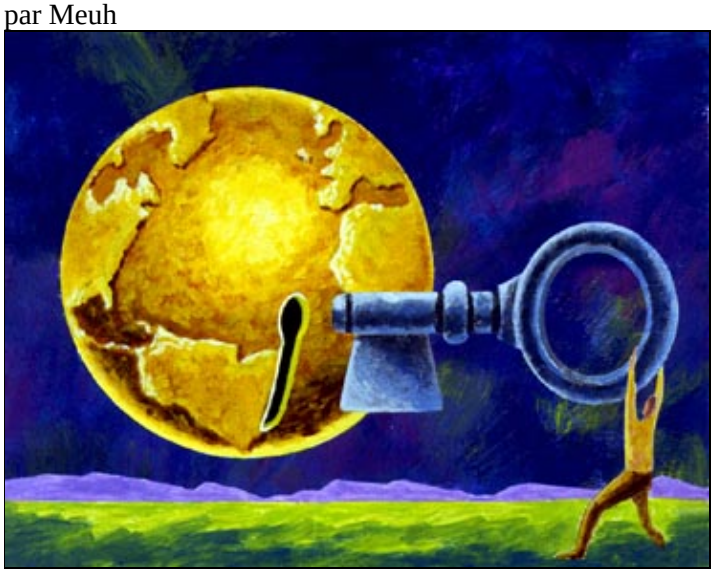
Sommaire

Crypter un système de fichiers

Crypter un système de fichiers
par Meuh



Depuis Linux version 2.4.22, le noyau stable intègre un ensemble d'algorithmes de cryptographie regroupés sous le nom de 'Scatterlist Cryptographic API'. Cette API est un 'backport' en provenance du kernel 2.5/2.6.

Grâce à cette API et à une extension du device loop, il est possible de crypter un système de fichiers. Nous allons apprendre ici à l'installer et à la configurer.

Il faut savoir qu'il existe deux versions d'API cryptographiques :

- celle de http://kerneli.org, disponible sous forme de patch pour les noyaux 2.2 et 2.4. Cette API est développée par AT guardian.no Alexander Kjeldaas, AT gnu.org Hebert Valerio Riedel , AT debian.org Kyle McMartin, AT certainkey.com Jean−Luc Cooke, AT mac.com David Bryson, AT endorphin.org Clemens Fruhwirth, AT ringstrom.mine.nu Tobias Ringstrom, AT gnumonks.org Harald Welte.
- celle du noyau 2.5/2.6 et maintenant 2.4.22. Cette version est un dérivé simplifié d'API de http://kerneli.org destinée à l'implémentation d'IPSec. Elle a été développée par AT intercode.com.au James Morris et AT redhat.com David S. Miller

À noter qu'il existe aussi la solution loop−AES développée par AT pp.inet.fi Jari Ruusu. Voir plus loin.

Comment ça marche

Le loop device permet de simuler un périphérique en mode bloc à partir d'un fichier. C'est grâce à ce driver que l'on peut monter les images de CD−ROM au format ISO9660 :

mount −t iso9660 −o loop monimage.iso /cdrom
La cryptoloop insère les fonctions de cryptage dans les fonctions de lecture et d'écriture du loop device. Le loop device est donc utilisé pour convertir un fichier crypté en *block device* que l'on peut ensuite monter comme un disque normal. Tout ce qui est écrit sur ce système de fichier est crypté, autant les données que les méta−données (arborescence, noms de fichiers, droits d'accès, ...). L'accès aux fichiers est entièrement transparent, à part un temps d'accès plus important que l'accès direct.

La cryptoloop utilise l'API de cryptographie afin d'avoir accès aux différents algorithmes de cryptage ('cypher') enregistrés. Parmi les cyphers disponibles dans la 'scatterlist cryptoapi', on citera : AES(Rijndael), Blowfish, Twofish, Serpent et l'antique DES.

Prérequis

Bien que les algorithmes de cryptage soient présents dans le kernel 2.4.22, il n'y a pour l'instant aucun driver permettant de crypter un fichier, un système de fichiers, un disque ou le swap. Il faut toujours un patch pour étendre les fonctionnalités du périphérique loop, ceci afin d'y insérer les fonctions de cryptage à la volée. Pour la version 2.4, sont donc nécessaires :

- Les sources du noyau 2.4.22,
- Un patch cryptoloop pour étendre le fonctionnement du périphérique loop et intégrer le support de la cryptographie : soit patch−cryptoloop−jari−2.4.22.0, soit patch−cryptoloop−hvr−2.4.22.0 .

