Tutorial Kerberos

Comprendre et mettre en place une architecture Kerberos

Sebastien.Varrette@imag.fr

Version : 0.2 – Avril 2004

Table des matières

1 Introduction : Kerberos, Kesako ?

Développé par le MIT, Kerberos [8, 4] est un système d'authentification sécurisé à tierce personne de confiance (ou TA pour Trusted Authority) conçu pour les réseaux TCP/IP.
Il ne s'agit pas en revanche d'un système d'authorisation d'accès aux ressources, bien que des extentions existent en ce sens : voir [6].
La distribution MIT [10] de Kerberos est libre. Au moment où ce document est écrit, la version courante, disponible sur `http://web.mit.edu/kerberos/dist/` est la 1.3.3.
Ce système est basé sur l'utilisation de la crytographie à clé privée. Kerberos partage avec chaque entité U du réseau une clé secrète K_U (un mot de passe dans le cas d'un utilisateur) et la connaissance de cette clé tient lieu de preuve d'identité. Des extentions existent pour l'utilisation de certificats qui permettent un meilleur passage à l'echelle (voir [11, 9, 3]).
L'authentification est négociée par le biais d'un tiers de confiance : le Key Distribution Center (KDC).

Kerberos possède un certain nombre d'avantages listés ici :
– l'authentification est sécurisée ;
– il n'y a pas de transmission de mot de passe sur le réseau ;
– le *Single Sign On* : il suffit que l'utilisateur s'authentifie une seule fois lors de la première authentification. Il n'a ainsi qu'un seul mot de passe à se souvenir et ne l'entre qu'une seule fois par jour typiquement ;

– une gestion centralisée de l'authentification ;
– c'est un standard IETF (RFC 1510 [5]) supporté par de nombreux OS (en particulier Windows 2000)

Ce document s'adresse à ceux qui souhaitent comprendre et/ou mettre en place une architecture de type Kerberos au sein d'un réseau d'entreprise.

Evidemment, comme tout tutorial, ce document est par essence incomplet. Il constitue néanmoins un bon début pour la compréhension général du principe de cette architecture et un déploiement fonctionnel.

2 Petit Lexique ...

L'utilisation de Kerberos fait appel à un vocabulaire dont les principaux termes sont énumérés ici :

KDC (Key Distribution Center) : Base de Données des clients et des serveurs (les 'principaux') ainsi que des clés privées associées.

Principal : Triplet <*primary name, instance, realm*>.

- *Primary name* : nom d'utilisateur ou du service ;
- *Instance* : rôle/groupe du primary ;
- *Realm* : domaine d'administration associé à au moins un serveur Kerberos qui stocke la 'master BD' du site/du domaine.

En pratique, on identifiera un principal de type utilisateur par une chaîne de la forme **login/staff@REALM**. Lorsqu'il s'agira d'un service, on utilisera plutôt la dénomination **service/host.imag.fr@REALM**

Client : Entité pouvant obtenir un ticket (utilisateur/hôte).

Service : Programme/ordinateur accédé sur un réseau. Ex : host (avec telnet, rsh), ftp, krbtgt (authentification), pop etc...

Ticket : Crédit temporaire permettant de vérifier l'identité du détenteur.

TGT : Ticket particulier permettant au détenteur d'obtenir d'autres tickets pour le même domaine.

keytab : Fichier contenant 1 ou plusieurs clés. Il est à la machine hôte/au service accédé ce que le mot de passe est aux utilisateurs

3 Kerberos en théorie

Cette section détaille les grandes lignes théoriques du protocole d'authentification utilisé dans Kerberos en tentant d'expliquer simplement comment et pourquoi ça marche.

Cependant, cette partie suppose un certain nombre de pré-requis mathématiques, notamment dans le domaine de la cryptologie. Le lecteur peu versé dans ce domaine et/ou qui souhaite simplement installer Kerberos ne doit pas hésiter à aller directement au §4, page 7, plus pratique.

3.1 Présentation Générale

Les étapes de l'authentification dans le modèle de Kerberos sont présentées de façon succinte dans la figure 1. Avant de détailler plus particulièrement ces étapes, il convient de donner les grandes lignes du protocole Kerberos.
Kerberos est basé sur l'utilisation de *tickets* qui serviront à convaincre une entité de l'identité d'une autre entité. Il crée également des *clés de session* qui sont données à deux participants et qui servent à chiffrer les données entre ces deux participants.

