4, ainsi que les options par défaut de SELinux dans la configuration du noyau. Ce patch devrait disparaître avec le noyau 2.6.
Une décision importante lors de la compilation du noyau SELinux est celle qui concerne l'option "NSA SELinux Development Support". Elle permet de faire fonctionner SELinux dans un mode permissive, dans lequel aucune action n'est bloquée par SELinux, mais les actions interdites sont enregistrées. Dans ce mode, il est possible de tester des politiques de sécurité sans que les

applications ne soient bloquées à cause d'une erreur. Il est possible de passer en mode enforcing par la commande avc_toggle et vice-versa. Si le noyau est destiné à une machine de développement pour SELinux, il vaut mieux activer cette option. Si le noyau doit être utilisé sur une machine de production, alors il est fortement conseillé de ne pas l'activer.
Après la compilation du noyau SELinux, une autre étape importante est la compilation et l'installation de la politique de sécurité. Au préalable, il est souhaitable de modifier le fichier policy/users contenant les rôles associés à chaque utilisateur, en particulier retirer les exemples d'utilisateurs. Il faut également désactiver les gestionnaires de session tels que gdm, kdm ou xdm, qui n'ont pas encore été modifiés pour utiliser l'environnement SELinux. Ensuite, dans le dossier policy, la commande make install-src install va installer les sources et la version compilée de la politique de sécurité dans le dossier /etc/security/selinux.
Le fichier policy.conf est la version texte de la politique de sécurité, obtenue à partir des fichiers du dossier policy. Les fichiers .te contiennent les règles de Type Enforcement, en rapport avec le nom du fichier (par exemple, admin.te contient des règles spécifiques à l'administrateur SELinux). Ensuite, le fichier policy.conf est compilé par la commande checkpolicy -o policy.version policy.conf, la version du langage de configuration de la politique étant obtenue avec checkpolicy -V (actuellement policy.12).
L'étape suivante est la compilation de libsecure, la bibliothèque contenant les fonctions d'interaction avec l'environnement SELinux, puis la compilation des démons et utilitaires modifiés. Ensuite vient la compilation optionnelle des outils spécifiques à la configuration de SELinux (dont il faudra avoir ajouté les fichiers de configuration dans la politique de sécurité). Ces outils permettent de visualiser et de modifier la politique de sécurité de façon légèrement simplifiée, mais il ne faut pas s'attendre à quelque chose de miraculeux.
Remarquons également l'installation de l'outil setfiles, très important puisqu'il met en place les PSID pour les fichiers. La mise en place de ces PSID est faite, dans le dossier policy, par la commande make reset, qui va en fait exécuter la commande setfiles -R en passant en paramètres la configuration des contextes de sécurité des fichiers et les points de montage des différents systèmes de fichiers. Les contextes de sécurité des fichiers sont contenus dans les fichiers .fc. Les PSID sont stockés dans le dossier ...security à la racine de chaque système de fichier. Les formats supportés actuellement sont Ext2, Ext3 et ReiserFS.

Après la création des labels des fichiers, il est enfin possible de relancer le système. Ne pas oublier d□ajouter le noyau SELinux dans la configuration de son boot manager, sans effacer les anciens !

Configuration et outils

Cette partie va décrire rapidement comment configurer les contextes de sécurité des fichiers et modifier la politique de sécurité, puis quels sont les utilitaires spécifiques à SELinux et leurs intérêts.
Comment modifier la politique de sécurité
Les fichiers de configuration se trouvent dans le sous-dossier policy du dossier de travail pour l'installation. Le plus simple actuellement est de modifier les fichiers directement dans ce dossier, puis d'utiliser le Makefile pour appliquer les changement. Le but de cette partie n'est pas d'expliquer la grammaire de la configuration de SELinux, mais seulement d'indiquer des étapes importantes dans la modification de cette configuration. Pour plus de détails sur les étapes, on pourra se reporter au fichier README. Le document [PolicyConf] contenu dans le dossier doc/policy explique en détail la grammaire de définition de la politique de sécurité.
Pour modifier ou ajouter la politique associée à une application, deux étapes sont nécessaires. Il

