Sécurité SI

Authentification Centralisée
LDAP / Radius

Vendredi 21 Avril 2006

Germain BAUVIN (bauvin_g)
Matthieu MICHAUD (michau_m)
Pierre-Yves ROFES-VERNIS (rofes-_p)

Table de Matières

Introduction

RADIUS est un protocole qui intègre les notions d'AAA (*Authorization, Authentication and Accounting*). Son but original était de permettre au FAI d'authentifier leurs clients de manière centralisée. *Lightweight Directory Access Protocol* (LDAP), quant à lui, est un protocole d'accès à des annuaires. Initialement conçu comme un frontal pour accéder à des bases de données externes, il est désormais devenu un annuaire standalone avec sa propre base de données.

Combinés, ils permettent de mettre en place une architecture d'authentification centralisée efficace pour de nombreux services : du ftp au ppp en passant par une simple authentification sur les postes de travail.

Ce dossier a pour but de peser le pour et le contre de l'utilisation de LDAP en association avec Radius par rapport à l'utilisation de la base d'utilisateur Unix.

Maquette Radius

Cette partie du document nous permettra d'expliquer les démarches nécessaires à la mise en place du serveur *FreeRadius* comme module d'authentification pour les postes de travail désignés. Ayant déjà rédigé un rapport sur la mise en place de ce type de serveur, certains des points de la partie 'Mise en place technique' en sont directement tirés.

Mise en place technique

Installation et configuration du serveur FreeRadius

La configuration de FreeRadius, une implémentation libre du protocole RADIUS, a été assez complexe. Ce service est par nature très modulaire et découpé en beaucoup de fichiers de configuration. Heureusement, des exemples de chaque fichier sont fournis dans la distribution officielle et ils sont largement commentés. Le processus de configuration s'est en grande partie résumé à copier les fichiers un par un, à comprendre leur utilité et à garder les parties utiles.

Le fichier suivant classifie les clients selon leur adresse d'émission pour définir un contrôle d'accès par groupe protégé par un mot de passe.

clients.conf :

```
client 10.226.3.0/24 {
        secret =        chiche
        shortname =     barry
        nastype =       other
}
```

radiusd.conf :

```
# Affectation de variable, aide a la configuration
prefix = /usr/local
exec_prefix = ${prefix}
sysconfdir = ${prefix}/etc
localstatedir = /var
sbindir = ${exec_prefix}/sbin
logdir = /var/log
raddbdir = ${sysconfdir}/raddb
radacctdir = ${logdir}/radacct
confdir = ${raddbdir}
run_dir = ${localstatedir}/run/radiusd
log_file = ${logdir}/radius.log
libdir = ${exec_prefix}/lib
pidfile = ${run_dir}/radiusd.pid

# Le nom (ou le #id) du proprietaire du processus radiusd
```

```
user = _radius
group = _radius

# limitations de ressources
max_request_time = 30
cleanup_delay = 5
max_requests = 1024

thread pool {
        start_servers = 5
        max_servers = 32
        min_spare_servers = 3
        max_spare_servers = 10
        max_requests_per_server = 0
}

bind_address = 10.226.3.10       # ou nom, ici goudale.srs.lab.epita.fr
port = 0                    # automatique

hostname_lookups = no
allow_core_dumps = no

# Quelles informations doit-on écrire dans le journal ?
log_stripped_names = no
log_auth = yes
log_auth_badpass = no
log_auth_goodpass = no

security {
        max_attributes = 200
        reject_delay = 1
        status_server = no
}
proxy_requests  = no       # Agir comme un proxy ?
snmp    = no               # Publier l'etat pour collecte SNMP ?

# Inclusion des options de contrôle d'acces
$INCLUDE  ${confdir}/clients.conf

# Configuration des modules
# (des qu'un module est declare, son nom peut être reutiliser)
modules {
 unix { }

  preprocess {
     huntgroups = ${confdir}/huntgroups
     hints = ${confdir}/hints
     with_ascend_hack = no
     ascend_channels_per_line = 23
     with_ntdomain_hack = no
     with_specialix_jetstream_hack = no
```

```
    with_cisco_vsa_hack = no
 }

    # Affecte des fichiers a des definitions de methodes
 files {
     usersfile = ${confdir}/users                # identification
     preproxy_users = ${confdir}/preproxy_users  # idem proxy
     acctusersfile = ${confdir}/acct_users       # session
    compat = no
 }

    # Champs RADIUS pour detecter qu'une session est unique
 acct_unique {
     key = "User-Name,NAS-IP-Address,Client-IP-Address,NAS-Port"
 }

 # Comme utmp d'UNIX pour une session RADIUS
 radutmp {
    filename = ${logdir}/radutmp
    username = %{User-Name}
    case_sensitive = yes
    check_with_nas = yes
    perm = 0600
    callerid = "yes"
 }

  expr { }
  digest { }

  exec {
     wait = no
    input_pairs = request
 }
 exec echo {
     wait = yes
     program = "/bin/echo %{User-Name}"
     input_pairs = request
     output_pairs = reply
 }

 ippool main_pool {
     range-start = 192.168.1.1
     range-stop =  192.168.1.253
     netmask = 255.255.255.0
     session-db = ${raddbdir}/db.ippool
     ip-index = ${raddbdir}/db.ipindex
 }
}

# Contenu des phases de dialogue
instantiate {
```

```
        exec
        expr
}

authorize {
        preprocess
        auth_log
        files
}

authenticate {
        unix
}

preacct {
        preprocess
        acct_unique
}

accounting {
        detail
        radutmp
        main_pool
}

session {
        radutmp
}

post-auth {
        main_pool
        reply_log
}
```

