Cookie Based SSO

Cookie based SSO in UserVoice requires the creation of an encrypted cookie on the client server. The cookie should be created for an pages that implement the UserVoice feedback widget or that have a link leading to the UserVoice forum.

For this to work domain aliasing will have to of been setup for your forum in order that the cookies set can be read by UserVoice.

The system used to encrypt the data in the cookie is a 2 stage process:

1. Encrypt the data (a JSON object) using AES (with your subdomain key as the password and your api_key as the salt)
2. Escape that string to make it websafe

N.B The domain for the cookie must be set to be cross subdomain. i.e for the domain example.com, with a uservoice forum at feedback.example.com the cookie must have a domain of .example.com

The JSON object inside the cookie can have the following attributes:

- expires - a timestamp indicating the period this cookie is valid for
- username - this user's username
- email - email address for this user
- url - an url to view this user
- avatar_url - the url for this users avatar
- profile_url - an url to this user's profile
- display_name - a display name
- guid - a unique identifier

The user will either be created or logged in depending on the username supplied. The url, avatar_url and profile_url attributes only have an effect if the subscription level includes 'Profile Integration'

Example

At a general level the process is as follows:

```
function create_sso_cookie {
   password = my_subdomain_key
   salt = my_api_key

   // Grab an encryption key using a password and salt
   aes_key = AESLibrary::create_key_using_password( password, salt )

   // Use the key to encrypt our JSON object
   data = aes_key.encrypt(my_json_object)

   // Escape the data to be websafe
   escaped_data = CGI::escape(data)

   // Create the cookie named _uservoice_sso, making sure the domain is correct
   new_cookie(data: escaped_data, name: '_uservoice_sso', domain: '.my_domain.com')
}
```

Rails Example

Using Ruby on Rails and the EZCrypto library the cookie can be implemented as follows:

in application.rb:

```
  def create_sso_cookie
    key = 'example'
    api_key = '49c54a3f7feeab5b91ceb4b8f70d2834'

    key = EzCrypto::Key.with_password key, api_key
    encrypted = key.encrypt({
      :expires => (Time.now + 5.minutes).to_s(:db),
      :username => 'example',
      :email => 'example@external.com',
      :url => "http://external.com/users/1",
      :avatar_url => "http://external.com/users/1.png",
      :display_name => "External User",
      :guid => "EXT001"
    }.to_json)
    value = CGI.escape(encrypted)
    cookies['_uservoice_sso'] = { :domain => '.external.com', :value => value, :expires =>
Time.now + 30.minutes }
  end
```

in environment.rb

```
ActionController::CgiRequest::DEFAULT_SESSION_OPTIONS[:session_domain] =
'.external.com'
```

You can then call the create_sso_cookie method from any controller action, or in a before_filter.

PHP Example (Using the full library from http://www.phpaes.com $9.95)

```
<?php
include("./AES.class.php");

$password = "example";
$salt = "49c54a3f7feeab5b91ceb4b8f70d2834";
$salted = $salt . $password;
$hash = hash('sha1',$salted,true);
$saltedHash = substr($hash,0,16);
$iv = "OpenSSL for Ruby";
$aes = new AES($saltedHash, 'CBC', $iv);

$data = array(
    "expires" => "2009-01-15 10:43:22",
    "username" => "example",
    "email" => "example@external.com",
    "url" => "http://external.com/users/1",
    "avatar_url" => "http://external.com/users/1.png",
    "display_name" => "External User",
```

```
   "guid" => "EXT001"
);
$data = json_encode($data);

// double XOR first block
for ($i = 0; $i < 16; $i++)
{
   $data[$i] = $data[$i] ^ $iv[$i];
}

$pad = 16 - (strlen($data) % 16);
$data = $data . str_repeat(chr($pad), $pad);

$encryptedData = $aes->encrypt($data);
$encryptedData = urlencode($encryptedData);

// setcookie will do a urlencode also
setcookie("_uservoice_sso",$encryptedData,time() + 60*5,"/",".ciety.net");
header("Location: http://feedback.external.com");
?>
```