Le premier patch apporte un nombre conséquent de changements au driver loop mais permet en contrepartie de crypter le swap. Le deuxième patch applique le minimum de changements pour permettre le cryptage/décryptage. L'auteur recommande l'usage du premier car il calcule correctement les tailles des volumes si l'on fait usage des offsets (voir plus loin).
Quant à la prochaine version stable du kernel, la 2.6, tout y est présent : API cryptographique, crypto loop device et IPSec, donc pas de patch à appliquer.
Ensuite, il est nécessaire de disposer des outils en adéquation avec la version de la cryptoloop/cryptoapi : vérifiez si vous votre système ne dispose pas des util−linux 2.12 : mount −V Si ce n'est pas le cas, il vous faut alors :

- les sources de util−linux−2.12 : la dernière version intégre un support minimal de la cryptoapi, et supporte la nouvelle version de la loop device présente dans le 2.6.

Ou bien :

- les sources de util−linux−2.11z
- le patch util−linux−2.11z−crypto−v2.diff.bz2 développé par Jari Ruusu. Le patch de Jari supporte loop−AES, la crypto api, et des fonctions évoluées pour la saisie du mot de passe.

À noter que Debian (3.0 Woody) fournit une version d'util−linux intégrant les patchs pour la cryptoapi de kerneli.org et est inutilisable pour notre usage.

Remarque : Attention, lors de la mise à jour de vos outils, il est possible que d'une version (patch) d'util−linux à l'autre, la méthode pour étendre et/ou hasher le mot de passe peut être différente. Ce changement produisant alors un mot de passe incapable de décrypter vos données. L'auteur vous conseille d'utiliser les util−linux 2.12, nouvelle version qui va fixer la norme assurant la compatibilité future.

Le kernel

Pour les gens déjà au kernel 2.6, allez directement en 2.

1. Appliquez le patch cryptoloop : patch−cryptoloop−jari−2.4.22.0 ou patch−cryptoloop−hvr−2.4.22.0 :
   $ cd /usr/src/linux
   $ bzcat <chemin>/patch−cryptoloop−<version>−2.4.22.0.bz2 | patch −p1 −E

2. Configurez le kernel : make menuconfig ou make xconfig Section [Block devices] : activez les modules correspondants aux périphériques loop et cryptoloop. Section [Cryptographic options] : activez le support de la crypto : tous les *cyphers* en modules
3. Si vous n'utilisiez pas déjà le kernel 2.6 ou 2.4.22 avec le support du loop device en module ou que l'opération de chargement a échoué, il faut recompiler un noyau : utilisez la commande correspondant à votre architecture, bien souvent :
   $ make bzImage
   Installez ensuite votre noyau suivant la procédure adaptée à votre distribution.
4. Déchargez si nécessaire le module loop.o, recompilez et installez les modules (suivez la procédure de votre distribution si possible) :
   $ make modules
   # make modules_install
   et pensez à rebooter votre système si vous avez changé de noyau.
5. Finalement (re)chargez loop.o, puis cryptoloop.o.
   # modprobe loop
   # modprobe cryptoloop
   Si le chargement échoue, recommencez à l'étape 3. Si cela ne marche toujours pas, attendez la prochaine version de ce guide.

Les outils

Pour manipuler le loop device patché, il faut une version adéquate de *lomount*. Cet outil, que l'on trouve généralement dans le répertoire /sbin, fait partie des util−linux. Comme précisé précédemment, la version 2.12 intègre un support de la cryptoloop, mais ne fournit pas tous les services que la version 2.11z avec le patch Jari.

Si actuellement vous n'avez pas les util−linux 2.12 :

1. Décompactez les sources d'util−linux, soit 2.12 soit 2.11z
   $ tar xzf util−linux−<version>.tar.gz
2. déplacez−vous dans le répertoire fraîchement créé
   cd util−linux−<version>
3. Si vous décidez d'utiliser la version 2.11z avec le patch de Jari, appliquez le patch :
   $ bzcat util−linux−2.11z−crypto−v2.diff.bz2 | patch −p1 −E
4. Configuration et compilation :
   $ ./configure
   $ cd lib; make ; cd ../mount; make
5. Copiez ensuite mount, umount dans /usr/local/bin, losetup dans /usr/local/sbin. Vous pouvez aussi écraser les fichiers originaux dans /bin et /sbin, mais ce n'est certainement pas ce que souhaite votre distribution.
   # cp mount umount /usr/local/bin/
   # cp losetup /usr/local/sbin/

Création et montage

Nous allons préparer un système de fichiers crypté pour un utilisateur en particulier. Le système de fichier sera monté dans le répertoire *home* de cet utilisateur.

