Documentation

- Téléchargement
  - Université Yale
  - Distribution esup-portail
  - Distribution RPM INT Evry
- Généralités
- Restrictions
  - Requêtes HTTP différentes de GET
  - Authentification, et non autorisation
  - Précautions lors de la configuration apache
    - Anomalie DirectoryIndex (1)
    - Anomalie DirectoryIndex (2)
- Modifications apportées
  - Paramètres de configuration
  - Gestion des connexions http(s)
  - Gestion des cookies
- Compilation du module
  - Si apache est compilé localement
  - Si apache est installé en package binaire
- Paramétrage Apache
  - Chargement du module
  - Configuration
    - AuthType CASVoir
    - CASLocalCacheFile
    - CASLocalCacheSize
    - CASLocalCacheTimeout
    - CASLocalCacheInsecure
    - CASEGDFile
    - CASLogoutParameter et CASLogoutLocalURL
    - CASHost
    - CASPort
    - CASLoginURL
    - CASTrustedCACert
    - CASValidate
    - CASCookieName
    - CASCookieDomain
    - CASCookiePath
    - CASDebug
- Exemple de configuration

Téléchargement

Université Yale

C'est le module initial. Le principal inconvénient est que le paramétrage se fait 'en dur', avant compilation.

Il est disponible ici :http://www.ja-sig.org/wiki/display/CAS/Yale+CAS+client+distribution

Distribution esup-portail

C'est la distribution précédente, modifiée par esup-portail. Ce document en décrit les différences.

Elle est disponible ici :http://sourcesup.cru.fr/frs/?group_id=214&release_id=711.

Distribution RPM INT Evry

C'est une http://www.ja-sig.org/wiki/display/CAS/RPM+Modules du package esup-portail, proposée par Jehan Procacci, de l'int d'Evry.

Généralités

mod_cas est un module apache V2 qui permet de filtrer des accès http à l'aide de directives 'require user' ou 'require valid-user' en utilisant le mécanisme de SSO CAS.

Il est distributé par l'Université de Yales avec les clients cas. Voir la documentation Yale associée.

esup-portail a apporté des modifications importantes à ce module.

Le fonctionnement global du module est le suivant :

Lors d'un accès http vers une url protégée par mod_cas,

- Si le navigateur n'a pas encore de 'session' http valide établie avec le serveur :
- mod_cas redirige le navigateur vers l'url de login CAS pour authentification
- Le navigateur revient du serveur CAS porteur du Service Ticket (ST), en paramètre de GET
- mod_cas valide directement le ST auprès du serveur CAS en https ; il connait donc le 'login' de l'utilisateur.
- mod_cas crée alors une session :
- création d'un identifiant local de session
- écriture dans un fichier 'cache' de cet id de session, du timestamp et du login correspondant
- écriture de cet identifiant de session dans un cookie auprès du navigateur
- Sinon, le navigateur a déja établi une session, et celle-ci est valide (le délai de fin de session n'est pas expiré) :
- lecture du cookie de session
- Validation de celui-ci (donc, récupération locale du login). Il n'y a alors pas de trafic spécifique lié à l'authentification CAS.
  L'utilisation de la session est facultative. Elle est en fait pratiquement indispensable pour des raisons de performances.

Restrictions

L'utilisation de ce module amène certaines restrictions.

La plupart des problèmes liés à ces restrictions n'apparaissent que lors d'un accès http en n'étant pas en session 'mod_cas'.

Requêtes HTTP différentes de GET

Pour authentifier CAS, il est nécessaire de rediriger le navigateur vers le serveur CAS ; si la requête initiale est un POST ou d'autre requêtes porteurs d'informations dans le corps de la requête, ces informations sont perdues.

Un exemple :

un POST vers le serveur apache, alors qu'il n'y a pas encore de session mod_cas.

mod_cas redirige le navigateur vers le serveur CAS, via un GET HTTP. Après authentification, le navigateur va revenir à l'URL initiale, porteur du Service Ticket (ST), mais par un GET HTTP et non par le POST d'origine.

Les informations du POST sont donc perdues.

Authentification, et non autorisation

CAS ne gère pas les autorisations. Aussi, il est possible de poser des directives "require user toto", ou "require valid-user", mais pas "require group machin".

