krb5.conf

Fichier de configuration Kerberos

Ce fichier contient la configuration Kerberos, incluant l'emplacement des serveurs KDC et admin pour les domaines Kerberos, les valeurs par défaut pour le domaine courant et pour les applications kerberos, et les mappages de nom d'hôte dans les royaumes Kerberos. Normallement, ce fichier est dans **/etc** mais son emplacement peut être spécifié dans la variable **KRB5_CONFIG**.

Structure

Les noms de sections sont entre crochets, chaque section peut contenir 0 ou plusieurs relations, sous la forme :

foo = bar

ou :

```
fubar = {
 foo = bar
 baz = quux
}
```

Placer un ***** à la fin d'une ligne indique que c'est la valeur final pour le tag. Cela signifie que ni le reste du fichier de configuration, ni un autre fichier de configuration ne sera vérifié pour une autre valeur pour ce tag.

Par exemple, les lignes suivantes :
foo = bar*
foo = baz
Alors la seconde valeur ne sera jamais lue.
Le fichier krb5.conf peut inclure d'autres fichiers en utilisant une des directives suivantes au début d'une ligne :
include FILENAME
includedir DIRNAME
FILENAME et **DIRNAME** doivent être des chemin absolus. Le fichier ou le répertoire nommé doit exister et lisible. Inclure un répertoire inclue tous les fichiers dans le répertoire dont le nom consiste uniquement de caractères alphanumériques, "-" ou "_". Les fichiers de profils sont syntaxiquement indépendants de leurs parent, donc chaque fichier inclus doit commencer avec un en-tête de section. Le fichier krb5.conf peut spécifier que la configuration devrait être obtenue depuis un module chargeable, au lieu du fichier lui-même, en utilisant le directive suivante au début de la ligne avant toute section headers :

module MODULEPATH :RESIDUAL

MODULEPATH peut être relatif au chemin de librairies de l'installation de krb5, ou peut être un chemin absolus. **RESIDUAL** est fournis au module à l'initialisation. Si krb5 utilise une directive module, kdc.conf devrait également l'utiliser.

Sections

krb5.conf peut contenir les sections suivantes :

[libdefaults] Paramètres utilisés par la libraire Kerberos v5

[realms] information et paramètres de contact spécifique au domaine

[domain_realm] Map les noms d'hôte serveur à des domaines Kerberos

[capaths] Chemins d'authentification pour les inter-domaines non-hiérarchiques

[appdefaults] Paramètres utilisés par certaines applications Kerberos

[plugins] Contrôle l'enregistrement des plugins

Additionnellement, krb5.conf peut inclure une des relations décrites dans kdc.conf, mais n'est pas une pratique recommandée.

[libdefaults]

allow_weak_crypto à false (défaut), les type de chiffrements faibles seront exclus des listes **default_tgs_enctypes**, **default_tkt_enctypes**, et **permitted_enctypes**.

ap_req_checksum_type Un entier qui spécifie le type de checksum AP-REQ à utiliser dans l'authentifiant. Doit être non définis pour le checksun approprié pour la clé de chiffrement soit utilisé.

canonicalize A true (défaut : false), la requête du ticket initial va demander la canonicalisation du nom du principal, et que répondre avec des principaux du client différent du principal demandé sera accepté.

ccache_type Détermine le format du cache d'accéditifs créés par kinit ou autre. défaut : 4, qui représente le format le plus courant.

clockskew Définis la quantité maximum permise de décalage d'horloge en seconde que la librairie va tolérer. Défaut : 300.

default_ccache_name Cette relation spécifie le nom du cache d'accréditifs par défaut. Défaut : DEFCCNAME.

default_client_keytab_name Cette relation spécifie le nom du keytab par défaut pour obtenir les accréditifs du client. Défaut : DEFCKTNAME.

default_keytab_name Cette relation spécifie le nom du keytab par défaut à utiliser par les serveurs d'application tels que sshd. Défaut : DEFKTNAME.