On pourra aussi utiliser une autre arborescence reprenant la même structure que /home mais dédiée aux données cryptées : /crypt/*user*/ contenant le container crypté et le système de fichiers crypté pour chaque

utilisateur. Ce choix est laissé au soin du lecteur.

Voici la liste des commandes à utiliser lors la création de votre container crypté :

$ signifie que la commande est lancée en tant qu'utilisateur lambda, **#** signifie que la commande requiert les droits du super utilisateur (root).

Définition des variables d'environnement :

On définit les chemins et les paramètres de cryptage. On spécifie le nom du fichier qui va contenir le système de fichiers crypté. Ce fichier est appelé par la suite *container*.

```
−− choix d'un périphérique loop disponible, entre 0 et 7
−− si vous utilisez devfs, vous pouvez utilisez /dev/loop/X
$ LOOP=/dev/loop0
−− point de montage du système de fichiers crypté
−− c'est aussi dans ce répertoire que l'on va placer le container
$ MOUNTPOINT=$HOME/crypt
−− nom du fichier crypté contenant le système de fichiers
$ CONTAINER=$MOUNTPOINT/.crypt.img
−− algorithme utilisé : AES(rijndael), le vainqueur du concours du NIST
$ CYPHER=aes
−− lecture de l'offset dans le fichier
−− utilisé en plus du mot de passe, peut sécuriser encore plus le container
−− l'offset est calculé en nombre de secteur pour satisfaire une contrainte
−− de la cryptoloop : les cyphers travaillent par bloc et non pas par flux.
$ read sector
$ OFFSET=$(($sector*512))
−− taille du container en Mo
$ read SIZE
```

Création du container :

```
−− création du container crypté
−− il est rempli de données pseudo−aléatoires afin de compliquer
−− les attaques par force brute
−− (surtout ne pas utiliser /dev/zero pour remplir le container crypté)
$ mkdir −p $MOUNTPOINT
$ dd if=/dev/urandom of=$CONTAINER bs=1M count=$SIZE
$ chmod 600 $CONTAINER
```

Activation :

C'est à ce moment que le système va vous demander de choisir un mot de passe.

Note : Avec les util−linux 2.12, il est impossible de choisir la taille de la clé, fixée à 256 bits (32 octets). De plus le mot de passe n'est pas étendu/hashé, donc utilisez un mot de passe d'une taille avoisinant les 32 caractères.

```
−− chargement des modules
# modprobe loop
# modprobe cryptoloop
# modprobe aes
```

−− activation de la cryptoloop, saisie du mot de passe
/usr/local/sbin/losetup −e $CYPHER −o $OFFSET $LOOP $CONTAINER

−− affichage des informations sur le loop device configuré
/usr/local/sbin/losetup $LOOP

Initialisation :

Création du système de fichier, on utilise ici ext3fs, la version journalisée du système de fichiers ext2fs. Cela a pour avantage de limiter les risques de corruption des méta données dans le container, mais c'est coûteux en ressources si le système de fichiers où se trouve le container est lui aussi journalisé.

−− création d'un système de fichiers utilisable
−− à 100% par les utilisateurs normaux, non privilégiés.
mkfs.ext3 −j −m 0 −L "home−crypt" $LOOP

−− montage de ce système de fichiers
−− ATTENTION : le container crypté devient inaccessible
−− pour le commun des mortels
mount −t ext3 $LOOP $MOUNTPOINT

−− on donne les droits à l'utilisateur
chown <user> :<group> $MOUNTPOINT

−− l'utilisateur sécurise l'accès à son système de fichiers crypté
$ chmod 700 $MOUNTPOINT

−− démontage
umount $MOUNTPOINT

−− suppression du device
−− note : n'importe quel losetup fera l'affaire ici
losetup −d $LOOP
Si la création du système de fichiers échoue à cause d'une écriture en dehors du périphérique, c'est que vous faites usage de l'offset avec une version de la loop device qui ne le gère pas correctement. Dans ce cas vous pouvez changer de patch ou alors calculer le nombre de blocs disponibles pour le système de fichiers et passer cette valeur à mkfs.ext3 : (taille du container / 512) − offset .

