Gestión de la Información en la Web
Seguridad y autenticación delegada

<u>Fecha de entrega</u>: martes 2 de febrero de 2016, 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 numero de grupo. Este fichero constará **únicamente** de **dos** ficheros: **auditoria.pdf** y **autenticacion_delegada.py**. Ambos documentos comenzarán con un comentario indicando los autores, el grupo y una **declaración de integridad** expresando que el trabajo es fruto del trabajo de sus miembros. Debido a la sencillez del servidor web no está permitido utilizar plantillas.

Objetivos mínimos
Detectar vulnerabilidades de seguridad basadas en las entradas del usuario; saber cómo explotarlas y conocer las técnicas principales para mitigarlas.

Objetivos opcionales
Realizar la autenticación delegada utilizando la API de Google en un servidor web implementado en Python con *bottle*.

Calificación
Ambos apartados contribuyen el 50% de la nota.

Otras consideraciones
Los documentos entregados será resultado exclusivamente del trabajo de sus miembros. No se permite ningún tipo de colaboración entre grupos. Cualquier fragmento que aparezca repetido en distintas prácticas o copiado de alguna fuente externa implicará **<u>automáticamente</u>** una calificación de cero y se trasladará dicha situación a las autoridades académicas competentes.

Apartado A: Auditoría de seguridad web

En este apartado vamos a auditar la seguridad de una aplicación web muy sencilla desarrollada con el *framework bottle*. Esta aplicación permite a los usuarios añadir preguntas y contestarlas. Internamente utiliza SQLite para almacenar los datos. Por simplicidad esta aplicación no tiene ningún tipo de gestión de usuarios ni sesiones.

Lo primero que hay que hacer es descargar la aplicación llamada **El Coladero** desde el Campus Virtual y descomprimirla. Antes de ejecutar la aplicación es imprescindible crear una base de datos SQLite llamada **database.db** a partir del fichero **database.sql** (en el Campus Virtual os dejo también un *database.db* generado en Linux por si os sirve en Windows). Una vez creada la base de datos podéis arrancar el servidor y acceder a su página principal http://localhost:8080/show_all_questions para ver el funcionamiento de la aplicación, que únicamente produce contestación en 5 rutas:

```
1. @get('/show_all_questions')
2. @post('/insert_question')
3. @get('/show_question')
4. @post('/insert_reply')
```

```
5. @get('/search_question')
```

La auditoría de seguridad generará un fichero llamado **auditoria.pdf** donde *cada página tendrá una cabecera incluyendo el grupo y el número de página*. Este documento contendrá una sección por cada ruta en la que se explicará **con detalle** las vulnerabilidades detectadas en dicha ruta. Por cada vulnerabilidad detectada en cada ruta debe aparecer:

1. **Nombre** de la vulnerabilidad con el máximo detalle posible.
2. Explicación de qué **situaciones peligrosas** o no deseadas puede provocar en esta aplicación web concreta.
3. **Ejemplo paso a paso** de cómo explotar esta vulnerabilidad. Se deben incluir **capturas de pantalla** en la que se vea con detalle cada uno de los pasos.
4. Explicación detallada de qué **medidas** hay que incluir en la aplicación web para mitigar esta vulnerabilidad. Tened en cuenta el principio de *defensa en profundidad* por el cual es preferible implantar varias medidas de seguridad si es posible. En caso de recomendar la **validación** de las entradas del usuario, debéis incluir el **formato** esperado de cada parámetro **con toda la precisión posible** (p. ej: *cadena de letras minúsculas de longitud entre 5 y 10 caracteres*). ***No es necesario*** *escribir código concreto aunque se puede incluir para acompañar la explicación detallada.*

Apartado B: Autenticación delegada utilizando el API de Google

En este apartado vamos a utilizar el API de Google para autenticar usuarios. Únicamente vamos a implementar una prueba de concepto, por lo que los pasos a seguir serán:

1. Darse de alta en la Consola de Desarrolladores de Google y crear los credenciales de la aplicación web (https://console.developers.google.com).
2. A la hora de autenticar un usuario, redirigirlo a la web de Google con los parámetros adecuados.
3. Recibir la petición del usuario, obtener el código generado por Google y canjearlo por un *id_token*.
4. Extraer el e-mail del usuario del *id_token* y devolver una página de bienvenida.

Aunque existen distintas librerías con funciones que facilitan la autenticación delegada con distintas redes sociales, el objetivo de este apartado es realizar todo el proceso **de manera "artesana"**, sin utilizar APIs ni funciones externas (salvo para decodificar el JSON Web Token o JWT).

Seguiremos el esqueleto que podéis descargar del Campus Virtual, llamado **autenticacion_delegada.py**. Este fichero contiene un servidor web que responde en **dos rutas**:

1.- /login_google
Genera una página HTML con un enlace o un botón que redirige al usuario a la página de autenticación delegada de Google. Esta petición debe contener las credenciales necesarias de nuestra aplicación web.

2.- /token
Recibe la petición redirigida desde Google con el código temporal que deberemos canjear por un *id_token*. Por tanto en las credenciales de nuestra aplicación deberemos configurar

el '*redirect_uri*' a '*http://localhost:8080/token*' tal y como aparece en el esqueleto. El *id_token* que obtendremos será un JWT con la información del usuario, del que extraeremos la dirección de e-mail. Finalmente se generará una página HTML de bienvenida con el mensaje "*Bienvenido <e-mail>*".

Para conseguir la máxima puntuación en el apartado B es necesario utilizar *tokens anti CSRF* para distinguir entre peticiones legítimas y maliciosas.

Referencias útiles:

- Transparencias sobre autenticación delegada.
- Documentación sobre OpenID Connect en Google: https://developers.google.com/identity/protocols/OpenIDConnect#server-flow