Gestión de la Información en la Web
Curso 2016-17
Práctica Autenticación & TOTP

Fecha de entrega: miércoles 18 de enero de 2017, 13:55h

Entrega de la práctica
La entrega de la práctica se realizará a través del Campus Virtual de la asignatura mediante un fichero **grupoXX.zip** donde **XX** es el número de grupo. Este ZIP constará de un fichero `autenticacion.py` con el código del servidor web, cuyo esqueleto se puede descargar del Campus Virtual. Además del servidor web, el fichero ZIP contendrá las vistas/plantillas necesarias para mostrar los datos adecuadamente (si las habéis utilizado).

Lenguaje de programación
Python 2.7 o **3.5**.

Calificación
El apartado 1 aporta el 40 % de la nota, y el apartado 2 aporta el 60 % restante.

Declaración de autoría e integridad
Todos los ficheros entregados contendrán una cabecera en la que se indique la asignatura, la práctica, el grupo y los autores. Esta cabecera también contendrá la siguiente declaración de integridad:

> *(Nombres completos de los autores)* declaramos que esta solución es fruto exclusivamente de nuestro trabajo personal. No hemos sido ayudados por ninguna otra persona ni hemos obtenido la solución de fuentes externas, y tampoco hemos compartido nuestra solución con nadie. Declaramos además que no hemos realizado de manera deshonesta ninguna otra actividad que pueda mejorar nuestros resultados ni perjudicar los resultados de los demás.

No se corregirá ningún fichero que no venga acompañado de dicha cabecera.

En esta práctica implementaremos distintas peticiones de gestión de usuarios (creación, acceso y cambio de contraseña) teniendo como prioridad **almacenar de la forma más segura las contraseñas** de los usuarios. De esta manera las contraseñas de nuestros usuarios estarán protegidas aunque un atacante acceda a nuestra base de datos.

Todos los datos de los usuarios se almacenarán en MongoDB, utilizando para ello la librería `pymongo` como en prácticas anteriores. En esta ocasión tenéis libertad para escoger el esquema que prefiráis, pero cada usuario se deberá almacenar como un único documento en la colección `users` de la base de datos `giw` **(es imprescindible respetar el nombre de la base de datos y de la colección)**.

Para realizar la práctica debéis descargar el esqueleto básico del servidor web desde el Campus Virtual y usarlo como base. Este esqueleto incluye 2 apartados con algunas funciones y distintas rutas en las que el servidor web debe responder a peticiones POST (no podéis cambiar las rutas, el método HTTP ni añadir nuevas rutas).

1. Autenticación básica mediante contraseñas [4pt]

El fichero `autenticacion.py` debe contener un comentario al inicio de este apartado en el que se **describa y explique detalladamente** el mecanismo escogido para el almacenamiento de contraseñas y se explique razonadamente por qué es seguro.

1. `/signup`
   Esta petición sirve para dar de alta a un usuario. Recibe los siguientes parámetros POST:

   - `nickname`: alias del usuario (único).
   - `name`: nombre completo del usuario (incluye apellidos).
   - `country`: país de residencia del usuario.
   - `email`: correo electrónico del usuario.
   - `password`: contraseña escogida por el usuario.
   - `password2`: contraseña repetida para comprobar que coincide.

   En respuesta a esta petición el servidor web hará lo siguiente:

   - Si las 2 contraseñas no coinciden no realizará ninguna modificación en la base de datos y devolverá una página web con el mensaje `"Las contraseñas no coinciden"`.
   - Si el alias de usuario ya existe en nuestra colección no realizará ninguna modificación en la base de datos y devolverá una página web con el mensaje `"El alias de usuario ya existe"`.
   - En otro caso insertará el usuario en la colección users y devolverá una página web con el mensaje `"Bienvenido usuario <name>"`, donde `<name>` es el nombre completo del usuario.