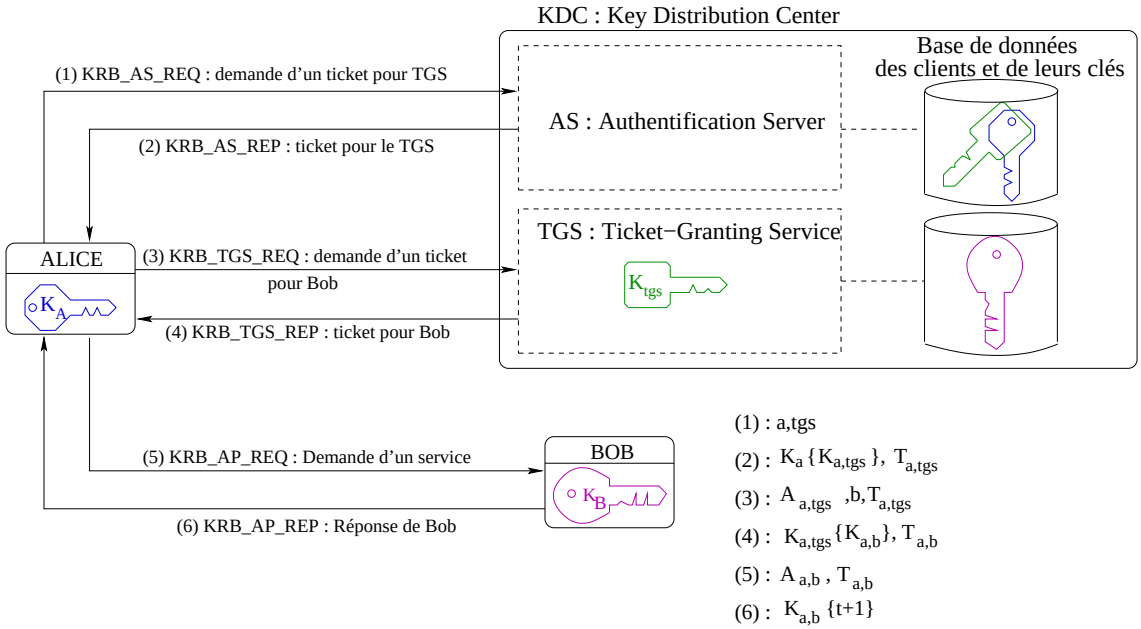

FIG. 1 – Etapes d'authentification Kerberos

On distingue deux types d'accréditations[1] :
- les **tickets** qui servent à donner au futur destinataire (Bob ou le TGS[2]), de manière sûre, l'identité de l'expéditeur (Alice) à qui le ticket a été émis. Il contient également des informations que le destinataire peut utiliser pour s'assurer que l'expéditeur qui utilise le ticket est bien celui à qui le ticket a été délivré.
- les **authentifiants** qui sont des accréditations supplémentaires présentées avec le ticket (détails dans le §3.2)

Dans la suite, on utilisera les notations données dans le tableau 1
Un ticket est de la forme suivante :

$$T_{a,s} = s, E_{K_s}(id_a, t, t_{end}, K_{a,s})$$

Il contient donc le nom du service qu'Alice souhaite utiliser (TGS ou Bob) et une liste d'informations chiffrées avec la clé secrète du serveur (donc qu'il sera le seul à pouvoir déchiffrer) :

[1] credentials en anglais
[2] Ticket-Granting Service ; dans ce cas, on parle de TGT : Ticket-Granting Ticket

a	Alice
b	Bob
id_u	information publique qui identifie u (ex : nom, @IP)
t	Date de la demande
t_{end}	Date d'expiration du ticket
K_u	Clé secrète de u
$K_{u,v}$	Clé de session pour u et v
$E_K(x)$	Fonction de chiffrement du texte x avec la clé K
$T_{u,v}$	Ticket de u pour utiliser v
$A_{u,v}$	Authentifiant de u pour v

TAB. 1 – Notations utilisées dans Kerberos

– l'identité d'Alice id_a,
– la date de la demande t,
– la date de fin de validité du ticket t_{end},
– enfin (et surtout) une clé de session $K_{a,s}$ qui sera utilisée d'une part pour l'authentifiant (voir ci-après) et d'autre part pour chiffrer les futures communications entre Alice (a) et le service s.

