

Una al Mes (Episodio 1)

@percu

15/05/18

Contenido

MISIÓN	1
RESOLUCIÓN	2
PARTE 1 – WEB	2
PARTE 2 - REVERSING	6
REFERENCIAS	9

Challenge

2 Solves

X

EPISODIO 1

100

Hemos conseguido entrar en la Fábrica Nacional de Moneda y Timbre. Pero una vez dentro, la lanza térmica que usaríamos para abrir la caja fuerte se ha roto. Debes descubrir los códigos para abrirla, y con ello conseguirás la contraseña para el zip del programa que genera la flag y el dinero ;).

Caja fuerte:
<http://34.253.233.243/lacasadepapel/episodio1/puerta.php>

Info: La flag tiene el formato UAM{md5}

TOP 3:

1. SPC
2. cukz

 episodio1.zip

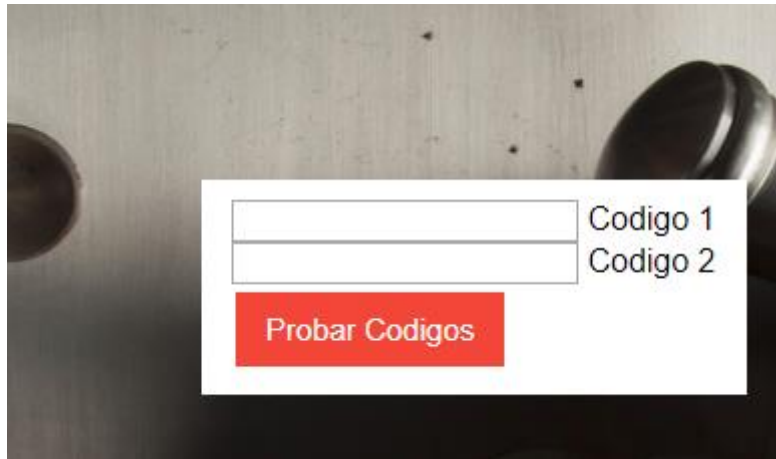
Flag

Submit

RESOLUCIÓN

PARTE 1 – WEB

El primer paso es acceder a la url <http://34.253.233.243/lacasadepapel/episodio1/puerta.php> indicada, donde vemos un formulario para acceder:



Codigo 1
Codigo 2
Probar Codigos

Probamos algunas queries para ver si es vulnerable a SQLi pero sin éxito.

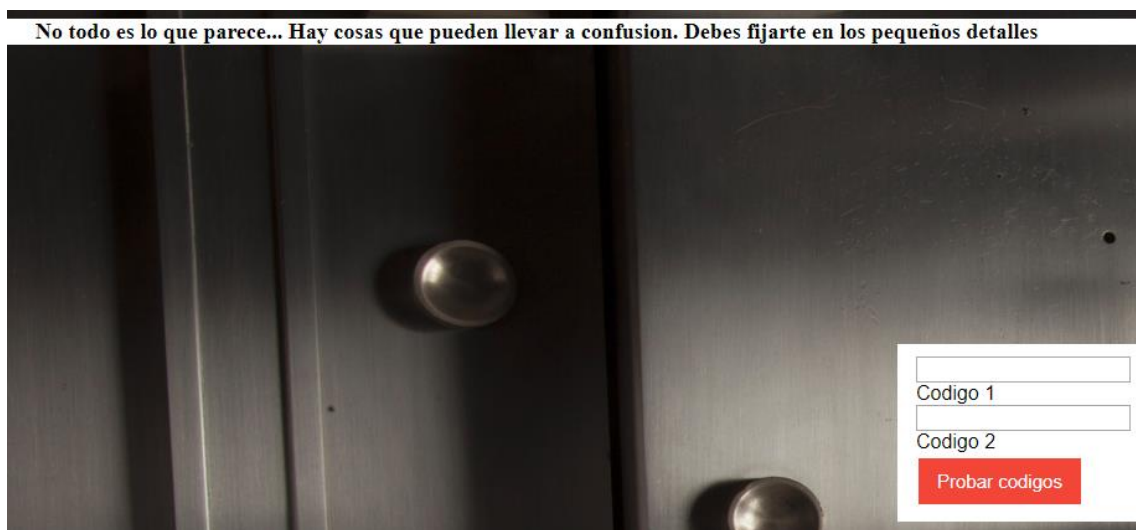
Si miramos el código fuente vemos:

```
<html>
  <head>...</head>
  <body> == $0
    <form action="puerta.php" method="post">...</form>
    <!-- --
      Soy mas de 1234/1234 que de admin/admin
    ---->
  </body>
</html>
```

Así que parece que va a ser 1234 / 1234.