Le fichier dictionnaire met en correspondance des noms d'attributs connus de RADIUS avec ceux donnés dans le dialogue avec ses clients qui diffèrent parfois selon les fabricants. Il n'est pas nécessaire de donner ce fichier ici, il est fourni dans la distribution officielle et doit simplement être place dans le répertoire de configuration.

L'échange avec le NAS comporte les étapes décrites dans l'ordre donné par le fichier de configuration. Nous avons essayé deux méthodes d'authentification « Unix » et « PAM ». Nous ne donnons pas la configuration avec le module PAM. Il suffirait d'ajouter la définition dans « modules » et de remplacer « Unix » par « PAM» dans l'étape « Authentication ». Ce module n'est pas recommandé par les développeurs de FreeRADIUS car les librairies sous-jacentes comporteraient des fuites de mémoire.

Critiques et remarques

Si cette solution est fonctionnelle, il reste cependant de nombreux points plus ou moins critiques qui jouent en sa défaveur, en voici quelques-uns.

Sécurisation des transactions

Le niveau de sécurisation des transactions laisse grandement à désirer. En effet, beaucoup d'informations filtrent au travers du trafic réseau sur les mots de passe et logins par exemple. Ces fuites mettent en danger toute l'architecture, puisque l'authentification est centralisée. Il suffit donc a un attaquant d'écouter simplement un peu sur le réseau pour récupérer l'accès à toutes les machines.

Montée en charge et nombre d'utilisateurs

L'utilisation d'un fichier texte (ici /etc/passwd) pour stocker les informations nécessaire au login d'un grand nombre d'utilisateurs est un très mauvais choix. En effet, si l'utilisateur lambda souhaite se logger et se trouve en toute fin de fichier, il risque d'attendre longtemps. Cela vaut également pour les autres utilisateurs qui désireraient se connecter au même moment.

Ajout de services

L'ajout de services implique, bien entendu, l'ajout de nouveaux utilisateurs. Et le problème de l'utilisation de la base d'utilisateurs Unix est qu'il n'est pas possible, ou alors d'une façon non intuitive, de filtrer quels utilisateurs ont accès à quels services. De plus, cela implique de gonfler encore le fichier de mots de passe en ajoutant de nouveaux utilisateurs systèmes et donc ralentir l'ensemble de la solution.

Le cas des postes banalisés

Un poste banalisé implique une journalisation beaucoup moins efficace des événements. Cela implique également un accès beaucoup moins bien filtré aux différents services utilisant le serveur Radius.