faut premièrement modifier les contextes des fichiers. Par convention, ces contextes sont placés dans un fichier application.fc situé dans le dossier file_contexts/program. On peut trouver des fichiers pour des applications non installées, ce n'est pas grave car le Makefile vérifie que chaque fichier .fc a bien un fichier .te correspondant, et la configuration des applications installées se fait donc plutôt à ce niveau-là.
Une fois que tous les contextes adéquats ont été ajoutés, on peut éditer les règles de Type Enforcement associées au programme. Le fichier concerné est application.te dans le dossier domains/program. Le sous-dossier unused contient des fichiers de configuration pour des programmes non installés, et il suffit de déplacer un fichier de configuration de ce dossier dans le dossier parent pour ajouter dans la politique de sécurité l'application correspondante. Bien sûr, si l'application que l'on veut ajouter n'a pas de configurarion disponible, il faudra l'écrire.
Concernant la déclaration des types pour les règles TE, il est intéressant de remarquer que le dossier types contient des types associés aux objets du système (fichiers, socket...) alors que le dossier domains contient plutôt les types associés aux processus. De plus, le dossier macroscontient toutes les macros que l□on peut utiliser dans les fichiers de configuration.

Les outils de configuration

Dans le dossier d'installation, le dossier tools/setools contient des applications dédiées à la manipulation de la politique de sécurité : apol, sepcut et seuser. Tous ces utilitaires sont basés sur Tcl/Tk, il faudra donc en disposer pour pouvoir les compiler et les utiliser.

apol est un outil d□analyse de la politique de sécurité. Il lit le fichier policy.conf, et permet de faire des recherches sur les types, les attributs, les rôles et les règles TE. sepcut sert à modifier et à tester des politiques de sécurité. Il suffit de lui faire ouvrir le dossier contenant les sources de la politique, et on peut alors lire et éditer tous les fichiers, puis charger la politique de sécurité modifiée pour faire des tests. Cependant, il ne mâche pas le travail : il faut savoir exactement ce qu□on fait lorsqu□on modifie un fichier. seuser est un gestionnaire d□utilisateurs pour SELinux, avec des fonctions d□ajout, de modification et de suppression d□utilisateurs dans la politique en cours d□utilisation. Il existe également trois outils en ligne de commande : seuseradd, seusermod, et seuserdel.

Enfin, on remarquera dans le dossier scripts la commande newrules.pl, qui peut générer un ensemble de règles à partir des logs générés par SELinux. Les logs de SELinux sont générés dans le noyau et remontés par klogd, ils sont donc situés dans le fichier de log destiné à recevoir les messages du noyau.

Les utilitaires classiques

Les utilitaires installés avec SELinux sont de deux sortes : les utilitaires classiques modifiés et les utilitaires spécifiques.
De nombreux utilitaires ont été modifiés pour afficher les informations relatives à SELinux. Voici une liste non exhaustive, mais déjà relativement longue : ps, ls, id, tar, stat, strace, find, logrotate, cron. Le but est d'afficher les SID ou les contextes de sécurité grâce à de nouvelles options.
Voici des exemples d'exécution de commandes modifiées :

```
moutane@node1:~% /usr/local/selinux/bin/ls □context /
drwxr-xr-x root bin system_u:object_r:bin_t bin
drwxr-xr-x root root system_u:object_r:boot_t boot
drwxr-xr-x root root system_u:object_r:device_t dev
```

```
drwxr-xr-x root root system_u:object_r:etc_t etc
drwxr-xr-x root root system_u:object_r:home_root_t home
drwxr-xr-x root root system_u:object_r:lib_t lib
drwxr-xr-x root root system_u:object_r:file_t mnt
dr-xr-xr-x root root system_u:object_r:proc_t proc
drwx--x--- root root system_u:object_r:sysadm_home_dir_t root
drwxr-xr-x root bin system_u:object_r:sbin_t sbin
drwxrwxrwt root root system_u:object_r:tmp_t tmp
drwxr-xr-x root root system_u:object_r:usr_t usr
drwxr-xr-x root root system_u:object_r:var_t var
moutane@node1:~% /usr/local/selinux/bin/ps ax □context
PID SID CONTEXT COMMAND
1 7 system_u:system_r:init_t init
2 1 system_u:system_r:kernel_t [keventd]
3 1 system_u:system_r:kernel_t [ksoftirqd_CPU0]
4 1 system_u:system_r:kernel_t [kswapd]
5 1 system_u:system_r:kernel_t [bdflush]
6 1 system_u:system_r:kernel_t [kupdated]
78 207 system_u:system_r:syslogd_t /usr/sbin/syslogd
83 208 system_u:system_r:klogd_t /usr/sbin/klogd -c 3 -x
85 210 system_u:system_r:inetd_t /usr/sbin/inetd
88 212 system_u:system_r:sshd_t /usr/sbin/sshd
112 217 system_u:system_r:local_login_t login □ moutane
113 216 system_u:system_r:getty_t /sbin/agetty 38400 tty1 linux
118 218 moutane:sysadm_r:sysadm_t -zsh
138 218 moutane:sysadm_r:sysadm_t -su
155 221 moutane:sysadm_r:honeyd_t /usr/local/bin/honeyd -f
config.default
moutane@node1:~% id
uid=1000(moutane) gid=1000(moutane) groups=1000(moutane)
context=moutane:sysadm_r:sysadm_t sid=218
moutane@node1:~% su -
Password:
root@node1:~# id
uid=0(root) gid=0(root) groups=0(root),10(wheel)
context=moutane:sysadm_r:sysadm_t sid=218
```