Probamos pero no podía ser tan fácil:

No todo es lo que parece... Hay cosas que pueden llevar a confusión. Debes fijarte en los pequeños detalles

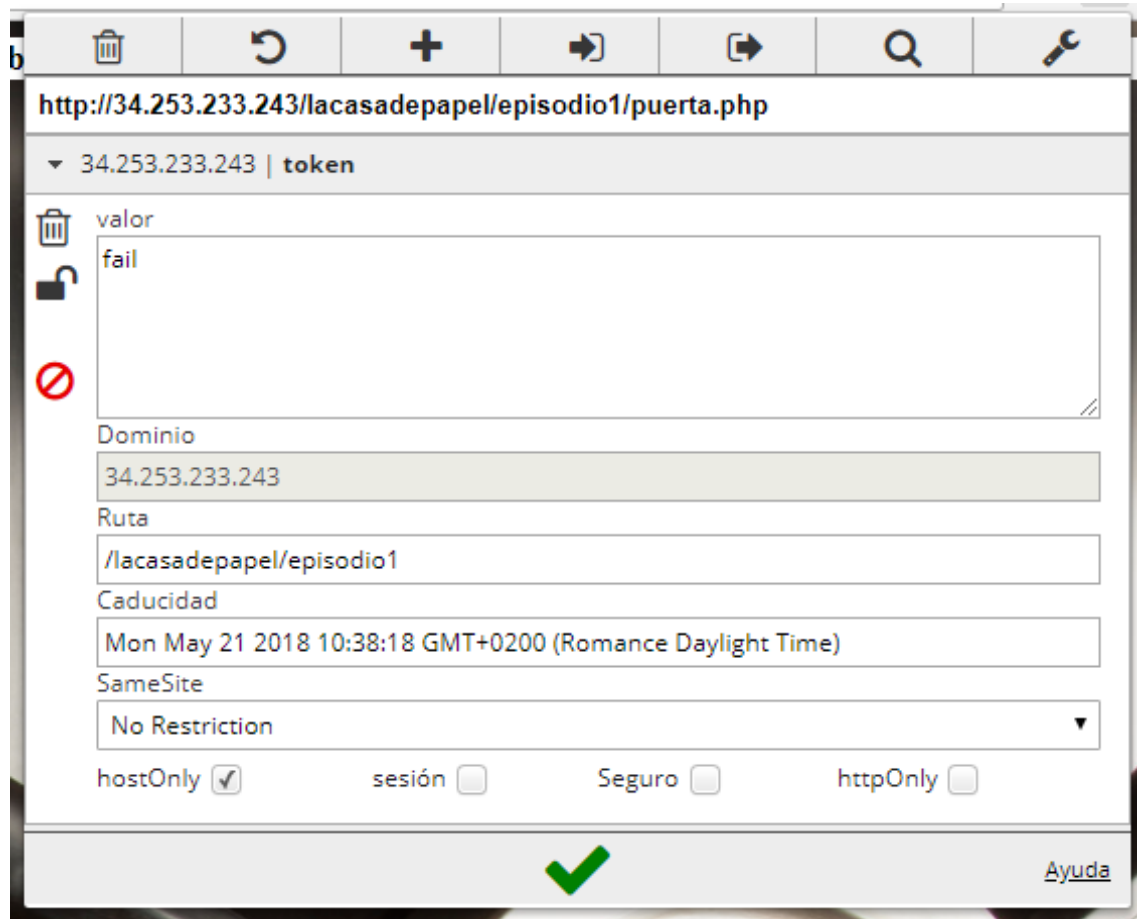


Codigo 1
Codigo 2
Probar codigos

Parece que debemos fijarnos en los pequeños detalles.