default_realm Identifie le domaine Kerberos par défaut pour le client. Si non définis, un domaine doit être spécifié avec tout principar Kerberos en invoquant un programme tel que kinit

default_tgs_enctypes Identifie la liste supportée de type de chiffrement de clé de session que le client devrait demander dans un TGS-REQ, par ordre de préférence. Défaut : aes256-cts-hmac-sha1-96 aes128-cts-hmac-sha1-96 des3-cbc-sha1 arcfour-hmac-md5 camellia256-cts-cmac camellia128-cts-cmac des-cbc-crc des-cbc-md5 des-cbc-md4

default_tkt_enctypes Identifie la liste supportée de type de chiffrement de clé de session que le client devrait demander dans un AS-REQ, par ordre de préférence. Défaut : aes256-cts-hmac-sha1-96 aes128-cts-hmac-sha1-96 des3-cbc-sha1 arcfour-hmac-md5 camellia256-cts-cmac camellia128-cts-cmac des-cbc-crc des-cbc-md5 des-cbc-md4

dns_canonicalize_hostname Indique si la recherche de noms sera utilisée pour canoniser les noms d'hôte à utiliser dans les nom principal de service. Définis ce flag à false pour améliorer la sécurité en réduisant le dépendance à DNS. Défaut : true.

dns_lookup_kdc Indique si les enregistrements DNS SRV dervaient pour localiser le KDC et d'autre serveurs pour un domaine, s'il ne sont pas listés par krb5.conf. (noter que l'entrée admin_server doit être dans krb5.conf pour pouvoir contacter kadmin, parce que l'implémentation DNS pour kadmin est incomplète).

extra_addresses Permet à un ordinateur d'utiliser plusieurs adresses locales, pour permettre à Kerberos de travailler dans un réseau qui utilise le NAT. Les adresses devraient être listée séparées par une virgule. n'a pas d'effet si **noaddresses** est à true.

forwardable À true, les tickets initiaux ne seront pas forwardable par défaut, si permis par le KDC. Défaut : false

ignore_acceptor_hostname En acceptant les contextes de sécurité GSSAPI ou krb5 pour les principaux de service basés sur l'hôte, ignore tout nom d'hôte passé par l'application appelante, et permet aux clients d'authentifier tout principal de service dans le keytab correspondant au nom du service et au nom du domaine. peut compromettre la sécurité des environnements d'hôte virtuels. Défaut : false.

k5login_authoritative À true, les principaux doivent listés dans le fichier k5login de l'utilisateur pour avoir un accès login, si un fichier .k5login existe. Si ce flag est à false, un principal peut avoir l'accès login au travers d'autre mécanismes même si un k5login existe mais ne liste pas le principal. Défaut : true.

k5login_directory Si définis, la librairie va rechercher le fichier k5login de l'utilisateur local dans le répertoire spécifié, avec un nom de fichier correspondant au nom de l'utilisateur local. Sinon, recherche un fichier nommé .k5login dans le home de l'utilisateur. Pour des raisons de sécurité, .k5login doit être possédé par l'utilisateur local ou par root.

kdc_req_checksum_type Un entier qui spécifie le type de checksum à utilise pour les demande KDC, pour la compatibilité avec de vieille version de KDC.

noaddresses À true, les demandes pour les tickets initiaux ne seront pas faites avec des jeux de restriction d'adresse, permettant aux tickets d'être utilisés via les NAT. Défaut : true.

permitted_enctypes Identifie tous les types de chiffrement permis pour le chiffrement de clé de session. Défaut : aes256-cts-hmac-sha1-96 aes128-cts-hmac-sha1-96 des3-cbc-sha1 arcfour-hmac-md5 camellia256-cts-cmac camellia128-cts-cmac des-cbc-crc des-cbc-md5 des-cbc-md4

plugin_base_dir Répertoire de base où se trouvent les plugins krb5. Défaut : krb5/plugins

preferred_preauth_types Permet de définir le type de pré-authentification préféré du client. Défaut : "17, 16, 15, 14" qui force libkrb5 à utiliser PKINIT si supporté.

