Utiliser le CAS (Recommandé)

Le CAS est ce qu'on appelle un système de Single Sign-On (SSO). Il permet de centraliser l'authentification des utilisateurs ce qui permet d'éviter à devoir se reconnecter à tous les sites qui demandent une authentification. De plus cela permet d'améliorer la sécurité en évitant de devoir taper son mot de passe sur de multiples sites web. **C'est la méthode recommandée par le CSN, elle est donc à privilégier.**

Utiliser le CAS dans un `.htaccess`

La méthode la plus simple pour restreindre l'accès à un page web avec l'authentification CAS est d'ajouter un fichier `.htaccess` à la racine du site. La configuration est la suivante :

```
AuthType Cas
AuthName "Authentication Required"
CASAuthNHeader CAS-User
Require valid-user
```

Lors d'une tentative d'accès au site, l'utilisateur sera alors redirigé vers le CAS pour s'authentifier.

Utiliser l'authentification CAS en PHP

Préparation

Avant toute chose, il vous faut autoriser les sessions pour votre espace personnel. Pour cela, vérifiez que vous possédez un dossier `sessions` à la base de votre dossier personnel (à coté de `public_html`).

Installation

Téléchargez ensuite la dernière version stable de la librairie phpCAS comme indiqué sur le guide d'installation, et extrayez cette archive dans votre dossier `public_html`. La liste de toutes les versions est disponible ici. Il existe d'autres méthodes d'installation mais elles n'ont pas été testées sur Etud.

Par exemple pour la version 1.3.6, si vous êtes connecté en SSH à votre espace club/étudiant, utilisez les commandes suivantes dans le dossier `public_html` :

```
wget https://github.com/apereo/phpCAS/archive/1.3.6.tar.gz
tar zxvf 1.3.6.tar.gz
```

Vous aurez ainsi les fichiers suivants dans votre dossier `public_html`, (affichés avec la commande `ls`) :

```
1.3.6.tar.gz  phpCAS-1.3.6
```

Vous pouvez supprimer le fichier `1.3.6.tar.gz` car il n'est plus nécessaire, et renommer le dossier `phpCAS-1.3.6` comme vous voulez.

L'installation est maintenant terminée !

Utilisation

Voici un exemple d'utilisation. Si vous avez installé comme expliqué précédemment, créez un fichier `index.php` à côté du dossier de `phpCAS-1.3.6`. Le contenu du dossier public_html est alors le suivant :

```
index.php  phpCAS-1.3.6
```

Ouvrez le fichier `index.php` et ajoutez le contenu suivant (adapté de cet exemple) :

```
<?php
// Load the CAS lib
require_once("phpCAS-1.3.6/CAS.php");

// Enable debugging
phpCAS::setDebug();
// Enable verbose error messages. Disable in production!
phpCAS::setVerbose(true);

// Initialize phpCAS
```

```
phpCAS::client(CAS_VERSION_2_0, "cas.insa-toulouse.fr", 443, 'cas', true);

// For production use set the CA certificate that is the issuer of the cert
// on the CAS server and uncomment the line below
// phpCAS::setCasServerCACert($cas_server_ca_cert_path);

// For quick testing you can disable SSL validation of the CAS server.
// THIS SETTING IS NOT RECOMMENDED FOR PRODUCTION.
// VALIDATING THE CAS SERVER IS CRUCIAL TO THE SECURITY OF THE CAS PROTOCOL!
//phpCAS::setNoCasServerValidation();

// force CAS authentication
phpCAS::forceAuthentication();

// at this step, the user has been authenticated by the CAS server
// and the user's login name can be read with phpCAS::getUser().

// logout if desired
if (isset($_REQUEST['logout'])) {
        phpCAS::logout();
}

// for this test, simply print that the authentication was successfull
?>
<html>
  <head>
    <title>phpCAS simple client</title>
  </head>
  <body>
    <h1>Successfull Authentication!</h1>
    <p>the user's login is <b><?php echo phpCAS::getUser(); ?></b>.</p>
    <p>phpCAS version is <b><?php echo phpCAS::getVersion(); ?></b>.</p>
    <p><a href="?logout=">Logout</a></p>
  </body>
</html>
```

