Archived:Token authentication in WidSets

Archived: This article is archived because it is not considered relevant for third-party developers creating commercial solutions today. If you think this article is still relevant, let us know by adding the template {{ReviewForRemovalFromArchive|user=~~~~|*write your reason here*}}.

The article is believed to be still valid for the original topic scope.

Authentication

Partner sites that require users to authenticate before providing user-specific information can use a shared secret based authentication where there is a "SHARED_SECRET" between WidSets and the Partner. The benefit of using a shared secret is that the Partner-specific user information of the user is not stored at WidSets in any point. An identifier for the user (TOKEN) is generated during the authentication process.

The secret is typically a randomly generated text string or MD5 hash. The SHARED_SECRET is delivered to the Partner using PGP encrypted email.

Every user must have a user-specific TOKEN to identify the user. The TOKEN can be in the same format as the SHARED_SECRET, but it can also be any identifier for a certain user of the partner site. A token is generated when the user accepts to use the Partner service, or more specifically, allows the WidSets server to act on behalf of the user towards the Partner service.

The user is directed to a special URL at the Partner site to accept the usage of these services. The Partner may want to explicitly draw the user's attention and receive an approval for the usage from the user. After the usage has been approved, the user is redirected to the WidSets site and the TOKEN and other Partner specific parameters will be included in HTTP request parameters. Other potential parameters include a user id and selections of the Partner widgets the user will receive.

Partner receiving an authenticated call

The WidSets server will start to make authenticated calls to the Partner service for content. An authenticated HTTP request call is made by including the necessary parameters in the request and three additional ones: a random seed, identifier for the user, that is, TOKEN and a hash signature. The signature consists of a MD5 hash of the concatenation of all the parameter values supplied, and a secret SHARED_SECRET that is not visible in plaintext.

For instance a personal/private comment feed could have the following parameters:

```
action=comments
maxcount=20
token=5F5132173341A8CFD1CA67EF0B90D843
seed=1205325181324
sig=md5('comments' + '20' + '5F5132173341A8CFD1CA67EF0B90D843' + '1205325181324' + SHARED_SECRET)
```

Parameters explained

- **action** specifies the type of the call.
- **maxcount** is the desired maximum number of items returned in the feed.
- **token** specifies the unique Partner site user id of the user.
- **seed** is a random seed. For example, a timestamp that is unique for every call for one user
- **sig** is a signature consisting of a hash of the parameter values concatenated in the order they were given and SHARED_SECRET between the parties.

It is possible to have any amount of parameters in a call as long as the signature is correctly formulated.

Example case of using TOKEN authentication

The following is an example use case for TOKEN authentication with an imaginary partner.

Partner specific information at WidSets

This information is stored on WidSets and negotiated beforehand.

```
name            = "Token authentication test"
shared_secret   = "aaaabbbbccccddddeeeeffff00001111"
api_url         = "http://testpartner.com/misc/widsets_token/api.php"
auth_url        = "http://testpartner.com/misc/widsets_token/auth.php"
```

Note:
Other info can be also added.

Partner widget setup

(C) Copyright Nokia 2013. All rights reserved.

In this example the widget gets data from the partner site through the HTTP widget service: (id=docall, type=http). This service is linked to the token authentication scheme (id=testpartner, type=lift).

As the parameters section has an *auth*-parameter (name=authparam, type=auth), the widget options page on the WidSets website presents a link to the user with the description: "Please login with this widget" or similar. This link leads the user to the Partner authentication URL specified earlier that is stored with partner specific information at WidSets. The URL in this case would be:

```
http://testpartner.com/widsets_auth?url=URLENCODE(callbackUrl)&referer=widsets
```

and

```
callbackUrl=<A callback URL where to return to>
```

At this location at the Partner website there must be a dialog for the user to authenticate using the Partner account information.