Conclusion

Le problème redondant est bien sûr la gestion des accès aux différents services ainsi que la montée en charge due à la gestion d'un grand nombre d'utilisateurs. Pour régler ce problème, l'utilisation d'un service d'annuaire semble une bonne idée. LDAP est une excellente réponse à ce besoin.

Maquette LDAP

La mise en place d'une authentification basée sur un annuaire LDAP nécessite la configuration d'un serveur de référence et des clients voulant se servir de cette référence. OpenLDAP, l'implémentation la plus répandue de LDAP, offre la possibilité de répliquer l'annuaire dit "maître" pour répartir la charge de travail. Notre maquette ne comporte pas de serveurs esclaves. Nous nous sommes restreints à la mise en place d'un unique maître et de plusieurs clients.

Comme cette architecture correspond aux besoins des utilisateurs du laboratoire, nous avons profité de l'exercice pour la mettre en place et a disposition de nos camarades. Les machines utilisées sont donc des machines du laboratoire.

Le serveur LDAP que nous avons utilisé est hébergé sur une machine faisant office de serveur de bases de données pour les autres services du laboratoire. Elle tourne sous FreeBSD 6. Ceci est contraire à l'énoncé de l'exercice, mais nous avons jugé que cette différence n'était pas gênante. Ce système est un système UNIX et, du point de vue de LDAP, seule son installation diffère comme elle diffèrerait d'une distribution Linux à une autre selon les outils de déploiement applicatifs.

Configuration du serveur

Nous avons configuré une autorité de certification et un serveur d'annuaire OpenLDAP sur une machine appelé "delerium". C'est une machine fiable et puissante, il s'agit d'un bi-xeon 2.8GHz. Les autres services n'étant pas sollicités de manière importante, elle suffira très largement dans notre cas de figure (20 utilisateurs, 6 stations de travail, réseau fast Ethernet commuté).

Autorité de certification avec OpenSSL

La communication entre les clients et le serveur LDAP est sécurisée par TLS. Nous avons mis en place une autorité de certification basique. Nous avons utilisé OpenSSL et un script très utile fourni dans la distribution officielle. Il s'agit du fichier "apps/CA.pl" qui n'est autre chose qu'une surcouche pour la commande OpenSSL dont la syntaxe est difficile.

En pratique, pour créer une autorité de certification basique nous avons procédé comme suit.

Création du répertoire de travail et copie du script :

```
$ mkdir /etc/ssl
$ cp /path/to/openssl/apps/CA.pl /etc/ssl
```

Ensuite il faut configurer OpenSSL.
/etc/ssl/openssl.cnf :

```
[ ca ]
default_ca      = delerium
```

```
[ delerium ]
dir             = /etc/ssl              # Where everything is kept
certs           = $dir/certs            # Where the issued certs are kept
crl_dir         = $dir/crl              # Where the issued crl are kept
database        = $dir/index.txt        # Database index file
unique_subject  = no
new_certs_dir   = $dir/newcerts         # Default place for new certs
certificate     = $dir/ca.crt           # The CA certificate
serial          = $dir/serial           # The current serial number
crl             = $dir/crl/crl.pem      # The current CRL
private_key     = $dir/private/ca.key   # The private key
RANDFILE        = $dir/private/.rand    # Private random number file
x509_extensions = usr_cert              # Extensions to add to the cert
name_opt        = ca_default            # Subject Name options
cert_opt        = ca_default            # Certificate field options
default_days    = 365                   # How long to certify for
default_crl_days= 30                    # How long before next CRL
default_bits    = 2048
default_md      = md5                   # Which md to use.
preserve        = no                    # Keep passed DN ordering
policy          = policy_match

[ policy_match ]
countryName             = match
stateOrProvinceName     = match
organizationName        = match
organizationalUnitName  = optional
commonName              = supplied
emailAddress            = optional

[ policy_anything ]
countryName             = optional
stateOrProvinceName     = optional
localityName            = optional
organizationName        = optional
organizationalUnitName  = optional
commonName              = supplied
emailAddress            = optional

[ req ]
default_bits            = 1024
default_keyfile         = privkey.pem
distinguished_name      = req_distinguished_name
attributes              = req_attributes
x509_extensions         = v3_ca
string_mask             = nombstr

[ req_distinguished_name ]
countryName                     = Country Name (2 letter code)
countryName_default             = FR
countryName_min                 = 2
```

```
countryName_max                 = 2
stateOrProvinceName             = State or Province Name (full name)
stateOrProvinceName_default     = France
localityName                    = Locality Name (eg, city)
localityName_default            = Kremlin-Bicetre
organizationName                = Organization Name (eg, company)
organizationName_default        = EPITA
organizationalUnitName          = Organizational Unit Name (eg, section)
organizationalUnitName_default  = Laboratoire SRS
commonName                      = Common Name (eg, YOUR name)
commonName_max                  = 64
emailAddress                    = Email Address
emailAddress_max                = 64

[ req_attributes ]
challengePassword               = A challenge password
challengePassword_min           = 8
challengePassword_max           = 24

[ usr_cert ]
basicConstraints                = CA:FALSE
nsComment                       = "OpenSSL Generated Certificate"
subjectKeyIdentifier            = hash
authorityKeyIdentifier          = keyid,issuer:always

[ v3_req ]
basicConstraints                = CA:FALSE
keyUsage                        = nonRepudiation, digitalSignature,
keyEncipherment

[ v3_ca ]
subjectKeyIdentifier            = hash
authorityKeyIdentifier          = keyid:always,issuer:always
basicConstraints                = CA:true

[ crl_ext ]
authorityKeyIdentifier          = keyid:always,issuer:always
```