Con el Burp miramos las peticiones y respuestas para ver si redirecciona a otra URL, pero no es el caso.

Solo vemos que nos pasa una cookie de nombre *token* y valor *fail*:



Probamos de cambiar a *success* el valor, pero el resultado es el mismo.

Inspeccionando el código vemos que carga un código javascript:

```
<html>
  <head>
    <title>Apertura con codigo</title>
    <script language="JavaScript" src="login.js"></script>
    <style>...</style>
    <link type="text/css" rel="stylesheet" href="chrome-extensio
  </head>
  <body> == $0
    <form action="puerta.php" method="post">
      <div> </div>
```

Procedemos a ver si podemos acceder a ese fichero .js
(<http://34.253.233.243/lacasadepapel/episodio1/login.js>)

Vemos que es una función comentada, por lo que no tiene aplicación en el código. Aún así nos da alguna pista:

```

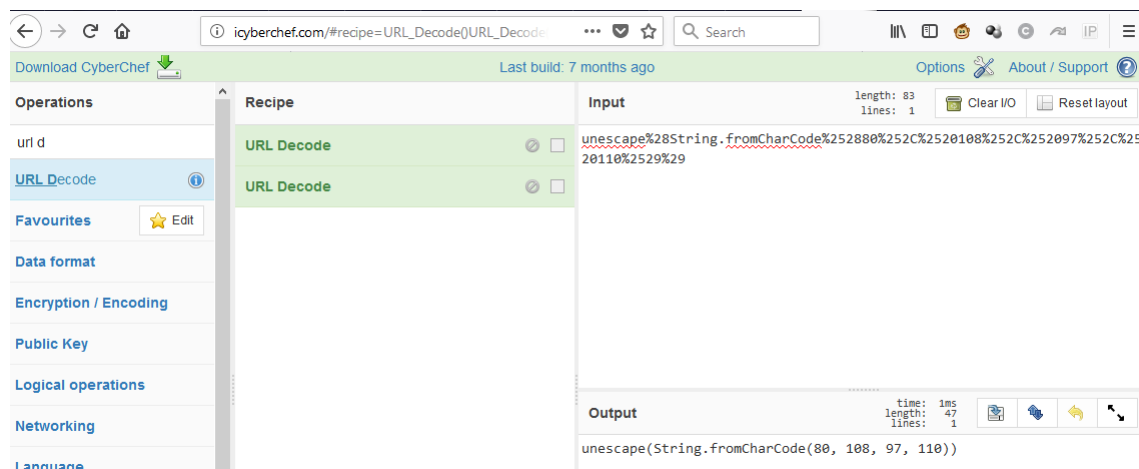
/*
function conexion(){
    var Password =
"unescape%28String.fromCharCode%252880%252C%2520108%252C%252097%252C%2520110%25
29%29:KZQWYZLOMNUWC===";
    for (i = 0; i < Password.length; i++)
    {
        if (Password[i].indexOf(code1) == 0)
        {
            var TheSplit = Password[i].split(":");
            var code1 = TheSplit[0];
            var code2 = TheSplit[1];
        }
    }
}
*/

```

Parece ser que los códigos correctos (code1 y code2) tienen que ser la primera y la segunda parte de la variable *Password*, que es una cadena separada por el símbolo ‘:’

La primera parte de la cadena

(unescape%28String.fromCharCode%252880%252C%2520108%252C%252097%252C%2520110%2529%29) la tendremos que pasar dos veces por un URL decoder para verla mejor:



Nos queda la función javascript:

`unescape(String.fromCharCode(80, 108, 97, 110))`

La función *String.fromCharCode* crea una cadena de caracteres a partir de sus códigos ASCII.