Alice ne peut pas déchiffrer le ticket mais peut le donner chiffré à s.

Un authentifiant ressemble quand à lui à ceci :

$$A_{a,s} = E_{K_{a,s}}(id_a, t)$$

Alice l'engendre chaque fois qu'elle veut utiliser un service (TGS ou Bob). Contrairement au ticket qu'Alice peut utiliser plusieurs fois pour accéder au service jusqu'à l'expiration du ticket, un authentifiant ne peut être utilisé qu'une seule fois. Cependant, comme Alice possède la clé de session partagée, elle peut en engendrer autant de fois qu'elle le souhaite.

3.2 Détails des messages Kerberos

On a vu sur la figure 1 les différents messages qui sont échangés. Quelques remarques préliminaires concernant ces messages :

– Les messages 1 et 2 permettent l'obtention du premier ticket TGT qu'Alice devra présenter ensuite au TGS à chaque fois qu'elle souhaite contacter un destinataire Bob.
– Les messages 3 et 4 permettent l'obtention d'un ticket de service qu'Alice devra présenter à Bob pour une demande de service.
– Les messages 5 et 6 correspondent à la demande de service qu'Alice formule à Bob et la réponse de ce dernier. Cette étape permet comme on va le voir de garantir l'authentification mutuelle d'Alice et de Bob et de leur fournir une clé de session qui leur permettra de chiffrer leurs futurs messages. C'est en ce sens qu'il faut prendre la notion de service.

Passons maintenant au contenu des messages proprement dit. Le tableau 2 en donne un apercu concis.

N^o	Message	Format
1	KRB_AS_REQ	$[a\,;\,tgs]$
2	KRB_AS_REP	$[E_{K_a}(K_{a,tgs})\,;\,T_{a,tgs}]$ $\hookrightarrow[E_{K_a}(K_{a,tgs})\,;\,tgs\,;\,E_{K_{tgs}}(id_a,t,t_{end},K_{a,tgs})]$
3	KRB_TGS_REQ	$[A_{a,tgs}\,;\,b\,;\,T_{a,tgs}]$ $\hookrightarrow[E_{K_{a,tgs}}(id_a,t)\,;\,b\,;\,tgs\,;\,E_{K_{tgs}}(id_a,t,t_{end},K_{a,tgs})]$
4	KRB_TGS_REP	$[E_{K_{a,tgs}}(K_{a,b})\,;\,T_{a,b}]$ $\hookrightarrow[E_{K_{a,tgs}}(K_{a,b})\,;\,b\,;\,E_{K_b}(id_a,t,t_{end},K_{a,b})]$
5	KRB_AP_REQ	$[A_{a,b}\,;\,T_{a,b}\,]$ $\hookrightarrow[E_{K_{a,b}}(id_a,t)\,;\,b\,;\,E_{K_b}(id_a,t,t_{end},K_{a,b})]$
6	KRB_AP_REP	$[E_{K_{a,b}}(t+1)]$

TAB. 2 – Messages dans Kerberos Version 5

Quelques remarques complémentaires sur ce tableau et les différents messages qu'il contient :

1. KRB_AS_REQ : Ce message sert simplement d'introduction à Alice. Elle y précise son nom et quel TGS elle souhaite rencontrer[3].
2. KRB_AS_REP : Le serveur d'authentification[4]cherche le client dans sa base de données. S'il le trouve, il engendre une clé de session $K_{a,tgs}$ qui devra être utilisée entre Alice et le TGS. Cette clé est d'une part chiffrée avec la clé secrète K_a d'Alice[5] : c'est la première partie du message ($E_{K_a}(K_{a,tgs})$). Ensuite, il crée un ticket $T_{a,tgs}$ pour Alice afin que celle ci puisse s'authentifier auprès du TGS. Comme on l'a déjà vu, ce ticket est chiffré avec la clé secrète K_{tgs} du TGS. Alice ne pourra pas le déchiffrer mais pourra le présenter tel quel à chaque requête au TGS. Dans ce cas particulier, le ticket est appelé TGT. Il est important de noter que seule la véritable Alice est capable de récupérer la clé de session $K_{a,tgs}$ (elle est la seule à posséder la clé secrète K_a). Ainsi, Alice dispose maintenant de la clé de session $K_{a,tgs}$ et du TGT $T_{a,tgs}$.
3. KRB_TGS_REQ : Alice doit maintenant obtenir un nouveau ticket pour chaque Bob qu'elle souhaite contacter. Pour cela, Alice contacte le TGS en lui fournissant d'une part le ticket TGT $T_{a,tgs}$ qu'elle possède déjà, et un authentifiant $A_{a,tgs}$ d'autre part (en plus du nom du serveur qu'elle souhaite contacter). L'authentifiant possède des informations formatées vérifiables à partir du ticket par le TGS et comme ces informations sont chiffrées avec la clé de session $K_{a,tgs}$, cela prouve au moins qu'Alice la connait et donc l'authentifie (d'où le nom d'authentifiant donné à $A_{a,tgs}$).
4. KRB_TGS_REP : Grâce à sa clé secrète K_{tgs}, le TGS déchiffre le ticket, récupère la clé de session $K_{a,tgs}$ et peut ainsi déchiffrer l'authentifiant $A_{a,tgs}$. Il compare le contenu de l'authentifiant avec les informations contenues dans le ticket et si tout concorde (Alice est authentifiée), il peut engen-

