Kollectiv' admin

intrigeri [intrigeri@squat.net]

7 juin 2003

Résumé

Ce document vise à proposer un schéma d'administration *collective* et *formalisée* d'un parc d'ordinateurs, dans un contexte nécessitant de prêter une attention particulière aux questions de sécurité informatique.

Table des matières

0.1 Changelog

- 2003 02 17 : version initiale
- 2003 04 04 :
  - quelques clarifications langagières
  - précision plus grande au niveau des relations entre *blocs* d'admin
- 2003 04 05 : annexes
- 2003 04 14 : début de conversion en TEX
- 2003 04 19 : fin de la conversion
- 2003 04 20 : écriture du Makefile
- 2003 05 10 : plein de clarifications/ajouts
- 2003 05 29 : réorganisation complète de ce document, recentrage sur les objectifs initiaux, peaufinages divers
- 2003 06 07 : version RFC

0.2 TODO

- préciser que les rôles des adminEs peuvent évoluer dans le temps
- comment gérer les dépendances entre blocs d'admin ?
- scripts permettant la gestion des admins ?
- paquet Debian « ala AlternC » ?

1 Analyse du problème

1.1 Politique

Le fait d'administrer des ordinateurs collectifs ou un serveur n'est pas anodin ; cela implique :

- un « contrat de confiance » avec les utilisateurices ;
- une grande disponibilité pour être réactif/ve en cas de problème ;
- du temps et de la régularité pour la maintenance ;

De plus, il est souhaitable, dans le cadre de l'administration collective d'un parc informatique, de mettre en place des solutions respectant les contraintes suivantes :

- non-concentration du pouvoir ;
- répartition du travail sur quelques cerveaux coopérants ;
- gestion et suivi aisés de la partie commune de l'administration ;
- simplicité de la transmission de tâches/responsabilités.

Quelques précisions avant de mettre les mains dans le cambouis...

- la confrontation de ce modèle théorique avec l'expérience vécue ne manquera pas, soyons-en sûrEs, de remettre en cause certains des choix de ce document... et tant mieux !
- l'auteur principal étant quelque peu novice en la matière, il est fort probable que des absurdités se soient glissées dans ce document... merci de les lui signaler.

1.2 Technique

1.2.1 Permissions des adminEs

Le choix qui a été fait est de ne pas divulguer les mots de passe root des machines, mais plutôt de donner aux adminEs les permissions nécessaires par l'intermédiaire de sudo.

D'autre part, on ne peut évidemment pas administrer un serveur indépendamment des machines clientes (exemple : cas d'un serveur d'authentification) ; il est donc logique de confier aux adminEs d'un service la gestion des logiciels correspondants sur les machines clientes.

Afin de simplifier notre modélisation, on supposera que, pour administrer un service donné, unE adminE a besoin des mêmes permissions sur le serveur que sur les machines clientes.

Une solution satisfaisant à ces contraintes est proposée dans l'annexe A.

1.2.2 Outils de partage d'information

Les réplications d'informations nécessaires entre les machines sont notamment sources :

- de complexification,
- de vulnérabilités,
- d'incohérences des données entre les machines.

Pour ces raisons, on veillera à éviter ces duplications au maximum, en choisissant autant que possible des solutions capables d'utiliser les informations centralisées sur un serveur ; cette direction donnée à notre réflexion vaut en particulier pour :

– la gestion et l'authentification des comptes utilisateurices ;
– le suivi des modifications de la configuration des machines ;
– le suivi des problèmes.

Pour une réflexion sur les outils qui seront utilisés pour partager ces informations, on se référera aux annexes B et C.

2 Et concrètement... comment procéder ?

Dans cette partie, nous décrirons un mode opératoire permettant de modéliser et de mettre en place le système d'administration collective proposé dans ce document.

2.1 Audit des besoins et moyens

La validité de la modélisation dépend, en bonne partie, de la rigueur avec laquelle on effectuera les audits suivants :

- lister les services à mettre en place sur la machine ;
- établir les relations de dépendance entre ces services ;
- en déduire l'étendue de la partie commune à touTEs les admin-e-s ;
- lister les moyens humains disponibles : affinités, disponibilités coïncidentes, temps disponible, compétences, etc.

2.2 Répartition des tâches

À partir des données rassemblées à l'étape 2.1 :

- définir des sous-ensembles de tâches à effectuer, qu'on nommera dorénavant des *blocs* d'admin, qui ne se recoupent pas entre eux ; pour les matheuxEs, il s'agit de réaliser une *partition* de l'ensemble des tâches à effectuer ;
- définir des sous-ensembles de personnes, qu'on nommera dorénavant des *équipes* d'admin ; le nombre de ces équipes doit évidemment être au maximum égal au nombre de blocs... ;
- assigner une équipe à chaque bloc ;
- déduire de l'étape 2.1 les relations de dépendance entre les blocs d'admin, selon la règle suivante :

  le bloc A dépend du bloc B
  $\leftrightarrow$
  $\exists$(un élément $a \in A$ et un élément $b \in B$) tels que a dépend de b

2.3 Conception de la communisation des informations

- choix des différents groupes/users/privilèges sudo, en fonction de la répartition des tâches effectuée à l'étape 2.2.
- en fonction de l'échelle du parc informatique (quantité de machines et de services à administrer) choix des procédures et outils de partage des informations ; cf. annexes B et C.