proxiable À true, les tickets initiaux seront mandatables, si permis par le KDC. Défaut : false

rdns À true, la recherche de noms inversée sera utilisée en plus de la recherche de nom pour canoniser les nom d'hôte à utiliser dans les noms de principaux de service. Si **dns_canonicalize_hostname** est à false, ce flag n'a pas d'effet. défaut : true.

realm_try_domains Indique si les composants de domaine des hôte devraient être utilisés pour déterminer de domaine Kerberos de l'hôte. La valeur de cette variable est un entier : -1 signifie ne pas rechercher, 0 tente le domaine de l'hôte, 1 tente le domaine immédiatement parent, etc. Le mécanisme usuel de la librairie pour localiser les domaines Kerberos est utilisé pour déterminer si un domaine est un domaine valide, qui peut impliquer de consulter DNS si **dns_lookup_kdc** est mis. Défaut : -1

renew_lifetime Définis la durée de vie de renouvellement pour les demandes de tickets initiaux. Défaut : 0

safe_checksum_type Un entier qui spécifie le type de checksum à utiliser pour les demandes KRB-SAFE. Défaut : 8 (RSA MD5 DES). Ce champ est ignoré quand ça valeur est incompatible avec le type de clé de session.

ticket_lifetime Durée de vie par défaut pour les demandes de tickets initiaux. Défaut : 1 jour

udp_preference_limit En envoyant un message au KDC, la librairie tente d'utiliser TCP avant UDP si la taille du message est supérieur à **udp_preference_limit**.

verify_ap_req_nofail À true, une tentative de vérifier des accréditifs initiaux va échouer si la machine client n'a pas de keytab. Défaut : false

[realms]

Chaque tag dans cette section est le nom d'un domaine Kerberos. La valeur du tag est une sous-section qui définis les propriétés de ce domaine. Pour chaque domaine, les tags suivants peuvent être spécifiés dans la sous-section :

admin_server Identifie l'hôte où le serveur d'administration réside. Typiquement, c'est le serveur maître. Ce tag doit avoir une valeur pour permettre la communication avec le serveur kadmind

auth_to_local Permet de définir un règle générale pour le mappage des noms des principaux aux nom des utilisateurs locaux. Est utilisé s'il n'y a pas de mappage explicite pour le nom principal. Les valeurs possible sont :

RULE :exp Le nom local sera formulé depuis exp. Le format est **[n :string](regexp)s/pattern/replacement/g**. L'entier **n** indique combien de composants le principal cible devrait avoir. Si cela matche, alors un chaîne sera formée depuis string, en substituant le domaine du principal pour **$0** et le **n**-ième composant du principal pour **$n** (exemple, si le principal était **johndoe/admin** alors **[2 :$2$1foo]** va résulter dans la chaîne **adminjohndoefoo**). Si la chaîne matche regexp, alors la substitution de commande **s//[g]** ne va jamais parcourir la chaîne. le **g** optionnel matche toutes les occurences.

DEFAULT Le nom du principal sera utilisé tel que le nom d'utilisateur. Si le principal a plus d'un composant ou n'est pas le domaine par défaut, cette règle n'est pas applicable et la conversion va échouer.

Par exemple :

```
[realms]
 ATHENA.MIT.EDU = {
  auth_to_local = RULE:[2:$1](johndoe)s/^.*$/guest/
  auth_to_local = RULE:[2:$1;$2](^.*;admin$)s/;admin$//
  auth_to_local = RULE:[2:$2](^.*;root)s/^.*$/root/
  auto_to_local = DEFAULT
 }
```

va résulter en tout principal sans **root** ou **admin** comme second composant à traduire avec la règle par défaut. Un principal avec un second composant **admin** va devenir son premier composant. **root** sera utilisé comme nom local pour tout principal avec un second composant **root**. L'exception à ces règles sont tout principal **johndoe/*** qui sera toujours le nom local **guest**.

auth_to_local_names Cette sous-section vous permet de définir explicitement des mappages de noms de principaux à des noms locaux. Le tag est le nom mappé, et la valeur est l'utilisateur local

default_domain Ce tag spécifie le domaine utilisé pour étendre les noms d'hôte en traduisant les principaux Kerberos v4 en principaux v5.