2. `/change_password`
   Cambia la contraseña de un usuario. Recibe como parámetros POST:

- `nickname`: alias del usuario.
- `old_password`: contraseña antigua.
- `new_password`: contraseña nueva.

En respuesta a esta petición el servidor hará lo siguiente:

- Si el alias del usuario no existe en nuestra base, o si `old_password` no coincide con la contraseña almacenada entonces devolverá una página web con el mensaje `"Usuario o contraseña incorrectos"`.
- En otro caso actualizará la contraseña en la base de datos y devolverá una página web con el mensaje `La contraseña del usuario <nickname> ha sido modificada`.

3. `/login`
   Autentica un usuario. Recibe como parámetros POST:

   - `nickname`: alias del usuario.
   - `password`: contraseña.

   En respuesta a esta petición el servidor hará lo siguiente:

   - Si el alias del usuario no existe en nuestra base de datos, o si `password` no coincide con la contraseña almacenada entonces devolverá una página web con el mensaje `"Usuario o contraseña incorrectos"`.
   - En otro caso devolverá una página web con el mensaje `Bienvenido <name>` donde `<name>` es el nombre completo del usuario.

2. Autenticación con TOTP

Muchas personas repiten su nombre de usuario y contraseña en múltiples aplicaciones web, por lo que su revelación puede comprometer la seguridad de todas las aplicaciones web. Lamentablemente esta situación no es tan inusual y varias veces al año se producen ataques a grandes webs que terminan con la publicación de inmensos listados de usuarios y contraseñas (`https://haveibeenpwned.com`, `https://hesidohackeado.com`). En este apartado vamos a incorporar un segundo factor de autenticación al servidor web mediante TOTP con Google Authenticator[1].

2.1. Alta de usuarios

Para integrar TOTPs en nuestro servidor web tendremos que modificar ligeramente el proceso de alta de un usuario:

1. Generar una semilla, que debe ser una cadena de 16 caracteres tomados aleatoriamente entre las 26 letras mayúsculas del inglés y los dígitos 2, 3, 4, 5, 6 y 7.

[1] También se puede usar *Latch*, *FreeOTP Authenticator*, `http://gauth.apps.gbraad.nl` o la librería `onetimepass`.

2. Almacenar la semilla en el documento del usuario dentro la base de datos para que el servidor web pueda generar el código temporal actual y compararlo con el que proporciona el usuario.

3. Comunicar la semilla al usuario para que añada una nueva cuenta a Google Authenticator. La app soporta dos maneras de añadir una nueva cuenta:

   - Manualmente a partir de un nombre de cuenta y la semilla. Este proceso es proclive a fallos ya que hay que introducir manualmente la cadena de 16 caracteres.
   - A partir de una URL convenientemente formada que Google Authenticator analiza y de la que extrae la información necesaria para añadir la cuenta. El formato de esta URL, que se explica en `https://github.com/google/google-authenticator/wiki/Key-Uri-Format`, es el siguiente:

     `otpauth://totp/<USERNAME>?secret=<SECRET>&issuer=<APP_NAME>`

     donde `<USERNAME>` es el nombre del usuario, `<SECRET>` la cadena de 16 caracteres generada automáticamente que sirve como semilla y `<APP_NAME>` el nombre del servidor web tal y como lo mostrará Google Authenticator. Un ejemplo de URL sería:

     `otpauth://totp/pepe_lopez?secret=JBSWY3DPEHPK3PXP&issuer=GIW_grupo89`

     Como esta URL tiene que ser procesada por la app del móvil, la opción más cómoda es generar un código QR (`https://es.wikipedia.org/wiki/C%C3%B3digo_QR`) que codifique dicha URL. De esta manera el usuario solo tendrá que escanear el código en su móvil y la cuenta se añadirá de manera automática a Google Authenticator.