La commande ls est lancée ici avec le modificateur context. En plus des informations classiques telles que les permissions, UID et GID, on obtient les contextes de sécurité des objets. Par exemple, le dossier ~~/root~~ possède le contexte de sécurité ~~system_u:object_r:sysadm_home_dir_t~~, avec la signification suivante : il est associé à l'utilisateur ~~SELinux system_u~~, qui représente le système d'exploitation ; il a le rôle ~~object_r~~, rôle générique des objets du système ; enfin il a le type ~~sysadm_home_dir_t~~, qui caractérise le dossier de l'administrateur. Ce contexte de sécurité est défini dans le fichier ~~types.fc~~ de la politique de sécurité.
La commande ~~ps~~ lancée ici avec le modificateur ~~context~~ affiche les contextes de sécurité ainsi que les SID pour chaque processus.
L'exemple de la commande ~~id~~ montre que même si ~~moutane~~ devient ~~root~~ avec la commande su, il reste moutane pour ~~SELinux~~.
De nouveaux utilitaires font également leur apparition. Certains sont très simples : ~~runas~~ va lancer une commande sous un certain contexte de sécurité, sous réserve d'accord dans la politique de sécurité. ~~run_init~~ peut lancer une commande dans le contexte de sécurité du programme init. newrole lance un shell sous un nouveau contexte de sécurité. ~~spasswd~~ est l'utilitaire ~~passwd~~ modifié pour SELinux, pour tenir compte du contexte particulier du fichier ~~/etc/shadow~~.
Les utilitaires de manipulation de la politique de sécurité sont très particuliers, et la plupart des opérations requiert le rôle ~~sysadm_r. checkpolicy~~ compile la politique de sécurité, et ~~load_policy~~ la charge. ~~setfiles~~ peut modifier les labels persistants des fichiers, avec deux modes : en temps normal, la commande déduit les SID des contextes de sécurité en faisant appel aux fonctions du noyau; mais lorsque l'on doit créer des labels sans que l'environnement SELinux ne soit chargé,

par exemple lors de l'installation, il faut utiliser setfiles -R. Enfin, lorsqu'on a compilé SELinux en mode "Development", la commande ~~avc_toggle~~ fait passer le système du mode ~~permissive~~ au mode ~~enforcing~~ et vice-versa. La commande ~~avc_enforcing~~ retourne le mode dans lequel se trouve SELinux.