Pour être en accord avec cette configuration d'OpenSSL, il faut redéfinir certaines variables au début du script CA.pl :

```
$CATOP="/etc/ssl";
$CAKEY="ca.key";
$CACERT="ca.crt";
```

Ensuite, il faut créer l'arborescence de travail d'OpenSSL et le certificat d'autorité :

```
$ /etc/ssl/CA.pl -newca
```


Nous pouvons préciser quel certificat d'autorité utiliser si celui défini dans la configuration n'existe pas. Ceci est utile si l'on dispose déjà d'un certificat à partir duquel nous allons signer.
Comme pour tout certificat, il faut renseigner les champs publics. Plusieurs questions seront posées sur l'identité du certificat. Remplissez les, et, comme proposé et plus que conseillé, mettez un mot de passe sur cette clef car elle est importante.

Puis, générons un certificat pour notre serveur LDAP :

```
$ /etc/ssl/CA.pl -newreq-nodes
```

Avant de renseigner les champs, vous pouvez protéger le nouveau certificat par un mot de passe. Il n'est pas crucial d'en mettre un ici, la clef prive sera protégée par le contrôle d'accès au système de fichier. Comme pour le certificat d'autorité, remplissez les champs. Si tout s'est bien déroulé, la demande de signature et la clef privé sont dans le fichier */etc/ssl/newreq.pem*. Il ne reste plus qu'a signer le certificat :

```
$ /etc/ssl/CA.pl -sign
```

Il en résulte deux fichiers : */etc/ssl/newreq.pem* et */etc/ssl/newcert.pem*, la clef prive et le certificat.

Service d'annuaire avec OpenLDAP

Comme ce n'est pas le sujet, nous ne nous attarderons pas sur la procédure d'installation d'OpenLDAP sur FreeBSD. La documentation de l'outil de déploiement d'application, c'est a dire "ports", existe et ne saurait être plus complète. Elle se trouve au chapitre 4 du manuel de l'utilisateur[1] et sous forme de page de manuel traditionnel[2].

OpenLDAP a été installé dans le préfixe habituel, */usr/local*. Les fichiers de configurations se trouvent donc dans */usr/local/etc/openldap*.

