01 - Installation et paramétrage de pam_cas

Généralités

pam_cas est un module qui permet à un 'service' UNIX sachant authentifier via pam d'utiliser le mécanisme de SSO du CAS.

La version originale a été dévelopée par l'Université de Yale. Elle est disponible ici.

Des modifications esup-portail ont été apportées afin de le rendre paramétrable.

Le fonctionnement global du module est le suivant :

- Il reçoit de pam deux informations : le nom d'utilisateur et le PT (Proxy Ticket) passé comme mot de passe.
- Il génère ensuite une connexion http(s) directe vers le serveur CAS, à l'url de validation de PT (en standard, /proxyValidate).
- Il analyse le retour de cette requête, reçue en xml : validation du PT, identité de l'utilisateur, hiérarchie de proxies par lesquels le PT a été obtenu.
- Il fait un controle du nom de proxy pour des raisons de sécurité, et s'assure que l'identité de l'utilisateur retournée par le serveur CAS correspond à celle passée par pam.
- Il retourne à pam le code PAM_SUCCESS en cas de réussite, PAM_AUTH_ERR dans le cas contraire.

Modifications esup

Les modifications apportées par rapport à la version de yale sont les suivantes :

Fichier de configuration

Il n'y a plus de paramètres 'en dur' dans les programmes.

La distribution proposée utilise un fichier de paramètres (par défaut, /etc/pam_cas.conf).

Gestion des connexions http(s)

La partie de code consacrée à la connexion http ou https a été entièrement ré-écrite. Elle s'appuie toujours sur l'API openssl, mais en utilisant des fonctions de plus haut niveau qu'auparavant (objet BIO).

Possibilités de debug

Il est possible maintenant de débuguer le module sans avoir à retoucher le code, par simple modification d'un paramètre du fichier de configuration.

De nombreuses informations de débugging ont été rajoutées. Les messages dans la log d'erreurs sont maitenant explicites.

Enfin, le binaire castest a été complétement ré écrit afin de pouvoir tester hors module pam la partie communication avec le serveur CAS.

Comportements optionnels

- L'administrateur peut maintenant choisir entre http ou https pour la validation des proxy ticket.
- Il est possible de ne plus contrôler le proxy qui a généré le Proxy Ticket.

Divers

- Le Makefile a été entièrement ré écrit
- Corrige un bug identifié avec certaines versions de redhat

Compilation du module

En pré-requis, openssl doit être installé.

Sur redhat, le package pam-devel doit également être installé.

- cd sources
- ln -s Makefile.Redhat Makefile (sous redhat, ou autre Makefile en fonction de l'OS).
- make (compilation et création du module pam_cas.so)
- make test (si vous désirez utiliser le binaire castest)

Faire un contrôle du binaire généré : ldd pam_cas.so.
Si des fichiers d'include ou des librairies manquent, adapter l'entête du Makefile.

Tenter un ./castest sans paramètre : il doit afficher un message d'erreur du genre :

```
unable to open config file "/etc/pam_cas.conf"
Error while reading config file. Error 1
cannot open it
```

Fichier de configuration

Par défaut, le fichier de configuration est /etc/pam_cas.conf.

Les directives doivent commencer en début de ligne.

Les lignes vides sont autorisées.

Une ligne 'active' se compose d'une directive, suivie d'espaces et/ou tabulation, puis d'une valeur.

Les directives ne sont pas sensibles à la casse.

Les directives de type booléen acceptent les valeurs "on" ou "off".

Les directives sont les suivantes :

