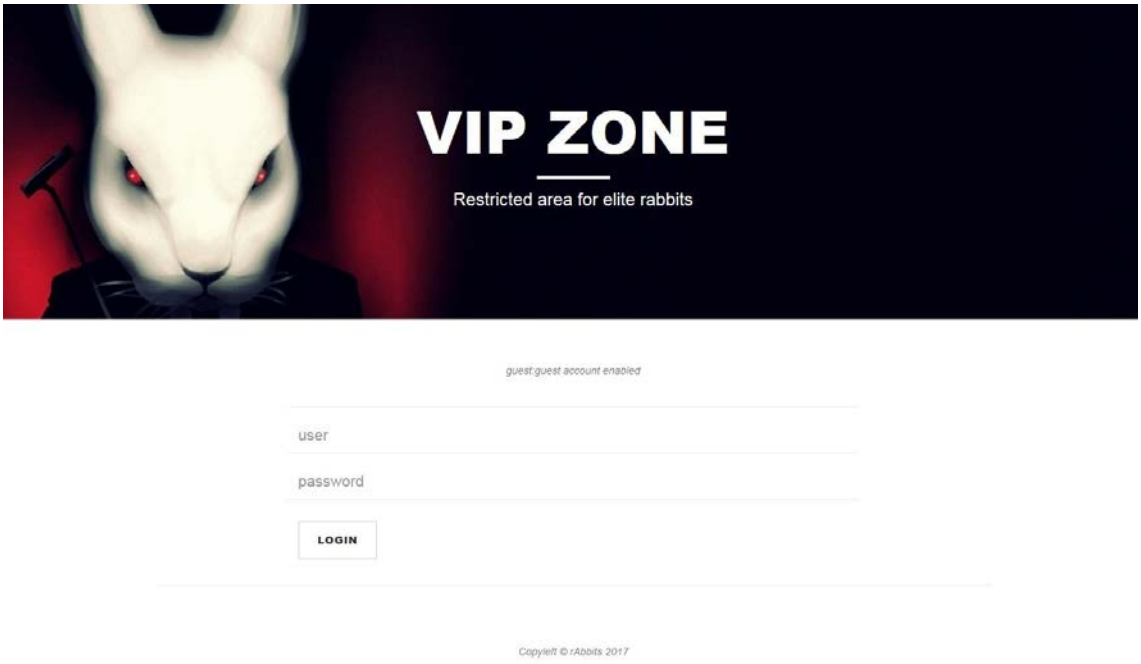
HACKING WEB
FWHIBBIT CTF TEAM

Writeup para el reto "VIP Area" (350 pts, Canadá) by Penkali

Link: http://web2.ctf.followthewhiterabbit.es:8002

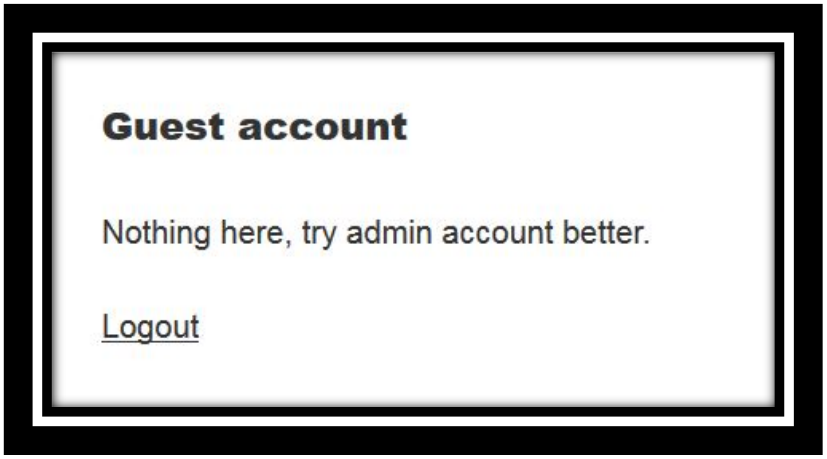
Description: Restricted area only for elite rabbits. Do you want to be part of this group? Then GO AHEAD!