[3]Il peut y en avoir plusieurs
[4]AS sur la figure 1
[5]s'il s'agit d'un utilisateur humain, K_a correspond au hachage de son mot de passe

drer une clé de session $K_{a,b}$ (qui sera utilisée entre Alice et Bob) qu'il chiffre avec la clé de session $K_{a,tgs}$ et un nouveau ticket $T_{a,b}$ qu'Alice devra présenter à Bob. Après réception de ce message et déchiffrement, Alice dispose donc en plus de $K_{a,tgs}$ et de $T_{a,tgs}$ (qu'elle conserve jusqu'à expiration du ticket pour dialoguer avec TGS) d'une nouvelle clé de session $K_{a,b}$ et d'un nouveau ticket $T_{a,b}$ qu'elle pourra utiliser avec Bob.

5. KRB_AP_REQ : Maintenant, Alice est prête à s'authentifier auprès de Bob ; cela s'effectue de la même manière qu'entre Alice et le TGS[6] (cf message KRB_TGS_REQ).
6. KRB_AP_REP : Il reste maintenant à Bob à s'authentifier en prouvant qu'il a pu déchiffrer le ticket $T_{a,b}$ et donc qu'il possède la clé de session $K_{a,b}$. Il faut donc qu'il renvoie une information vérifiable par Alice chiffrée avec cette clé. Pour éviter les attaques de type replay, l'information choisie est $t+1$. En vérifiant cela, Alice est maintenant sûre de l'identité de Bob et dipose d'une clé de session $K_{a,b}$ utilisable pour chiffrer les communications entre Alice et Bob.

3.3 Les faiblesses de Kerberos

Il n'y a pas de système parfait et il s'agit d'etre bien conscient des limitations de ce système. Les documents[1, 7], quoiqu'ancien, donne les grandes lignes des faiblesses du système Kerberos :

- *attaque par répétition* : bien que les datations soient supposées éviter cela, les messages peuvent être rejoués pendant la durée de vie des tickets (qui est d'environ 8 heures).
- *services de datation* : les authentifiants dépendent du fait que toutes les horloges du réseau soient plus ou moins synchronisées. Si l'on peut tromper un ordinateur quand à l'heure réelle, alors les anciens authentifiants peuvent être rejoués. La plupart des protocoles de maintien du temps en réseau ne sont pas sûrs, ce qui peut donc être un sérieux défaut.
- *Paris de mots de passe* : il s'agirait pour un intrus de collectionner les premières moitié du message KRB_AS_REP ($E_{K_a}(K_{a,tgs})$ pour prévoir la valeur de K_a (en général, $K_a = H(Password)$). En pariant sur le mot de passe $\tilde{P}$, l'intrus peut calculer $\tilde{K}_a$, déchiffrer et obtenir $\tilde{K}_{a,tgs}$ et vérifier la pertinence de son choix en déchiffrant l'authentifiant $A_{a,tgs} = E_{K_{a,tgs}}(id_a, t)$ dont il connait le contenu (il connait au moins id_a).
- *spoofing login* : on peut envisager une attaque où tous les logiciels Kerberos clients sont remplacés par une version qui non seulement réalise le protocole Kerberos mais enregistre également les mots de passe.