PHP Example (Using mcrypt)

```
<?php
$password = "example";
$salt = "49c54a3f7feeab5b91ceb4b8f70d2834";
$salted = $salt . $password;
$hash = hash('sha1',$salted,true);
$saltedHash = substr($hash,0,16);
$iv = "OpenSSL for Ruby";

$data = array(
   "expires" => "2009-01-15 10:43:22",
   "username" => "example",
   "email" => "example@external.com",
   "url" => "http://external.com/users/1",
   "avatar_url" => "http://external.com/users/1.png",
   "display_name" => "External User",
   "guid" => "EXT001"
);
$data = json_encode($data);

// double XOR first block
for ($i = 0; $i < 16; $i++)
{
   $data[$i] = $data[$i] ^ $iv[$i];
}

$pad = 16 - (strlen($data) % 16);
$data = $data . str_repeat(chr($pad), $pad);

$cipher = mcrypt_module_open(MCRYPT_RIJNDAEL_128,'','cbc','');
mcrypt_generic_init($cipher, $saltedHash, $iv);
$encryptedData = mcrypt_generic($cipher,$data);
```

```
mcrypt_generic_deinit($cipher);

$encryptedData = urlencode($encryptedData);

// setcookie will do a urlencode also
setcookie("_uservoice_sso",$encryptedData,time() + 60*5,"/",".ciety.net");
header("Location: http://feedback.external.com");
?>
```

C# Example (ASP.NET)

```
string passPhrase = "external";        // your account key
string saltValue = "12322asd10bd4e6050142134c463fc53";        // your api key
string initVector = "OpenSSL for Ruby"; // must be 16 bytes DO NOT CHANGE!!!!!!

string original = "Here is some data to encrypt!"; // REPLACE WITH JSON ENCODED USER
DATA
byte[] initVectorBytes = Encoding.UTF8.GetBytes(initVector);

SHA1 sha = new SHA1CryptoServiceProvider();
byte[] keyBytesLong = sha.ComputeHash(Encoding.UTF8.GetBytes(saltValue +
passPhrase));
byte[] keyBytes = new byte[16];
Array.Copy(keyBytesLong, keyBytes, 16);

byte[] textBytes = Encoding.UTF8.GetBytes(original);
for (int i = 0; i < 16; i++) {
    textBytes[i] ^= initVectorBytes[i];
}

// Declare the stream used to encrypt to an in memory
// array of bytes.
MemoryStream msEncrypt = null;

// Declare the RijndaelManaged object
// used to encrypt the data.
RijndaelManaged aesAlg = null;

try
{
   // Create a RijndaelManaged object
   // with the specified key and IV.
   aesAlg = new RijndaelManaged();
   aesAlg.Mode = CipherMode.CBC;
   aesAlg.Padding = PaddingMode.PKCS7;
   aesAlg.KeySize = 128;
   aesAlg.BlockSize = 128;
   aesAlg.Key = keyBytes;
   aesAlg.IV = initVectorBytes;

   // Create an encrytor to perform the stream transform.
   ICryptoTransform encryptor = aesAlg.CreateEncryptor();

   // Create the streams used for encryption.
```

```
   msEncrypt = new MemoryStream();
   CryptoStream csEncrypt = new CryptoStream(msEncrypt, encryptor,
CryptoStreamMode.Write);

   csEncrypt.Write(textBytes, 0, textBytes.Length);
   csEncrypt.FlushFinalBlock();
}
finally
{
    // Clear the RijndaelManaged object.
    if (aesAlg != null)
        aesAlg.Clear();
    }
}
byte[] encrypted = msEncrypt.ToArray();
string urlEncoded = HttpUtility.UrlEncode(encrypted);
```