Il faudrait pour celà chainer mod_cas avec un autre module traitant des autorisations ... à faire.

Précautions lors de la configuration apache

Anomalie DirectoryIndex (1)

Des anomalies dans le traitement de l'option "Indexes" ont déja été rencontrées lorsque les directives mod_cas sont incluses dans un container "Location" plutot que dans un container "Directory".

Le problème rencontré est le suivant :

Si on n'est pas encore en session mod_cas et qu'on tente d'accéder directement à un répertoire, apache ne traite pas les éventuels fichiers d'index (index.html, ...).

Si vous rencontrez ce problème et que les directives CAS sont dans un container "Location", essayez de les mettre dans un container "Directory" (ou dans un .htaccess).

Anomalie DirectoryIndex (2)

Une autre anomaile relevée ; mise en évidence avec le module php.

Voici l'exemple de config :

```
<Directory />
DirectoryIndex index.html index.htm index.php
...
</Directory>

<Files *.php>
   SetOutputFilter PHP
   SetInputFilter PHP
</Files>
```

Dans cet exemple, index.php ne sera pas 'exécuté' lorqu'on n'est pas encore en 'session mod_cas'.
Il faut remplacer le <Files *.php> par :

```
AddType application/x-httpd-php .php
```

De la même facon, il faut retirer tout <Files *.php> qui seraient dans la config apache. Par exemple, mod_ssl donne de base :

```
<Files ~ ".(cgi|shtml|phtml|php|php3?)$">
   SSLOptions +StdEnvVars
</Files>
```

il faut le retirer (au moins pour php)

Modifications apportées

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

Paramètres de configuration

La plupart des paramètres de configuration de la distribution yale sont 'en dur' dans un fichier include (cas.h). Ils sont donc figés lors de la compilation.

Les modifs esup-portail permettent de gérer ces paramètres au niveau de la config apache.

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).

En fait, le code est repris du Module PAM pour CAS d'esup-portail.

Le code devient donc plus simple.

La gestion des certificats devient également plus souple :

Si le certificat du serveur CAS a été délivré par une Autorité de Certification, il était nécessaire avec le mod_cas initial de passer en paramètre tous les certificats de la chaine de certification ; avec mod_cas esup-portail, seul le certificat de l'autorité de certification racine est nécessaire.

Gestion des cookies

Il est possible de forcer le domaine et le 'path' du cookie de mod_cas.

Cette fonctionnalité peut être nécessaire, par exemple dans le cas d'un apache gérant de multiples serveurs virtuels ; par exemple, www1.univ.fr, www2.univ.fr, ...

Le domaine peut être forcé à ".univ.fr", qui est valide pour tous les domaines enfants de univ.fr.

Compilation du module

Il faut compiler le module mod_cas ; a partir de la version 2.0.11-esup-2, il n'est plus nécessaire de patcher mod_dir.

Deux types de compilation sont possibles, selon qu'apache a été installé en 'package binaire', ou qu'il soit compilé localement.

en pré-requis, openssl doit être installé.

Si apache est compilé localement

Il suffit de recopier le répertoire sources vers le répertoire modules de la distribution source d'apache.

Ensuite, lancer ./buildconf dans le répertoire des sources apache.

Puis, ajouter --enable-cas=shared lors du 'configure'.

Si apache est installé en package binaire

Cette procédure fonctionne sour redhat ou fedora, elle est à adapter pour d'autres configurations.

- cd sources
- apxs -I/usr/kerberos/include/ -i -l ssl -c mod_cas.c ssl_client.c
  L'inclusion de /usr/kerberos/include n'est pas nécessaire sur tous les systèmes

Paramétrage Apache

Lire le fichier README.yale du répertoire sources/cas.

C'est un fichier distribué avec le package cas-client de Yale.

Chargement du module

Il faut indiquer à Apache de charger le module mod_cas. Ceci se fait par la directive :

```
LoadModule cas_module modules/mod_cas.so
```

Configuration

La configuration se fait dans un 'container' Directory, ou dans un fichier .htaccess.

Les différents paramètres possibles sont les suivants :

AuthType CASVoir

README.yale

Indique que l'on veut authentifier à l'aide du mécanisme CAS.