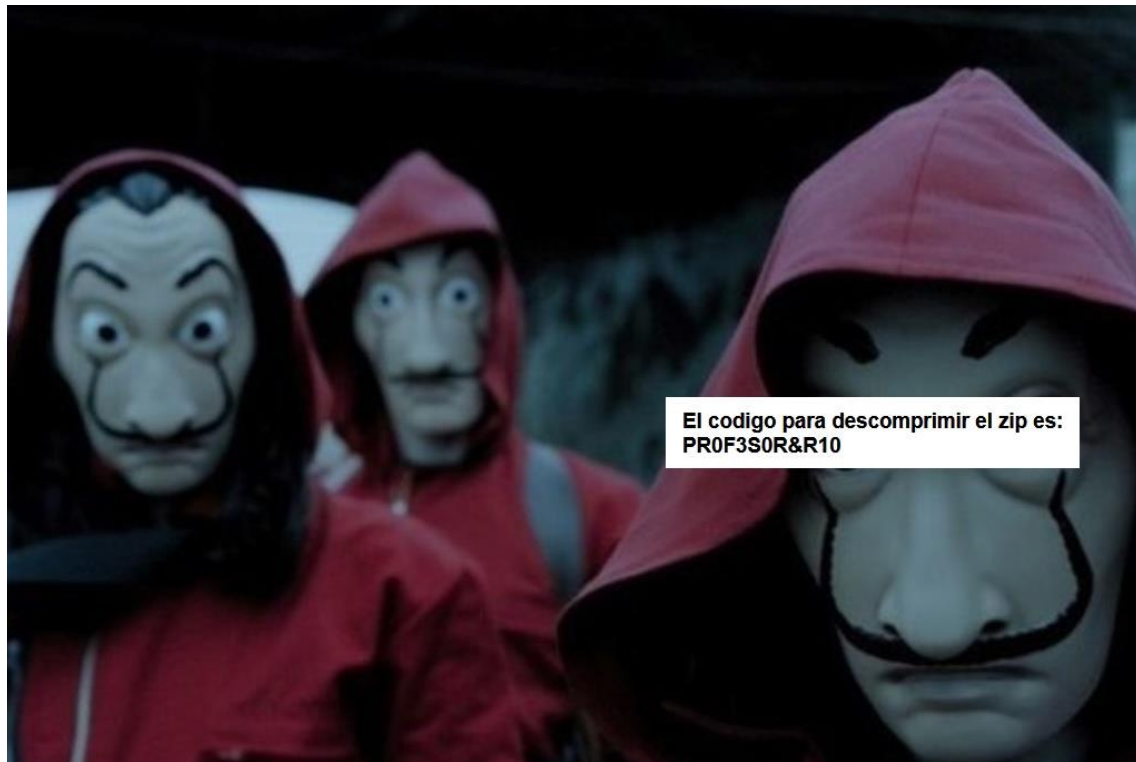
Así que la cadena codificada asignada posteriormente a la variable *code1* es: **Plan**

La segunda parte de la variable *Password* parece una cadena codificada en base 64 (KZQWYZLOMNUWC==). Si la decodificamos en base64 nos queda la cadena “)..a.Î0Õ..” que no parece arrojar nada claro. Probamos a decodificarla en **base32** y la cadena resultante es **Valencia**.

Así que parece ser que las claves a introducir en el formulario son **Plan** y **Valencia**:

Plan	Codigo 1
Valencia	Codigo 2
Probar codigos	

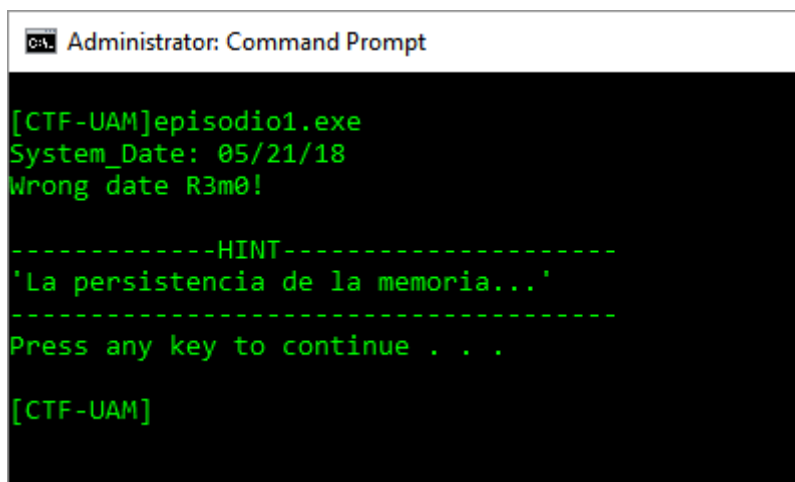
Al introducirlos nos muestra la siguiente web:



Así que ya tenemos el código para descomprimir el fichero .zip del enunciado. Y es:
PROF3S0R&R10

PARTE 2 - REVERSING

Una vez descomprimido el fichero obtenemos un ejecutable de nombre episodio1.exe. Lo ejecutamos. Nos llevamos un remazo con un mensaje que nos informa de que la fecha no es correcta y nos da una pista “La persistencia de la memoria”:



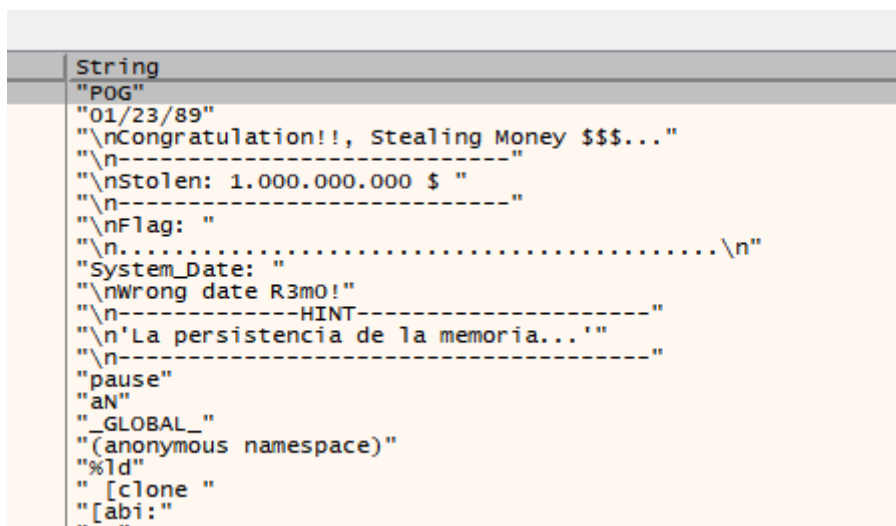
```
Administrator: Command Prompt

[CTF-UAM]episodio1.exe
System_Date: 05/21/18
Wrong date R3m0!

-----HINT-----
'La persistencia de la memoria...'
-----HINT-----
Press any key to continue . . .

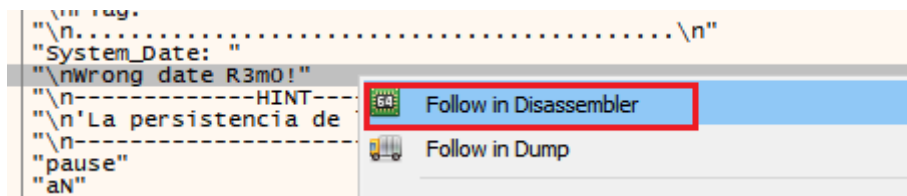
[CTF-UAM]
```

Utilizando el x64dbg abrimos el ejecutable *episodio1.exe* y buscamos las referencias de cadenas en todos sus módulos:



```
String
"POG"
"01/23/89"
"\nCongratulation!!, Stealing Money $$$..."
"\n-----HINT-----"
"\nStolen: 1.000.000.000 $ "
"\n-----HINT-----"
"\nFlag: "
"\n.....\n"
"System_Date: "
"\nWrong date R3m0!"
"\n-----HINT-----"
"\nLa persistencia de la memoria..."
"\n-----HINT-----"
"pause"
"aN"
"GLOBAL_"
"(anonymous namespace)"
"%ld"
"[clone "
"[abi:"
```