Protection apportée

Sans détailler tous les aspects techniques présentant ce qu'apporte SELinux en sécurité, nous allons succinctement montrer en quoi une machine avec un tel système d'exploitation pourra résister face à la plupart des agressions connues.
Voici une question qui revient souvent quand on compare SELinux et d'autres solutions comme Grsecurity utilisant PAX : est-ce que SELinux bloque les buffers overflows et gêne les pirates avec des aléas dans les adresses de fonctions référencées dans des bibliothèques dynamiques, etc. ? La réponse est négative. En effet, si SELinux ne bloque ou ne gère certaines possibilités utilisées par les agresseurs, il faut comprendre que le concept est différent : une fois l'attaque lancée vers un programme contenant des failles, elle sera limitée à ce que le programme est capable de faire. On a donc une assurance du principe dit de containment permettant de contenir un intrus ayant réussi à percer certaines barrières qui ne dépendent pas de SELinux directement.
L'exemple donné dans [SYSADMIN03] rappelle ainsi que si un service Apache qui tournerait en root se faisait pirater sur un serveur SELinux (via buffer overflow, etc.), alors ce dernier ne pourrait modifier par exemple la configuration du serveur Bind local à ce serveur. En effet, le Security Server de SELinux, utilisant les politiques de sécurité qui ont été chargées dans le noyau, interdira au processus Apache agressé d'agir autrement que ce qu'on lui a imposé. Usuellement, lorsqu'un pirate rentre sur une machine à distance, il essaie de lancer un shell pour pouvoir visiter à sa guise sa nouvelle conquête. Avec SELinux, l'appel à ~~execl("/bin/sh"...)~~ devra être autorisé à un processus pour que ce dernier puisse lancer un shell. Typiquement, on va associer un type ~~shell_exec_t~~ aux programmes de type shell et n'autoriser leur exécution qu'aux processus qui ont le besoin, comme les programmes de login.
Par conséquent, pour assurer la sécurité, il suffit de configurer SELinux en respectant le principe de moindre privilège : les politiques de sécurité doivent donner le minimum nécessaire aux processus et utilisateurs pour qu'ils puissent effectuer les actions auxquelles ils peuvent normalement prétendre. En cas d'intrusion à cause d'un problème applicatif, le noyau pourra contenir l'attaque. Et quand bien même Apache aurait besoin de lancer un shell (par exemple pour des scripts CGI), SELinux peut autoriser des changements de contexte pour ajuster au mieux les privilèges nécessaires en fonction des rôles et actions menées.
En guise d'exemple, nous avons étudié le démon honeyd (cf [MISC8]) sur un SELinux (sur une Slackware). Il s'agit d'un démon jouant le rôle de pot à miel en répondant à des requêtes réseau pour simuler des services sur lesquels les pirates perdront leur temps et montreront leurs techniques. Honeyd est un démon Unix classique qui lit des fichiers de configuration, écoute le réseau, gère des requêtes/réponses réseau, lance des sous-programmes et écrit des traces de sécurité.
Concernant la "labellisation" des fichiers utilisés spécifiquement par honeyd, nous avons donc défini des contextes de fichiers suivants (honeyd.fc) :

```
# [attention on rappelle que ceci est montré pour illustration]
# Le binaire honeyd
/usr/local/bin/honeyd system_u:object_r:honeyd_exec_t
# Les fichiers de configuration de honeyd
/usr/local/share/honeyd system_u:object_r:honeyd_conf_t
```

```
/usr/local/share/honeyd/nmap.assoc system_u:object_r:honeyd_conf_t
/usr/local/share/honeyd/xprobe2.conf system_u:object_r:honeyd_conf_t
/usr/local/share/honeyd/nmap.prints system_u:object_r:honeyd_conf_t
# Le fichier de paramétrage de honeyd que nous utilisions
# sachant qu□il faut développer le sien
/usr/local/share/honeyd/config.sample system_u:object_r:honeyd_conf_t
# Les scripts lancés par honeyd
/usr/local/share/honeyd/scripts(/.*)?
system_u:object_r:honeyd_script_exec_t
# Les logs des scripts test.sh et web.sh vont dans /tmp/log
/tmp/log system_u:object_r:honeyd_script_log_t
# ...
```