Au final :

Maintenant que tout est prêt, on peut monter le container de façon *intégré* en une seule opération :

−− ensuite, activation de la loop et montage, en une seule commande
/usr/local/bin/mount −t ext3 −o defaults,user,exec,loop,encryption=$CYPHER,offset=$OFFSET $CONTAINER $MOUNTPOINT

−− et démontage si nécessaire
/usr/local/bin/umount $MOUNTPOINT
Malheureusement, seul root a la possibilité de monter le container crypté. Dans /etc/fstab, une option user=xxx ou group=xxx autorisant seulement un utilisateur ou un groupe d'utilisateur à monter/démonter un système de fichier serait intéressante ici : une idée à creuser.

Pour l'instant, l'administrateur peut créer un script contenant toutes les commandes nécessaires, et autoriser l'utilisateur à exécuter ce script grâce à 'sudo'.

NOTE : attention, ici on monte le container crypté par dessus le répertoire le contenant. Quand il est monté, cela rend son accès impossible pour l'utilisateur lambda. Cela évite que l'on puisse effectuer une recherche de clé de cryptage par force brute en comparant le système de fichier décrypté et le container crypté. Mais cette astuce permet surtout d'éviter de modifier le container pendant qu'il est monté.

Changement de clé, d'algorithme, etc

Vous serez probablement amenés à changer de mot de passe ou d'algorithme de cryptage. L'opération n'est pas transparente : elle nécessite la création d'un nouveau container et la recopie des informations contenues dans l'ancien.

Deux solutions sont disponibles : la copie brute du container ou la copie des fichiers.

Copie brute

Ici on effectue une copie bloc pour bloc des loop devices :

−− Déplacement de l'ancien container
$ mv $CONTAINER ${CONTAINER}.old
−− Création du nouveau container au minimum de la même taille que l'ancien
−− en sachant qu'une taille supérieure ne sera pas directement utilisée
−− Attention aussi au changement d'offset : un offset plus grand implique
−− une taille plus importante.
$ dd if=/dev/urandom of=$CONTAINER bs=1M count=<size>
−− configuration du loop device pour le nouveau container
/usr/local/sbin/losetup −e <NEWCYPHER> [−o <NEWOFFSET>] $LOOP $CONTAINER
−− configuration du loop device pour l'ancien container
/usr/local/sbin/losetup −e <OLDCYPHER> [−o <OLDOFFSET>] /dev/loop1 ${CONTAINER}.old
−− recopie des données
dd if=/dev/loop1 of=$LOOP bs=1M
−− Vérifiez que vos données sont bien présentes
mount $LOOP $MOUNTPOINT
−− Suppression de l'ancien container
losetup −d /dev/loop1
$ rm ${CONTAINER}.old
Il est possible de spécifier une taille plus grande, de copier les données avec dd, et ensuite d'utiliser un outil comme *resize2fs* pour agrandir le système de fichiers. Cet exercice est laissé au soin des lecteurs aventureux.

Copie de fichiers

Cette fois, nous allons copier les fichiers présents dans le container :

−− Déplacement de l'ancien container
$ mv $CONTAINER ${CONTAINER}.old
Ensuite il faut créer un nouveau container en suivant la même marche à suivre qu'au début. Ce nouveau container doit être suffisament grand pour contenir un système de fichiers contenant les fichiers de l'ancien container. De plus, la remarque précédente au sujet des offsets s'applique ici aussi.

Avant de monter le nouveau container, activez l'ancien :

−− configuration du loop device pour l'ancien container
losetup −e <OLDCYPHER> [−o <OLDOFFSET>] /dev/loop/1 ${CONTAINER}.old
Montez ensuite le nouveau container et l'ancien :

mount $LOOP $MOUNTPOINT
−− montage de l'ancien container dans un répertoire temporaire
mount /dev/loop1 <tmp_mountpoint>
Recopiez ensuite les fichiers :

−− recopie des fichiers
$ cd <tmp_mountpoint>
$ cp −Rdp .??* * $MOUNTPOINT/
$ cd ..
Supprimez ensuite l'ancien container :

−− suppression de l'ancien container
umount <tmp_mountpoint>
losetup −d /dev/loop1
umount $MOUNTPOINT
$ rm ${CONTAINER}.old
La deuxième solution permet un changement de type de système de fichiers et un redimensionnement simple du système de fichiers, mais nécessite de monter deux fois le container, exposant deux fois les données non cryptées.