After the user authenticates him/herself at the Partner website he/she is directed back to the WidSets website through the callback URL. The returning request contains the user's specific TOKEN. The user is returned back to the widget's options page and in place of the authentication link there is a message "Authentication OK (link=disable here)" or similar.

Every time the user's authenticated widget communicates with the Partner using the widget service 'docall', the call is automatically signed as described before. The Partner can now authenticate the call by calculating the signature separately and comparing it to the one supplied.

A Widget using a token authenticated HTTP widget service

Widget definition stored in widget.xml (see Archived:Widget Configuration 2.1 for more details on widget.xml):

```
<widget spec_version="2.1">

  <info>
    <name>token auth test</name>
    <version>2.0</version>
    <author>anttihei</author>
    <shortdescription>Tests http token auth for</shortdescription>
  </info>

  <services>

    <service type="http" version="1" id="docall">
      <authentication type="lift" id="testpartner"/>
    </service>

  </services>

  <parameters>

    <parameter type="auth"
               name="authparam"
               desc="the value must be the id of the auth plugin specified in one or more services"
               value="lift/testpartner" />
    <parameter name="token"/>
    <parameter name="useapiurl" value="false"/>

    <parameter type="string"
               name="widgetname"
               description="Widget Name"
               editable="false">My Testpartner Widget</parameter>

  </parameters>

  <resources>
    <img scale="true" src="bkg.png" />
    <img scale="true" src="logo.png" />
    <code src="TestToken.he"/>
    <stylesheet>

     mini {
       align: hcenter vcenter;
       background: image "bkg.png" transparent left top repeat-x repeat-y;
     }

      bkg {
       align: hcenter vcenter;
      }

    </stylesheet>
  </resources>

  <layout minimizedheight="65sp">
    <view id="miniView">
      <img src="logo.png" class="mini" top="0px" right="100%" bottom="100%" left="0px" />
    </view>
    <view id="menuView" class="mini">
      <script id="message" class="bkg" top="0sp" right="100%" bottom="100%" left="0sp" />
    </view>
 </layout>

</widget>
```

TestToken.he code file. The code runs in the widget and makes a HTTP request toward the partner API using the HTTP widgetservice (service id=docall). This service has been configured to use the token authentication scheme.

```
class TokenTest {

  // Setup the URL to your "api.php"
  String url = "";
```


(C) Copyright Nokia 2013. All rights reserved.

```
  void startWidget()
  {
    setMinimizedView(createMinimizedView("miniView", null));
  }


  Shell openWidget()
  {
    setBubble(null, "Making auth call");
    Value arg = [
      "url" => url,
      "params" => ["testparam1" => "yeah", "testparam2" => "oujee", "testparam54" => "testing"]
    ];
    call(new Object(), "docall", "get", arg, onSuccess, onFailure);
    return null;
  }


  void onFailure(Object state, String ret)
  {
    printf("error: "+ret);
    setBubble(null, "Failed: "+String(ret));
  }


  void onSuccess(Object state, Value ret)
  {
    printf("success: "+String(ret));
    setBubble(null, "Success:"+String(ret));
  }


}
```

Here is another example widget that uses a token authenticated syndication-widgetservice: token_widget_rss.zip

API call error messages

Errors in API-calls are reported by a HTTP status code **403 Forbidden**. The implementor of the partner-side API can choose whether or not to supply a describing error message.

For example:

```
403 Bad signature
```

or

```
403 User not found
```

The normal error case is that the user has not been authenticated at the Partner site. When an error code is received, the developer of the widget should give instructions how to authenticate the widget.

Simple test/example partner side implementation

The implementation consists of the authentication part (*auth.php*) and the content fetching part (*api.php*) and a utility (*namevaluedb.php*) for storing user=token relations.