kdc Le nom ou l'adresse d'un hôte hébergeant un KDC pour ce domaine. Un numéro de port optionnel peut être inclus. Si le nom ou l'adresse contient des " :", utiliser des crochets pour distinguer le " :" de l'adresse du numéro de port. Pour que la machine puisse communiquer avec le KDC, ce tag doit être donné, ou il doit y avoir des enregistrements DNS SRV.

kpasswd_server Pointe vers le serveur où tous les changements de mot de passe sont effectués. S'il n'y a pas de telles entrées, le port 464 de l'hôte **admin_server** sera tenté.

master_kdc Identifie le KDC maître. Actuellement, ce tag est utilisé dans un cas : si la tentative d'obtention des accréditifs échoue à cause d'un mot de passe invalide, le client va tenter de contacter le KDC maître, dans le cas où le mot de passe de l'utilisateur vient juste d'être changé.

[domain_realm]

Cette section fournis une traduction depuis un nom de domaine ou un nom d'hôte en un nom de domaine Kerberos. Le nom du tag peut être un nom d'hôte ou un nom de domaine, où les noms de domaine sont identifiés par un préfixe point (**.**). La valeur est le nom de domaine Kerberos pour un hôte ou un domaine particulier. Une relation de nom d'hôte fournis implicitement la relation de nom de domaine correspondante, sauf si une relation de nom de domaine explicite est fournie. Le domaine Kerberos peut être identifié soit dans la section realm ou en utilisant les enregistrements DNS SRV. Les noms d'hôte et domaine devraient être en minuscules. Par exemple :

```
[domain_realm]
crash.mit.edu = TEST.ATHENA.MIT.EDU
.dev.mit.edu = TEST.ATHENA.MIT.EDU
mit.edu = ATHENA.MIT.EDU
```

mappe l'hôte **crash.mit.edu** dans le domaine Kerberos **TEST.ATHENA.MIT.EDU**. La seconde entrée map tous les hôtes sous le domaine **dev.mit.edu** dans le domaine Kerberos **TEST.ATHENA.MIT.EDU** sauf l'hôte **dev.mit.edu** qui est mappé au domaine **ATHENA.MIT.EDU**.

Si aucune traduction ne s'applique au nom d'hôte utilisé pour un principal de service pour une demande de ticket de service, la librairie va tenter d'obtenir un référant au domaine approprié depuis le KDC du domaine du client. Si cela échoue, le domaine de l'hôte est considéré être la portion de domaine du nom d'hôte convertit en majuscules, sauf le paramètre **realm_try_domains** dans la section libdefaults qui force un domaine parent différent à utiliser.

[capaths]

Pour pouvoir effectuer une authentification inter-domaine directe (non hiérarchique), une configuration est nécessaire pour déterminer les chemins d'authentification entre les domaines.

Un client va utiliser cette section pour trouver le chemin d'authentification entre son domaines et le domaine du serveur. Le serveur va utiliser cette section pour vérifier le chemin d'authentification utilisé par le client, en vérifiant le champ transited.

Il y a un tag pour chaque domaine client participant, et chaque tag a des sous-tags pour chaque domaine serveur. La valeur des sous-tags est un domaine intermédiaire qui peuvent participer dans l'authentification inter-domaine. Les sous-tags peuvent être répétés s'il y a plus d'un domaine intermédiaire. Une valeur "." signifie que le 2 domaines partagent les clés directement, et aucun intermédiaire ne devrait être autorisé à participer.

Seul ces entrées qui sont nécessaire dans le client ou le serveur doivent être présent. Un client a besoin d'un tag pour son domaine local avec des sous-tags pour tous les domaines des serveurs auquel il aura besoin de s'authentifier. Un sereur a besoin d'un tag pour chaque domaine des clients qu'il désert, avec un sous-tag pour chaque domaine serveur.