Une fois ces ressources labellisées, il faut déterminer l'ensemble des privilèges et possibilités de ~~honeyd~~. Si un bogue de sécurité existait dans ~~honeyd~~ ou dans un des sous-programmes simulant un service et lancé par ~~honeyd~~, les pirates seraient bloqués et ne pourraient pas par exemple contrôler la machine visée (~~honeyd~~ tournant pourtant sous root !).
Extrait du paramétrage du Type Enforcement pour honeyd (~~honeyd.te~~) :

```
 # [attention on rappelle que ceci est montré pour illustration]
# Declarons honeyd comme daemon réseau
daemon_domain(honeyd)
can_network(honeyd_t)
# Type declarations pour honeyd
type honeyd_conf_t, file_type, sysadmfile;
type honeyd_script_exec_t, file_type, sysadmfile, exec_type;
type honeyd_script_log_t, file_type, sysadmfile;
type honeyd_script_t, domain, privlog;
# Honeyd peut être exécuté par init ou l□admin
role system_r types honeyd_t;
role sysadm_r types honeyd_t;
# Quand sysadm lance honeyd, il passe dans le domaine honeyd_t
domain_auto_trans(sysadm_t, honeyd_exec_t, honeyd_t)
# Autorisation à honeyd de lire les fichiers de configuration
# Meme si ces fichiers étaient en écriture (rw-), honeyd ne pourrait
# les modifier, à cause du MAC assuré grâce à ces lignes de politiques
allow honeyd_t honeyd_conf_t:dir { search };
allow honeyd_t honeyd_conf_t:file { getattr read };
# /dev/urandom utilisé par honeyd pour générer des nombres aléatoires
allow honeyd_t random_device_t:chr_file { read };
# Honeyd a besoin du réseau
# La capability net_raw est nécessaire (sniffer réseau à base de libpcap)
allow honeyd_t honeyd_t:capability { net_raw };
# reste du réseau :
allow honeyd_t honeyd_t:packet_socket { bind create getopt ioctl read
setopt write };
allow honeyd_t honeyd_t:rawip_socket { sendto write create getopt setopt };
# ...
```

Si ~~honeyd~~ a été choisi ici comme exemple, c'est par originalité (pas de politique existante à l'heure actuelle) et pour la richesse des actions effectuées par ce démon (réseau, système, etc.). Le fait de créer une politique pour ce programme nous a permis de mieux comprendre la configuration de SELinux. Nos fichiers de tests sont disponibles sur [RSTACK].
En conclusion par rapport à la sécurité apportée par SELinux, on peut dire que cela permet de contenir des agressions grâce à la mise en place de politiques de sécurité bas niveau appliquées à toutes les interactions système (sorte de "pare-feu" système). Citons par exemple que les attaques classiques d'écriture dans /dev/kmem seront donc facilement bloquées (quels processus root ont réellement besoin du droit d'écrire dedans sur une machine sécurisée ?), les tags de site Web pourront aussi être interdits (le noyau bloquant l'écriture dans la configuration et dans les pages

Web d'un serveur), et votre serveur FTP (~~wu-ftpd~~ au hasard ;-)) ne pourra plus être utilisé comme outil d'administration à distance par des attaquants, etc.