Déplaçons notre clef prive et notre certificat dans ce répertoire :

```
$ mv /etc/ssl/newreq.pem /usr/local/etc/openldap/server.key
$ mv /etc/ssl/newcert.pem /usr/local/etc/openldap/server.crt
```

Modifions le fichier de configuration du serveur, slapd.conf :

```
# schema indispensable
include         /usr/local/etc/openldap/schema/core.schema

# schema supplementaire adapte aux comptes unix
include        /usr/local/etc/openldap/schema/cosine.schema
include        /usr/local/etc/openldap/schema/inetorgperson.schema
include        /usr/local/etc/openldap/schema/nis.schema
```

[1] http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/ports.html
[2] http://www.freebsd.org/cgi/man.cgi?query=ports&format=html

```
pidfile         /var/run/openldap/slapd.pid
argsfile        /var/run/openldap/slapd.args
loglevel        0 # rien

# imposons un chiffrement de 112 bit minimum pour l'ecriture
# et de 64 bit pour la lecture.
security ssf=1 update_ssf=112 simple_bind=64

TLSCertificateFile      /usr/local/etc/openldap/server.crt
TLSCertificateKeyFile   /usr/local/etc/openldap/server.key

# certificat d'authorite creer a l'etape precendente
TLSCACertificateFile    /etc/ssl/ca.crt

####################
# controle d'acces #
####################

access to dn.base=""
    by * read

access to dn.subtree="dc=srs,dc=lab,dc=epita,dc=fr"
    attr=userPassword
    by dn="cn=manager,dc=srs,dc=lab,dc=epita,dc=fr" write
    by self write
    by anonymous auth
    by * none

access to dn.subtree="dc=srs,dc=lab,dc=epita,dc=fr"
    by dn="cn=manager,dc=srs,dc=lab,dc=epita,dc=fr" write
    by * read

access to *
    by dn.base="cn=manager,dc=srs,dc=lab,dc=epita,dc=fr" write
    by self write
    by * auth

#######################################
# configuration du module de stockage #
#######################################

# moteur de base de donnees
database        bdb
directory       /var/db/openldap-data

# pair identifiant/secret du super utilisateur
rootdn          "cn=manager,dc=srs,dc=lab,dc=epita,dc=fr"
rootpw          {SSHA}... # obtenu par slappasswd -h {SSHA}

suffix          "dc=srs,dc=lab,dc=epita,dc=fr"
```

```
# index de la base de donnees
index           objectClass             eq
index           cn,sn,uid,displayName   pres,sub,eq
index           uidNumber,gidNumber     eq
```

L'installation d'OpenLDAP via les "ports" crée un utilisateur et un groupe *ldap* propriétaires des données critiques et du processus. Comme le fichier de configuration, slapd.conf, contient le mot de passe chiffré du super utilisateur, il ne doit être lisible que par l'utilisateur *ldap*. De même, la clef prive du serveur est sensible. Par contre le certificat, peut et doit être lisible par tous. Corrigeons les droits sur ces fichiers :

```
$ cd /usr/local/etc/openldap
$ chmod 755 .
$ chown ldap:ldap server.key slapd.conf
$ chmod 600 server.key slapd.conf
$ chmod 644 server.crt
```

Il ne reste plus qu'à lancer le service. Il devrait écouter sur les ports TCP 389 (LDAPv3) et 636 (LDAPv3 over SSL). Si cela ne fonctionne pas correctement, la commande *slaptest* permet de vérifier le fichier de configuration. Attention, si vous n'avez pas les droits de lecture sur le fichier de configuration du serveur, il considérera qu'il est mauvais et non inaccessible ! Si cela ne vous donne pas suffisamment d'information, la journalisation peut être active. C'est le paramètre *loglevel* de *slapd.conf*, -1 active le niveau maximum de verbosité.

OpenLDAP envoie ses messages vers le service de journalisation standard, *syslog*. Pour qu'il les accepte ajoutez les lignes suivantes dans son fichier de configuration :

```
!slapd
*.*                             /var/log/ldap.log
```

N'oubliez pas de créer le fichier destination et de redémarrer *syslog*.

Configuration des clients

D'un point de vue globale, la configuration des clients se résume a spécifier l'utilisation de la source de donnée de référence, c'est a dire notre annuaire. Cette source de donnée peut être utilise par les differents services que nous voulons base sur cette référence. L'authentification d'un utilisateur sur un système en est un exemple. Il existe beaucoup d'autres services qui ont les modules logicielles développe pour servir de LDAP. En effet, le serveur HTTP Apache, les serveurs de messagerie comme Qmail et Postfix sont des exemples classiques.

Voyons, en pratique, comment nous avons configure nos postes de travail pour creer des comptes UNIX partage via un annuaire LDAP.

