Autenticación y Autorización en gvHIDRA

Gaspar Quiles – DSIC - UPV

24 marzo 2011

Autenticación/Autorización en gvHIDRA

- Objetivos
- Introducción
- Esquema en gvHIDRA
- Sin autenticación
- Autenticación mediante usuario y contraseña
- Autenticación externa
- Autorización: Módulos y Roles

Objetivos

- Conocer el funcionamiento de la autenticación / autorización en gvHidra y en PEAR::Auth
- Implementar distintas soluciones usando contenedores Auth predefinidos y personalizados
- Proporcionar las bases para integrar con la infraestructura de usuarios existente en la organización

Introducción

- El control de acceso significa qué usuarios pueden acceder a qué recursos. Básicamente trata los dos siguientes problemas:
  - **Autenticación**: Procedimiento de comprobación de la identidad de un usuario
  - **Autorización**: Definición granular de permisos de acceso concedidos a un determinado usuario, dispositivo o sistema
- La autenticación se basa en algo que el usuario sabe (p.e. contraseña), tiene (huella, retina, tarjeta inteligente, …) o se pueden usar distintos canales (web, email, movil, …).
- Autenticación de 2 o 3 factores

Esquema en gvHIDRA

- Tenemos dos fases:
    1. Comprobar credenciales (opcional)
    2. Carga de la información del usuario/aplicación
- En gvHIDRA definimos una clase (que hereda de gvhBaseAuth) que implementa la lógica usando PEAR:Auth. Es usada desde:
    1. login*.php → punto de entrada (autenticación)
    2. validacion::valida → inicializa params (autorización)
- Excepto el fichero de login que ha de situarse en la raiz de la aplicación, los otros dos pueden estar en cualquier sitio

Esquema en gvHIDRA

- Podemos especificar el módulo de validación a usar accediendo directamente a su fichero de login.
- Si no indicamos el fichero de login hay una redirección al primero de los ficheros con nombre login*.php
- Lo normal es que sólo haya uno por aplicación

Sin Autenticación

- Accedemos directamente con usuario prefijado
- Incluir en AuthBasic::autenticate (antes del start)

```
$aut->setAuth('invitado');
```

- Sólo recomendado para aplicaciones monousuario y acceso controlado por IP

Autenticación mediante usuario/contraseña

- Auth incluye muchos gestores de donde obtener los usuarios (BD, Array, LDAP, Radius, SOAP, IMAP, …) y permite definir nuevos.
- También podemos componer varios
- Para crear nuevos gestores se recomienda seguir los pasos en AuthGva en la doc.

Autenticación mediante usuario/contraseña

- La solución más simple seria tener los usuarios directamente en un array:

```
static $options = array(
  'cryptType'=>'md5',
  'users'=>array(
    'usu1'=>'529113007b15005637b3dad6d9ba2f10',   //md5(usu1)
    'usu2'=>'9c60c45d8440e2ece3442fed8fe4c5c2',   //md5(usu2)
));
$aut = new Auth('Array',self::$options);
$aut->start();
```

Autenticación mediante usuario/contraseña

- Se crea una entrada en $_SESSION:

```
    [_authsession] => Array (
            [challengekey] => d16758bcf878f81fb31f705ebbd5526a
            [data] => Array( )
            [sessionip] => 127.0.0.1
            [sessionuseragent] => Mozilla/5.0 (Windows; U; Windows NT 6.1;
es-ES; rv:1.9.2.13) Gecko/20101203 Firefox/3.6.13
            [sessionforwardedfor] =>
            [challengecookie] => 4be98d3d40a9b186c771519c313d1f07
            [registered] => 1
            [username] => usu1
            [timestamp] => 1300749848
            [idle] => 1300749848
        )
```

- Si vamos a tener varias aplicaciones en mismo dominio hay que cambiar el nombre:

```
$aut->setSessionName($p_apli); // entrada _auth_demo
```

Autenticación mediante usuario/contraseña

- El formulario de login es personalizable:

```
function milogin($username, $status, &$auth) {
  echo <<<cadena
<html>
<head>
<title>Formulario de acceso</title>
…
</html>
cadena;
}

$aut = new Auth('Array',self::$options,'milogin');
```

Autenticación mediante usuario/contraseña

- Una vez validado el usuario, hay que inicializar su información en esa aplicación (para aplicar autorización)
- Se hace en método validacion::valida. La información ha de seguir cierta estructura en la sesión, y se debe validar con:

```
$auth_container->checkData($_SESSION, $apli);
```

Autenticación mediante usuario/contraseña

- ¿De donde obtenemos la información?
  - Podemos conectarnos a un sistema externo (BD, ldap, …) y recuperarla
  - De forma estática en la aplicación (xml, php, ini...). !!No recomendable si hay muchos usuarios!!