4 Kerberos en pratique

Cette section détaille la procédure d'installation de la distribution MIT de Kerberos sur une machine Linux. Au moment où ce document est écrit, La version

[6] Bob n'est rien d'autre qu'un serveur particulier

courante est la 5.1.3.3. Celle-ci inclut les logiciels clients/serveurs et est fournie pour la plupart des plate-formes (UNIX World, Mac OS et Windows).

4.1 Choix du matériel

– Il faudra dédié une machine au serveur KDC. En tant que tel, ce sera un point privilégié d'attaque et devra donc être aussi sûr que possible. **Si le KDC est compromis, toute l'architecture est compromise !**. Pour rentrer un peu plus dans les détails, ce serveur devra vérifier les propriétés suivantes :

  1. La machine devra être physiquement sécurisée ;
  2. le système d'exploitation devra être mis a jour avec les derniers patchs ;
  3. aucun utilisateur, hormis l'administrateur du serveur Kerberos, ne doit avoir de compte sur cette machine ;
  4. le nombre de processus (démons) qui tournent sur cette machine doit être limité au maximum ;
  5. Il faudra éventuellement prévoir d'autres machines qui feront office de serveurs secondaire. Des mécanismes de duplications existent et permettent de pallier aux éventuels problèmes hardware du serveur principal. Pour les utiliser, il convient de disposer d'une seconde machine qui jouera le rôle de KDC esclave.

  Au niveau hardware, ce serveur n'a pas besoin d'être une machine dernier cris. Par exemple, on pourra se contenter d'un PIII 500 Mhz.
– les autres machines sont simplement celles des utilisateurs du réseaux sur lesquels les applications kerberos clientes devront être installé (la procédure d'installation est détaillée dans le §4.3)

4.2 Installation du serveur Kerberos

Dans Kerberos, il convient en fait de distinguer deux types de serveurs :

1. le KDC d'une part (démon `krb5kdc` écoutant sur le port 88 par défaut), en charge de l'"Authentication Service" (AS) et du "Ticket Granting Service" (TGS) (voir figure 1, page 4). Comme expliqué dans le §3, le KDC possède une copie de chaque mot de passe associé à chaque principal.
2. le serveur d'administration (`kadmind`, utilisant le port 749 par défaut) qui permet de manipuler la base de donnée Kerberos. La plupart du temps, ce serveur d'administration est lancé sur la même machine que le KDC. C'est ce qui sera supposé ici et c'est pourquoi on désignera cette machine comme étant "le" serveur Kerberos.

On utilise aussi le terme de *serveur d'application* qui désigne généralement au programme kerberisé avec lequel un client va communiquer à partir de tickets Kerberos. Le démon telnet Kerberos (`telnetd`) est un exemple de serveur d'application.

4.2.1 Installation de l'OS

Il convient déjà d'installer un Linux minimal, sans X ni aucun programme GUI. (de bons guides pour La distribution Debian pourront être trouver dans [12, 2]). SSH est optionnel. S'il permettra ensuite une administration a distance, l'absence de toute possibilité de login à distance augmentera significativement la sécurité du KDC.

4.2.2 Installation par package

C'est sans doute l'installation la plus aisée.
- *Sous Mandrake/Redhat* : les packages concernés sont `krb5-server` et `krb5-libs`. La commande `urpmi <nom_du_package>` devrait permettre de les installer.
- *Sous Debian* : il s'agit cette fois-ci de considérer les packages `krb5-kdc` et `krb5-admin-server`. Ici, il faut utiliser la commande `apt-get install <nom_du_package>`.

4.2.3 Installation par compilation des sources

Les sources de la dernière version disponible peut être trouvée sur le site du MIT : `http://web.mit.edu/kerberos/dist/`
vous récupérerez ainsi par exemple le fichier `krb5-1.3.3.tar`.