2.2. Funciones y rutas a definir

Para este apartado de la práctica será necesario implementar las siguientes funciones y rutas del servidor web:

1. `def gen_secret()`
   Genera una cadena aleatoria de 16 caracteres a escoger entre las 26 letras mayúsculas del inglés y los dígitos 2, 3, 4, 5, 6 y 7. Ejemplo:

```
>>> gen_secret()
'7ZVVBSKR22ATNU26'
```

2. `def gen_gauth_url(app_name, username, secret)`
   Genera la URL para insertar una cuenta a Google Authenticator a partir de sus fragmentos. Ejemplo:

```
>>> gen_gauth_url( 'GIW_grupoX', 'pepe_lopez', 'JBSWY3DPEHPK3PXP')
'otpauth://totp/pepe_lopez?secret=JBSWY3DPEHPK3PXP&issuer=GIW_grupoX'
```

3. `def gen_qrcode_url(gauth_url)`
   Para simplificar la generación del código QR a partir de la URL utilizaremos el servicio externo *goqr.me*[2]. Este servicio nos proporciona una API que genera imágenes con códigos QR a partir de los datos que queremos codificar. Por ejemplo para generar el código QR con contenido "GIW" realizaríamos una petición a `https://api.qrserver.com/v1/create-qr-code/?data=GIW`. Podéis encontrar más información sobre el resto de parámetros para la generación de códigos QR en `http://goqr.me/api/doc/create-qr-code/`.

   A partir de la URL de Google Authenticator, la función `gen_qrcode_url` genera la URL de la petición a *goqr.com* que generará el código QR. Por ejemplo:

```
>>> gen_qrcode_url('otpauth://totp/pepe_lopez?secret=
    JBSWY3DPEHPK3PXP&issuer=GIW_grupoX')
'https://api.qrserver.com/v1/create-qr-code/?data=otpauth%3A%2F%2
    Ftotp%2Fpepe_lopez%3Fsecret%3DJBSWY3DPEHPK3PXP%26issuer
    %3DGIW_grupoX'
```

   Nota: Es importante codificar los parámetros adecuadamente, ya que formarán parte de una URL.

4. `/signup_totp`
   Esta petición sirve para dar de alta a un usuario. Recibe los siguientes parámetros POST:

   - `nickname`: alias del usuario (único).
   - `name`: nombre completo del usuario (incluye apellidos).
   - `country`: país de residencia del usuario.
   - `email`: correo electrónico del usuario.
   - `password`: contraseña escogida por el usuario.
   - `password2`: contraseña repetida para comprobar que coincide.

   En respuesta a esta petición el servidor web hará lo siguiente:

   - Si las 2 contraseñas no coinciden no realizará ninguna modificación en la base de datos y devolverá una página web con el mensaje `Las contraseñas no coinciden`.
   - Si el alias de usuario ya existe en nuestra colección no realizará ninguna modificación en la base de datos y devolverá una página web con el mensaje `El alias de usuario ya existe`.
   - En otro caso insertará al usuario en la colección `users` y devolverá una página web con el **código QR** para configurar Google Authenticator. Esta página web contendrá también el **nombre de usuario** y la **semilla** generada por si el usuario quiere configurar Google Authenticator manualmente o utilizar otra aplicación TOTP.

[2]En producción es preferible generar nosotros mismos los códigos QR y evitar compartir las semillas con terceros.

5. `/login_totp`
   Autentica un usuario utilizando dos factores. Recibe como parámetros POST:

   - `nickname`: alias del usuario.
   - `password`: contraseña.
   - `totp`: código TOTP generado por Google Authenticator o cualquier aplicación similar.

   En respuesta a esta petición el servidor hará lo siguiente:

   - Si el alias del usuario no existe en nuestra base de datos, la contraseña no coincide o el código TOTP no es válido devolverá una página web con el mensaje `Usuario o contraseña incorrectos`.
   - En otro caso devolverá una página web con el mensaje `Bienvenido <name>` donde `<name>` es el nombre completo del usuario.

   Para poder comprobar la validez del código temporal enviado por el usuario nuestro servidor web usará la biblioteca onetimepass (ver transparencias).