Par exemple, **ANL.GOV**, **PNL.GOV** et **NERSC.GOV** veulent utiliser le domaine **ES.NET** comme domaine intermédiaire. ANL a un sous-domaine **TEST.ANL.GOV** qui va s'authentifier avec **NERSC.GOV** mais pas **PNL.GOV**. La section capaths pour les systèmes **ANL.GOV** ressembleraient à :

```
[capaths]
 ANL.GOV = {
  TEST.ANL.GOV = .
  PNL.GOV = ES.NET
  NERSC.GOV = ES.NET
  ES.NET = .
 }
 TEST.ANL.GOV = {
  ANL.GOV = .
 }
 PNL.GOV = {
  ANL.GOV = ES.NET
 }
 NERSC.GOV = {
  ANL.GOV = ES.NET
 }
 ES.NET = {
  ANL.GOV = .
 }
```

La section capaths du fichier de configuration utilisé dans les systèmes **NERSC.GOV** ressembleraient à :

```
[capaths]
 NERSC.GOV = {
  ANL.GOV = ES.NET
  TEST.ANL.GOV = ES.NET
  TEST.ANL.GOV = ANL.GOV
  PNL.GOV = ES.NET
  ES.NET = .
 }
 ANL.GOV = {
  NERSC.GOV = ES.NET
 }
 PNL.GOV = {
  NERSC.GOV = ES.NET
 }
 ES.NET = {
  NERSC.GOV = .
 }
 TEST.ANL.GOV = {
  NERSC.GOV = ANL.GOV
  NERSC.GOV = ES.NET
 }
```

Quand un sous-tag est utilisé plus d'une fois dans un tag, les clients utilisent l'ordre des valeurs pour déterminer le chemin. L'ordre des valeurs n'est pas important pour les serveurs.

[appdefault]

Chaque tag dans cette section nomme une application Kerberos v5 ou un option qui est utilisé par certaines applications Kerberos v5. La valeur du tag définis le fonctionnement par défaut pour cette application.

par exemple :

```
[appdefaults]
 telnet = {
  ATHENA.MIT.EDU = {
   option1 = false
  }
 }
 telnet = {
  option1 = true
  option2 = true
 }
 ATHENA.MIT.EDU = {
  option2 = false
 }
 option2 = true
```

Les 4 manières présentées de spécifier la valeur d'une option sont affichés dans l'ordre de précédence descendante. Dans cet exemple, si telnet fonctionne dans le domaine **EXAMPLE.COM**, il devrait, par défaut, avoir option1 et option2 à true. Cependant, un programme telnet dans le domaine **ATHENA.MIT.EDU** devrait avoir l'option1 à false et option2 à true. Tout autre programme dans **ATHENA.MIT.EDU** devrait avoir l'option2 à false par défaut. Tout programme fonctionnant dans d'autre domaine devraient avoir option2 à true.

La liste d'options spécifiable pour chaque application peut être trouvée dans les man des applications.

[plugins]

Les tags dans cette section peuvent être utilisées pour enregistrer les modules dynamique et pour activer/désactiver les modules. Tous les interface activable n'utilisent pas cette section. Celles qui l'utilise sont documentées ici. Chaque interface pluggable correspond à une sous-section. Toutes les sous-sections supportent les même tags :

[SECTION]

[SECTION] name="-" table="paragraphes" imbrication="0"

Ce tag peut avoir plusieurs valeurs. S'il y a des valeur pour ce tag, alors les modules nommés seront désactivés pour l'interface pluggable.

Ce tag peut avoir plusieurs valeurs. S'il y a des valeur pour ce tag, alors les modules nommés seront activés pour l'interface pluggable.

Ce tag peut avoir plusieurs valeurs. Chaque valeur est une chaîne sous la forme **modulename :pathname**. chaque module est enregistré comme module dynamique pour l'interface pluggable. Si pathname n'est pas un chemin absolus, il est considéré relatif à **plugin_base_dir**.

Pour les interfaces pluggable où l'ordre des modules est importante, les modules enregistrés avec un tag module viennent en premier, dans l'ordre qu'il sont enregistrés. Si **enable_only** est utilisé, alors l'ordre de ces tags remplace l'ordre des modules normal.