1. **Décompressez ce fichier** :
   `[seb@falkor] ~> tar xvf krb5-1.3.3.tar`
   Cette décompression fournit deux fichiers :
   - `krb5-1.3.3.tar.gz` : l'archive compressée des sources ;
   - `krb5-1.3.3.tar.gz.asc` : la signature associée.
2. **Vérifiez la signature** (cela suppose que gnupg[7] est installé) :

   ```
   [seb@falkor] ~> gpg --verify krb5-1.3.3.tar.gz.asc
   gpg: Signature faite mar 06 avr 2004 22:38:27 CEST avec la clé RSA ID F376813D
   gpg: Impossible de vérifier la signature: clé publique non trouvée
   ```

   Il faut ici commencer par récupérer la clé publique (d'identifiant $OxF376813D$) qui a servie à signer le fichier. Pour cela, il suffit de taper la commande suivante :

   ```
   [seb@falkor] ~> gpg --keyserver pgp.mit.edu --recv-key F376813D
   gpg: clé F376813D: clé publique "Tom Yu <tlyu@MIT.EDU>" importée
   gpg:        Quantité totale traitée: 1
   gpg:                       importée: 1  (RSA: 1)
   ```

   Vous pouvez alors enfin vérifier la validité de l'archive récupérée :

   ```
   [seb@falkor] ~> gpg --verify krb5-1.3.3.tar.gz.asc
   gpg: Signature faite mar 06 avr 2004 22:38:27 CEST avec la clé RSA ID F376813D
   gpg: Bonne signature de "Tom Yu <tlyu@MIT.EDU>"
   ```

3. **Décompresser les sources** :

[7] `apt-get install gnupg` sous Debian

```
[seb@falkor] ~> tar xvzf krb5-1.3.3.tar.gz
[seb@falkor] ~> cd krb5-1.3.3
[seb@falkor] krb5-1.3.3> ls -l
total 48
drwxr-xr-x    9 seb      equipar      4096 2004-04-06 22:07 doc
-rw-r--r--    1 seb      equipar     37729 2004-04-01 00:44 README
drwxr-xr-x   18 seb      equipar      4096 2004-04-06 22:06 src
```

La lecture du `README` est recommandée. Le répertoire `doc/` contient la documentation sous différent format, en particulier :
- `doc/install-guide.ps` : la doc d'installation ;
- `doc/admin-guide.ps` : la doc administrateur ;
- `doc/user-guide.ps` : la doc utilisateur.

4. **Lancer la compilation** : Commencer par créer un répertoire racine où tous les fichiers générés seront installés[8] :
`[root@falkor] # mkdir -p /usr/local/Kerberos`.
Cela complique un peu les choses mais a l'avantage de centraliser tous les fichiers installés.
ATTENTION ! Il faut avoir installer le package ncurses-devel[9] sinon la compilation va produire une erreur[10] !
Ensuite, voici les commandes de compilation qui devraient fonctionner :

```
[root@falkor] # cd src/
[root@falkor] # ./configure CC=gcc LDFLAGS=-lncurses \
                --prefix=/usr/local/Kerberos --without-krb4
[root@falkor] # make
[root@falkor] # make check
[root@falkor] # make install
```

Le répertoire `/usr/local/Kerberos/` doit maintenant ressembler à cela :

```
[root@falkor] # ls -l /usr/local/Kerberos/
total 28
drwxr-sr-x    2 root     staff        4096 avr 21 17:54 bin
drwxr-sr-x    4 root     staff        4096 avr 21 17:53 include
drwxr-sr-x    2 root     staff        4096 avr 21 17:53 lib
drwxr-sr-x    5 root     staff        4096 avr 21 18:57 man
drwxr-sr-x    2 root     staff        4096 avr 21 17:54 sbin
drwxr-sr-x    4 root     staff        4096 avr 21 17:53 share
```

Quelques manipulations de post-compilation sont nécessaires :
- Faire correspondre les pages du man : il suffit d'ajouter la ligne :
`MANDATORY_MANPATH  /usr/local/Kerberos/man`
dans le fichier `/etc/manpath.config`.
- Ajouter le chemin des binaires dans la variables d'environnement `$PATH` :
pour cela, completer dans `/etc/profile` la valeur de PATH par `/usr/local/Kerberos/bin`.