Autenticación mediante usuario/contraseña

- También podemos usar más de una fuente de datos para los usuarios
- Veamos ejemplo con:
  - Base de datos (MDB2 mysql)
  - Array
- Auth prueba con cada contenedor

Autenticación mediante usuario/contraseña

```
static $options = array(
  array(
    'type'=>'MDB2',
    'options'=>array(
      'dsn'=>'mysql://root:gaspar@localhost/auth',
      'table'=>'usuarios',
      'usernamecol'=>'usuario',
      'passwordcol'=>'pwd',
      //'cryptType'=>'md5', // por defecto
  )),
  array(
    'type'=>'Array',
    'options'=>array(
      'cryptType'=>'md5',
      'users'=>array(
        'usu1'=>'529113007b15005637b3dad6d9ba2f10',   //md5(usu1)
        'usu2'=>'9c60c45d8440e2ece3442fed8fe4c5c2',   //md5(usu2)
)))));
$aut = new Auth('Multiple',self::$options);
```

Autenticación mediante certificados

- Otra posibilidad es identificarnos mediante un certificado digital (accv, dnie)
- Del certificado se obtiene el dni, y en aplicación hay que definir si se permite el acceso y a que usuario corresponde
- De momento en pruebas

Autenticación externa: 1. GET/POST

- Tenemos un sistema externo para comprobar las credenciales del usuario
- Recibimos el usuario como parámetro
- También podemos pasar otros parámetros usados en la aplicación
- Plataformas single sign on (sso)

Autenticación externa: 1. GET/POST

- Incluir en AuthBasic::autenticate (antes del start)

```
if (isset($_POST['user']))
    $aut->setAuth($_POST['user']);
```

- Invocar con formulario:

```
<form action="login.php" method="post">
 Soy <input type="text" size="20" name="user">
 <input type="submit" name="Entrar" value=" Entrar ">
</form>
```

Autenticación externa: 1. GET/POST

- Si no recibe parámetro muestra formulario de login
- Podemos evitarlo:

```
$aut->setAllowLogin(false);
if (isset($_POST['user']))
   $aut->setAuth($_POST['user']);
else
   return 'Usuario no identificado!!';
```

Autenticación externa: 1. GET/POST

- También podemos recibir otros parámetros para inicializar la sesión
- Los parámetros hay que redirigirlos por la sesión de Auth:
  - En autenticate se coge el parámetro:

```
$aut->setAuthData( 'apl', $_POST['apl'], true );
```

  - En postLogin se recupera:

```
$sess['daplicacion'] = $aut->getAuthData('apl');
```

Autenticación externa: 2. GET/POST encriptado

- Con lo anterior es muy sencillo suplantar a un usuario
- Para evitarlo usamos criptografia simétrica:
  - Se usa la misma clave para encriptar en origen y desencriptar en destino
  - Escogemos algoritmo encriptación (Cipher)
  - En servidor externo se encriptan los parámetros (paso 1)
  - En aplicación se desencriptan los parámetros (paso 2)

Autenticación externa: 2. GET/POST encriptado

- Paso 1: Invocar desde servidor externo:

```
include('Cipher.php');
$user = $_REQUEST['user'];
echo 'conectamos a usuario '.$user;
$cifrar = new Cipher('clave secreta');
$user_crypt = $cifrar->encrypt($user);
echo <<<eof
<form action="login.php" method="post">
 <input type="hidden" name="user_crypt" value="$user_crypt">
 <input type="submit" name="Entrar" value=" Confirmar ">
</form>
eof;
```

Autenticación externa: 2. GET/POST encriptado

- Paso 2: en aplicación recogemos parámetros

```
include('Cipher.php');
if (isset($_POST['user_crypt'])) {
    $cifrar = new Cipher('clave secreta');
    $user = $cifrar->decrypt($_POST['user_crypt']);
    // falta comprobar si user es valido
    $aut->setAuth($user);
}
```

Autenticación externa: 2. GET/POST encriptado

- Inconvenientes:
  - El algoritmo de cifrado ha de estar disponible tanto para la aplicación gvHIDRA como para el servicio externo
  - La clave de encriptación debe ser secreta, por lo que hay que establecer una política adecuada de seguridad (custodia, fortaleza, cambios periódicos,...)
  - El personal informático de una aplicación puede acceder a todas las aplicaciones que usen este sistema, lo cual puede ser un problema para algunas aplicaciones

