pam_pkcs11.conf

fichier de configuration pour pam_pkcs11

Le fichier de configuration utilise la librairie scconf. Les paramètres et données sont groupées dans des blocks. ils peuvent être imbriqués.

Un fichier de configuration pam_pkcs11 ressemble à :

```
pam-pkcs11
 global options
 [...]
 use_pkcs11_module = pkcs11 module to be used
 pkcs11_module module1 {
  module1 specific options
 }
 pkcs11_module module2 {
  module2 specific options
 }
 [...]
 use_mappers = mapper1, mapper2,... ;
 mapper mapper1 {
  mapper1 specific options
 }
 mapper mapper2 {
  mapper2 specific options
 }
 [...]
 mapper mapperN {
  mapperN specific options
 }
}
```

Exemple de fichier pam_pkcs11.conf

```
pam_pkcs11 {
 nullok = true ; # Allow empty passwords
debug = true ; # Enable debugging support.
 use_first_pass = false ; # Do not prompt for the passwords but take them from the PAM_ items instead
 try_first_pass = false ; # Do not prompt for the passwords unless PAM_(OLD)AUTHTOK is unset.
 use_authtok = false ; # Like try_first_pass, but fail if the new PAM_AUTHTOK has not been previously set
(intended for stacking password modules only).
 use_pkcs11_module = opensc ; # Filename of the PKCS #11 module. The default value is "default"

pkcs11_module opensc {
 module = /usr/lib/opensc-pkcs11.so ;
 description = "OpenSC PKCS#11 module" ;
 slot_description = "none" ; # spécifier le slot à utiliser. préférer cette option à slot_num.
 # slot_num = 0 ; # spécifier soit slot_description, soit slot_num.
 ca_dir = /etc/pam_pkcs11/cacerts ; # répertoire contenant les certificats de l'autorité au format pem ou
asn1 oui des liens hash.
 crl_dir = /etc/pam_pkcs11/crls ; # répertoire contenant la CRL offline
 support_threads = false ; # certaines librairies supportent le multi-thread
 cert_policy = ca,signature ; # définis la vérification de certificat
 # none aucune vérification
 # ca vérfication de la CA
```

```
 # crl_online télécharge la CRL
 # crl_offline utilise la crl offline
 # crl_auto tente d'abord la online, puis la offline
 # signature vérifie la signature
 token_type = "Smart card" ; # spécifie le type de token. sera utilisé dans le message de prompt.
 pkcs11_module etoken { # Aladdin eTokenPRO 32
  module = /usr/local/lib/libetpkcs11.so
  description = "Aladdin eTokenPRO-32" ;

  slot_num = 0 ;
  support_threads = true ;
  ca_dir = /etc/pam_pkcs11/cacerts ;
  crl_dir = /etc/pam_pkcs11/crls ;
  cert_policy = ca,signature ;
 }
 pkcs11_module nss { # NSS (Network Security Service) config
  nss_dir = /etc/ssl/nssdb ;
  crl_policy = none ;
 }
 pkcs11_module default { # Default pkcs11 module
  module = @libdir@/pam_pkcs11/pkcs11_module.so ;
  description = "Default pkcs#11 module" ;
  slot_num = 0 ;
  support_threads = false ;
  ca_dir = /etc/pam_pkcs11/cacerts ;
  crl_dir = /etc/pam_pkcs11/crls ;
  cert_policy = none ;
 }
 use_mappers = digest, cn, pwent, uid, mail, subject, null ; # spécifie les mappers à utiliser:
  # subject Sujet du certificat
  # pwent  CN ou gecos
  # ldap  mapper LDAP
  # opensc Cherche le certificat dans ${HOME}/.eid/authorized_certificates
  # openssh Cherche la clé public du certificat dans ${HOME}/.ssh/authorized_keys
  # mail  compare le champs email du certificat
  # ms  Utilise l'UPN
  # krb  Compare avec le Kerberos Principal Name
  # cn  Compare le CN
  # uid  Compare l'UID
  # digest Certificat Digest
  # generic Contenu définis par l'utilisateur
  # null  blind access/deny mapper
 mapper_search_path = @libdir@/pam_pkcs11 ; # chemin des module mapper.
 mapper subject {    # Mapper le sujet du certificat pour le login, fournis dans un fichier sous la forme
"Subject -> login"
  debug = false ;
  #module = @libdir@/pam_pkcs11/subject_mapper.so ;
  module = internal ;
  ignorecase = false ;
  mapfile = file :///etc/pam_pkcs11/subject_mapping ;
 }
 mapper pwent {    # map le CN à getpwent()
  debug = false ;
  ignorecase = false ;
  module = internal ;
  #module = @libdir@/pam_pkcs11/pwent_mapper.so ;
 }
 mapper ldap {    # mapper d'annuaire
  debug = false ;
```

```
  module = @libdir@/pam_pkcs11/ldap_mapper.so ;