En premier lieu, il faut installer et configurer un client LDAP sur chaque poste. Encore une fois, comme ce n'est pas le sujet de ce rapport, je vous renvoie a la documentation de votre systeme favoris ou a celle d'OpenLDAP. Quel qu'il soit, le client se configure grace au fichier <prefix>/openldap/ldap.conf.

La communication entre les clients et le serveur LDAP necessite TLS. Recuperons les certificats d'autorite :

```
$ mkdir /etc/ssl
$ chmod 755 /etc/ssl
$ scp delerium:/etc/ssl/ca.crt /etc/ssl/ca.crt
```

Ecrivons le fichier de configuration du client LDAP :

```
#
ssl start_tls
ssl on
tls_cacert      /etc/ssl/ca.crt

# partie de l'arbre
suffix          dc=srs,dc=lab,dc=epita,dc=fr

uri             ldaps://delerium.srs.lab.epita.fr:636/
ldap_version    3
```

A ce stade, le client est capable de se connecter au serveur et de l'interroger. Les commande ldapwhoami, ldapadd doivent donc fonctionner.

```
$ ldapwhoami
anonymous
$ ldapwhoami -D "cn=manager,dc=srs,dc=lab,dc=epita,dc=fr" -W
Enter LDAP Password: *******
dn:cn=manager,dc=srs,dc=lab,dc=epita,dc=fr
```

Initialisons notre annuaire avec nos donnees. Il faut utiliser la commande slapadd avec un fichier au format LDIF (le format d'echange unifie utilise avec OpenLDAP) :

```
$ cat base.ldif
dn: dc=srs,dc=lab,dc=epita,dc=fr
dc: srs
objectClass: top
objectClass: domain
objectClass: domainRelatedObject
associatedDomain: srs.lab.epita.fr
structuralObjectClass: domain

dn: ou=Groups,dc=srs,dc=lab,dc=epita,dc=fr
ou: Groups
objectClass: top
objectClass: organizationalUnit
structuralObjectClass: organizationalUnit

dn: ou=Users,dc=srs,dc=lab,dc=epita,dc=fr
ou: Users
objectClass: top
```

```
objectClass: organizationalUnit
structuralObjectClass: organizationalUnit

dn: cn=srs2007,ou=Groups,dc=srs,dc=lab,dc=epita,dc=fr
objectclass: top
objectclass: posixGroup
cn: srs2007
gidNumber: 20070
memberUid:
$ slapadd -l base.ldif
```

Et, ajoutons un utilisateur :

```
$ cat michau_m
dn: cn=Matthieu Michaud,ou=Users,dc=srs,dc=lab,dc=epita,dc=fr
cn: Matthieu Michaud
objectClass: top
objectClass: person
objectClass: posixAccount
objectClass: shadowAccount
uid: michau_m
userPassword: michau_m
uidNumber: 17254
gidNumber: 20070
gecos: Matthieu Michaud
loginShell: /bin/bash
homeDirectory: /nfs/home/michau_m
$ slapadd -l michau_m
```

Ensuite, demandons a notre systeme d'authentifier les utilisateurs via l'annuaire. Ici, c'est PAM (Pluggable Authentification Module) qui intervient. Le module PAM-LDAP n'est pas inclus dans le systeme de base de FreeBSD. Une installation a encore ete necessaire via les ports. Ce module se base sur le bibliotheque client LDAP, ici celle d'OpenLDAP. Je ne sais pour quelle raison, ce module essaie de lire le fichier de configuration dans <prefix>/etc aulieu de <prefix>/etc/openldap. Il a donc ete utile de creer un lien symbolique :

```
$ ln -s <prefix>/etc/openldap/ldap.conf <prefix>/etc/
```

Ensuite, il suffit de rajouter les directives de configuration suivantes :

```
pam_filter              objectClass=posixAccount
pam_login_attribute     uid
pam_member_attribute    memberUid
pam_password            exop
```

Elle ne perturbe pas le fonctionnement du reste des clients LDAP.