- **host**
  obligatoire. C'est le nom de host du serveur CAS (ex : secure.its.yale.edu).
- **port**
  facultatif. C'est le port TCP pour contacter le serveur CAS en http ou https. Par défaut : 80 si http, ou 443 si https
- **uriValidate**
  facultatif. C'est l'URI utilisée pour la validation d'un Ticket.
  Par défaut,/proxyValidate
- **ssl**
  facultatif, booléen. Si on, la connexion se fera en https, si off en http
  Par défaut, prend la valeur on
- **debug**
  facultatif, booléen.
  Si on, des informations de debug sont envoyées dans le syslog (niveau LOG_DEBUG).
  par défaut, prend la valeur off
- **proxy**
  facultatif. On y paramètre le nom du proxy CAS qui a obtenu le Proxy Ticket.
  Il peut y avoir plusieurs directivesproxy.
  Si pas de directive proxy, pam_cas n'effectue pas ce contrôle.
  ex :proxy https://uportal1.its.yale.edu/CasProxyServlet
- **trusted_ca**
  facultatif si ssl a la valeur off.
  Contient le nom d'un fichier qui contient un ou plusieurs certificats au format pem.
  Ce certificat sert à valider la connexion https.
  Si le certificat X509 du serveur CAS est auto-signé, ce certificat doit être dans le fichier.
  Si le certificat a été signé par une Autorité de Certification, c'est le certificat de l'AC de plus haut niveau de la chaine de certification qui doit être dans le fichier.

Voici un exemple de fichier de configuration : pam_cas.conf.

Utilisation du module

On va prendre pour exemple l'intégration pour le serveur imap

- Recopier le module pam_cas.so dans /lib/security/.
  ajuster le propriétaire et les droits
- Adapter le fichier /etc/pam.d/imap (voir exemple ci-après)
- Arrêter - Relancer le démon utilisateur (dans le cas de cyrus-imap, c'est le démon saslauthd).
  En fait, cette dernière étape n'est à priori pas nécessaire.

Paramètres du module

- -s : paramètre obligatoire. Contient l'URL du service tier. Cette URL doit être identique à celle que le proxy CAS a passé au serveur CAS lors de la demande du PT.
- -f : paramètre facultatif. C'est le nom du fichier de paramètres (par défaut, /etc/pam_cas.conf).

Exemple de fichier /etc/pam.d/imap

Ce fichier suppose que imap tente l'authentification CAS, puis LDAP, puis UNIX

```
auth sufficient /lib/security/pam_cas.so \-simap://imap.univ.fr \-f/etc/pam_cas.conf
auth sufficient /lib/security/pam_ldap.so
auth required /lib/security/pam_pwdb.so shadow nullok
account required /lib/security/pam_pwdb.so shadow nullok
```

A remarquer qu'on peut mettre pam_cas sans craintes de surcharge du serveur imap : il ne fait de requêtes auprès du serveur CAS que si le mot de passe reçu ressemble à un PT ou un ST.

En cas de problème

Logs 'normaux'

Des logs sont produits en cas d'erreur, aux niveaux LOG_NOTICE et LOG_ERR

Des logs de debug sont générés si debug=on dans le fichier de configuration. Ils sont alors produits au niveau LOG_DEBUG.

castest

C'est un utilitaire fourni dans le 'package' pam_cas. Il permet de générer des requêtes de validation de ticket auprès du serveur CAS. Il utilise les mêmes librairies que le module pam_cas, et utilise également le fichier de config.

Il est vivement conseillé de l'utiliser avant la mise en service du module pam_cas ; cela permet de debuguer la configuration.

Il affiche sa configuration, puis tente de valider le ticket. Il affiche alors les messages que sort pam_cas en mode debug, plus d'autres informations.

Il supporte les arguments suivants (tous facultatifs) :

- le service CAS. Par défaut, https://foo.fr
- un ticket (ST ou PT) à valider. Par défaut, PT-1-xxx
- le nom du fichier de configuration. Par défaut, /etc/pam_cas.conf
  Donc, pour l'utiliser, préparez déja votre fichier de configuration.

exemple d'utilisation :

```
$PAM_CAS_SOURCES/castest <service> <ticket> <fichier_de_configuration>
```

Il est possible de le lancer sans paramètres. Avec un fichier de configuration correctement paramétré, la sortie est la suivante :

```
------------------------------------------------------------\-
Parameters from test :
host = auth.univ-nancy2.fr
port = 443
uri = /proxyValidate
ssl = on
trusted_ca = /Cert/ac-racine.pem
debug = localtest
proxy = [https://imp.univ.fr/cas/casProxy.php]
proxy = [https://ent.univ.fr/CasProxyServlet]
service = [https://foo.fr]
ticket = PT-1-xxx
\------------------------------------------------------------\-
authentication failure
<cas:serviceResponse xmlns:cas='http://www.yale.edu/tp/cas'>
<cas:authenticationFailure code='INVALID_TICKET'>
ticket 'PT-1-xxx' not recognized
</cas:authenticationFailure>
</cas:serviceResponse>
\------------------------------------------------------------\-
invalid ticket : bad CAS ticket
```

On a donc validé les paramètres de communication avec le serveur CAS très simplement.