Observamos que la respuesta devuelta por el ejecutable está codificada en el programa. Así que iremos a la dirección del programa donde nos dice “Wrong date R3m0!”:



```
String
"POG"
"01/23/89"
"\nCongratulation!!, Stealing Money $$$..."
"\n-----HINT-----"
"\nStolen: 1.000.000.000 $ "
"\n-----HINT-----"
"\nFlag: "
"\n.....\n"
"System_Date: "
"\nWrong date R3m0!"
"\n-----HINT-----"
"\nLa persistencia de la memoria..."
"\n-----HINT-----"
"pause"
"aN"
"GLOBAL_"
"(anonymous namespace)"
"%ld"
"[clone "
"[abi:"

Follow in Disassembler
Follow in Dump
```

Situados en este punto del código analizamos:

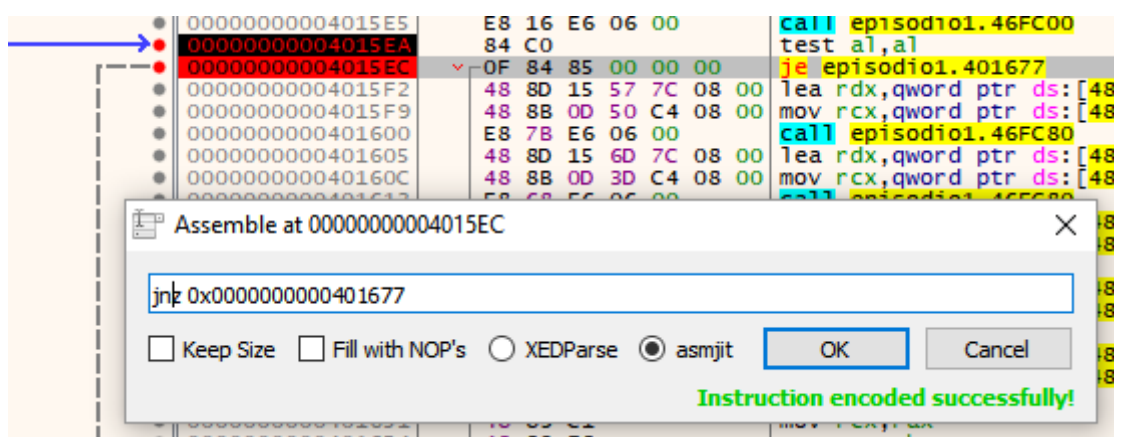
00000000004015E5	E8 16 E6 06 00	call episodio1.46FC00	
00000000004015EA	84 C0	test al,al	
00000000004015EC	0F 84 85 00 00 00	je episodio1.401677	
00000000004015F2	48 8D 15 57 7C 08 00	lea rdx,qword ptr ds:[489250]	0000000000489250:"\nCongratulation!!, Stealing Money \$\$\$..."
00000000004015F9	48 8B 0D 50 C4 08 00	mov rcx,qword ptr ds:[48DA50]	
0000000000401600	E8 7B E6 06 00	call episodio1.46FC80	
0000000000401605	48 8D 15 6D 7C 08 00	lea rdx,qword ptr ds:[489279]	0000000000489279:"\n-----"
000000000040160C	48 8B 0D 3D C4 08 00	mov rcx,qword ptr ds:[48DA50]	
0000000000401613	E8 69 E6 06 00	call episodio1.46FC80	
0000000000401618	48 8D 15 78 7C 08 00	lea rdx,qword ptr ds:[489297]	0000000000489297:"\nStolen: 1.000.000.000 \$ "
000000000040161F	48 8B 0D 2A C4 08 00	mov rcx,qword ptr ds:[48DA50]	
0000000000401626	E8 55 E6 06 00	call episodio1.46FC80	
0000000000401632	48 8D 15 47 7C 08 00	lea rdx,qword ptr ds:[489279]	0000000000489279:"\n-----"
0000000000401639	48 8B 0D 17 C4 08 00	mov rcx,qword ptr ds:[48DA50]	
0000000000401651	E8 42 E6 06 00	call episodio1.46FC80	
0000000000401654	48 8B 0D 04 C4 08 00	mov rcx,qword ptr ds:[48DA50]	0000000000489281:"\nFlag: "
000000000040164C	E8 2F E6 06 00	call episodio1.46FC80	
0000000000401651	48 89 C1	mov rcx,rcx	
0000000000401654	48 89 E8	mov rcx,rbp	
0000000000401657	E8 61 E8 06 00	call episodio1.46FC80	
000000000040165A	48 8D 15 5A 7C 08 00	lea rdx,qword ptr ds:[4892C0]	00000000004892C0:"\n.....\n"
0000000000401666	48 8B 0D E3 C3 08 00	mov rcx,qword ptr ds:[48DA50]	
000000000040166D	E8 0E E6 06 00	call episodio1.46FC80	