Les sections suivantes sont actuellement supportées dans la section plugins.

interface ccselect

La sous-section ccselect contrôle les modules pour la sélection du cache d'accréditifs dans une collection de cache. En plus des modules dynamique enregistrés, les modules embarqués suivant existent :

k5identity Utilise un fichier **.k5identity** dans le home de l'utilisateur pour sélectionner un principal

realm Utilise le domaine de service pour deviner un cache approprié depuis la collection.

interface pwqual

Cette sous-section contrôle les modules pour l'interface de qualité de mot de passe, qui est utilisé pour rejeter les mots de passe faibles lors des changements de mot de passe. Les modules embarqués suivant existent pour cette interface :

dict Vérifie avec un fichier dictionnaire du domaine

empty Rejète les mots de passe vide

hesiod Vérifie avec les informations utilisateurs stockés dans Hesiod (seulement si Kerberos est construit avec le support de Hesiod)

princ Vérifie avec les composant du nom du principal

interface kadm5_hook

Fournis des plugins avec des informations sur la création de principaux, la modification, changement de mot de passe et suppression. Cette interface peut être utilisée pour écrire un plugin pour synchroniser Kerberos MIT avec celui de Microsoft.

interface clpreauth et kdcpreauth

Ces interfaces permettent aux modules de fournir des mécanisme de pré-authentification au client et au KDC. Les modules intégrés suivant existent pour ces interfaces :

pkinit Implémente le mécanisme de pré-authentification pkinit

encrypted_challenge Implémente FAST

encrypted_timestamp Implément le mécanisme de timestamp chiffré

interface hostrealm

Cette section contrôle les modules pour l'interface host-to-realm, qui affecte le mappage locale des noms d'hôte et le choix du domaine par défaut. Les module intégré suivant existent pour cette interface :

profile Consulte la section [domain_realm] du profile pour les mappages host-to-realm autoritatifs, et la variable **default_realm** pour le domaine par défaut.

dns Recherche les enregistrements DNS pour les mappages et le domaine par défaut. n'opère que si **dns_lookup_realm** est à true.

domain Applique de l'heuristique pour les mappages host-to-realm. Implémente la variable **realm_try_domains**, et utilise le domaine parent en majuscule du nom d'hôte si cela ne produit pas de résultat.

interface localauth

Contrôle les modules pour l'interface d'autorisation local, qui affecte la relation entre les principaux Kerberos et les comptes système locaux. Les modules intégrés suivant existent pour cette interface :

default Implémente le type DEFAULT pour les valeurs **auth_to_local**

rule Implémente le type RULE pour les valeur **auth_to_local**

names Recherche un mappage **auth_to_local_names** pour le nom du principal

k5login Autorise un principal à un compte local en accord avec le .k5login du compte

an2ln Autorise un principal à un compte local si le nom du principal se mappe au nom local

Options PKINIT

Les options suivante sont spécifique à PKINIT. Ces valeur peuvent être spécifiées dans [libdefaults] comme valeurs globales par défaut, ou dans une sous-section spécifique au domaine, ou peuvent être spécifiés comme valeurs spécifique au domaine dans la section [realms]. Les valeurs spécifique au domaine écrase, n'ajoutent pas à une spécification libdefaults par défaut. L'ordre de recherche est :

1. sous-section spécifique au domaine de [libdefaults]

```
[libdefaults]
 EXAMPLE.COM = {
  pkinit_anchors = FILE:/usr/local/example.com.crt
 }
```

2. Valeurs spécifique au domaine dans [realms]

```
[realms]
 OTHERREALM.ORG = {
  pkinit_anchors = FILE:/usr/local/otherrealm.org.crt
 }
```

3. Valeur générique dans [libdefaults]

```
[libdefaults]
 pkinit_anchors = DIR:/usr/local/generic_trusted_cas/
```

Spécifier les informations d'identité PKINIT

FILE :filename [,keyfilename] Cette option est dépendante du contexte.