Voici une petite explication ligne par ligne de ce fichier :

```
require_once("phpCAS-1.3.6/CAS.php");
```

Cette ligne est essentielle ! Elle permet de charger la librairie CAS dans votre script PHP. Sans cette ligne vous ne pourrez pas utiliser les fonctions du CAS. Dans cet exemple, la librairie est dans le dossier `phpCAS-1.3.6`, mais si vous avez renommé ce fichier ou l'avez placé dans un autre dossier, entrez le chemin vers ce dossier à la place.

```
phpCAS::setDebug();
phpCAS::setVerbose(true);
```

À supprimer pour un site en production ! Ces lignes sont utiles quand vous développez votre site pour voir les erreurs CAS. Quand vous pensez avoir un système robuste, pensez à enlever ces lignes.

```
phpCAS::client(CAS_VERSION_2_0, "cas.insa-toulouse.fr", 443, 'cas', true);
```

Cette ligne permet d'initialiser le CAS et le lier au serveur de l'INSA. Le dernier argument permet d'activer les sessions PHP (rappelez vous le dossier `sessions` que vous aviez créé au début).

```
phpCAS::setNoCasServerValidation();
```

Théoriquement, il faudrait pas utiliser cette ligne mais je sais pas où trouver le certificat du serveur de l'INSA donc bon.

```
phpCAS::forceAuthentication();
```

Cette ligne permet de n'afficher la page que si l'utilisateur est authentifié avec le CAS.

```
if (isset($_REQUEST['logout'])) {
        phpCAS::logout();
}
```

Ce bloc permet de déconnecter l'utilisateur si le paramètre `logout` est présent dans l'url. Par exemple, si votre site est à l'adresse `https://etud.insa-toulouse.fr/~CAStest`, alors `https://etud.insa-toulouse.fr/~CAStest?logout=` déclenchera la déconnexion.

```
<html>
  <head>
    <title>phpCAS simple client</title>
  </head>
  <body>
    <h1>Successfull Authentication!</h1>
    <p>the user's login is <b><?php echo phpCAS::getUser(); ?></b>.</p>
    <p>phpCAS version is <b><?php echo phpCAS::getVersion(); ?></b>.</p>
```

```
    <p><a href="?logout=">Logout</a></p>
  </body>
</html>
```

Ce bloc est le HTML affiché sur la page. Comme il ne sera affiché que si l'utilisateur est connecté, il est donc possible d'utiliser sans problème les méthodes de `phpCAS` pour récupérer des informations sur l'utilisateur. Ici, le nom d'utilisateur et la version de `phpCAS` sont récupérés avec les méthodes `phpCAS::getUser()` et `phpCAS::getVersion()`. Il y a aussi un lien déclenchant la déconnexion comme expliqué précédemment.

Pour des utilisations plus complexes, regardez la liste de tous les exemples.

Si vous avez des problèmes, vous pouvez visiter le code source sur github et le wiki de phpCAS.

Laravel

Voici maintenant un exemple d'utilisation de l'authentification CAS avec le framework PHP Laravel.

Vous devez au préalable suivre l'installation via Composer comme indiqué en haut du guide d'installation de phpCAS.

Dans le fichier `config/auth.php`, vous devez rajouter :

```
/*
|--------------------------------------------------------------------------
| INSA CAS properties
|--------------------------------------------------------------------------
|
| This option controls the INSA CAS authentification system used
| to log in a different way the users.
|
*/
'cas' => [
  'debug'  => env('INSA_CAS_DEBUG', false),
  'server' => [
    'hostname' => env('INSA_CAS_HOSTNAME', ''),
    'port'     => (int) env('INSA_CAS_PORT', 443),
    'uri'      => env('INSA_CAS_URI', ''),
  ],
```

```
],
```