Autenticación externa: 3. GET/POST con clave pública/privada

- El servidor externo encripta los parámetros usando la clave pública de la aplicación gvHIDRA (paso 1)
- La aplicación desencripta los parámetros con su clave privada e inicia sesión con el usuario recibido (paso 2)
- Es la opción más segura en autenticación externa
- También conviene firmar la solicitud externa, para confirmar la procedencia

Autenticación externa: 3. GET/POST con clave pública/privada

- Paso 1: el servidor externo encripta:

```
$public_key = file_get_contents('apl.crt');
$token = $_REQUEST['user'].'::'.time();
echo 'conectamos con parametros '.$token;
openssl_public_encrypt($token,$cifrar,$public_key);
$encoded = base64_encode($cifrar);
echo <<<cadena
<form action="login.php" method="post">
 <input type="hidden" name="user_crypt" value="$encoded">
 <input type="submit" name="Entrar" value=" Confirmar ">
</form>
cadena;
```

Autenticación externa: 3. GET/POST con clave pública/privada

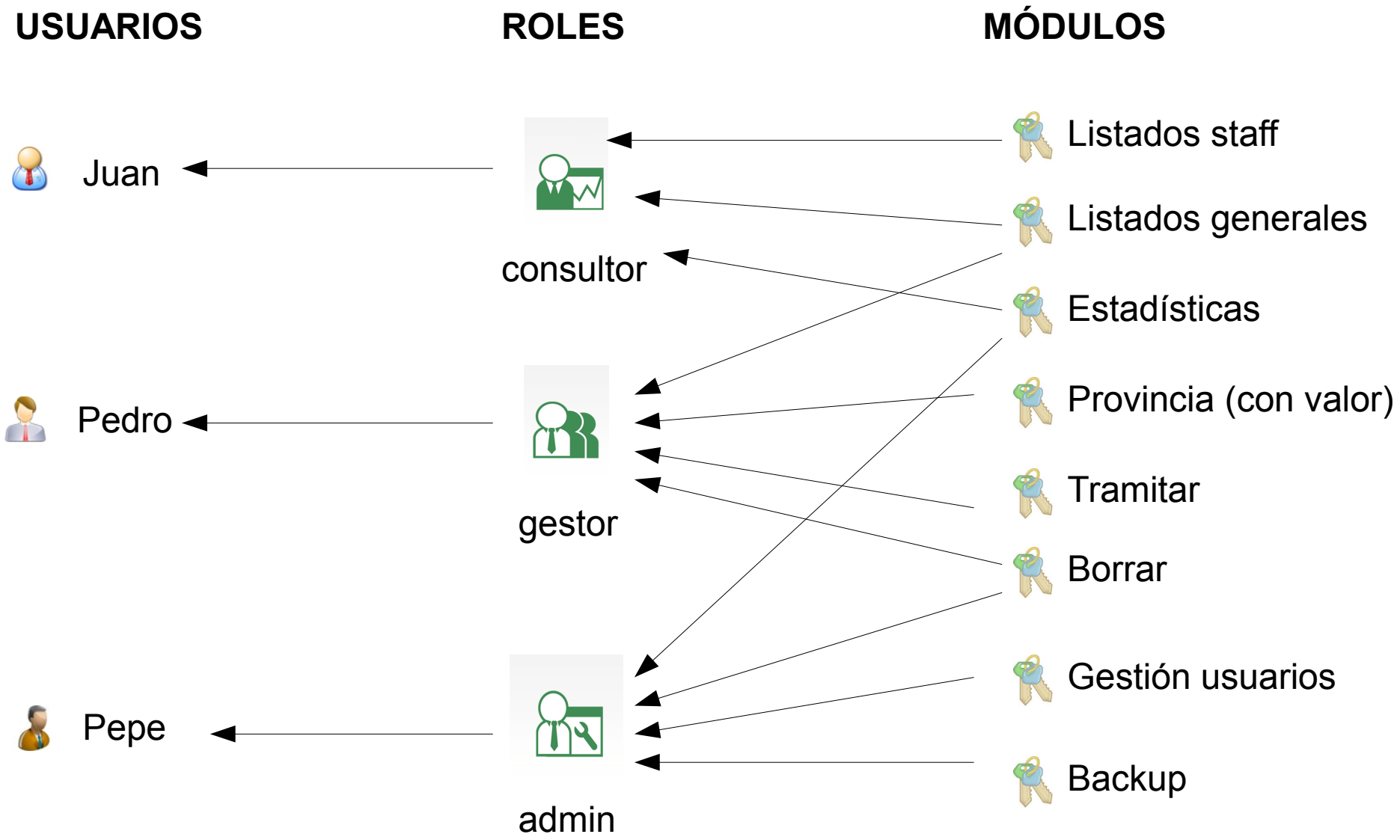
- Paso 2: la aplicación desencripta parámetros:

```
if (isset($_POST['user_crypt'])) {
   $private_key = file_get_contents('apl.key');
   $user_crypt = base64_decode($_POST['user_crypt']);
   if (!openssl_private_decrypt($user_crypt, $decrypted, $private_key))
      return 'Error en parámetros de autenticación';
   $tokens = explode('::', $decrypted);
   if ($tokens[1] + 300 < time())
      return 'Error en parámetros de autenticación por caducidad del token';
   // falta comprobar si usuario es valido
   $aut->setAuth($tokens[0]);
}
```