2.4 Implémentation du modèle

Il s'agit de :

- installation et paramétrage du système d'exploitation et de la partie commune
- installation, par les adminEs concernéEs, des services nécessaires pour la suite
- mise en place des outils de partage d'information prévus à l'étape 2.3

A Distribution des droits des adminEs

On l'a vu, les adminEs doivent avoir accès aux machines clientes, avec les mêmes permissions que sur le serveur, pour pouvoir les administrer.

Comme toutE autre utilisateurice, unE adminE se connectant sur une machine cliente se verra authentifiéE sur le serveur. Puis vient le moment où ille utilise sudo ; sudo, tout d'abord, authentifie l'adminE sur le serveur central, via l'utilisation de PAM (Pluggable Authentication Modules) sur la machine cliente. Mais la difficulté est ailleurs : sudo ne sait stocker *que* dans le fichier `/etc/sudoers` les permissions détaillées attribuées aux adminEs ; afin de gérer tout ceci de façon centralisée, il nous faut utiliser un système sécurisé de partage ou réplication de fichiers pour `/etc/sudoers`. Le logiciel cfengine (http ://www.cfengine.org/) semble remplir cette tâche à merveille.

B Suivi et diffusion des changements

Dans notre cadre de travail collaboratif, les membres d'une équipe d'adminEs doivent être informéEs des modifications de configuration effectuées par l'unE d'entre elleux. Il faut donc mettre en place un système permettant non seulement d'enregistrer ces modifications, mais aussi de les faire parvenir à qui de droit.

B.1 Suivi des changements

Pour enregistrer les modifications de configuration effectuées, différentes méthodes s'offrent à nous :

- fichiers Changelog édités à la main par les adminEs ; cette solution a l'indéniable avantage de la simplicité, mais elle part du principe que les adminEs seront rigoureuxes et renseigneront systématiquement et correctement le Changelog approprié... cette supposition est peut-être un tantinet optimiste ;)
- GNU RCS (Revision Control System) sauvegarde les modifications effectuées sur les fichiers qu'on lui confie ; c'est sur ce logiciel que se base CVS, qui lui est plutôt conçu pour gérer, à travers le réseau, des arborescences complètes, alors que RCS en reste à l'échelle du fichier et d'une machine.

B.2 Diffusion des Changelogs

Afin de diffuser les modifications des Changelogs, une solution commode consiste en la création d'une liste de discussion par équipe d'admin, plus une ML commune à tou-te-s les adminEs ; ainsi, il suffira de mettre en place un système qui envoie des mails aux listes concernées en cas de modification des Changelogs ; si RCS est choisi, rcs-report fait exactement ce travail ; sinon, diffmon est une solution à étudier.

Pour éviter que ces emails ne passent en clair sur le réseau, plusieurs solutions existent, selon le cas on choisira la plus adaptée :

- inscrire aux ML l'utilisateurice de chaque adminE sur le serveur d'administration... c'est le plus simple, on n'a pas besoin de cryptage dans ce cas là car les mails ne sortent pas du serveur ;
- utiliser une clé gpg par équipe d'admin et crypter les mails avant qu'ils ne soient envoyés sur la liste ;

– utiliser la version patchée de Mailman :
cf. http ://www.nah6.com/products/secure-list/.

C Suivi des problèmes

Tout d'abord, pourquoi suivre les problèmes avec un logiciel spécialisé ? On pourrait en effet considérer cette méthode comme trop lourde, trop compliquée, cependant... un tel logiciel sert :

– pour ne pas oublier dans un coin certains problèmes, en se promettant d'y revenir plus tard ;
– pour que toutE unE chacunE puisse signaler des problèmes.

Pour ce faire, d'excellents outils libres existent ; par exemple :

– RT (perl/mysql), utilisé sur stallman.indymedia.org ; il n'est hélas pas packagé en Debian, mais sait s'authentifier sur le serveur web :
http ://www.bestpractical.com/rt/
– Bugzilla (perl/mysql), très complet en version 2.14.2 sur Woody ; cette version n'est plus supportée par l'équipe de développement... et a des trous de sécurité ; la dernière version stable (2.16.3) est dans Sarge :
http ://www.mozilla.org/projects/bugzilla/
– Debian bug tracking software (perl/ ?) ; un peu roots :
http ://www.chiark.greenend.org.uk/ ian/debbugs/
– Mantis (php/mysql) :
http ://mantisbt.sourceforge.net/
– GNU GNATS (?/txt) :
http ://www.gnu.org/software/gnats/

Idéalement, le logiciel choisi doit être capable de s'authentifier sur l'annuaire centralisé d'utilisateurices qui aura été choisi. Seul RT remplit cette condition, car il sait s'authentifier sur Apache, donc, par transitivité, sur PAM et/ou LDAP.