0000000000401672	E9 94 00 00 00	jmp episodio1.401708	
0000000000401677	48 8D 15 70 7C 08 00	lea rdx,qword ptr ds:[4892EE]	00000000004892EE:"System_Date: "
000000000040167E	48 8B 0D CB C3 08 00	mov rcx,qword ptr ds:[48DA50]	
0000000000401685	E8 F6 E5 06 00	call episodio1.46FC80	
000000000040168A	48 89 C1	mov rcx,rcx	
000000000040168D	48 8D 45 10	lea rax,qword ptr ss:[rbp+10]	
0000000000401691	48 89 C2	mov rdx,rcx	
0000000000401694	E8 E7 E5 06 00	call episodio1.46FC80	
0000000000401699	48 8D 15 3C 7C 08 00	lea rdx,qword ptr ds:[4892FC]	00000000004892FC:"\nwrong_date R3m0!"
00000000004016A0	48 8B 0D A9 C3 08 00	mov rcx,qword ptr ds:[48DA50]	
00000000004016A7	E8 D4 E5 06 00	call episodio1.46FC80	
00000000004016AC	48 8D 15 58 7C 08 00	lea rdx,qword ptr ds:[48930E]	
00000000004016B3	48 8B 0D 96 C3 08 00	mov rcx,qword ptr ds:[48DA50]	
00000000004016BA	E8 C1 E5 06 00	call episodio1.46FC80	
00000000004016B8	48 8D 15 4A 7C 08 00	lea rdx,qword ptr ds:[489310]	0000000000489310:"\n-----HINT-----"
00000000004016B8	48 8B 0D 83 C3 08 00	mov rcx,qword ptr ds:[48DA50]	
00000000004016CD	E8 AE E5 06 00	call episodio1.46FC80	
00000000004016D2	48 8D 15 5F 7C 08 00	lea rdx,qword ptr ds:[489338]	0000000000489338:"\nLa persistencia de la memoria..."
00000000004016D9	48 8B 0D 70 C3 08 00	mov rcx,qword ptr ds:[48DA50]	
00000000004016E0	E8 35 E5 06 00	call episodio1.46FC80	
00000000004016E5	48 8D 15 74 7C 08 00	lea rdx,qword ptr ds:[489360]	0000000000489360:"\n-----"
00000000004016EC	48 8B 0D 5D C3 08 00	mov rcx,qword ptr ds:[48DA50]	
00000000004016F3	E8 85 E5 06 00	call episodio1.46FC80	
00000000004016F9	48 8B 0D 0F 7C 08 00	mov rcx,qword ptr ds:[48930E]	
00000000004016FF	48 8B 0D 4A C3 08 00	mov rcx,qword ptr ds:[48DA50]	
0000000000401706	E8 75 E5 06 00	call episodio1.46FC80	
000000000040170B	48 8D 15 76 7C 08 00	lea rdx,qword ptr ds:[489388]	0000000000489388:"pause"
0000000000401717	FR A9 A6 01 00	call centinela.system	

Se aprecia un salto en la dirección 0x4015EC donde comprueba si el valor de un mismo registro (AL) es igual. Evidentemente esa comprobación será siempre cierta. Si se cumple esa condición hay un salto hacia el apartado donde nos dice que la fecha del sistema es incorrecta y finaliza el programa.

