Inhaltsverzeichnis

Dokuwiki mit Forward Authentification

Wenn man Dokuwiki in einer Docker- Umgebung mit vorgeschaltetem Single-Sign-On (SSO) Proxy betreibt, übernimmt ja der Proxy schon alle Authentifizerungsarbeiten für den jeweiligen User und Dokuwiki muß nur noch wissen, wer sich da denn nun angemeldet hat.

Dazu gibt's (aus unbekannter Quelle) ein forwardauth- Plugin, was genau diesen Job übernimmt und quasi an Dokuwiki übermittelt, welcher User sich im reinkommenden HTTP- Request- Header denn versteckt hat.

Dazu legt man die untrige auth.ph in einen lib/plugins/forwardauth- Ordner und trägt in seiner conf/local.php (oder über das Admin- Panel) für den authtype den Wert forwardauth ein

```php
<?php
/**
 * Dokuwiki's Main Configuration File - Local Settings
 * Auto-generated by install script
 * Date: Tue, 25 Oct 2022 05:40:09 +0000
 */
$conf['title'] = 'Meine Wissensdatenbank';
$conf['lang'] = 'en';
$conf['license'] = '0';
$conf['useacl'] = 1;
$conf['superuser'] = '@admin';
$conf['savedir'] = '/app/www/public/data';
$conf['authtype'] = "forwardauth";
$conf['plugin']['forwardauth']['userHeader'] = 'Remote-User';
```

und hier das Plugin selber:

auth.php

```php
<?php
/**
 * ForwardAuth DokuWiki Auth Plugin.
 *
 * @licence    Public Domain, use how you wish, I don't caare.
 * @author     Alexander Dean-Kennedy
 * @version    0.0.1
 */

// must be run within Dokuwiki
if(!defined('DOKU_INC')) die();

/**
 * Provides support for using ForwardAuth middlewares in reverse
 * proxies for providing authentication to DokuWiki docker containers.
 *
```

```php
 * NOTE: This is an extremely simple implementation that only focuses
 * on looking-up HTTP Headers and does not support logout or even
 * header-validation.
 *
 * USAGE: Configure via conf/local.php like so:
 *
 * $conf['authtype'] = "forwardauth";
 * $conf['plugin']['forwardauth']['userHeader'] = 'Remote-User';
 *
 * // Optionally, you can also provide header names for the following:
 * $conf['plugin']['forwardauth']['nameHeader'] = 'Remote-Name';
 * $conf['plugin']['forwardauth']['mailHeader'] = 'Remote-Email';
 * $conf['plugin']['forwardauth']['groupsHeader'] = 'Remote-Groups';
 *
 * By default, if no value is provided for the name, we will default to
 * the value provided by 'userHeader'. Thus setting both the User's
 * Name and the username to the value of 'userHeader'.
 *
 * If no headers are provided, we default to the header names used by
 * Authelia, as it is what I use. Referenced here:
 *  https://github.com/authelia/authelia/blob/
 *      7c6a86882f93515a148cadcce8eccd77c3f24433
 *      /internal/handlers/const.go#L20-L23
 **/
class auth_plugin_forwardauth extends DokuWiki_Auth_Plugin
{
    // Defaults for Authelia.
    const USER_HEADER = "Remote-User";
    const NAME_HEADER = "Remote-Name";
    const MAIL_HEADER = "Remote-Mail";
    const GRPS_HEADER = "Remote-Groups";

    public function __construct()
    {
        parent::__construct();

        $this->cando['external'] = true;  // Assumes redirect already
happened.
        $this->cando['logout']   = false; // Logout happens elsewhere.
    }

    public function trustExternal($user, $pass, $sticky = false) {
        // We assume $user is ALWAYS nil and overwrite with header
value.
        $data = $this->getUserData($user);
        if ($data) {
            return $this->fillGlobals($data);
        }
        return false;
    }
    private function fillGlobals($data) {
```

```php
        global $USERINFO;
        $USERINFO['name'] = $data['user'];
        $USERINFO['mail'] = $data['mail'];
        $USERINFO['grps'] = $data['groups'];
        $_SERVER['REMOTE_USER'] = $data['user'];
        $_SESSION[DOKU_COOKIE]['auth']['user'] = $data['user'];
        $_SESSION[DOKU_COOKIE]['auth']['info'] = $USERINFO;
        return true;
    }

    public function getUserData($user, $requireGroups = true)
    {
// If no user is provided, no auth was done. Admin will
    // need to fix their ForwardAuth in the reverse proxy.
// For example, setting the ForwardAuth middleware.
// I do not believe there is a course of action we can
// perform here as we don't know the middleware path.
$user = $this->getAuthUser();
        if (empty($user)) { return false; }

        $data = array();
        $data['user']   = $user;
        $data['name']   = $this->getAuthName();
        $data['mail']   = $this->getAuthMail();
        $data['groups'] = $this->getAuthGroups();
        return $data;
    }


    /* Extract header values from HTTP Request. */
    private function getAuthUser() {
    return sane_lookup($this->getAuthUserHeader());
    }
    private function getAuthMail() {
    return sane_lookup($this->getAuthMailHeader());
    }
    private function getAuthName() {
    $name = sane_lookup($this->getAuthNameHeader());
    if (empty($name)) { return $this->getAuthUser(); }
    return $name;
    }
    private function getAuthGroups() {
    $groups = sane_lookup($this->getAuthGroupsHeader());
    return explode(',', $groups);
    }


    /* Generate PHP HTTP Header names from Config values. */
    private function getAuthUserHeader() {
        return header_to_php_name(
        plugin_conf('userHeader', self::USER_HEADER)
    );
    }
```

```php
    private function getAuthNameHeader() {
    return header_to_php_name(
        plugin_conf('nameHeader', self::NAME_HEADER)
    );
    }
    private function getAuthMailHeader() {
    return header_to_php_name(
        plugin_conf('mailHeader', self::MAIL_HEADER)
    );
    }
    private function getAuthGroupsHeader() {
    return header_to_php_name(
        plugin_conf('groupsHeader', self::GRPS_HEADER)
    );
    }
}

function plugin_conf($name, $default="") {
    global $conf;
    if (!array_key_exists('plugin', $conf)) { return $default; }
    if (!array_key_exists('forwardauth', $conf['plugin'])) { return
$default; }
    if (!array_key_exists($name, $conf['plugin']['forwardauth'])) {
return $default; }
    return $conf['plugin']['forwardauth'][$name];
}

function sane_lookup($header) {
    // Simplify header lookup by making sure it is set.
    return ($header === '') ? "" : $_SERVER[$header];
}

function header_to_php_name($header) {
    // Headers are all uppercase, and all '-' must be underscores.
    return ($header === '') ? "" : "HTTP_" . strtoupper(str_replace('-
', '_', $header));
}
```