  ldaphost = "" ;      # fqdn du serveur ldap (utilise l'URI ldap pour en spécifier plusieurs)
  ldapport = ;      # port du serveur ldap s'il n'est pas donné dans l'URI
  URI = "" ;     # list d'URI utilisés dans l'ordre
  scope = 2 ;      # scope de recherche : 0 (base), 1 (one), 2 (sub)
  binddn = "cn=pam,o=example,c=com" ;  # DN du bind. doit avoir un accès en lecture
  passwd = "" ;      # password du binddn
  base = "ou=People,o=example,c=com" ; # base de recherche
  attribute = "userCertificate" ;  # attribut qui contient le certificat
  filter = "(&(objectClass=posixAccount)(uid=%s))" # Filtre de recherche pour les entrées utilisateurs ( doit
seulement laisser passer l'entrée pour le login utilisateur )
  ssl = tls     # off (désactiver), tls (activer tls), on|ssl (activer ssl)
  tls_randfile =      # chemin de la source entropie
  tls_cacertfile = /etc/ssl/cacert.pem # Chemin vers le certification pour l'authentification du pair
  tls_cacertdir =     # Répertoire contenant les certificats X509 pour l'authentification du pair
  tls_checkpeer = 0    # 0 ou 1. spécifie de vérifier le certificat.
  tls_ciphers =      # Chiffrement à utiliser
  tls_cert = ...     # Chemin du fichier contenant le certificat local pour l'authentification TLS client
  tls_key = ...     # Chemin du fichier contenant la clé privée pour l'authentification TLS client
 }
 mapper opensc {      # recherche les certificats depuis $HOME/.eid/autorized_certificates qui match
l'utilisateur
  debug = false ;
  module = @libdir@/pam_pkcs11/opensc_mapper.so ;
 }
 mapper openssh {     # Cherche les clés publique dans $HOME/.ssh/authorized_keys qui match l'utilisateur
  debug = false ;
  module = @libdir@/pam_pkcs11/openssh_mapper.so ;
 }
 mapper mail {     # Compare le champ email du certificat
  debug = false ;
  module = internal ;
  # module = @libdir@/pam_pkcs11/mail_mapper.so ;
  mapfile = file :///etc/pam_pkcs11/mail_mapping ; # Déclarer un fichier de map, 'empty' ou vide pour ne pas
utiliser de fichier de map
  ignorecase = true ;    # ignorer la casse
  ignoredomain = false ;    # Vérifie le domaine mx. en utilisant un fichier de map, cette option est
ignorée.
 }
 mapper ms {      # utilise l'UPN (login@AD_domain)
  debug = false ;
  module = internal ;
  # module = @libdir@/pam_pkcs11/ms_mapper.so ;
  ignorecase = false ;
  ignoredomain = false ;
  domain = "domain.com" ;
 }
 mapper krb {     # Compare avec le principal kerberos
  debug = false ;
  module = internal ;
  # module = @libdir@/pam_pkcs11/krb_mapper.so ;
  ignorecase = false ;
  mapfile = "none" ;
 }
 mapper cn {      # le CN est le login
  debug = false ;
  module = internal ;
  # module = @libdir@/pam_pkcs11/cn_mapper.so ;
  ignorecase = true ;
```

```
  # mapfile = file :///etc/pam_pkcs11/cn_map ;
  mapfile = "none" ;
 }
 mapper uid {     # l'uid (s'il existe) est le login
  debug = false ;
  module = internal ;
  # module = @libdir@/pam_pkcs11/uid_mapper.so ;
  ignorecase = false ;
  mapfile = "none" ;
 }
 mapper digest {     # Evalue le digest du certificat et le map dans un fichier
  debug = false ;
  module = internal ;
  # module = @libdir@/pam_pkcs11/digest_mapper.so ;
  algorithm = "sha1" ;     # Algorithme utilisé pour évaluer le digest
("null","md2","md4","md5","sha","sha1","dss","dss1","ripemd160")
  mapfile = file :///etc/pam_pkcs11/digest_mapping ;
  # mapfile = "none" ;
 }
 mapper generic {     # mapper de contenu générique
  debug = true ;
  #module = @libdir@/pam_pkcs11/generic_mapper.so ;
  module = internal ;
  ignorecase = false ;    # ignirer la casse
  cert_item = cn ;     # ( "cn" , "subject" , "kpn" , "email" , "upn" ou "uid" )
  mapfile = file :///etc/pam_pkcs11/generic_mapping ;
  use_getpwent = false ;    # utiliser petpwent() pour mapper le login
 }
 mapper null {     # Pas de mappage. Quand un utilisateur match à NULL ou nobody
  debug = false ;
  # module = @libdir@/pam_pkcs11/null_mapper.so ;
  module = internal ;
  default_match = false ;    # match toujours (true) ou échoue toujours (false)
  default_user = nobody ;    # en cas de match, retourner cet utilisateur
 }
}
```