En primer lugar nos dirigimos a la web y nos encontramos con lo siguiente:



Básicamente nos invita a entrar con la cuenta guest:guest. ¿Qué puede fallar?

Si entramos con esa cuenta nos encontramos este mensaje:

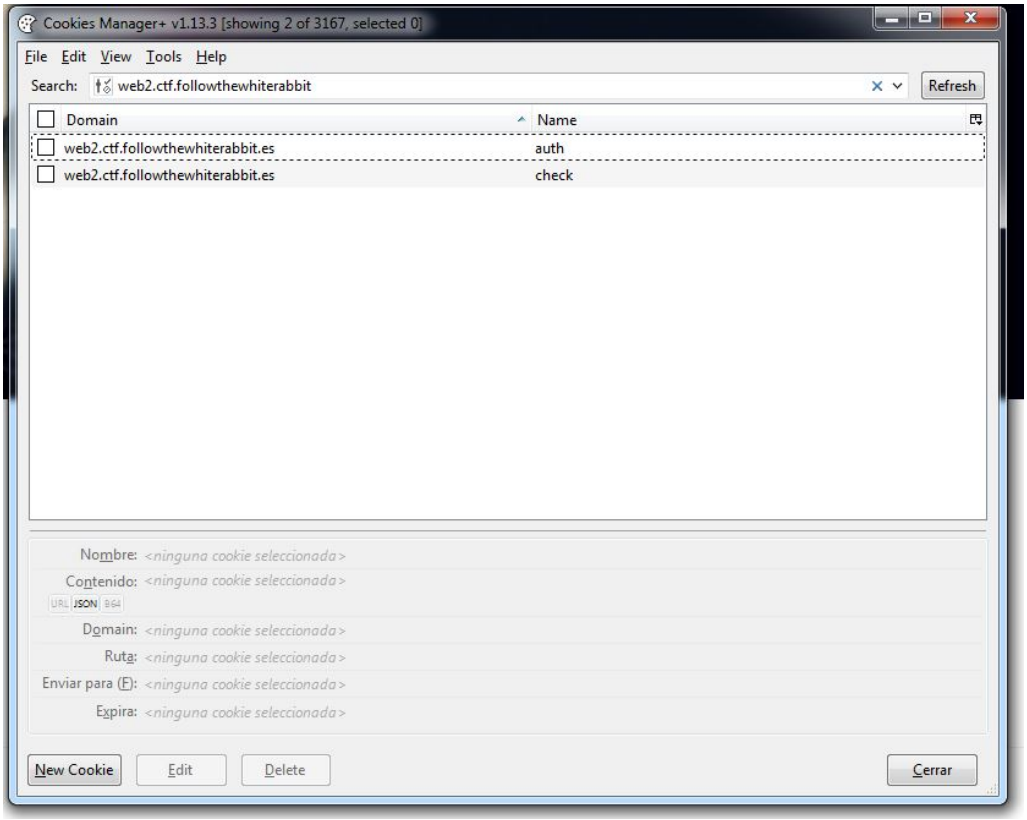


Vaya. Parece que tendremos que encontrar de otro modo nuestra flag. Qué decepción si no fuese así, ¿Verdad?

Después de mirar el código de la página por si hubiese alguna pista y trastear un poco podemos caer un detalle:

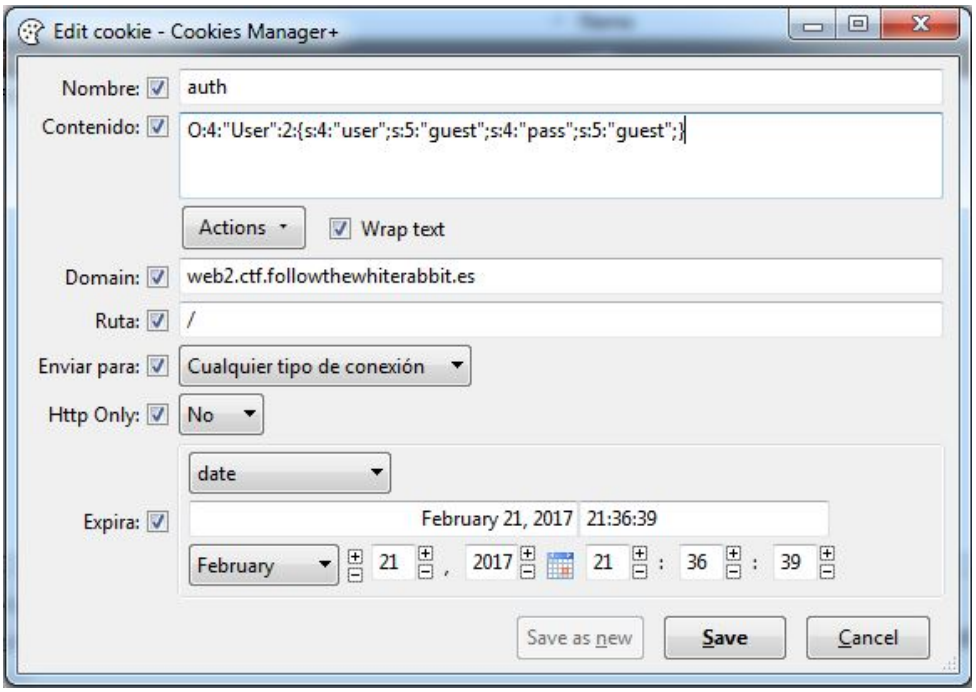
Nos han hecho entrar, algo quieren que veamos...

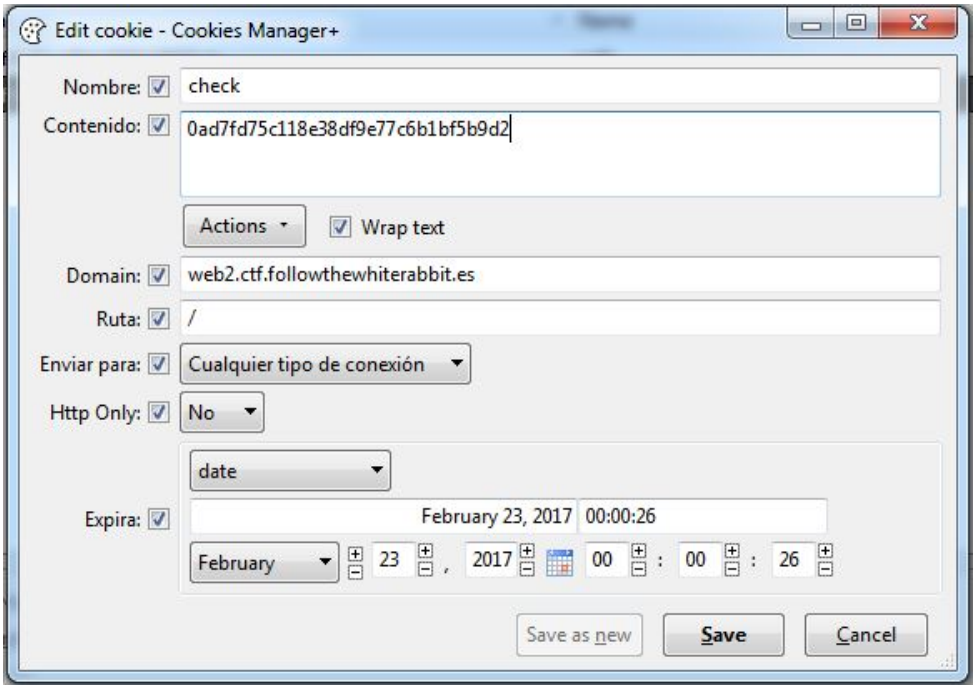
Una de las formas más utilizadas para seguir al usuario y guardar su sesión suelen ser las cookies, por lo que vamos a echar un ojo no sea que tengamos alguna en esta página. Personalmente para ello siempre me ha gustado el plugin the Firefox "Cookies Manager+".