Así que lo que haremos será establecer un breakpoint en la dirección 0x4015EA y en tiempo de ejecución modificaremos la instrucción de la dirección 0x4015EC (je episodio1.401677) por jne episodio1.401677:

00000000004015E5	E8 16 E6 06 00	call episodio1.46FC00	
00000000004015EA	84 C0	test al,al	
00000000004015EC	0F 84 85 00 00 00	je episodio1.401677	
00000000004015F2	48 8D 15 57 7C 08 00	lea rdx,qword ptr ds:[489250]	0000000000489250:"\nCongratulation!!
0000000000401600	48 8B 0D 50 C4 08 00	mov rcx,qword ptr ds:[48DA50]	
0000000000401605	E8 7B E6 06 00	call episodio1.46FC80	
000000000040160C	48 8D 15 6D 7C 08 00	lea rdx,qword ptr ds:[489279]	0000000000489279:"\n-----"

Le damos a run y en cuanto llegue al breakpoint modificamos la instrucción:



Seguimos la ejecución con Run... y vemos como efectivamente entra en el bucle correcto:

00000000004015E5	E8 16 E6 06 00	call episodio1.46FC00	
00000000004015EA	84 C0	test al,al	
00000000004015EC	0F 85 85 00 00 00	jne episodio1.401677	
00000000004015F2	48 8D 15 57 7C 08 00	lea rdx,qword ptr ds:[489250]	0000000000489250:"\nCongratulation!!, Stealing Money \$\$\$..."
00000000004015F9	48 8B 0D 50 C4 08 00	mov rcx,qword ptr ds:[48DA50]	


```
\episodio1.exe

Congratulation!!, Stealing Money $$$..._
```

Finalizamos la ejecución y obtenemos la flag (e30f35ad8d9cb6efc0778539a669fa85)

```
\episodio1.exe

Congratulation!!, Stealing Money $$$...
-----
Stolen: 1.000.000.000 $
-----
Flag: e30f35ad8d9cb6efc0778539a669fa85
.....
Press any key to continue . . . _
```

Analizando la pista “La persistencia de la memoria” que nos da en la primera ejecución, descubrimos que es un cuadro de Eugenio Salvador Dalí pintado en 1931. Dalí murió el 23/01/1989.

Si nos fijamos otra vez en las variables del .exe vemos que hay una que tiene por valor “01/23/89”, que es la fecha de su muerte:

All Modules (Strings)		
Address	Disassembly	String
00000000004010AF	mov rax,qword ptr ds:[48D560]	"000"
000000000040155D	lea rdx,qword ptr ds:[489240]	"01/23/89"
00000000004015F2	lea rdx,qword ptr ds:[489250]	"\nCongratulation!!, Stealing Money \$\$\$..."
0000000000401605	lea rdx,qword ptr ds:[489279]	"\n-----"
0000000000401618	lea rdx,qword ptr ds:[489279]	"\nStolen: 1.000.000.000 \$ "
0000000000401628	lea rdx,qword ptr ds:[489279]	"\n-----"
000000000040163E	lea rdx,qword ptr ds:[489281]	"\nFlag: "
000000000040165F	lea rdx,qword ptr ds:[4892C0]	"\n....."
0000000000401677	lea rdx,qword ptr ds:[4892EE]	"System_Date: "
0000000000401699	lea rdx,qword ptr ds:[4892FC]	"\nWrong date R3m0!"
00000000004016BF	lea rdx,qword ptr ds:[489310]	"\n-----HINT-----"
00000000004016D2	lea rdx,qword ptr ds:[489338]	"\n'La persistencia de la memoria...'"
00000000004016E5	lea rdx,qword ptr ds:[489360]	"\n-----"
0000000000401708	lea rcx,qword ptr ds:[489388]	"pause"
0000000000401780	lea r8,qword ptr ds:[489748]	"sk"

Ese valor 01/23/89 codificado en MD5 nos da e30f35ad8d9cb6efc0778539a669fa85 que es la flag a encontrar.

Por lo tanto el exe va codificando a MD5 esa variable, que es la que nos muestra finalmente el programa al conseguir modificar el salto condicional.

UAM{e30f35ad8d9cb6efc0778539a669fa85}

REFERENCIAS

<https://portswigger.net/burp/communitydownload>

<http://icyberchef.com/>

<http://multiencoder.com/>

<https://x64dbg.com/>