Note. This implementation is not considered as secure or efficient. It is only for test purposes.

auth.php

User=Token relations are stored in the file *testpartner.db*. The login to the partner site is simulated with an input box for a username and a button labeled 'Login'.

```
<?php

  include 'namevaluedb.php';

  $db = new NameValueDB("testpartner.db");

  $username = $_GET["userid"];
  $url = $_GET["url"];

  if (isset($_GET["userid"]) && isset($_GET["url"])) {

    $set = "0123456789ABCDEF";
    $token = "";
    for ($i=0;$i<32;$i++) {
      $token = $token.$set[rand(0,15)];
    }

    $db->update($username, $token);
    echo "Token updated for $username.<br>\n";
    $fw = fopen("auth.log","a");
    fwrite($fw, "Token added: ".$token." userid: ".$username." date: ".date("M j G:i:s")."\n");
    fclose($fw);

    $returl = $url;
    if (!ereg('\?', $returl)) {
      $returl = $returl . '?';
    }
```


(C) Copyright Nokia 2013. All rights reserved.

```
    $returl = $returl . "token=" . $token;
    echo "CLICK TO CONTINUE: <a href=". $returl . ">" . $returl . "</a><br>\n";

  } else {
    echo "<form method='get' action='$PHP_SELF' enctype='multipart/form-data'>
          Username:<input type='text' name='userid' /><br>
          <input type='hidden' name='url' value=$url />
          <input type='submit' value='Login'><br></form>";

  }

?>
```

api.php

The API uses the same relation file *testpartner.db* for looking up users with tokens provided in the calls for content. Additionally, it stores all requests coming into the file *repeat.db*. By doing so, it can prevent a repeat attack if somebody has been able to intercept the HTTP request. Every HTTP request coming to the API is, in theory, different if the random seed and token combination is globally unique.

The API returns a simple success message if everything is all right, or an RSS feed if the request parameter "rss=true" is used. Errors are reported with the HTTP status code 403.

```
<?php

  include 'namevaluedb.php';

  $db = new NameValueDb("testpartner.db");
  $repeatdb = new NameValueDb("repeat.db");

  $token = $_GET["token"];
  $sig = $_GET["sig"];
  $rss = $_GET["rss"];

  $ss = 'aaaabbbbccccddddeeeeffff00001111';
  $plaintext = "";

  $date = date("M j G:i:s");
  $qs = $_SERVER['QUERY_STRING'];

  $fl = fopen("api.log","a");
  fwrite($fl, "call: ".$qs."\n");
  fclose($fl);

  if ($repeatdb->get($qs)) {

    httpError("Reuse of request not allowed");
    exit;

  }

  $result = $db->getByRight($token);

  if (!$result) {
    httpError("User not found");
    exit;

  } else {

    list($userid, $token) = explode("|", $result);

    foreach($_GET as $key => $value) {
      if ($key != "sig") {
        $plaintext = $plaintext.$value;
      }
    }

    $plaintext = $plaintext.$ss;
    $calculated_sig = md5($plaintext);

    $fl = fopen("api.log","a");
    fwrite($fl, "date: ".$date.
                ", userid: ".$userid.
                ",calc signature: ".$calculated_sig.
                ", plaintext: ".$plaintext.
                ", sig: ".$sig."\n\n");
    fclose($fl);

    if ($calculated_sig == $sig) {

      $repeatdb->update($qs, $date);

      if ($rss) {

        header('Content-type: text/xml; charset=utf-8');
        $doc = new DOMDocument('1.0', 'utf-8');
        $root = $doc->createElement('rss');
        $root->setAttribute('version', '2.0');
        $doc ->appendChild($root);
        $channel = $doc->createElement('channel');
        $root->appendChild($channel);
        $item = $doc->createElement('item');

        $channel->appendChild($item);
        $title = $doc->createElement('title');
        $item->appendChild($title);
        $titleValue = $doc->createCDATASection("success for user $userid");
        $title->appendChild($titleValue);

        $desc = $doc->createElement('description');
        $item->appendChild($desc);
        $descValue = $doc->createCDATASection($date);
        $desc->appendChild($descValue);

        echo $doc->saveXML();

      } else {
        echo "success for user $userid at $date";

      }

    } else {
      httpError("Bad signature");
```