Limites

Complexité d'administration

En regardant en détail le fonctionnement de SELinux, vous aurez sûrement compris qu'une des premières difficultés rencontrées concerne l'exploitation de cette solution.
En effet, si la mise en place est envisageable sur une machine jouant un rôle particulier (serveur Web et DNS, etc.), le déploiement de SELinux sur un réseau de plusieurs machines peut demander un travail important (par exemple sur un parc de postes de travail, etc.).
Notons par exemple qu'il n'existe pas encore de moyens pour gérer le déploiement et la configuration des règles de sécurité à distance. Cela ne serait de toutes façons pas suffisant, car la complexité des règles et fichiers à gérer pour SELinux est telle qu'une compréhension de ce qui est installé sur tout un parc informatique semble illusoire.
Certains outils encore pauvres ou en développement par différents sites (console Webmin, etc.) permettront peut-être de pallier cela, à moins que l'on ne voit plutôt apparaître des solutions clef en main (distributions blindées basées sur SELinux) qui nécessiteraient peu de maintenance et de paramétrage de la part des administrateurs. D'autres outils manquent encore, comme l'intégration du backup des contextes de sécurité des fichiers (labels).
Notons de plus qu'aucun retour d'expérience véritable et reconnu n'existe à ce jour sur la mise en production de SELinux, la NSA elle-même se gardant un droit de réserve compréhensible sur la question (Never Say Anything).
Enfin, dans le cadre de l'ajout d'une application qui n'aurait pas été prévue initialement, donc sans politique de sécurité, l'écriture de cette dernière peut s'avérer très difficile si l'on veut être certain d'appliquer le principe de moindre privilège. Il semble en effet nécessaire de bien connaître à l'avance le fonctionnement de l'application à intégrer dans SELinux. Pour générer des politiques de sécurité, on peut utiliser l'outil ~~newrules.pl~~ analysant les logs systèmes et couplé avec le mode permissive de SELinux : cela fournit les lignes de politique nécessaires au fonctionnement d'une application (ce qui manquerait pour qu'elle fonctionne si SELinux tournait en mode réel), mais ces dernières sont souvent trop englobantes (on ne peut pas automatiser la définition de type pour les ressources concernées, donc ces lignes sont souvent avec des types génériques, ce qui peut au final poser un problème de sécurité).
Ce travail est donc compliqué et demande encore une forme d'expertise (imaginez les problèmes posés par une mise à jour de nombreuses applications..). A contrario, le blindage de ~~honeyd~~ avec Grsecurity ou Systrace [MISC8] fut pour nous très rapide (quelques minutes) par rapport au même travail sous SELinux. Mais il est tout à fait normal que la définition d'une politique de sécurité soit une démarche fastidieuse, étant donné que l'approche retenue par SELinux est de déclarer l'ensemble des actions autorisées.

Robustesse

A priori, à ce jour, aucune faille majeure sur le projet SELinux n'a été annoncée publiquement. Par contre, s'il peut transformer le noyau Linux avantageusement, il ne faut pas oublier que la sécurité qu'il apporte sera basée notamment sur la qualité des politiques de sécurité (et leur maintenance, car les applications évoluent rapidement, surtout dans le logiciel libre, et les politiques de sécurité

doivent donc suivre). Aussi, nous avons pu constater que certaines proposées par des contributeurs SELinux ne sont pas peaufinées pour limiter fortement un pirate (les démons peuvent souvent lire tout ~~/etc~~, à l'exception de fichiers particuliers comme ~~shadow~~ et la configuration SELinux).
On peut donc s'attendre à un travail de fond sur les politiques de sécurité, notamment sur les applications nécessitant beaucoup de droits. En effet, si un programme requiert des règles assez souples, un pirate ne pourra peut-être pas obtenir un shell, mais il pourra lancer d'autres actions malveillantes entrant dans le cadre des actions légitimes du programme violé.
Par exemple, prenons le cas d'un serveur FTP qui a besoin de pouvoir ouvrir des sockets TCP vers des ports supérieurs à 1024 n'importe où sur Internet (notamment pour envoyer les fichiers demandés par un client légitime en mode FTP actif). Si, par exemple, un buffer overflow modifie son comportement, le pirate pourra peut-être essayer d'utiliser le serveur FTP comme rebond vers d'autres services sur Internet.
Autre exemple, que se passe-t-il si un serveur Web se met à ne plus utiliser ses privilèges pour lire les pages Web demandées par des clients, mais qu'il décide (via un buffer overflow) de répondre toujours la même chose pour chaque requête : "ce site Web a été piraté lamentablement" ?
On peut ainsi penser que de nouvelles formes de shellcodes et attaques verront le jour, non pas pour contourner les politiques de sécurité mais pour abuser des possibilités offertes dans le cadre des actions légitimes d'une application. On touche ici la limite de ce genre de techniques de containment, et d'autres solutions comme Grsecurity, qui répondent différemment à ces problèmes liés aux vulnérabilités logicielles, et méritent d'être étudiées.

Un dernier point concernant la sécurité est celui de la surveillance. La lecture des traces de sécurité de SELinux et son interprétation n□est pas encore chose aisée (aucun outil associé) en cas de gros volumes de traces.
De plus le système antiflood de logs fait qu□une limite de vitesse est imposée aux logs de SELinux : passé cette frontière, des logs seront perdus plutôt que de prendre le risque de trop remplir le disque ou perdre en performances (serait-ce utilisable pour cacher la fin d□une séquence d□attaques ?).