CASLocalCacheFile

Voir README.yale. Facultatif.

C'est le nom (chemin complet) d'un fichier qui va servir au cache local pour le mécanisme de session.

S'il n'est pas spécifié, mod_cas ne gère pas de mécanisme de session (déconseillé).

Par exemple :

```
CASLocalCacheFile /tmp/CAScache
```

CASLocalCacheSize

Facultatif.

C'est la taille en nombre d'entrées du cach local de session.

Par défaut : 1000

CASLocalCacheTimeout

Voir README.yale. Facultatif.

C'est la durée de validité du cache de session, en minutes.

Par défaut, 3600 (une heure)

CASLocalCacheInsecure

Voir README.yale. Facultatif.

Force un cookie 'secure' si Off. Mette à On si on ne désire pas ce type de cookie.

Par défaut, Off

CASEGDFile

Voir README.yale. Facultatif.

CASLogoutParameter et CASLogoutLocalURL

Pas documenté. Sert à forcer un logout CAS

CASHost

Ajout esup-portail. Obligatoire.

C'est le nom de host du serveur CAS

Par exemple :

```
CASHost auth.univ.fr
```

CASPort

Ajout esup-portail. Facultatif

C'est le port d'accès au serveur CAS.

Par défaut, 443.

CASLoginURL

Ajout esup-portail. Obligatoire

C'est l'url complète de login CAS.

Par exemple :

```
CASLoginURL https://auth.univ-nancy2.fr
```

CASTrustedCACert

Ajout esup-portail. Obligatoire

C'est un nom de fichier complet qui contient le certificat (format PEM) permettant de valider (coté SSL) la requête https de validation de ticket CAS.

Si le certificat du serveur CAS est autosigné, c'est celui-ci qui est présenté.

Si le certificat du serveur CAS a été délivré par une Autorité de Certification, c'est le certificat de l'Autorité racine qui doir être présenté.

Par exemple :

```
CASTrustedCACert /Cert/ac-racine.pem
```

CASValidate

Ajout esup-portail. Facultatif

C'est l'uri du serveur CAS utilisée pour faire valider un ticket CAS (en protocole CAS V1)

Par défaut, /cas/validate

CASCookieName

Ajout esup-portail. Facultatif

C'est le nom du cookie de session mod_cas.

Par défaut,MODCASID

CASCookieDomain

Ajout esup-portail. Facultatif

C'est le domaine du cookie de session CAS.

Par défaut, aucun domaine n'est indiqué lors de la création du cookie ; le domaine est donc celui d'accès au serveur virtuel apache. Si le serveur est accédé en "http://www.unserv.univ.fr", le domaine par défaut sera : "www.univ-nancy2.fr".

Exemple d'utilisation :

```
CASCookieDomain .unserv.univ-nancy2.fr
```

(ceci crée un cookie accessible à tout site dans la hiérarchie deunserv.univ.fr, commewww1.unserv.univ.fr).

CASCookiePath

Ajout esup-portail. Facultatif

C'est le "path" du cookie de session CAS.

Par défaut, /

CASDebug

Ajout esup-portail.

Valeur booléenne (On ou Off) ; facultatif

Permet d'écrire des traces de debug dans la log d'erreur d'apache.

Ne pas utiliser en production, le volume de log est très important.

Exemple de configuration

On veut limiter l'accès à http://exemple.univ-fr/essai à toute personne authentifiée CAS, et à http://exemple.univ-fr/essai/toto à l'utilisateur toto

```
LoadModule cas_module modules/mod_cas.so
...

<IfModule mod_cas.c>
CASLocalCacheInsecure On
CASLocalCacheFile /tmp/CAScache
CASTrustedCACert /Cert/ac-racine.pem
CASLoginURL [https://auth.univ.fr]
CASHost auth.univ.fr
CASPort 443
CASValidate /validate
# CASDebug on
</IfModule>
...
Directory /home/essai>
<IfModule mod_cas.c>
AuthType CAS
AuthName "bases en ligne"
Require valid-user
</IfModule>
</Directory>
Directory /home/essai/toto>
Require user toto
</Directory>
```