Puis, pour que les applications utilise l'annuaire pour les authentifications, si elles utilisent PAM, il suffit de configurer le service associe (au sens de PAM). OpenPAM, une implementation de PAM, est

integre au systeme de base de FreeBSD et ne necessite aucune installation supplementaire. Elle utilise les fichiers de configuration contenu dans /etc/pam.d.

Modifions la configuration du service PAM system, /etc/pam.d/system :

```
#
# System-wide defaults
#

# auth
auth sufficient  <prefix>/lib/pam_ldap.so   no_warn try_first_pass
auth required    pam_unix.so                no_warn try_first_pass nullok

# account
account         sufficient      <prefix>/lib/pam_ldap.so
account         required        pam_login_access.so
account         required        pam_unix.so

# session
session         sufficient      <prefix>/lib/pam_ldap.so   no_fail
session         required        pam_lastlog.so             no_fail

# password
password sufficient   <prefix>/lib/pam_ldap.so   no_warn try_first_pass
password required     pam_unix.so                no_warn try_first_pass
```

A ce stade, un utilisateur declare dans l'annuaire peut s'authentifier sur un terminal et la commande su doit fonctionner. En modifiant la configuration service PAM sshd, le fichier /etc/pam.d/sshd, de la meme maniere, l'utilisateur peut egalement acceder au poste par SSH. Il faut verifier que le serveur SSH utilise bien PAM. Pour OpenSSH, il faut verifier la presence des deux directives suivantes dans son fichier de configuration, sshd_config :

```
UsePAM yes
ChallengeResponseAuthentication yes
```

Il reste encore un point a faire fonctionner. La correspondance entre identifiants numeriques et leurs noms n'est plus etablie. Des commandes comme ls -l ou id y echoue. Les systemes UNIX relativement recents integre une interface d'abstraction pour ces resolutions. Il s'agit de Name Service Switching, ou nsswitch, de la librairie C standard. Une fois de plus, le modules NSS-LDAP n'est pas present dans le systeme de base de FreeBSD, a nouveau une installation s'est impose.

Pour finir, ce module, comme PAM-LDAP, est un client LDAP base sur la configuration globale des clients LDAP. A nouveau, il suffit de creer un lien symbolique entre le fichier de configuration du module et celui du client LDAP :

```
$ ln -s <prefix>/etc/openldap/ldap.conf <prefix>/etc/nss_ldap.conf
```

et d'ajouter les directives de configuration suivantes :

```
scope sub

nss_base_passwd         ou=Users,dc=srs,dc=lab,dc=epita,dc=fr
nss_base_group          ou=Groups,dc=srs,dc=lab,dc=epita,dc=fr
```

Cette ajout n'influe toujours pas sur le reste de la configuration des clients LDAP.

Voici le resultat final :

```
delerium$ ssh michau_m@st-louis
Password:
Last login: Thu Apr 20 18:37:22 2006
Copyright (c) 1980, 1983, 1986, 1988, 1990, 1991, 1993, 1994
        The Regents of the University of California.  All rights reserved.

FreeBSD 6.0-RELEASE-p6 (HP) #1: Fri Apr  7 18:56:37 CEST 2006

Welcome to FreeBSD!

[michau_m@st-louis:~]
$ id
uid=17254(michau_m) gid=20070(srs2007) groups=20070(srs2007)
```

Cette solution, comme les autres solutions d'authentification, a les défauts de ses qualités. En effet, les comptes sont centralisés, donc un compte compromis l'est sur l' ensemble des machines soumis a l'autorité centrale. Il faut donc prévoir des contre-mesures et limiter la confiance accordée aux postes clients. De plus, les stations de travail sont logiquement contrôlées par la même autorité que le ou les serveurs centraux d'authentification. Sinon, la chaîne de confiance est rompue et la sécurité compromise.

Conclusion

Si la mise en place de cette architecture utilisant LDAP peut paraître quelque peu difficile, le jeu en vaut la chandelle. L'authentification centralisée simplifie considérablement le travail de l'administrateur système, même si elle soulève des problèmes de sécurité évidents.

Ce dossier nous aura permis de découvrir les fondations de la centralisation des informations de login sur un réseau et permis la mise en place d'une telle solution au sein même du laboratoire de la spécialisation.