(C) Copyright Nokia 2013. All rights reserved.

```
    exit;

  }

}


function httpError($msg)
{
  header('HTTP/1.0 403 $msg');
  echo "403 $msg";
  $fl = fopen("api.log","a");
  fwrite($fl, "User not found $msg\n");
  fclose($fl);
}


?>
```

namevaluedb.php

Stores name-value pairs into a text file.

```
<?php

class NameValueDB
{

  private $dbfilename;


  public function __construct($fn)
  {
    $this->dbfilename = $fn;
    touch($this->dbfilename);
  }


  public function get($name)
  {
    return $this->getEntry($name, true);
  }


  public function getByRight($right)
  {
    return $this->getEntry($right, false);
  }


  public function update($name, $value)
  {
    $lines = $this->getDbLines();
    $s = count($lines);
    $val = "$name|$value";

    for ($i=0; $i<$s; $i++) {
      list($n, $t) = explode("|", $lines[$i]);
      if ($n == $name) {
        $lines[$i] = $val;
        $this->writeArrayLocked($lines);
        return;
      }
    }

    $lines[] = $val;
    $this->writeArrayLocked($lines);
  }


  private function getEntry($target, $left)
  {
    global $dbfilename;

    $lines = $this->getDbLines();
    foreach($lines as $v) {
      list($n, $t) = explode("|", $v);
      $c = $left ? $n : $t;
      if ($c == $target) {
        return $v;
      }
    }
    return null;
  }


  private function getDbLines()
  {
    $data = $this->readLocked();
    $lines = explode("\n", trim($data,"\n"));
    return $lines;
  }


  private function readLocked()
  {
    $fp = fopen("$this->dbfilename", "r");
    $this->waitFree($fp);
    $size = filesize($this->dbfilename);
    $data = "";
    if ($size > 0) {
      $data = fread($fp, $size);
    }
    flock($fp, LOCK_UN);
    fclose($fp);
    return $data;
  }


  private function writeArrayLocked($arr)
  {
    $fp = fopen($this->dbfilename, "w+");
    $this->waitFree($fp);
    foreach($arr as $key => $val) {
      fwrite($fp, "$val\n");
```


(C) Copyright Nokia 2013. All rights reserved.

```
    }
    flock($fp, LOCK_UN);
    fclose($fp);
  }


  private function waitFree($fp)
  {
    while (!flock($fp, LOCK_EX)) {
      usleep(50000);
    }
  }


}

?>
```

Summary

- WidSets and the partner make a contract and agree on the SHARED_SECRET.
- A partner widget is uploaded to the WidSets server.
  - The widget is configured to use the SHARED_SECRET agreed in the first step.
  - The options page displays a special link for authentication.
  - The widget is not usable yet. It needs to be authenticated by the user.

Activation of a widget

- In the widget options page, click the Activate your widget here link.

  The link leads to a Partner site login page that states that the user needs to input his/her authentication information
  – OR –
  The user is automatically authenticated though Web browser session (cookie).
  The authentication procedure depends on whether or not the Partner wants to draw special attention to the authentication event that in practice gives WidSets access to the Partner API using the user credentials.
  If a login is displayed, it should state that by entering a username and password the user is giving WidSets permission to possibly retrieve personal data from the Partner to be sent into the user's mobile.
- After successful authentication the HTTP request is directed back to the WidSets server with credentials. The HTTP request back to WidSets contains the following data:
  - TOKEN to be used in signing calls, and
  - The widget options will indicate that the widget has been authenticated/activated.

  The widget in question can now access the user's personal data on the partner site by using token authenticated widget services.

See also

- Archived:Integrating WidSets with Web Sites
  - Archived:External linking in WidSets
  - **Archived:Token authentication in WidSets**

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.