Dans le fichier `.env`, vous pouvez rajouter :

```
INSA_CAS_DEBUG=false
INSA_CAS_HOSTNAME=cas.insa-toulouse.fr
INSA_CAS_PORT=443
INSA_CAS_URI=cas
```

Le fichier `CAS.php` ci-dessous permet de regrouper dans un seul fichier l'utilisation de l'objet `phpCAS`.

```
<?php

namespace App\Traits;

use phpCAS;
use Illuminate\Http\RedirectResponse;

/**
 * Class CAS
 * @package App\Traits
 *
 * @author: Damien MOLINA
 */
class CAS {

    /**
     * Determine whether the CAS auth system is currently
     * in a debugging mode.
     *
     * @return bool
     */
    private static function isDebuggingMode() {
        return boolval(
            config('auth.cas.debug')
        ) ;
    }

    /**
```

```
    * Initialize the CAS object.
    *
    * @return void
    */
    private static function initialize() {
        if(CAS::isDebuggingMode()) {
            // Enable debugging.
            phpCAS::setDebug() ;

            // Enable verbose error messages.
            phpCAS::setVerbose(true) ;
        }

        phpCAS::client(
            CAS_VERSION_2_0, config('auth.cas.server.hostname'), config('auth.cas.server.port'),
config('auth.cas.server.uri'), true
        ) ;
    }

    /**
    * Make the authenticate request.
    *
    * @param string $returnRoute
    */
    private static function authenticate(string $returnRoute) {
        // We initialize the phpCAS component
        CAS::initialize() ;

        // We put the return URL.
        phpCAS::setFixedServiceURL($returnRoute) ;

        // Disable the CAS server verification. UNCOMMENT TO DEBUG.
        //phpCAS::setNoCasServerValidation() ;

        // force CAS authentication
        phpCAS::forceAuthentication() ;
    }

    /**
    * Make the authentification request.
```

```
    *
    * @param string $returnRoute
    * @return RedirectResponse|void
    */
    public static function request(string $returnRoute) {
        CAS::authenticate($returnRoute) ;

        /*
        * Now we are redirecting the user
        * to the login page.
        */
        return redirect()->to(
            phpCAS::getServerLoginURL()
        ) ;
    }

    /**
    * Get all the data from the request.
    *
    * @param string $returnRoute
    * @return array
    */
    public static function response(string $returnRoute) {
        /*
        * We make the request again to take
        * the server data without redirecting
        * the user.
        */
        CAS::authenticate($returnRoute) ;

        return [
            'login'   => phpCAS::getUser(),
            'version' => phpCAS::getVersion(),
        ] ;
    }

}
```

Vous pourrez ainsi utiliser l'objet `CAS` dans un contrôleur en définissant, par exemple, les méthodes `CASRequest` et `CASResponse` :

```
    /**
     * Make the CAS authentification request and
     * redirect the user to the login page.
     *
     * @return RedirectResponse
     */
    public function CASRequest() {
        return CAS::request(
            $this->getCasReturnRoute()
        ) ;
    }

    /**
     * Manage the user return.
     *
     * @return RedirectResponse
     */
    public function CASResponse() {
        $data = CAS::response(
            $this->getCasReturnRoute()
        ) ;

        $login = $data['login'] ;

        // Do something with the login
    }

    /**
     * Get the CAS return route.
     *
     * @return string
     */
    private function getCasReturnRoute() {
        return route('your-route') ;
    }
```

Dans ce cas, la variable `$login` contiendra l'identifiant utilisé dans le service CAS.

Utiliser l'authentification CAS dans d'autres langages

Le CAS propose des clients dans d'autres langages que le PHP. Ces utilisations ne sont pas couvertes par ce wiki car peu utilisées sur le serveur Etud. Si vous voulez tout de même utiliser l'un de ces clients, vous pouvez visiter le wiki du CAS.

Révision #18
Créé 24 août 2020 15:44:31 par Arnaud
Mis à jour 27 octobre 2022 15:59:16 par Manah