Autorización: Módulos y Roles

- Usados para asignar a usuarios permisos sobre recursos o funcionalidades
- **Módulos**: representan a los recursos, y pueden asignarse muchos por usuario
- Cada módulo tiene un código, una descripción y opcionalmente puede tener un valor
- **Roles**: representan los perfiles de usuarios y pueden incluir muchos módulos. Cada usuario sólo tiene un role

Autorización: Módulos y Roles



Autorización: Módulos y Roles

- Se inicializan la primera vez que se accede a la aplicación, en método postLogin:
- Estructura a seguir en la sesión:

$_SESSION[<apl>]['modulos']

$_SESSION[<apl>]['modulos'][<modulo x>]['valor']

$_SESSION[<apl>]['modulos'][<modulo x>]['descrip']

...

$_SESSION[<apl>]['rolusuar']

Autorización: Módulos y Roles

- Consulta de módulos:
  - IgepSession::hayModulo(string) return boolean
  - IgepSession::dameModulo(string) return array
  - IgepSession::dameModulos() return array de arrays
- Módulos dinámicos: pueden cambiar durante la ejecución de la aplicación usando métodos. Métodos en IgepSession: anyadeModuloValor, quitaModuloValor, …

  !!! el menú sólo se actualiza al pasar por ventana inicial !!!

Autorización: Módulos y Roles

- Módulos en menús:

```
<controlAcceso>
    <moduloPermitido id="Tramitar"/>
</controlAcceso>

<controlAcceso>
   <moduloPermitido id="Provincia">
            <valorModulo valor="Valencia" />
            <valorModulo valor="Alicante" />
   </moduloPermitido>
</controlAcceso>
```

Autorización: Módulos y Roles

- Consulta de role:
  - IgepSession::dameRol() return string
- Roles en menús:

```
<controlAcceso>
   <rolPermitido valor="consultor"/>
   <rolPermitido valor="gestor"/>
</controlAcceso>
```

Autorización: Módulos y Roles

- Recomendaciones:
  - Se pueden asignar módulos a usuarios aunque mejor hacerlo a través de roles
  - Si en la aplicación sólo autorizamos usando módulos, los cambios de permisos no afectan al código fuente de la aplicación, sólo a la asignación de módulos a roles (o usuarios)

Otras funcionalidades Auth

- Log: acceso a información interna de log
- Métodos callback: login, logout, failed login, check login
- setExpire: fija tiempo para expiración
- setIdle: fija tiempo máximo entre dos acciones
- Seguridad avanzada (detecta cambios de ip, cookie para la siguiente petición, ...)
- ...

Documentación

- gvHIDRA: http://www.gvhidra.org/
- PEAR Auth: http://pear.php.net/package/Auth
- Apress - Pro PHP Security 2nd Edition - Dic 2010: conceptos generales de seguridad
- Apress - Pro PHP Security – 2005: implementación de un single sign on
- Marco Tabini - phparchitect's Guide to PHP Security - Sep 2005: conceptos generales de seguridad
- Addison Wesley - Securing PHP Web Applications - Dic 2008
- Apress - Pro PHP Patterns Frameworks Testing and More - Mar 2008: Autenticación con certificados

Logout

Gracias

Comentarios a:

gquiles@dsic.upv.es
gvhidra_soporte@listserv@gva.es