Dans **pkinit_identity** ou **pkinit_identities**, filename spécifie le nom d'un fichier PEM contenat le certificat de l'utilisateur. Si Keyfilename n'est pas spécifié, la clé privée de l'utilisateur est attendue dans filename.

Dans **pkinit_anchors** ou **pkinit_pool**, filename est assumé être le nom d'un fichier ca-bundle style OpenSSL.

DIR :dirname Cette option est dépendante du contexte.

Dans **pkinit_identity** ou **pkinit_identities**, dirname spécifie un répertoire avec des fichiers .crt et .key.

Dans **pkinit_anchors** ou **pkinit_pool**, dirname est assumé être un répertoire CA hashé style OpenSSL où chaque certificat CA est stocké dans un fichier nommé **hash-of-ca-cert.#**

PKCS12 :filename filename est le nom d'un pkcs#12, contenant le certificat de l'utilisateur en la clé privée.

PKCS11 : [module_name=] modname [:slotid=slot-id] [:token=token-label] [:certid=cert-id] [:certlabel=cert-label] Tous les mots clé sont optionnels. modname spécifie l'emplacement de la librairie PKCS#11. Si une valeur est rencontrée sans mot clé, il est assumé être le modname. Si aucun nom de module n'est spécifié, le défaut est **opensc-pkcs11.so**. **slotid=** et/ou **token=** peuvent être spécifiés pour forcer l'utilisation d'un lecteur de carte ou un token spécifique. **certid=** et/ou **certlabel** peuvent être utilisés pour forcer la sélection d'un certificat particulier sur un périphérique.

ENV :envvar envvar spécifie le nom d'une variable d'environnement qui a été définie à une valeur se conformant à une des valeur précédente. Par exemple, **ENV :X509_PROXY** où la variable d'environnement **X509_PROXY** a été définie à **FILE :/tmp/my_proxy.pem**

options krb5.conf pour PKINIT

pkinit_anchors Spécifie l'emplacement des certificats de confiance racine que le client utilise pour signer les certificats KDC. Peut être spécifié plusieurs fois.

pkinit_cert_match Spécifie les règles de correspondance que le certificat client doit matcher avant d'être utilisé pour l'authentification PKINIT. Peut être spécifié plusieurs fois. Tous les certificats sont vérifiés avec chaque règle jusqu'à ce qu'un certificat matche. La comparaison de chaîne Issuer et Subject sont des représentations chaîne rfc2253 des valeurs Issuer DN et Subject DN du certificat. La syntaxe de la règle est **[relation-operator]component-rule ...** où :

relations-operator peut être soit **&&** signifiant toutes les composantes des règles doivent matcher, ou || signifiant que seul un composant de règle doit matcher. Défaut : &&

component-rule Peut être un des suivant. Noter qu'il n'y a pas de ponctuation ou d'espace blanc entre les composantes de règle :

<SUBJECT> regular-expression

<ISSUER> regular-expression

<SAN> regular-expression

<EKU> extended-key-usage-list

<KU> key-usage-list

extended-key-usage-list est une liste de valeurs Extended Key Usage. Toutes les valeurs dans la liste doivent être présents dans le certificat. Les valeurs sont :

pkinit

msScLogin

clientAuth

emailProtection

key-usage-list est une liste de valeurs requises Key Usages. Toutes les valeurs dans la liste doivent être présentes dans le certificat. Les valeurs sont :

digitalSignature

keyEncipherment

exemple :

```
pkinit_cert_match = ||<SUBJECT>.*DoE.*<SAN>.*@EXAMPLE.COM
pkinit_cert_match = &&<EKU>msScLogin,clientAuth<ISSUER>.*DoE.*
pkinit_cert_match = <EKU>msScLogin,clientAuth<KU>digitalSignature
```

pkinit_eku_checking spécifie quelle valeur d'Extended Key Usage le certificat KDC présenté au client doit contenir. (Noter que si le certification du KDC a le pkinit SubjectAlternativeName encodé comme nom Kerberos TGS, la vérification EKU n'est pas nécessaire vu que la CA fournie a certifié cela comme certificat KDC). Les valeurs reconnues sont :

kpKDC C'est la valeur par défaut et spécifie que le KDC doit avoir le id-pkinit-KDKdc EKU comme définis dans la rfc4556.