¡Premio! Tenemos dos cookies. Por su nombre podemos intuir que una va a guardar nuestra autentificación y la otra va a realizar algún tipo de comprobación. No siempre hay que fiarse, pero se puede tirar por ahí.

El contenido de ambas cookies se presenta en base64 con URL encoding. Si deshacemos estas codificaciones encontraremos esto:





En primer lugar tenemos un objeto de PHP serializado, es decir, una representación en string del objeto que puede ser cargada para reconstruirlo.

Vamos a hacerle una pequeña disección para dejar claro qué es cada cosa:

O:4:"User":2:{s:4:"user";s:5:"guest";s:4:"pass";s:5:"guest";}

O:4:"User":2 - Objeto

Objeto:<Long_del_nombre_de_la_clase>:"Nombre_de_la_clase":<Número_de_propiedades>

{s:4:"user";s:5:"guest";s:4:"pass";s:5:"guest";} - Propiedades

s:4:"user";s:5:"guest" - Primera propiedad

String:<Long_del_nombre_de_la_propiedad>:"Nombre_de_la_propiedad";

<Tipo_del_valor>:<Long_del_valor_de_la_propiedad>:"Valor_de_la_propiedad"

Lo mismo se aplica a la segunda propiedad. Ambas nos dan nuestro valor para "usuario" y "contraseña".

Conocemos una vulnerabilidad típica en PHP como viene siendo el uso de comparaciones flexibles en lugar de estrictas. Es decir, cuando alguien en su código de PHP escribe "Contraseña == Input" en lugar de "Contraseña === Input". En el segundo caso se compara primero el tipo de las variables comparadas y después su valor, mientras que en el primero podemos obtener resultados como 'True == "Texto" --> True'.

Vamos a explotar esa vulnerabilidad alterando el objeto:

O:4:"User":2:{s:4:"user";s:5:"**admin**";s:4:"pass";**b:1**;}

Con ese objeto hemos alterado nuestro usuario y nuestra contraseña. El nombre del usuario pasa a ser "admin" y nuestra contraseña se ha convertido en una propiedad booleana de valor "True".

Por desgracia, si almacenamos este objeto en la cookie y la codificamos (1º base64; 2º Url Encoding) nos iremos de patitas a la pantalla de login pese a que nuestras cookies siguen activas.

Algo nos estamos dejando. Es como si hubiese algún tipo de comprobación más o... Check.

¿Recordáis esa otra cookie? Pues resulta que sí que parece significar algo. De algún modo debe de tener alguna relación con la otra cookie, por lo que cogemos el objeto serializado sin modificar y comprobamos varios hash para él. El resultado no es muy alentador pues ninguno tiene nada que ver con lo que esperábamos encontrar.

En mi caso, mientras le daba vueltas a qué podía estar faltando jugueteaba con las opciones del Cookie Manager+ y cuando hice clic sobre las acciones volvía ver: URL encode/decode, JSON encode/decode y base64 encode/decode...

"Why? People always ask 'why?' but they never ask 'why not?'"

Probando las distintas codificaciones y sus hashes, de pronto ahí estaba. Ese hash correspondía al objeto serializado y pasado a base64 en md5.

Ya tenemos todas las piezas, por lo que tras coger nuestro objeto y pasarlo a base64 obtenemos el siguiente hash:

6897f0060a84ecb0600e4167d2a748e4

Lo introducimos en la cookie, codificamos en base64 y en url y la guardamos.

Ya estamos listos para cargar de nuevo la página y... "Voilà!"

Más información:

- Comparaciones en PHP:

  http://php.net/manual/es/types.comparisons.php

- Practical PHP Object Injection: Este se va más allá de lo necesario para esta flag, pero explica el formato de objetos serializados muy bien:

  https://www.insomniasec.com/downloads/publications/Practical%20PHP%20Object%20Injection.pdf