Si problème, s'assurer avec wget que la machine qui supporte pam_cas arrive bien à communiquer avec le serveur CAS. Par exemple :

```
wget \--ca-certificate=/Cert/ac-racine.pem \-O /tmp/cas.log "https://auth.univ.fr:443/proxyValidate?ticket=PT-1-
xxx&service=[https://foo.fr]"
```

Le contenu du fichier /tmp/cas.log contient la réponse du serveur CAS ; il devrait être :

```
<cas:serviceResponse xmlns:cas='http://www.yale.edu/tp/cas'>
<cas:authenticationFailure code='INVALID_TICKET'>
ticket 'PT-1-xxx' not recognized
</cas:authenticationFailure>
</cas:serviceResponse>
```

Rem : si on ne veut pas valider le certificat du serveur https lors du wget, il suffit de remplacer --ca-certificate=/Cert/ac-racine.pem par --no-check-certificate ; un warning sera néammoins affiché lors du wget.

testImap.php

Cet utilitaire permet de tester le fonctionnement d'un serveur IMAP CAS-ifié.

Cache des tickets

pam_cas permet de valider un ticket (PT ou ST) présenté comme un simple mot de passe.

Pour des raisons de performance, il est souvent nécessaire de "cacher" ce ticket afin de pouvoir le rejouer plusieurs fois.

Ce document décrit l'exemple d'imap. Imap est utilisé par les webmail (cassifiés), l'ent (cassifié) et les clients de messagerie (non cassifié).

Les Webmails ont la particularité de générer de nombreuses connexions imap, au moins une par chargement de page.

Il serait très couteux que le webmail redemande un nouveau ticket pour chaque ouverture de connexion IMAP ; un cache doit donc être installé côté serveur, et le code du client webmail doit être modifié afin de gérer au mieux ce mécanisme de cache.

Plusieurs mécanismes de cache coté serveur sont connus. Nous conseillons l'utilisation de **cacheDirectory** dans esup-pam-cas (fonctionnalité ajoutée dans la version 2.0.11-esup-2.0.6)

esup-pam-cas cacheDirectory

Avant l'activation :

- configurer pam_cas_expire.conf (optionnel)

- mettre en place un cron exécutant le script pam_cas_expire

Puis activer l'option cacheDirectory dans pam_cas.conf

cyrus - saslauthd

C'est un démon qui est livré avec la distribution cyrus-imap.

cyrus-imap peut être paramétré pour authentifier les connexions (imap, pop, ...) à l'aide de ce démon, via une socket unix.

saslauthd peut être paramétré pour authentifier à l'aide de pam. Il est alors possible d'utiliser pam_cas pour cassifier saslauthd, donc cyrus-imap.

Et il est possible de paramétrer saslauthd pour qu'il implémente un cache de mot de passe (donc, éventuellement, de tickets).

Un patch permet de le rendre capable de conserver en cache plusieurs mots de passe pour un utilisateur ; ceci permet d'optimiser le mécanisme lorsque des sources différentes (ent, webmail, client lourd) génèrent en parallèle des requêtes IMAP, avec chacune un mot de passe différent.

pam_ccreds

C'est un module pam qui permet de cacher les mots de passe.

Il est développé par PADL Software.

Des patchs ont été proposés par Nicolas Bouillis fin de l'adapter au cache de ticket, et à apporter des fonctionnalités supplémentaires.

Une documentation de mise en oeuvre est proposée par Damien Mascré (IUT Villetaneuse - Paris 13).

Voir également un échange de mails à ce sujet sur la liste esup-devel.

Pour les personnes qui ne sont pas abonnées à cette liste, ce mail est également disponible, ainsi que le document attaché (patchs).

imapproxy

C'est une solution utilisée par certains établissements, qui semble donner satisfaction.

Voir les documents suivants :

- http://shib.kuleuven.be/docs/horde3-cas/proxyCAS.pdf
- http://shib.kuleuven.be/docs/horde3-cas/
- http://wiki.horde.org/CASAuthHowTo