kpServerAuth Un certificat KDC avec le id-kp-serverAuth EKU tel qu'utilisé par Microsoft sera accepté

none Le certificat du KDC ne sera pas vérifié pour savoir s'il a un EKU acceptable.

pkinit_dh_min_bits Taille de la clé Diffie-Hellman que le client va tenter d'utiliser. les valeur acceptable sont 1024, 2048 (défaut), et 4096.

pkinit_identities Spécifie les emplacements à utiliser pour trouver les information d'identité X.509 du client. Peut être spécifié plusieurs fois. Ces valeur ne sont pas utilisées si l'utilisateur spécifie **X509_user_identity** sur la ligne de commande.

pkinit_kdc_hostname La présence de cette option indique que le client est prêt à accepter un certificat KDC avec un dNSName SAN au lieu de nécessiter de id-pkinit-san comme définis dans la rfc4556. peut être spécifié plusieurs fois.

pkinit_longhorn À true, on parle au KDC Longhorn

pkinit_pool Spécifie l'emplacement des certificats intermédiaires qui peuvent être utilisés par le client pour compléter la validation de la chaîne. Peut être spécifié plusieurs fois.

pkinit_require_crl_checking Vérifie les informations de révocation.

pkinit_revoke Spécifie l'emplacement de la CRL à utiliser pour vérifier les informations de révocations.

pkinit_win2k Spécifie si le domaine cible est supporte l'ancienne version, pré-rfc, de Kerberos

pkinit_win2k_require_binding À true, le KDC cible attendu est patché pour retourner une réponse avec un checksum. Défaut : false

Expansion de paramètres

De nombreuses variables, telles que **default_keytab_name**, permettent d'étendre les paramètres :

%{TEMP} Répertoire temporaire

%{uid} User ID réel ou Windows SID

%{euid} User ID effectif ou Windows SID

%{USERID} Idem à %{uid}

%{null} Chaîne vide

%{LIBDIR} Répertoire d'installation des librairies

%{BINDIR} Répertoire d'installation des binaires

%{SBINDIR} Répertoire d'installation des binaires administratifs

%{username} nom de l'utilisateur unix ou user ID effectif

%{APPDATA} (Windows) Roaming application data for current user

%{COMMON_APPDATA} (Windows) Application data for all users

%{LOCAL_APPDATA} (Windows) Local application data for current user

%{SYSTEM} (Windows) Windows system folder

%{WINDOWS} (Windows) Windows folder

%{USERCONFIG} (Windows) Per-user MIT krb5 config file directory

%{COMMONCONFIG} (Windows) Common MIT krb5 config file directory

Exemple de fichier krb5.conf

```
[libdefaults]
 default_realm = ATHENA.MIT.EDU
 default_tkt_enctypes = des3-hmac-sha1 des-cbc-crc
 default_tgs_enctypes = des3-hmac-sha1 des-cbc-crc
 dns_lookup_kdc = true
 dns_lookup_realm = false

[realms]
 ATHENA.MIT.EDU = {
```

```
  kdc = kerberos.mit.edu
  kdc = kerberos-1.mit.edu
  kdc = kerberos-2.mit.edu:750
  admin_server = kerberos.mit.edu
  master_kdc = kerberos.mit.edu
  default_domain = mit.edu
 }
 EXAMPLE.COM = {
  kdc = kerberos.example.com
  kdc = kerberos-1.example.com
  admin_server = kerberos.example.com
 }

[domain_realm]
 .mit.edu = ATHENA.MIT.EDU
 mit.edu = ATHENA.MIT.EDU

[capaths]
 ATHENA.MIT.EDU = {
   EXAMPLE.COM = .
 }
 EXAMPLE.COM = {
   ATHENA.MIT.EDU = .
 }
```