Si vous souhaitez regarder rapidement à quoi ressemble un SELinux et sa robustesse, n'hésitez pas à vous connecter via SSH sur les machines suivantes laissées en libre service avec le compte ~~root~~ (vous verrez que ce dernier est bloqué et ne peut rien modifier/lire de concret à cause des politiques utilisées par le Security Server de SELinux) :

```
Gentoo SELinux, root en libre service, mot de passe : □gentoo□
$ ssh -a -x root@selinux.dev.gentoo.org
Debian SELinux, root en libre service, mot de passe : □azerty□
$ ssh -a -x root@cose.coker.com.au
```

Pendant le FOSDEM 2003 à Bruxelles, une machine Debian SE Linux était installée de la même manière pour tous les attaquants potentiels désirant tester la fiabilité de SE Linux (concours monté sur place et non annoncé sur le site du FOSDEM) : personne ne perça la machine.

Conclusion et perspectives

Si nous devions coller un seul adjectif à SELinux, nous choisirions sans hésiter le suivant : ambitieux. En effet, les possibilités quant à la sécurité sont très complètes et la NSA a ainsi offert au logiciel libre un bel exemple et élan en termes de recherche et de développement sur la façon de construire un système d'exploitation robuste, fiable et stable.

La contre-partie de ce caractère ambitieux réside dans la difficulté de définir des politiques de sécurité fines pour l'ensemble des applications utilisées par un système d'exploitation standard. Cela nécessite un investissement important pour tenir à la fois compte des spécificités des applications, de la grammaire de SELinux et des contextes de déploiement. Par ailleurs, il faut garder à l'esprit que ce genre de solution ne couvre que l'aspect "modélisation" du problème de containment, et à ce titre, d'autres solutions plus simples à déployer, comme [GrSecurity], intégrant notamment des protections contre l'abus des failles logicielles (PaX), semblent complémentaires et méritent d'être prises en compte.
Quel futur attend donc ce projet libre très avancé techniquement ? On voit déjà poindre à l'horizon des solutions de sécurité utilisant SELinux (plateformes robustes dédiées à certaines fonctions, etc.). Par ailleurs, certaines parties de SELinux sont encore en développement, comme selopt qui pourrait permettre de transporter les SID au travers d'un réseau sur un environnement homogène composé de SE Linux (un parc informatique, un cluster de calcul...). Le site de SELinux sur [SourceForge] représente une excellente source d'information.
Avec la NSA, force de renseignement américaine colossale derrière un tel projet, ainsi que l'avènement des LSM et du noyau 2.6, gageons que cette solution fera parler d'elle, à moins que des restrictions politiques ne s'y opposent (certains mouvements de pensée aux USA ont annoncé leur étonnement de voir qu'une solution de sécurité était développée grâce aux impôts américains et distribuée gratuitement (sous licence GPL) à tout le monde, y compris leurs "ennemis"...).

Références

- [SELinux] Security-Enhanced Linux. http://www.nsa.gov/selinux/
- [Assemblée2000] Rapport d'information numéro 2623 de l'Assemblée Nationale française du 11 octobre 2000 intitulé "Les systèmes de surveillance et d'interception électroniques pouvant mettre en cause la sécurité nationale". Voir le chapitre I-A nommé "Une organisation vraisemblablement détournée de sa finalité militaire initiale".
- [Spencer99] Ray Spencer, Stephen Smalley, Peter Loscocco, Mike Hibler, David Andersen and Jay Lepreau. The Flask Security Architecture: System Support for Diverse Security Policies. In Proceedings of the 8th USENIX Security Symposium, 1999.
- [Smalley01] Stephen Smalley, Chris Vance and Wayne Salamon. Implementing SELinux as a Linux Security Module. 2001. http://www.nsa.gov/selinux/module-abs.html
- [Bell73] D. E. Bell and L. J. LaPadula. Secure Computer Systems: Mathematical Foundations and Model. Technical Report, The MITRE Corp., Number M74-244, May 1973.
- [Amoroso94] Edward Amoroso. Fundamentals of Computer Security Technology. 1994. ISBN 0-13-108929-3.
- [PolicyConf] Stephen Smalley. Configuring the SELinux Policy. Last revised: January 2003. http://www.nsa.gov/selinux/policy2-abs.html
- [SYSADMIN03] Kerry Thomson. SELinux. In SysAdmin Magazine, March 2003, Volume 12, Number 3
- [MISC8] Arnaud Guignard, Laurent Oudot, V. G. Honeyd. In MISC Magazine n°8, Juillet-Août 2003.
- [RSTACK] Configuration de SELinux pour honeyd (pour test) http://www.rstack.org
- [FAQ]The UnOfficial SELinux FAQ. http://www.crypt.gen.nz/selinux/faq.html
- [Coker]Security Enhanced Linux information. http://www.coker.com.au/selinux/
- [SourceForge] SELinux Distribution Integration News. http://selinux.sourceforge.net/
- [HOWTO]Getting Started with SE Linux HOWTO. http://sourceforge.net/docman /display_doc.php?docid=15285&group_id=21266