[8] Il s'agit par défaut de `/usr/local/`

[9] `apt-get install libncurses5-dev` sous Debian

[10] `[...]telnet.c :783 :undefined reference to 'tgetent'` C'est ce qui explique que l'option '-lncurses' soit passée à l'éditeur de lien dans le configure

Faites de même dans le fichier `/root/.profile` en ajoutant cette fois ci `/usr/local/Kerberos/sbin`

– placer les fichiers de configuration au bons endroits :

```
[root@falkor] # ls
doc README src
[root@falkor] # cp src/config-files/krb5.conf /etc/
[root@falkor] # mkdir -p /usr/local/Kerberos/var/krb5kdc
[root@falkor] # cp src/config-files/kdc.conf /usr/local/Kerberos/var/krb5kdc/
```

4.2.4 Choix d'un nom de domaine Kerberos

Le serveur Kerberos gèrera un domaine qu'il faudra nommer. La convention veut que ce nom de domaine soit en majuscules et corresponde au nom de domaine réseau (par exemple, les machines du domaine `exemple.imag.fr` seront dans le domaine Kerberos `EXEMPLE.IMAG.FR`).
La topologie des domaines Kerberos doit refléter l'administration système plutôt que la topologie physique du réseau.
mirror system management topology rather than physical network topology.

4.2.5 Configuration de /etc/krb5.conf

4.3 Installation sur les machines clientes

5 Quelques liens utiles...

Si les (nombreuses) références proposées dans ce document ne vous suffisent pas (ou si vous ne voulez pas perdre de temps à les analyser), voici les liens les plus utiles pour comprendre/installer Kerberos. Tous ces liens référencent des sites en anglais (c'est d'ailleurs l'une des raisons d"être de ce tutorial)...

– `http://web.mit.edu/kerberos/` : le site officiel pour la distribution Kerberos du MIT
– `http://www.cmf.nrl.navy.mil/CCS/people/kenh/kerberos-faq.html` : FAQ sur Kerberos
– `http://www.isi.edu/~brian/security/kerberos.html` : Brian Tung, "The Moron's Guide to Kerberos" ; un tutorial en anglais.
–

Références

[1] S. M. Bellovin and M. Merritt. Limitations of the kerberos authentication system. *SIGCOMM Comput. Commun. Rev.*, 20(5) :119–132, 1990. `http://csrc.nist.gov/publications/secpubs/kerblim.ps`

[2] A. De Lattre. Formation debian gnu/linux. `http://people.via.ecp.fr/~alexis/formation-linux/`

[3] A. Harbitter and D. A. Menasce. The performance of public key-enabled kerberos authentication in mobile computing applications. In *ACM Confe-*

rence on Computer and Communications Security, pages 78–85, 2001. citeseer.ist.psu.edu/article/harbitter01performance.html

[4] B. Kohl and T. Neuman, C.and T'so. The evolution of the kerberos authentication system. In *Distributed Open Systems*, pages 78–94, Tromso, Norway, 1994. IEEE Computer Society Press. ftp://athena-dist.mit.edu/pub/kerberos/doc/krb_evol.PS

[5] J. Kohl and C. Neuman. RFC 1510 : The Kerberos Network Authentication Service (Version 5). Technical report, Massachusetts Institute of Technology, September 1993. ftp://ftp.isi.edu/in-notes/rfc1510.txt

[6] C Neuman. Proxy-based authorization and accounting for distributed systems. In *Proceedings of the 13th International Conference on Distributed Computing Systems*, pages 283–291, May 1993.

[7] C. Neuman and S. Stubblebine. A note on the use of timestamps as nonces. *Operating Systems Review*, 27(2) :10–14, 1993. http://www.alw.nih.gov/Security/FIRST/papers/authent/ntn.ps

[8] C. Neuman and T. Ts'o. Kerberos : An authentication service for computer networks. *IEEE Communications Magazine*, 32(9) :33–38, September 1994. http://gost.isi.edu/publications/kerberos-neuman-tso.html

[9] M.A. Sirbu and J.C.-I. Chuang. Distributed authentication in kerberos using public key cryptography. In *Symposium on Network and Distributed System Security*, pages 134–143, San Diego, California, 1997. http://www.isoc.org/isoc/conferences/ndss/97/sirbu_sl.pdf

[10] MIT Kerberos Team. Kerberos : The network authentication protocol. http://web.mit.edu/kerberos/

[11] B. Tung, C. Neuman, M. Hur, A. Medvinsky, S. Medvinsky, J. Wray, and J. Trostle. Public key cryptography for initial authentication in kerberos (rfc 1510bis). Technical report, USC/ISI – Microsoft, 2004. draft-ietf-cat-kerberos-pk-init-19.txt

[12] S. Varrette. Tutorial d'installation d'une debian sur votre machine, 2003. http://www-id.imag.fr/~svarrett/Tutorial/install_debian.html