Présentation succinte de Loop−AES

Loop−AES développée par .ruusu@pp.inet.fi Jari Ruusu, n'est pas une API de cryptographie générique, ce patch ne fournit que les services de cryptage de volumes. Loop−AES offre plus d'algorithmes que son nom ne le laisse croire : twofish, blowfish, mars et rc6 sont supportés en plus d'AES. Cette solution de cryptage est moins générique mais plus performante que les autres de par sa spécialisation.

Quelques remarques concernant la sécurité

Quelques rappels sur la sécurité...

1. Soyez paranoïaque.
2. Vérifiez que les personnes qui vous parlent de cryptographie ne sont pas des petits malins qui cherchent à vous induire en erreur.
3. Utilisez GnuPG et/ou MD5 pour vérifier les signatures de toutes les sources et patches que vous utilisez.
4. à l'intérieur du système de fichiers crypté, ne pas crypter des fichiers avec le même algorithme et le même mot de passe : ces fichiers risqueraient d'apparaitre en clair$ dans le container. On fait usage d'algorithme de cryptage symétrique, la même clé est utilisée pour crypter et décrypter, et bien souvent la même fonction assure le cryptage et le décryptage : on crypte une fois pour obtenir l'information cryptée, on recrypte une deuxième fois pour obtenir l'information en clair.
5. Ne pas exporter votre container ni votre système de fichiers crypté : NFS, FTP, Samba, etc... sont à proscrire. Quel intéret de crypter vos données si vous les faites circuler en clair. Utilisez des protocoles sécurisés (SSH, VPN) pour transmettre vos fichiers, cryptés eux−mêmes si besoin.

6. Protégez votre ordinateur :

- empêchez l'ouverture de l'ordinateur,
- empêchez l'utilisation de disque d'amorçage autre que le disque dur,
- sécurisez l'amorçage du système d'exploitation : une personne non autorisée ne doit pas pouvoir passer de paramètres au noyau.
- ne laisser jamais de session ouverte sans protection, verrouillez votre terminal dès que vous le quittez.
- archivez vos données de façon cryptée.

1. Ne stockez vos mots de passe que dans votre tête.
2. Et ne croyez pas qu'en sachant tout cela vous êtes en sécurité.

Conclusion

Nous avons vu comment crypter un système de fichiers contenu dans un fichier. Pour allez plus loin, on peut utiliser une partition en lieu et place du fichier container.

Il est maintenant 'facile' de crypter ses données avec le noyau Linux, même s'il faut toujours un patch pour la version 2.4. À terme, la version 2.6 de linux plus les util−linux 2.12 offriront à tout un chacun la possibilité de crypter des systèmes de fichiers.

Ensuite à chacun de voir s'il a besoin de crypter ses données et s'il a confiance dans les algorithmes du kernel.

Liens

- Crypto API (original) http://www.kerneli.org/
- Util Linux 2.12 : http://www.kernel.org/pub/linux/uti...
- Patchs cryptoloop pour le kernel 2.4.22 :
  http://www.kernel.org/pub/linux/ker...
  http://www.kernel.org/pub/linux/ker...
  http://encryptionhowto.sourceforge.net/
- patch util−linux 2.11z (Jari Ruusu) :
  http://therapy.endorphin.org/patches/
  http://www.kerneli.org/pipermail/cr...
- patches pour étendre les util−linux 2.12 : http://www.stwing.org/~sluskyb/util...
- explication cryptoapi : http://www.kerneli.org/pipermail/cr...
- quelques informations :
  http://www.abeowitz.com/crypto/
  http://sourceforge.net/projects/cry...
- crypto api 2.5 par James Morris : http://samba.org/~jamesm/crypto/
- Le projet de cryptographie (loop + crypto) de Jari : http://loopaes.sourceforge.net/

Copyright

Copyright (C) 2003 @meuh.eu.org Yann Droneaud Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front−Cover Texts, and no Back−Cover Texts.

Cette page est issue de la documentation 'pré−wiki' de Léa a été convertie avec HTML::WikiConverter. Elle fut créée par Meuh le 09/09/2002.

Copyright

Copyright © 09/09/2002, Meuh

Ce document est publié sous licence Creative Commons Attribution, Partage à l'identique, Contexte non commercial 2.0 : http://creativecommons.org/licenses/by−nc−sa/2.0/fr/