- [RedHat]Red Hat on Security: SELinux. http://www.redhat.com/solutions/security/SELinux.html
- [GrSecurity] http://www.grsecurity.org/

~~Retrouvez cet article dans :~~ Linux Magazine Hors série 17

Posté par (La rédaction) | Signature : Mathieu Blanc, Laurent Oudot | Article paru dans

Laissez une réponse

Vous devez avoir ouvert une session pour écrire un commentaire.

« Précédent Aller au contenu »

Identifiez-vous
Inscription
S'abonner à UNIX Garden

Articles de 1ère page

- Inkscape 0.45 : plus rapide, plus facile !
- KTorrent : le client BitTorrent pour KDE
- La légalité du spam : une question ouverte
- Pear et les librairies PHP
- FreeDOS
- Mise en œuvre d'une passerelle Internet sous Linux
- Les pseudo-classes en CSS
- GNU/Linux Magazine N°108 - Septembre 2008 - Chez votre marchand de journaux
- Linux Pratique N°49 - Septembre/Octobre 2008 - Chez votre marchand de journaux
- Donner du style à son texte : utiliser les lettrines

Actuellement en kiosque :

- # Il y a actuellement

- 721 articles/billets en ligne.

Recherche

- # Catégories

- - Administration réseau
  - Administration système
  - Agenda-Interview
  - Audio-vidéo
  - Bureautique
  - Comprendre
  - Distribution
  - Embarqué
  - Environnement de bureau
  - Graphisme
  - Jeux
  - Matériel
  - News
  - Programmation
  - Réfléchir
  - Sécurité
  - Utilitaires
  - Web

- # Archives

- - septembre 2008
  - août 2008

- juillet 2008
- juin 2008
- mai 2008
- avril 2008
- mars 2008
- février 2008
- janvier 2008
- décembre 2007
- novembre 2007
- février 2007

GNU/Linux Magazine

- GNU/Linux Magazine 108 - Septembre 2008 - Chez votre marchand de journaux
- Edito : GNU/Linux Magazine 108
- GNU/Linux Magazine HS 38 - Septembre/Octobre 2008 - Chez votre marchand de journaux
- Edito : GNU/Linux Magazine HS 38
- GNU/Linux Magazine 107 - Juillet/Août 2008 - Chez votre marchand de journaux

GNU/Linux Pratique

- Linux Pratique N°49 -Septembre/Octobre 2008 - Chez votre marchand de journaux
- Edito : Linux Pratique N°49
- À télécharger : Les fichiers du Cahier Web de Linux Pratique n°49
- Linux Pratique Essentiel N°3 - Août/Septembre 2008 - Chez votre marchand de journaux
- Edito : Linux Pratique Essentiel N°3

MISC Magazine

- Misc 38 : Codes Malicieux, quoi de neuf ? - Juillet/Août 2008 - Chez votre marchand de journaux
- Edito : Misc 38
- Références de l'article « Détection de malware par analyse système » d'Arnaud Pilon paru dans MISC 38
- Références de l'article « La sécurité des communications vocales (3) : techniques numériques » d'Éric Filiol paru dans MISC 38
- Misc 37 : Déni de service - Mai/Juin 2008 - Chez votre marchand de journaux